UNIVERSITY OF THESSALY
SCHOOL OF TECHNOLOGY
DEPARTMENT OF DIGITAL SYSTEMS

MASTER'S DEGREE PROGRAM
"MODERN COMMUNICATION SYSTEMS
AND THE INTERNET OF THINGS"

*"**AllERT**: Flood disaster management system using IoT for prompt response and recovery efforts"*

Dimitra A. Kokkinou - 2024
Supervisor: Panagiotis Fitsilis

UNIVERSITY OF THESSALY

School of Technology

Department of Digital Systems

"MODERN COMMUNICATION SYSTEMS AND THE INTERNET of THINGS"

"Master of Science in Modern Communication Systems and the Internet of Things"

"*AllERT*[*]: Flood disaster management system using IoT for prompt response and recovery efforts"

Postgraduate Thesis

submitted to the Department of Digital Systems of the University of Thessaly as part of the requirements for the acquisition of a Master's Degree in Modern Communication Systems and Internet of Things by

Dimitra A. Kokkinou

Declaration of Authenticity, Copyright issues

[*] ***All Emergency in Real Time**: The AllERT protocol is designed to facilitate coordination among "**all emergency" services.***

Tripartite Commission of Inquiry

This thesis has been unanimously approved by the three-member examination committee appointed by the Assembly of the Department of Digital Systems of the University of Thessaly, according to the law and the approved Study Guide of the MSc "Modern Communication Systems and Internet of Things".

The members of the Committee were:

- Prof. Panagiotis Fitsilis (Supervisor)
- Prof. George Karetsos (Member)
- Dr.-Ing. Fotios Gioulekas (Member)

The approval of the thesis by the Department of Digital Systems of the University of Thessaly does not imply acceptance of the author's views

Acknowledgments

It is my strong belief that our goals themselves equip us with the energy needed to reach them. This project took many hours over an extended period researching and working seamlessly, as well as many sacrifices. The challenge of resignation has always been present. Choosing the subject of my thesis was due to a strong motivation to offer help. Ironically, almost immediately after that, Storm Daniel hit Thessaly and caused extensive destruction to large areas of land. It then became a matter of personal responsibility. Unexpected events occurred, requirements that had not been foreseen intervened and new issues arose. However, the excitement of accomplishment and the achievement is special, for which I wish first and foremost to thank my mother and my children. Each of them made a significant contribution in their own way and without them I would not have succeeded.

I would like to express my gratitude to my supervisor Mr. Panos Fitsilis for his guidance and the opportunity to work under his mentorship during my research. I would also like to thank all the professors of the postgraduate program "CSIoT" for providing valuable information and guidance. Each individually put in their own building block to build *AllERT*.

Abstract

Natural disasters, in particular floods, pose major threats to human life and infrastructure in affected areas and require rapid and effective response strategies. This thesis presents a new Decision Support System (DSS) protocol called *AllERT*, which is designed for real-time coordination of emergency services during flood events. The system integrates Internet of Things (IoT) devices, Unmanned Aerial Vehicles (UAVs) and satellite communications to enhance situational awareness and improve the allocation of resources needed for response.

The AllERT protocol is triggered upon detection of flood indicators, using a network of flood sensors strategically placed in urban and rural areas. The sensors provide real-time data, which is immediately processed by the DSS to generate actionable information. UAVs enter the system to capture high-resolution images of the affected areas, allowing accurate mapping of the extent of the damage. This information, combined with pre-existing resource inventories and route planning algorithms implemented in MATrix LABoratory (MATLAB) and ArcGIS, supports the system in making updated decisions for responding to the mandated emergency.

The system architecture places a strong emphasis on ensuring reliable communication practices, proposing a structure for seamless transmission that resolves scenarios where traditional networks fail. The methodology includes a well-organized simulation and pilot demonstration test, ensuring the reliability and effectiveness of the system in real applications.

The findings of the research suggest that the AllERT protocol provides a superior solution for flood emergency management compared to existing systems. The system's ability to process and analyze data in real-time ensures that emergency responders can act quickly and efficiently, saving lives and allocating human and material resources with accuracy.

Keywords: Flood Response, Decision Support System, IoT, UAV, Satellite Communication, Real-Time Data Processing, Emergency Management.

Περίληψη

Οι φυσικές καταστροφές, ιδίως οι πλημμύρες, αποτελούν σημαντικές απειλές για την ανθρώπινη ζωή και τις υποδομές των περιοχών που πλήττονται και απαιτούν ταχείες και αποτελεσματικές στρατηγικές αντιμετώπισης. Η παρούσα διατριβή παρουσιάζει ένα νέο πρωτόκολλο συστήματος υποστήριξης αποφάσεων (DSS) με την ονομασία *AllERT*, το οποίο έχει σχεδιαστεί για το συντονισμό των υπηρεσιών έκτακτης ανάγκης σε πραγματικό χρόνο κατά τη διάρκεια πλημμυρικών φαινομένων. Το σύστημα ενσωματώνει συσκευές του Διαδικτύου των Πραγμάτων (IoT), μη επανδρωμένα αεροσκάφη (UAVs) και δορυφορικές επικοινωνίες για την ενίσχυση της επίγνωσης της κατάστασης και τη βελτίωση της κατανομής των πόρων που απαιτούνται για την αντιμετώπιση των συνεπειών.

Το πρωτόκολλο *AllERT* ενεργοποιείται κατά την ανίχνευση δεικτών πλημμύρας, χρησιμοποιώντας ένα δίκτυο αισθητήρων πλημμύρας που είναι στρατηγικά τοποθετημένοι σε αστικές και αγροτικές περιοχές. Οι αισθητήρες παρέχουν δεδομένα σε πραγματικό χρόνο, τα οποία υπόκεινται αμέσως σε επεξεργασία από το DSS για να δημιουργήσουν πληροφορίες που μπορούν να χρησιμοποιηθούν. Τα UAV εισέρχονται στο σύστημα για τη λήψη εικόνων υψηλής ανάλυσης από τις πληγείσες περιοχές, επιτρέποντας την ακριβή χαρτογράφηση της έκτασης της ζημιάς. Οι πληροφορίες αυτές, σε συνδυασμό με προϋπάρχουσες απογραφές πόρων και αλγορίθμους σχεδιασμού διαδρομών που έχουν υλοποιηθεί σε MATLAB και ArcGIS, υποστηρίζουν το σύστημα στη λήψη βέλτιστων αποφάσεων για την αντιμετώπιση της επιβεβλημένης έκτακτης ανάγκης.

Η αρχιτεκτονική του συστήματος δίνει μεγάλη έμφαση στη διασφάλιση αξιόπιστων πρακτικών επικοινωνίας, προτείνοντας μια δομή για απρόσκοπτη μετάδοση που επιλύει σενάρια όπου τα παραδοσιακά δίκτυα αποτυγχάνουν. Η μεθοδολογία περιλαμβάνει μια καλά οργανωμένη προσομοίωση και πιλοτική δοκιμή επίδειξης, διασφαλίζοντας την αξιοπιστία και την αποτελεσματικότητα του συστήματος σε πραγματικές εφαρμογές.

Τα ευρήματα της έρευνας υποδηλώνουν ότι το πρωτόκολλο AllERT, παρέχει μια ανώτερη λύση για τη διαχείριση έκτακτης ανάγκης σε περίπτωση πλημμύρας σε σύγκριση με τα υπάρχοντα συστήματα. Η ικανότητα του συστήματος να επεξεργάζεται και να αναλύει δεδομένα σε πραγματικό χρόνο διασφαλίζει ότι οι φορείς αντιμετώπισης εκτάκτων αναγκών μπορούν να δράσουν γρήγορα και αποτελεσματικά, σώζοντας ζωές και κατανέμοντας ανθρώπινους και υλικούς πόρους με ευστοχία.

Table of Contents

1. Introduction

1.1 Background

Disasters can be broadly categorized into two main types: *anthropogenic* and *natural* [1]. Anthropogenic or human-made disasters result from human actions and include events such as industrial accidents, oil spills, nuclear explosions and wars. These disasters often result from negligence, lack of foresight, or the deliberate actions of individuals or groups, leading to significant environmental, economic and human losses. On the other hand, natural disasters are caused by natural processes and phenomena, including earthquakes, hurricanes, floods, wildfires and volcanic eruptions. Unlike anthropogenic disasters, natural disasters are largely beyond human control, although their impact can be exacerbated or mitigated by human activities, such as urban planning and environmental management [2, 3]. Both categories of disasters significantly disrupt the functioning of societies worldwide, thus requiring comprehensive preparedness, response and relief efforts, to minimize their impact on human life and property [1]. Figure 1, depicts the main categories of disasters; however, additional categories could be included.

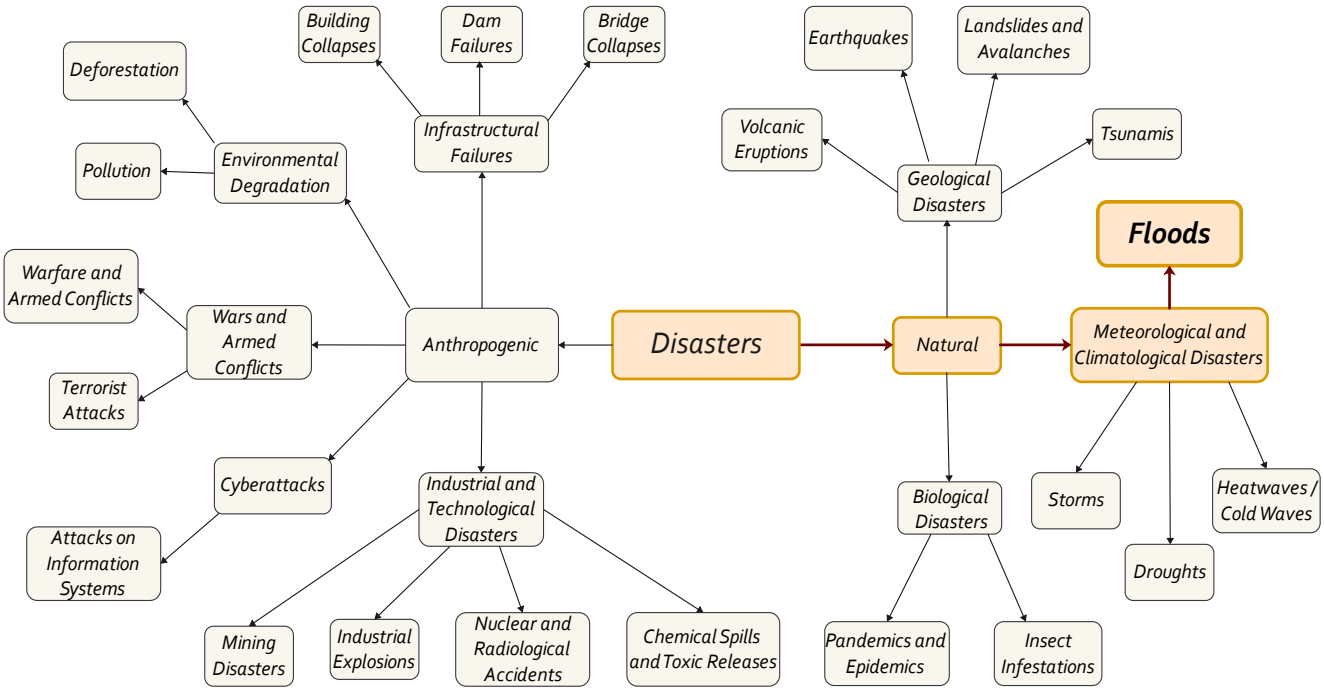


Figure 1: Types of Disasters [1-3]

Amongst the various types of natural disasters, floods are particularly notable for their high frequency of occurrence and their far-reaching impact on communities and ecosystems alike. By definition, floods occur when an area that is usually dry becomes inundated with water, often causing damage to infrastructure, displacement of populations or, at worst, loss of life. They manifest in different forms, such as flash floods, river floods, or urban floods, each with its own unique characteristics and challenges [4]. The causes of floods are also multifaceted and can vary depending on the geographic location and climatic conditions of a particular region. Heavy rainfall is a common trigger for floods, especially in areas with inadequate drainage systems or in regions prone to monsoons. When the ground becomes saturated and cannot absorb any more water, excess runoff accumulates on the surface, leading to flooding. Snowmelt can also contribute to floods, particularly in mountainous regions where large amounts of snow accumulate during the winter months. As temperatures rise, the snow melts and the resulting water runoff can overwhelm rivers and streams, causing them to breach their banks and flood their surrounding areas [4].

In addition to natural causes, human activities can greatly affect the occurrence and impact of floods. Urbanization alters the natural landscape and disrupts the water cycle as the construction of impervious surfaces, such as roads and buildings, reduces the amount of land available for water absorption, which in turn leads to increased runoff during rainfall events that increases the risk of flooding [4]. Deforestation, on the other hand, removes trees and vegetation that help regulate water flow and stabilize soil, further increasing the vulnerability of an area to flooding [5]. Although flooding can result from human failure, this thesis considers its approach only as a natural disaster, otherwise any solution has to be reviewed from a different perspective.

Without a doubt, the impacts of floods can be devastating, affecting both human societies and natural systems. In terms of human impact, floods can result in the loss of lives, injuries and displacement of communities. The destruction of infrastructure, including houses, roads, bridges and utilities, can severely disrupt daily life and hinder economic development. Floodwater can also contaminate water sources, leading to the spread of waterborne diseases and posing long-term health risks to affected populations [4]. An equally important affect is the fact that floods can have significant psychological and emotional effects on individuals and communities, causing trauma, distress and uncertainty [6].

Their environmental impact is not less significant as floods can cause erosion, sedimentation and the destruction of habitats. The excessive water flow can wash away fertile topsoil, massively degrading agricultural lands and reducing productivity. Floods can also lead to the release of pollutants, such as chemicals and sewage, into water bodies, causing threats to aquatic ecosystems and biodiversity. The long-term consequences of floods on the environment can be far-reaching, affecting both terrestrial and aquatic ecosystems thus disrupting ecological balance [7].

Given their destructive nature and the immense challenges they pose to societies worldwide, effective management strategies are of high importance in reducing vulnerability especially in habitable environments, tuning down their intensity when they occur and limiting their impacts. Flood management must involve a combination of structural (dams, levees, flood control channels) [8] and non-structural (land-use planning, early warning systems, community engagement and education) measures aimed at prevention, preparedness, response and recovery by enhancing resilience and at the same time, reducing exposure to flooding risks [9].

1.2 Problem Statement

It is not surprising that over the past few years, the impact of climate change has further deteriorated the occurrence and intensity of natural disasters in general and floods in particular. As a result, in the recent years there has been a growing recognition of the need for the development of effective disaster management systems to enhance response and recovery efforts not only at local and national but also at international levels [10].

However, traditional disaster management systems that are still widely in use today, face multiple challenges in terms of their ability to provide immediate and efficient response and recovery efforts [11–13]. Indeed, the lack of timely availability of qualified personnel and an inability to exploit the potential for skill reutilization are deficiencies which in the field of disaster management can result in devastating consequences [10]. At this level, they fail to meet the real-time needs for communication, data access and analysis amongst all stakeholders which, at the stages when the phenomenon unfolds, is of grave importance [14]. For example, during a disaster, decision-makers, incident commanders, emergency responders and city managers require immediate access to spatial data for coordination and decision-making purposes. Concurrently, they need to conduct mission-critical data analysis to support search and rescue operations. Given the critical nature of

timing in all response operations, it is essential that all potentially necessary datasets are either pre-stored in a centralized repository or pre-connected to an integrated repository. In short, current traditional disaster management systems are unable to process disaster-related information within the framework of an integrated system [14].

1.3 Significance of the Study

In response to those challenges, the integration of the IoT technology has shown promise in revolutionizing these much needed systems with state-of-the-art assistance [15, 16].

IoT, by design, enables the transmission and collection of real-time data from various sensors and devices (Things), thus allowing for enhanced situational awareness and rapid response coordination [17]. IoT devices offer great versatility as they can be deployed in infrastructure monitoring, early warning systems and urban planning, both reducing the negative effects of disasters and improving the ability to recover and adapt. An interconnected network of such IoT-enabled devices facilitates proactive measures in pre-flood periods and provides a comprehensive and real-time mapping of the disaster-affected areas. By taking advantage of the power of such network, disaster management teams can gain critical insights regarding the conditions on the field, enabling them to make informed, precise decisions, targeted interventions as well as allocate resources more effectively and efficiently [16].

As such, the recent application of IoT in disaster management represents a paradigm shift in how the authorities responsibly approach and respond to catastrophic events, especially nowadays that their numbers increase due to the rapid climate change. Its integration offers the potential to improve the speed, performance and efficacy of response and recovery efforts in a notable extend, ultimately saving lives and reducing the overall impact of disasters in society [18, 19].

As technology continues to advance, its role in disaster management growingly becomes a necessity in safeguarding communities and fostering sustainable development. However, it must not be overlooked that in order to utilize the full potential of IoT in this field and to develop more proactive, responsive and resilient disaster management systems, several considerations must be addressed first. These include ensuring the security and privacy of IoT devices and data, promoting interoperability and standardization across different IoT platforms, addressing the challenges posed by limited connectivity in disaster-stricken areas and providing training and capacity building for

stakeholders involved in the implementation and management of IoT-based disaster management systems [20].

As far as the adaptation and integration of the IoT in disaster management systems specific for flood incidents is concerned, one can only be skeptical. A look at the major floods over the past five years internationally, is more than enough to ring the alarm bell that the capabilities of IoT must be used to their fullest as soon as possible. Cyclone Idai Flooding in 2019 [21], Germany and Belgium Floods in 2021 [22], South Africa, Nigeria and Pakistan Floods in 2022 [23–25], and Greece's Floods in 2023 [26] are the sad reminders that there is a lot of work to be done. These events highlight the critical necessity for improved technological integration to enhance disaster preparedness and response. For that reason, the main focus of this review is on flood management strategies in Greece.

Greece's geographical location and geological features make it vulnerable to several types of natural disasters, including earthquakes, wildfires, floods and heatwaves. Among these disasters, flooding is particularly noteworthy due to its frequency and impact on urban and densely populated areas. The recent flooding in Thessaly caused by the Daniel storm [27], underscores the need for prompt development of solutions. The traditional disaster management paradigm, heavily reliant on human intervention and conventional communication technologies (even power outages), often falls short in addressing the immediacy and unpredictability of such crises [28].

The present thesis explores the potential of IoT technologies to revolutionize disaster management, emphasizing immediate response and recovery efforts. Its main objective is the development and implementation of an integrated disaster management and Search And Rescue (SAR) Protocol. By executing this protocol, all the data collected using IoT technologies are used to feed a Decision Support System (DSS). Decision-makers can mobilize resources, evacuate vulnerable populations and initiate response plans long ahead of a flood's onset. Thus, targeted interventions will be carried out by rescue teams and emergency services, practically simulating a Control Tower through which the corresponding instructions will be given. The aim is to improve coordination, communication and intervention in flood scenarios.

While existing protocols and practices have made significant progress, there are still considerable gaps in coordination and communication, particularly in the critical first hours after a flood. The proposed Protocol aims to bridge these gaps and provide a well-

coordinated, technology-enhanced approach to disaster response. By creating a practical, real-world solution, this thesis contributes to the field of emergency management and provide a model for more effective, life-saving response operations.

It is important to note that despite the promising prospects of IoT in enhancing disaster management, the integration of such technologies is not devoid of challenges. Issues related to interoperability, data privacy and security, as well as the need for reliable infrastructure, present significant hurdles. On the top of that, the reliance on technology underscores the importance of establishing reliable IoT systems that can withstand the very disasters they seek to mitigate [29]. Addressing these challenges requires a multidisciplinary approach, combining technological innovation with policy development, stakeholder engagement and community participation.

The Federal Emergency Management Agency (FEMA), part of the U.S. Department of Homeland Security, is one of the top organizations that have played a significant role in defining and promoting the use of National Response Framework (NRF) framework [30] in the United States and influencing its adoption globally. Its structure is designed to provide a comprehensive approach to managing disasters and emergencies, encompassing activities that should be undertaken before, during and after a disaster occurs. It is distinguished into four phases: *preparedness, response, recovery and mitigation* [31]. Adopting this framework, the IoT-based system under study in this thesis named *AllERT* (an abbreviation of All Emergency in Real Time), primarily focuses on the **response phase**. All the steps outlined below are designed to facilitate rapid and coordinated response efforts. In the aftermath of a flood disaster, the need for provably better coordination of relief efforts and management of aid distribution is monumental. In Figure 2: FEMA's four-phase Emergency Management Cycle or “the Emergency Cycle” is shown, with the focus on the *AllERT* protocol.

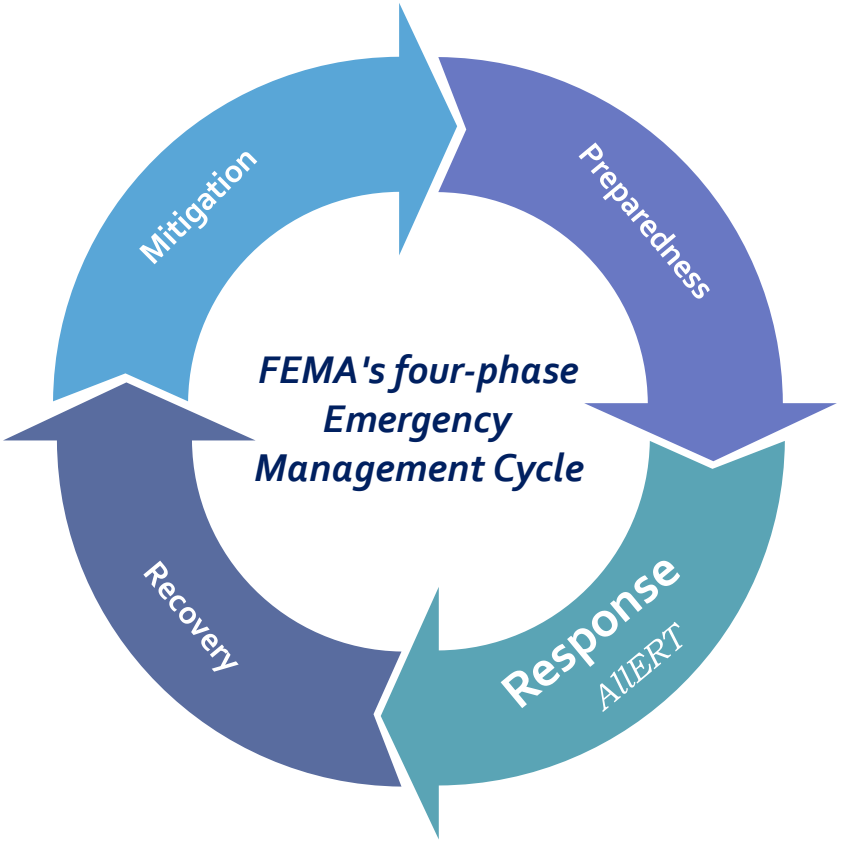


Figure 2: FEMA's four-phase Emergency Management Cycle [31] – AllERT's focus

At the heart of the *AllERT* protocol lies the integration of UAVs, or drones, equipped with advanced imaging capabilities, suitable to operate in challenging environmental conditions, to capture real-time data regarding the effect on disaster hit areas. By obtaining sufficient visual information, responders can quickly assess the extent of the damage, prioritize areas for immediate assistance and strategize the allocation of resources [32].

The core tool for exploiting to the utmost of these visual information is the use of sophisticated image processing algorithms. They are employed to analyze UAV-captured data swiftly and accurately, providing actionable intelligence to decision-makers. Geographic Information Systems (GIS) technology is then incorporated to optimize response efforts, by determining the route of best obtaining access to areas of immediate attention. By combining data from IoT sensors, UAVs and infrastructure analysis, the system provides real-time information for SAR operations.

A significant challenge faced during emergencies is the failure of conventional communication systems such as Global System for Mobile Communications (GSM),

often due to infrastructure damage or network overload. To address this issue, all communication channels between authorities and response teams are rerouted through satellite phones, thus ensuring uninterrupted communication and coordination even in the most adverse conditions and challenging environments. By circumventing GSM issues, satellite communication not only guarantees the timely dissemination of critical information but also enhances the overall effectiveness of emergency response efforts. However, the high cost of satellite communication and the need for specialized training for responders are hurdles that need to be addressed prior in order for satellite phones to be used effectively [33].

Satellite internet serves as an alternative communication link to the cloud, activated during emergencies if the primary connection suffers damage. This setup ensures that critical data is transmitted to the Control Tower without delay, maintaining a seamless operational workflow throughout the response phase. This kind of network is already in place and can be quickly deployed to provide connectivity in disaster-affected areas.

Once the protocol is activated, a powerful platform for gathering real-time data and running algorithms is indispensable. This platform is cloud-based, allowing for scalable storage and processing capabilities. It then incorporates Machine Learning (ML) algorithms to continuously analyze data and identify patterns that can aid in decision-making. In addition to real-time data collection and analysis, this flood management system also emphasizes the importance of collaboration and coordination among different stakeholders [34]. This includes integrating the data from various sources, such as government agencies, Non-Governmental Organizations (NGOs) and the public, to create a comprehensive situational awareness dashboard. Such dashboard provides a unified view of the flood situation, allowing responders to make informed and valuable decisions.

Ultimately, the success of any disaster response effort depends on making swift and well-informed decisions that maximize contribution and minimize the impact on affected communities. Implementing a comprehensive DSS will enhance the above statement.

1.4 Research Questions and Objectives

The primary objective of this research is to develop and evaluate a real-time IoT-based DSS, designed to enhance the speed and efficiency of response during flood disasters. This system will act as a procedural protocol to provide emergency services with

immediate, actionable data. The ultimate goal is to streamline response efforts, improve resource allocation and reduce the time taken to reach those in need.

To address this objective, the research is guided by the following research question:

- ***How can a real-time IoT-based DSS, significantly reduce response times and enhance the optimal resource allocation during flood disaster management?***

This primary research question encapsulates the core focus of the study, emphasizing the importance of real-time data and its impact on emergency response operations. By answering this question, this thesis attempts to demonstrate the practical benefits of integrating ***Modern Communication Systems and IoT*** into disaster management systems.

In addition to the primary research question, some secondary questions have been formulated to facilitate a full examination of the suggested approach:

- What are the main constituents and the architecture of an effective IoT-based DSS for flood disaster management?
- How does the proposed system compare to existing flood disaster management solutions in terms of response time, resource allocation and overall effectiveness?

The objectives of this research are multi-faceted. Firstly, it intends to design and develop a comprehensive system architecture that integrates IoT technology, UAVs and geospatial tools in the form of a procedural protocol. Secondly, it seeks to implement real-time data processing algorithms that can enhance decision-making processes during flood events. And finally, the research intends to test the system through simulated flood scenarios based on historical data, to evaluate its performance and effectiveness. By achieving these objectives, the research will contribute to the development of more responsive and immediate disaster management systems, ultimately leading to better preparedness and quicker response times in flood-prone areas. And why not? It intends to be broadly established for all countries and all types of disasters.

1.5 Structure of the Thesis

A roadmap for the entire thesis is presented below:

Chapter 1, provides an overview of the research, outlining the background, the problem statement and the study's significance. It introduces the research questions and objectives and the importance of the potential impact on emergency management practices.

Chapter 2, surveys the existing literature relevant to IoT technologies, satellite communications, UAVs, GIS, image processing, decision support systems and procedural protocols in emergency management. The review highlights existing solutions and identifies gaps which it aims to address. The final sub-chapter introduces and describes the *AllERT* protocol for flood management.

Chapter 3, explains the research design and the system architecture of the *AllERT* protocol. It describes the hardware and software components, the selected communication protocols and the role of Node-RED in the system's dashboard functionality. Lastly, it examines the limitations of the study as well as some ethical considerations related to human welfare and data security.

Chapter 4, features an Implementation Use Case applied in Thessaly's flood disaster caused by storm Daniel for a practical application of the proposed system. It consists of the ten defined steps of the *AllERT* protocol, a description of the Dashboard and the analysis of the DSS functionality. Finally, the results are interpreted, addressing how the system performed in the context of the research questions and objectives. System performance and contribution of the study are also presented.

Chapter 5, offers recommendations for future work or improvements to the *AllERT* protocol and its constituent factors. The contribution of volunteers is particularly acknowledged.

Chapter 6, outlines the achievements of the research, with an emphasis on the system's ability to make reliable real-time flooding decisions. It also suggests improving interdisciplinary collaboration and integrating local population in the development of the system. Finally, emphasis is given to data management and privacy.

The final parts of the structure are: a list of all references cited throughout the thesis, indexes of all figures and tables included and a number of appendices with additional material, such as codes or supplementary information, that support the research findings.

2. Literature Review

2.1 IoT in Emergency Management: Review of how IoT technologies, including drones and sensors, are used in disaster response.

Through a comprehensive examination of the literature, this part of the study aims to explore the state-of-the-art IoT-based (based on the Internet of Things) systems and the utilization of UAVs for area scanning in flood management contexts. Several studies have highlighted the potential of improving different aspects of emergency interventions and reviewing these findings reveals that almost all stages are eligible to be handled by IoT. By synthesizing existing research findings and identifying gaps in knowledge, this review will assist in the development of a holistic approach to flood disaster preparedness and response, ultimately contributing to more effective management practices. Due to the rapid evolution of IoT technologies, only the most current research on their application in handling emergencies and crises will be explored.

Over the years, the integration of IoT technologies has emerged as a promising innovative approach for early warning systems in flood-prone regions. By leveraging a network of interconnected sensors deployed in key locations, IoT-based systems can continuously monitor various environmental parameters such as water levels, rainfall intensity and river flow rates in real-time. Numerous studies have explored the effectiveness of such technological interventions, highlighting their ability to enhance preparedness, reduce response times and mitigate the impact of flooding on vulnerable populations by helping in managing evacuation processes. Powered by data regarding the levels of flooding, safe routes and the status of evacuation shelters can be easily deduced [15–17, 29].

The advancement in solutions that are built on IoT for flood detection and water monitoring signifies a leap towards real-time, accurate and efficient management strategies, meeting the urgent need for enhanced flood crisis preparedness and prediction efforts globally.

2.1.1 The Use of Internet of Things (IoT) – Sensors and related devices

Although IoT technologies cannot prevent flood disasters, they can be used effectively to distribute information on corresponding preparedness. In the content of a research that provides a comprehensive review of the technologies and methodologies involved, the importance of utilizing technological assets to gather and process data close to the source

is highlighted. This approach ensures timely decision-making by providing swifter access to valuable information. It demonstrates sensing, networking and processing capabilities with low response times, to communicate and share data for advanced services in near real-time. The research also underscores the evolution from traditional flood monitoring methods to more advanced solutions, emphasizing the potential for improved flood detection, impact assessment and faster response mechanisms [35].

Other systems use sensors and networked devices to enhance flood forecasting through advanced data analysis capabilities supported by community engagement, also known as crowdsourcing, to enhance the functionality and efficiency of the methodology [36]. Needless to say, the collaborative implementation of intelligent IoT-based flood monitoring systems in flood-prone regions worldwide, holds the potential to significantly reduce the harmful impacts of floods. Continuous research, development and teamwork between all those involved in this important area is obviously necessary.

A very common structure met while reviewing academic material, is Wireless Sensor Networks (WSNs) that consist of spatially distributed autonomous sensors. These are used to monitor physical or environmental conditions and send their data through the network to a main location [37–39]. A nice example of it is in this paper, where the authors present a conceptual design of a Search and Rescue system using WSNs intended to enhance such operations in disaster-hit areas. The WSNs are used to track and identify survivors, coordinate search efforts and assess the on-ground situation in real time. The system involves equipping individuals in danger zones with personal Victim Sensor Nodes (VSNs) that communicate with Network Aggregation Hubs (NAHs) spread across the hazardous area. NAHs next aggregate the data of sensor nodes in their proximity and relay it back to a central server. The WSNs, in this case, are used to form a flexible network where nodes use different routing algorithms to transmit information to the network aggregation hub. The results can supply rescue teams with invaluable assistance [39].

Further studies suggest that mesh networks can be beneficial in response during floods by providing a reliable and decentralized communication infrastructure. These networks, based on the concept of deployable wireless communication infrastructure, ensure connectivity even when traditional networks are unavailable due to power loss or physical damage [40]. Mesh networks enable critical network capacity through offline first communication, allowing for data transmission without the need for prior existing

infrastructures like base stations [41]. The collected data can be used for flood prediction and alerting the appropriate emergency services and end users. Mesh networks also offer the advantage of multi-hop and self-organized features, making them suitable for the domain under this study [42] and at the same time, responders can establish reliable communication channels and coordinate their actions more effectively in the affected areas.

In disaster management conditions, effective communication is required for prompt reactions and accurate coordination. Thus, communication protocols that prioritize reliability, minimal delay, scalability and resilience are preferable amongst others. This is why the most commonly used protocols in IoT for disaster management include Message Queuing Telemetry Transport (MQTT), CoAP, LoRaWAN, cellular networks 4G/5G and mesh networking protocols like Zigbee [34, 43]. LoRaWAN in particular, is highly valued in disaster management for its long-range capabilities, low power consumption and ability to penetrate through obstacles, making it ideal for transmitting signals in dense urban areas or remote locations. On top of that, its network architecture allows for flexible deployment models, which can be rapidly deployed in cases of emergency [44]. Apparently, building redundant and resilient communication technology capacities to provide uninterrupted connectivity during catastrophes, is something that definitely requires attention [45].

One size does not fit all, which is why the selection of the most suitable protocol depends on heterogeneous factors such as the nature of the disaster, the available infrastructure, requirements in range and coverage, possible power constraints and the device types involved. A combination of multiple protocols utilized to ensure dependable communication under diverse conditions could be a wise decision.

Edge computing, a technology rooted back in the 1990s that has seen lightning-fast development and adaptation over the past five years [46], has the potential to offer great assistance in flood response by bringing services closer to affected areas, reducing latency and giving the opportunity for real-time data processing and analysis. By processing data locally at the borders of the network, faster actions and decisions are provided to emergency managers since they can receive early warnings and real-time updates about flood situations upcoming. This equips them with the tool to give better directions to the field crews [47]. Edge computing also supports the dissemination of flood alerts and warnings to the public, ensuring that accurate information reaches the affected population

on time. It also allows a great portion of the data processing to be carried out by UAVs on airborne surveillance devices and thus rescue operations during floods are enhanced in terms of situational awareness and response capabilities [48].

In similar way, Fog Computing can further enhance the effectiveness of disaster management systems by acting as an intermediary layer between the Edge Computing infrastructure and the Cloud. With Fog Computing, the concept of edge computing is extended by adding a decentralized computing infrastructure that is closer to the end devices but not necessarily on the device itself. It is a hybrid approach that offers several additional benefits that complement and expand upon the advantages provided by only the edge computing. Choosing this solution, allows for efficient data processing, analysis and storage, improving the overall effectiveness of disaster management efforts. The combination of Edge and Fog computing ensures that data is processed and acted upon as close as possible to the spot it was generated, while still providing all the benefits of centralized analysis and when needed the most [49].

However, the vast amount of data generated by IoT devices during disasters can be overwhelming thus effective data management and analysis are essential. The need for efficient and accurate data processing and analysis techniques to support real-time decision-making during and after disasters is of supreme importance and needs further research.

2.1.2 The Use of UAVs

UAVs, or drones as they are more commonly known, now come equipped with high-resolution cameras and a suite of sensors, deployed to rapidly assess flood-affected areas and make a quick and early assessment of the situation. As they soar above areas laid waste by floods, they become our eyes in the sky, capturing intricately detailed aerial imagery and swiftly scanning vast territories. Such an advantageous position provides emergency responders with essential insights into the magnitude of flood damage, the state of vital infrastructure, and the whereabouts of individuals cut off by water. Each flight not only maps the physical landscape but also charts a course for rescue and recovery, becoming a necessary tool in the strategic arsenal of disaster management [50].

To date, UAVs have been instrumental across various facets of SAR operations, proving to be a groundbreaking technological enhancement. Academic research underscores how UAVs, particularly when used in conjunction with advanced technologies like

Convolutional Neural Networks for real-time object detection, have revolutionized the detection of survivors and the evaluation of disaster impacts. Having such potential can boost the efficiency of survey operations. Provided with a good balance between speed and accuracy, they are suitable for resource-constrained environments a topic extensively documented in scholarly research [43].

An important feature that drones possess is their capability to gather and transmit data in real-time [34], via their equipped sensors before, during and after disasters [51]. The collected data can therefore be uploaded to the cloud for further analysis and storage using IP-based communication which permits prompt information access, even in remote and inaccessible regions[52]. They can also be used for tasks such as aerial photography and mapping, providing mobile real-time images of disaster areas [53] thus supporting the decision-making.

With regards to victims' localization, Search-And-Rescue DrOne-based solution (SARDO) is a drone-based solution that uses mobile phones and wearables to locate missing people in disaster scenarios. Its techniques combines novel concepts of pseudo-trilateration[1] and ML to efficiently locate smart devices in a given area that can lead to a person in peril [55].

Additional study seeks to enhance the capabilities of a UAV by integrating an onboard microcomputer, image processing for human detection, Global Positioning System (GPS) technology for location tagging and LiDAR sensors for obstacle detection [56]. They are applicable for various tasks in agriculture, disaster surveillance, 3D mapping, military operations, healthcare, transportation and more. However, the functionality of a single UAV is restricted due to high energy consumption and coverage limitations [57]. Multi-UAV systems were developed to offer cooperative and layered capabilities to overcome these limitations and provide more benefits [58, 59].

Among other studies, some propose equipping UAVs with communication payloads (wireless transceivers and transmission modules) that can serve as temporary mobile networks to facilitate uninterrupted communication between responders and communities. This is based on their serving as aerial base stations (ABSs), which can be deployed quickly to temporarily rebuild any damaged communication network and

[1] **Pseudo-trilateration** is a technique used in location-based systems, such as wireless networks or GPS, to estimate the position of a target object or device. It is called "pseudo" because it relies on approximations or incomplete data, rather than the exact distances typically required in traditional trilateration [54].

restore connectivity for users in disaster-hit areas [60]. The same studies have explored the feasibility and effectiveness of using drones as aerial communication transmitters, assigning them a role in ensuring the continuity of communication [61].

Then there are the autonomous UAVs that represent an additional advancement in the field, due to their ability to perform tasks without direct human intervention, resulting in a high degree of independence. Unlike regular ones, they don't require a human pilot to control their flight path and actions. Instead, they are equipped with software and sensors that enable them to make decisions based on pre-programmed instructions or real-time data [62]. This means they can perform with greater precision, from mapping and surveillance to delivery in complex and dynamic environments like post-flood response scenarios. Quickly evolving into a very useful tool for a wide range of industries and applications, they seem to provide flexibility, cost savings and less risk in circumstances when human-controlled UAVs might not be feasible or reliable, such as in unknown or hazardous areas [63]. The findings of the literature review indicate that autonomous UAVs offer considerable potential in a range of applications, with particular promise in post-flood crisis scenarios. For example, prementioned research suggests that system which was tested for its ability to autonomously survey areas and detect humans with high accuracy [56].

Finally, a comprehensive review was conducted on the utilization of drones during disaster management in a research study that was quite intriguing [64]. According to it, the use of UAVs for delivering supplies in disaster response depicts a game-changing shift in how aid is provided to affected areas. Their agility and ability to navigate through challenging terrains, have been recognized for their success in transporting medical supplies, food and other essentials to areas that are otherwise inaccessible because of natural disasters. The incorporation of UAVs into disaster management logistics has opened new avenues, ensuring that help reaches those in need promptly. As this technology continues to evolve, their operational capabilities are expected to expand further.

Below a table of the common classifications of UAVs:

Classification	***Types of UAVs***
Flying Principle	*Fixed-wing, Rotary-wing, Hybrid, Flapping-wing*

Classification	***Types of UAVs***
Mission	*Reconnaissance, Surveillance, Attack, Transport, Search and Rescue*
Weight	*Micro-UAVs, Small UAVs, Tactical UAVs, Medium-Altitude Long-Endurance (MALE) UAVs, High-Altitude Long-Endurance (HALE) UAVs*
Propulsion	*Electric, Fuel, Solar*
Control	*Remotely Piloted, Autonomous, Semi-autonomous*
Altitude Range	*Low-altitude UAVs, High-altitude UAVs, Stratospheric UAVs*
Configuration	*Mono-rotor, Multi-rotor, Tilt-rotor, Tilt-wing*
Purpose	*Military, Civilian, Commercial, Industrial, Scientific*
Launch Method	*Ground-launched, Air-launched, Sea-launched*
Payload	*Sensors, Cameras, Communication Systems, Weapons, Cargo*
Autonomy Level	*Fully Autonomous, Semi-autonomous, Human-operated*
Size	*Mini-UAVs, Handheld UAVs, Man-portable UAVs, Vehicle-mounted UAVs*
Endurance	*Short-endurance UAVs, Long-endurance UAVs, Ultra-long-endurance UAVs*
Range	*Short-range UAVs, Intermediate-range UAVs, Long-range UAVs*

Table 1: Common classifications of UAVs [65]

2.2 Satellite Communication: Exploration of satellite phone and internet technologies in emergency management scenarios.

Satellite communication involves the use of satellites orbiting the Earth to enable communication between devices that are too far apart to be connected by conventional means, such as cables or terrestrial wireless systems. It can support a variety of services, including voice communication through satellite phones and data services through satellite internet [66]. By providing coverage in remote, rural and maritime areas where

other forms of communication infrastructure may be limited or nonexistent, it implies a lot in global telecommunications [33].

In today's interconnected world, telecommunications companies prioritize uninterrupted connectivity and contingency solutions tailored to specific sectors. Satellite phones have emerged as vital tools across various industries, including government, where they ensure communication continuity during high-impact operations, such as emergency situations during military activities or in case of natural disasters [67]. In mining, satellite connectivity aids in remote camp locations and worker monitoring while also helping to avoid regulatory fines through timely accident reporting. Similarly, the maritime sector benefits from satellite phones by enhancing communication between port areas and vessels, as demonstrated by initiatives in the fishing industry. In the oil and gas sector, satellite phones and their global connectivity provision, optimize processes like drilling in remote areas. Infrastructure projects in regions with challenging terrestrial infrastructure can’t be overlooked. For instance, in Latin America where terrestrial infrastructure showcases geographical challenges and dispersed populations, engineering industries rely on satellite voice connectivity for real-time information exchange, regulatory compliance and efficient project management. All these sectors demonstrate the diverse applications and indispensability of satellite phones in overcoming communication barriers worldwide [68].

Satellite phones have been employed in cases of disaster response, as they have been considered to be of great significance in terms of reinstating communication channels in the immediate aftermath of a calamity, particularly when the terrestrial infrastructure has suffered damage. Satellite technology in general is presented as a solution to ensure reliable communication in adverse environments, so as to support emergency managers in maintaining a common operational picture, saving lives and protecting emergency responders during mass rescue operations [69]. Nevertheless, the above statement presents certain challenges. The affordability factor relating to satellite calls under normal circumstances has a negative impact on their applicability in all cases. While the cost of satellite calls can be mitigated during emergencies, the rarity of such devices and the maintenance of up-to-date contact lists are major issues [33, 70]. The limited availability of satellite phones in the impacted regions may render them insufficient to effectively address the requirements at hand [33]. Despite these aforementioned hurdles, the field of satellite technology, including satellite telephony, still remains as a valuable instrument

in disaster response [33, 69–74]. Aside from having the ability to bridge communication connectivity gaps that arise during disasters, it operates independently of terrestrial infrastructure, regardless of the distance between the parties involved.

New developments involve the use of High-Throughput Satellites (HTS), whereby more recent satellites provide considerably enhanced data rates, increasing the applicability of satellite internet in the domain of emergency management [73]. Low Earth Orbit (LEO) Satellite Constellations, such as SpaceX's Starlink initiative, take steps to reduce latency and increase bandwidth by deploying several small-sized satellites in a low earth orbit [75]. Therefore, in the years to come, all of the previously listed factors could increase accessibility and efficacy.

Satellite communication however, may experience challenges with limited coverage and reception due to factors like terrain, weather conditions and obstructions that affect signal quality [76]. This can result in unstable connectivity and poor reception. Combining satellite communication technologies with traditional terrestrial communication networks can help bypass the gaps and limitations created by such environmental conditions, expanding coverage and dependability overall. From a hybrid point of view, the combined strengths of each system provide a more resilient communication infrastructure [77].

Terrestrial Networks, such as cellular and fiber-optic systems, offer high-speed, low-latency communication in areas with existing infrastructure [78]. Hybrid Systems automatically switch between satellite and terrestrial networks based on availability and Quality of Service (QoS) factors, ensuring the best possible connection at any given time. For example, when satellite signals are weak or obstructed, the system can switch to terrestrial networks if available. Conversely, in situations where terrestrial networks are compromised due to disasters, satellite communication can provide an alternative path, serving as a backup, ensuring continuous communication for emergency services and affected populations [79].

The integration of satellite and traditional terrestrial communication networks offers a comprehensive solution that moderates the limitations posed by environmental conditions, infrastructure damage and network congestion [80]. Thus resilience and flexibility in the coverage of communication systems, enhance robustness, which is particularly critical in emergency management and disaster response scenarios. Taking advantage of the complementary strengths of satellite and terrestrial technologies,

emergency responders, governments and communities can maintain communication in the most challenging conditions.

For the successful management of all possible consequences during a flood in real time, it is imperative to strengthen cooperation between experts, which should be achieved in a very short timescale. The proposed system aims to achieve the shortest response times by effectively utilizing a combination of various existing communication methods and technologies as already featured in the academic review.

2.3 FINDER MK4: Life Sign Detection System

The Finding Individuals for Disaster Emergency Response (FINDER) technology, developed by National Aeronautics and Space Administration (NASA) (Jet Propulsion Laboratory) in collaboration with the Department of Homeland Security (DHS), uses microwave radar to detect heartbeats and respiration through debris. The device was initially created in response to the 2010 Haiti earthquake, where the need for an efficient method to locate survivors trapped under rubble was imperative. FINDER can distinguish between human movements and other motions, such as those made by machinery or animals, making it a valuable tool in all disaster scenarios.

While it is primarily designed to detect heartbeats and breathing through rubble, its technology can also be adapted for other disaster scenarios, including floods. The microwave radar it uses can penetrate through various materials, which makes it precious for locating individuals trapped in different environments, such as those submerged or hidden by flood debris. Without a doubt, it is a valuable tool for rescuers in tracking survivors who are otherwise hidden to the naked eye.

The device's ability to penetrate deep through rubble and its incorporation into a familiar ruggedized container (Pelican case) ensures that first responders can quickly deploy it and receive exploitable results. Although the initial prototype had limitations, such as difficulty in determining the number of individuals and challenges with certain materials, it proved holding great promise. SpecOps Group Inc. was one of the companies that received a license to build commercial versions of FINDER. They then improved the device, as prementioned, and it can now perform a full scan in under 30 seconds weighing only around 3.6 kg. The latest MK4 version of X3 FINDER, interfaces with iOS and Android devices [81–83].

FINDER has been deployed in various international disasters, including Hurricane Dorian in the Bahamas, and is used by public safety departments globally. SpecOps' sales to local and national government authorities have funded further development of detection devices for police use, such as Cyclops ID [84], which detects movement through walls. The collaboration between NASA and private industry, exemplified by FINDER, continues to evolve.

2.4 Image Processing for Damage Assessment: Review of existing methods for analyzing images to assess infrastructure damage by floods.

2.4.1 Image-based techniques for the purpose of damage detection and analysis

In the expansive domain of remote sensing in respect of disaster management, a wide array of techniques is utilized for the classification and analysis of images, highly assisting disaster recovery initiatives. These techniques often encompass both supervised and unsupervised classification algorithms, extending to more complex methods such as deep learning and Object-Based Image Analysis (OBIA) [85–87]. To identify and evaluate flood scenarios, a comprehensive range of algorithms and methodologies is put to use, analyzing and classifying images. This analysis capitalizes on data sourced from satellite imagery, aerial photos and UAVs captures, occasionally incorporating carefully chosen video footage datasets [88].

Conducted research explores the utilization of image processing, computer vision and machine learning methodologies in the realm of structural health monitoring and disaster management. Such studies emphasize the shift from conventional inspection techniques to cutting-edge automated systems that deploy high-resolution imagery to achieve efficient and precise assessment of damages in civil infrastructures and post-disaster situations [89].

Supervised classification algorithms, such as Maximum Likelihood Classification (MLC), Support Vector Machines (SVMs) and Random Forests (RF), rely on pre-labeled training data to identify and categorize different land cover types or damage levels in satellite imagery [85, 90, 91]. Unsupervised classification, including K-means and

ISODATA[2] clustering, discovers natural groupings in data without prior labels, useful for identifying unforeseen patterns in disaster-affected areas [90, 93].

Deep learning, particularly through the use of Convolutional Neural Networks (CNNs), represents a significant advancement in the field, enabling the automatic extraction of complex features from high-resolution images. This method has shown exceptional performance in detecting intricate patterns and changes in disaster-stricken regions [94, 95]. Similarly, OBIA groups pixels into meaningful objects rather than analyzing them individually, offering a more detailed perspective on the ground situation [85].

Change detection algorithms offer powerful tools for monitoring temporal changes, which are crucial for assessing the extent of damage and the progress of recovery efforts. Techniques such as Post-Classification Comparison play a vital role in disaster management by comparing satellite images taken before and after an event. This method allows for the identification of changes in land cover and land use, enabling a quick and accurate evaluation of the impact of natural disasters [96]. These strategies are instrumental in guiding emergency response, recovery operations and in planning for future disaster resilience.

The ceaseless march of technology, where once-separate innovations now intertwine, is shaping a future where disaster management might evolve in both surprising and profound ways. Researchers armed with the sharpest of images, the keen eyes of UAVs and the boundless potential of deep learning, sketch out strategies that are not only tougher and smarter but also ready to perform to the unpredictable rhythm of the world. They are not just stepping ahead; they're building a resilient framework that can adapt on the fly. The focus on datasets like FloodNet along with the investigation into diverse computer vision methods for identifying damage and segmenting areas, illustrates a wider trend towards incorporating technology. It's a movement that seeks to weave technology into the very fabric of response to natural disasters, striving to stay one step ahead of the ever-more-complex challenges that nature and time throws [87, 97, 98].

The following step of this review looks at the FloodNet dataset which represents an important step towards the development of advanced technologies for enhancing flood response strategies. It is unique for its high-resolution UAV imagery captured after

[2] "The ISODATA Classification method is similar to the K Means method, but incorporates procedures for splitting, combining, and discarding trial classes to obtain an optimal set of output classes. The ISODATA algorithm analyzes a sample of the input to determine a specified number of initial class centers" [92].

Hurricane Harvey, tailored for the evaluation of image classification, semantic segmentation and Visual Question Answering (VQA) tasks in post-flood scenarios. It provides a valuable resource for the development and benchmarking of algorithms aimed at interpreting post-disaster scenes, eventually contributing to more efficient and effective flood management and recovery efforts [98]. Semantic segmentation is a valuable method in computer vision, particularly significant for remote sensing. It includes dividing an image into segments, assigning each pixel to a specific category. This technique allows for a detailed analysis of images by categorizing all pixels into classes like roads, buildings, water, vegetation and affected areas during disasters [99].

Concurrently, substantial research is dedicated to comparing four computer vision methods for identifying damage. The study delves into each technique's specific approach to segmenting and classifying images to detect damage. The Otsu method is utilized to transform grayscale images into binary images for further analysis. Markov random fields are applied to label image pixels, taking into account the spatial relationships among them. Red, Green and Blue (RGB) color detection is used to assess variations in color, while K-means clustering groups images together based on similarities in color [87]. Within the same study, the evaluation of computer vision techniques also included the use of an evaluation metric that is widely accepted in object detection benchmarks: the Intersection over Union (IoU) index. The formula for IoU index is the ratio of true positives to the union of true positives, false positives and false negatives [100, 101]. This metric quantifies the overall similarity between the predicted and target masks by measuring the number of pixels common between the two masks divided by the total number of pixels present across both masks.

Fusion of aerial images with other data sources like LiDAR (Light Detection and Ranging) or radar data enhances the accuracy of flood mapping to a significant degree and improves the identification of impacted areas [102]. This process, known as data fusion or sensor fusion, combines information from multiple sources to provide a more comprehensive and detailed understanding of the environment [103].

All of the aforementioned techniques aid in various aspects of disaster management, including preparation, response, recovery and mitigation. Together, they provide a holistic understanding of the impact of disasters and they enable quick and accurate assessments, which inform effective response mechanisms. In addition, they have significantly improved the ability to monitor environmental changes, assess vulnerability

and risk, and plan for disaster resilience. With continuous advancements in these methodologies and tools, even more progress in efficiently managing the effects of natural disasters can be expected.

Challenges persist in flood damage detection, as while image processing techniques have shown success in identifying damage in structures such as bridges and roads, the specific hurdles posed by flood damage remain largely unaddressed. Flood-induced damage encompasses a broad spectrum of issues, including water-induced structural weakening, debris accumulation and erosion, necessitating specialized detection methods beyond current image processing algorithms [104]. The complexity of post-flood environments, characterized by dynamic landscapes altered by debris and sediment deposits, presents challenges for traditional image processing algorithms to distinguish between pre-existing damage and flood-induced damage. Enhancements in image analysis techniques are imperative to explore these complexities and accurately assess flood-induced damage [105].

Capturing flood events is obscured and limited by the fact that image-based methods cannot detect flood damage in areas obscured by cloud cover in post-event imagery. As noted, "if an area is not visible in the post-event image due to cloud cover, the flood attribute is omitted" [106]. Apart from that, repercussion of floods estimates is subject to significant uncertainty due to variability in depth-damage curves and land use data. As described, "the variation between the damage models is regarded as an indication of the uncertainty inherently present in depth-damage functions and values of elements at risk" [107]. This indicates that image-based techniques may miss critical flood-affected areas, leading to inaccurate flood damage assessments and hindering the effectiveness of any relief operation based on this methodology. To address these gaps, future research should explore methods to incorporate alternative data sources, such as radar imagery, to overcome cloud cover limitations [108]. Efforts to improve depth-damage models and land use data quality could help reduce the uncertainty of the outcome.

2.4.2 MATLAB and Similar Tools

The availability of sophisticated computational tools such as MATLAB, enables researchers and practitioners to implement and experiment with complex algorithms efficiently. Given this type of tools, a powerful framework for data analysis, model building and simulation is built, making advanced image analysis much more feasible.

Particularly MATLAB, offers a wide array of functionalities tailored for processing and analyzing remote sensing data, suitable for applications like flood mapping and the identification of affected areas. With its extensive toolboxes for image processing, computer vision and deep learning [109–113], it is highly capable of implementing all the methodologies of the previous subchapter. It supports a comprehensive workflow from image preprocessing to model training and evaluation, exploiting specialized functions for handling remote sensing data, machine learning algorithms and parallel computing capabilities for large datasets. Its integration with other programming environments and support for Graphics Processing Unit (GPU) acceleration make it a powerful and scalable option for highly sophisticated image analysis.

While MATLAB is a preferred choice for many, alternatives like Python with TensorFlow, PyTorch and OpenCV also offer powerful capabilities, with the choice largely depending on specific project needs and user preferences.

A comparative overview of MATLAB, TensorFlow, PyTorch and OpenCV in various dimensions such as usability, primary focus, language support and typical applications follows as an example:

Feature	***MATLAB***	***TensorFlow***	***PyTorch***	***OpenCV***
Primary Focus	*Broad (image processing, numerical analysis, engineering)*	*Deep learning, machine learning*	*Deep learning, machine learning*	*Computer vision and real-time image processing*
Usability	*High, with an Integrated Development Environment (IDE)*	*Moderate, API-focused*	*High, Pythonic API, dynamic computation graphs*	*High, especially for traditional computer vision tasks*
Language Support	*Proprietary MATLAB language*	*Primarily Python, with C++ support*	*Python, with C++ interfaces*	*C++, Python, Java and others*
Performance	*High in numerical computations and simulations*	*High, with GPU acceleration and TPU[3] support*	*High, with GPU acceleration*	*High, optimized for real-time applications*

[3] Tensor Processing Units (TPUs) are Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads [114].

Feature	***MATLAB***	***TensorFlow***	***PyTorch***	***OpenCV***
Community & Support	*Large, with extensive documentation and user forums*	*Very large, open-source community and support*	*Large, growing rapidly, open-source community*	*Very large, extensive community and library support*
Typical Applications	*Engineering simulations, data analysis, algorithm development*	*Machine learning models, deep learning applications*	*Machine learning research, deep learning applications*	*Image processing, video capture and analysis*
Learning Curve	*Moderate to high, depending on background*	*High, due to complex ML concepts*	*Moderate, Python-friendly and intuitive for researchers*	*Moderate, requires knowledge of computer vision concepts*
License	*Proprietary, commercial*	*Open Source (Apache 2.0)*	*Open Source (BSD[4])*	*Open Source (BSD)*

Table 2: Tools for Damage Detection and Analysis Comparison [109–113, 116–119]

Each of these tools/platforms has its strengths and is suited for different types of projects and user profiles, ranging from academic research and prototyping to industrial applications in computer vision and machine learning.

2.5 GIS and Resource Allocation in Emergency Management: Benefits from GIS for Route Optimization and Integrating Broader Resource Allocation Strategies During Floods.

A large number of existing studies cover a wide range of topics related to disaster management, with a focus on flood events and their impact on transportation infrastructure. Their main objective is to address the development and implementation of technological tools and frameworks to improve situational awareness, identify safe routes for evacuation and assess road network accessibility. A number of case studies conducted, draw attention to the increasing exposure of road transportation systems to flood events due to climate change and the need for integrated and adaptable solutions that can support decision-making and emergency response efforts in flood-prone environments [120–123]. GIS applications in emergency management have revolutionized the way authorities respond to crises. However, this field requires further refinement and

[4] "BSD licenses are a low restriction type of license for open source software that does not put requirements on redistribution" [115].

improvement to enhance the overall effectiveness of these applications. This section aims to provide an overview of GIS technologies specifically tailored for route optimization and resource allocation in emergency management scenarios.

Until recently, resource allocation during emergencies has often been marred by inefficiencies and disorganization. While routes for emergency response were typically planned, the lack of synchronized coordination with available resources, often led to significant delays and the wastage of precious time. Emergency responders would be dispatched to affected areas without a clear understanding of where resources were most urgently needed, resulting in inefficient exploitation of critical materials and personnel [124, 125]. Consequently, valuable time that could have been used to save lives and detect influence was squandered due to the absence of integrated systems for aligning route optimization with resource availability. This unlinked tactic underscored the lack of cohesion and coordination and the pressing need for more organized and efficient resource allocation strategies supported by advanced GIS technologies, in order to synchronize routes with the timely arrangement of essential materials and personnel required during emergency situations. By integrating spatial data with real-time information, GIS will enable emergency responders to analyze, visualize and interpret critical information swiftly. A final condition that should be considered in terms of resource allocation is information on population distribution, which would greatly enhance intervention decisions. This will be explored in this paper in an approximate manner, leaving the refinement for future exploration.

2.5.1 Route Optimization

One of the key functionalities of GIS in emergency management is route optimization. During emergencies, such as natural disasters or public health crises, time is of the essence. GIS-based route optimization algorithms analyze various factors including road conditions, traffic congestion and the location of critical facilities to determine the most efficient routes for emergency vehicles. If travel time is minimized and resource utilization maximized at the same time, GIS-enabled route optimization will result in more efficient responses to emergencies across all areas.

The following system, already proposed, has shown that GIS software can retrieve vital data such as water sources, road networks and elevation, all of which are essential to understand and navigate the terrain during floods. The data is structured in raster layers,

making it computationally efficient for routing algorithms. The A* optimization technique, tailored for GIS, calculates the most feasible paths in flood circumstances by analyzing environmental characteristics like road accessibility and elevation. Additional techniques such as step size crossing and bidirectional search are applied to further enhance the computational efficiency and accuracy of the found routes. The system is dynamic and capable of modifying and redirecting pathways as flood conditions change using real-time data. This is of paramount importance, as it can be the catalyst in sustaining effective evacuation and resource distribution routes as environmental conditions fluctuate [123].

Advanced web-based platforms integrate GIS data with flood modeling tools to provide decision support in real-time. At the same time, they employ algorithms like Dijkstra[5] for pathfinding in normal and flood circumstances, as well as visual tools for planning and real-time adjustment of routes, which is fundamental during ongoing flood events [120].

As with most cutting-edge technologies, while GIS has significantly improved emergency management practices, several challenges like data interoperability issues, limited access to real-time data and the need for continuous training and skill development among emergency responders still persist. However, looking ahead, advancements in remote sensing technologies, Artificial Intelligence and Machine Learning hold promise for enhancing the capabilities of GIS in emergency management.

Even though GIS have been extensively employed in flood preparedness efforts, their utilization in real-time response remains relatively limited. In the field of flood management, GIS technology primarily serves as a valuable tool for preemptive mapping and analysis of vulnerable areas susceptible to inundation [127]. Through the integration of spatial data such as topography, land use, hydrology and infrastructure, it facilitates the identification and visualization of flood-prone zones, helping authorities create well-informed mitigating plans and evacuation routes in advance of an impending flood event [120, 123]. However, despite its efficacy in flood preparedness, the application of GIS for real-time response remains somewhat underexplored in the literature. Few studies have focused on exploiting the capacity of GIS to optimize response times during floods, highlighting a gap in research that warrants further investigation. As such, there is a need for greater attention to this topic.

[5] "An algorithm that examines the connectivity of a network to find the shortest path between two points. Conceived by Dutch computer scientist Edsger Dijkstra (1930–2002)" [126].

A significant body of research has focused on the development of predictive systems designed to identify roads that are at risk of flooding and subsequent closure. They are studies which essentially employ statistical models that incorporate historical flood data, meteorological observations and hydrological parameters to forecast the likelihood of road inundation under varying weather conditions. From the above, the critical role of data-driven knowledge in enhancing the resilience of cities to flood-related disturbances is highlighted [120]. By providing the ability to anticipate potentially affected routes, the authorities' preparedness in terms of transport networks is strongly equipped.

2.5.2 Resource Allocation

In addition to route optimization, GIS also plays an indispensable role in the allocation of resources during an emergency. Spatial analysis and modeling techniques may provide administrators with the ability to identify areas with the highest concentration of affected populations and prioritize resource allocation accordingly. This includes deploying medical supplies, personnel and equipment to areas with the greatest need. Inter-agency collaboration is also made possible through GIS, by providing common platforms for sharing spatial data resulting in coordinate response efforts with regards to resource management in real-time [128].

Different strategies to optimize resource distribution during floods, highlight a combination of hierarchical decision-making frameworks and technological innovations. Hierarchical Resource Allocation is a method that includes a multi-level approach where resources are assigned through a hierarchical structure, ensuring that the distribution is managed systematically from higher to lower administrative levels. Considering the requirements and priorities of these different levels within the administrative structure, DSSs can be employed for resource and task allocation in a structured manner. Managing resources and tasks through a software that allows decision-making based on current demands and available resources, point at the ultimate goal of optimizing the outcome [129].

In instances like floods where logistics and delivery services are vital, systems prioritize routes depending on the criticality of products that must be transported. For example, medical supplies or food delivery to affected areas are given higher priority over less significant products. This ensures that essential supplies reach people in urgent need quickly and efficiently. A game-theoretic model designed for resource allocation during

floods, is the deployment of Navy helicopters for rescue and relief in Kerala, which displays a strategic distribution of resources based on criticality and proximity to resource stations. The proposed methodology entails the construction of a crisis management system that takes into account multiple events [130].

The seamless cooperation among government services is paramount to ensure the efficient distribution of resources during emergencies [131]. Thus, it is highly important for a unified catalog with records of all currently available resources to be constructed. In times of crisis, various government agencies, including emergency management, health, transportation and public safety, must collaborate closely in order to pool their resources effectively. An accurate inventory will serve as a centralized repository where all available resources, such as medical supplies, personnel and equipment, are registered and regularly updated. Such a common database can allow authorities to have a thorough overview of the assets available and thereby make informed decision-making in emergency situations easier to accomplish. Of course it can also foster transparency, eliminate duplication of efforts and enhance inter-agency coordination, thereby maximizing the beneficial impact of operations.

2.6 Surveying DSS Platforms & Dashboards: An examination of existing applications in decision support systems and the use of dashboards in flood incidents

A DSS is an interactive, computerized application that aids an organization in making informed decisions by analyzing large volumes of data and presenting viable options. It supports determinations, judgments and courses of action at various management, operations and planning levels. A DSS is almost imposed in compiling comprehensive information that assists in problem-solving and decision-making processes, particularly in assessing the significance of uncertainties. As a component of business intelligence systems, it often includes a specific industry-related database, enhancing decision-making capabilities across multiple industries [132–138].

Since the late 1950s and early 1960s, Carnegie Institute of Technology theoretical studies on organizational decision-making have inspired Decision Support Systems. They became a distinct field of research in the mid-1970s, gaining a boost in the 1980s with the rise of Executive Information Systems (EIS), Group Decision Support Systems

(GDSS) and Organizational Decision Support Systems (ODSS). In the late 1980s, DSSs evolved into more interactive computer-based solutions to help decision-makers use databases and models to solve complicated problems. DSSs expanded their capabilities in the 1990s with data warehousing and Online Analytical Processing (OLAP), extending their scope and uses and have evolved over the years to fit the needs of organizations and decision-makers [137].

Initially, DSSs evolved from the business data processing tradition, focusing on financial and operational data associated with business use. By the early 1980s, DSS technology started to incorporate relational database technologies, leading to a significant growth in the importance of spatial DSSs. This shift highlighted the need for practical topics like data warehousing issues, data extraction, data quality and effective user interface design, which were not adequately addressed in the early stages of DSS development [136, 139].

Traditional DSSs and Business Intelligence (BI) tools used primarily historical data that was preconfigured and structured for analysis. These systems could not provide real-time data, so decisions were based on past events and trends, making it difficult for users to respond quickly to dynamic situations. The focus was on analyzing what had already happened rather than enabling immediate actions based on current information, which limited the agility and responsiveness of organizations [140, 141].

In contrast, modern DSSs make use of advanced tools that enable "active intelligence", allowing continuous gathering, analysis and action upon real-time data. These systems ensure that decision-making is based on up-to-date information through an end-to-end analytics data pipeline. This shift empowers users with immediate insights and the ability to act promptly, moving organizations from reactive decision-making to a proactive approach. Consequently, modern DSSs make organizations more agile and responsive, enhancing their ability to make quick decisions in a fast-paced business environment [140, 141].

2.6.1 Components of DSSs

A Decision Support System varies across research sources but typically consists of *four components* that work together to provide a comprehensive framework for decision-making, enabling users to analyze information, create models and make informed decisions based on the insights generated by the system [138, 141–145]:

1. *Data Management Component*

   This is the core component of the DSS, responsible for collecting, storing and managing data. It usually includes a database that contains the information that is relevant to the decision-making process. The data management component can access data from internal (organizational), external and local (personal) information sources and it ensures that the data is accurate, up-to-date and easily accessible by the other components of the DSS [142].

2. *Model Management Component*

   As its name indicates, this component manages the models used by the DSS. Models are representations of events, facts, or situations. They can take various forms and are used for experimentation and analysis. The model management system stores, maintains and allows for quick and easy creation and manipulation of the DSS models [142, 144].

3. *Knowledge Management Component*

   This component is not explicitly mentioned in the academic material, but it is implied as a part of the overall DSS framework. Knowledge management involves the organization, maintenance and dissemination of intelligence within an organization. It is very important for effective decision-making and is often integrated with other components of the DSS [141].

4. *User Interface Management Component*

   Finally, the User Interface Management Component, is responsible for allowing users to interact with the DSS. It includes the user interface management system, which is the part of the system that users directly interact with. For that purpose it should be designed to be flexible, consistent, simple and adaptable to facilitate effective decision-making [141, 142].

2.6.2 Types of DSS Platforms

Decision Support Systems can be categorized into several types, each serving specific decision-making needs. They can vary in complexity, functionality and purpose, depending on the type of problem, data and user involved:

1. *Data-Driven* DSSs

   Data-driven Decision Support Systems organize, retrieve and analyze large volumes of data using database queries and OLAP technologies. They process data from various sources and present it meaningfully, using data warehouses, data mining, OLAP and business intelligence (BI) to generate reports or dashboards. These systems help discover patterns, trends and insights from historical or current data, thus supporting descriptive or diagnostic analysis. Used by managers, staff and suppliers, data-driven DSSs are deployed via mainframe systems, client/server links, or the web. They provide access to extensive internal and external data, helping users make informed decisions about businesses, inventories and products [138, 145–148].

2. *Model-Driven DSSs*

   Model-driven Decision Support Systems use formal representations of problems through models and methods like decision analysis, optimization, statistics and simulation. These systems rely on mathematical models or algorithms to simulate, optimize, or evaluate different scenarios and alternatives. They utilize tools such as spreadsheets, simulation software, optimization software, or Artificial Intelligence (AI) for calculations, projections, or predictions based on input data and parameters. Model-driven DSSs thus help explore the consequences of actions, compare trade-offs, or find optimal solutions for specific objectives or constraints. Used within an organization, they can be deployed on stand-alone PCs, client/server systems, or the web and assist with tasks such as scheduling and decision analyses [138, 145, 146, 149, 150].

3. *Knowledge-Driven DSSs*

   Knowledge-based Decision Support Systems provide expertise for solving problems in specific domains, like healthcare or finance. They are also known as knowledge-driven DSSs or knowledgebase systems because they use expert knowledge (often by the use of data mining) to align with a company's processes and can assist both internal users and external customers. They are deployed through client/server setups, web platforms, or standalone software and continuously monitor updated organizational data to support decision-making by diagnosing, predicting, interpreting and classifying information. They can be extremely helpful for managers as they perform tasks faster than humans and guide consumers in selecting products and services that best suit their needs [138, 140, 145, 146, 150, 151].

4. *Document-Driven DSSs*

Document-driven Decision Support Systems focus on retrieving and analyzing documents stored in various electronic formats within company systems, such as shared drives, cloud storage, or Data Asset Management (DAM) solutions. They are designed to organize and manage relevant documents, enabling users to search for specific information using keywords or search terms. Commonly used in industries where documentation is critical, such as law or academia, document-driven DSSs allow for efficient retrieval of unstructured information from a range of electronic sources. They search web pages, database documents and other information based on user queries to gather relevant data. These systems can be implemented through web-based platforms or client/server systems, making them accessible to a wide range of users. In essence, any use of a database's search function or an internet search engine involves a document-driven DSS [138, 140, 146, 150, 151].

5. *Communication-Driven DSSs*

Communication-driven Decision Support Systems are a very unique type of DSSs as they are designed to facilitate collaboration and communication among decision-makers who may be separated by time or space. These systems help companies manage tasks that require input from multiple people, providing tools for real-time communication such as email, instant messaging, voice chat and online collaboration platforms. Aimed primarily at internal teams and partners, communication-driven DSSs assist in conducting meetings, sharing documents and working collaboratively on projects, thus improving efficiency and effectiveness. Common deployment technologies include web-based platforms and client-server systems. These systems are ideal for virtual business meetings, online chat and video conferences given that they provide digital communication and shared access to decision tools, therefore supporting a collaborative decision-making process. Also known as group DSSs, communication-driven systems generally help streamline decision-making by allowing team members to share information and work together seamlessly, regardless of their physical location [138, 140, 145, 146, 150, 151].

6. *Intelligent DSSs*

An Intelligent Decision Support System (IDSS) integrates AI techniques in order to enhance decision-making processes. IDSS examples include Flexible Manufacturing Systems (FMS), intelligent marketing DSSs and medical diagnosis systems. These

systems aim to mimic human consultants by gathering evidence, diagnosing problems, suggesting actions and evaluating outcomes. Often based on expert systems, which encode knowledge and emulate human expert behavior using predicate logic rules, IDSSs can outperform human experts in certain situations but may struggle with novelty or uncertainty. Research in IDSSs focuses on developing flexible responses to uncertainty, employing techniques like intelligent agents, case-based reasoning and fuzzy logic. Managers, diagnosticians and other decision-makers use IDSSs to identify patterns, resolve problems and analyze solutions, using AI components like data mining and machine learning. Examples of IDSS software include smart manufacturing systems and medical diagnostic systems [145, 152].

7. *Manual DSSs*

A Manual Decision Support System is a more traditional, old-school type of decision support system that relies on the human factor rather than computer algorithms, to aid in decision-making processes within an organization or a project. In a manual DSS, a group of experts (economists, executives, managers, etc.) typically collaborates to analyze various aspects of this organization or project, including its strengths, weaknesses, opportunities and threats commonly known as a Strengths, Weaknesses, Opportunities and Threats (SWOT) analysis. By this way of working a comprehensive evaluation of the situation at hand is achieved. While manual DSSs are effective in involving human expertise and insights, they are often slower in processing information compared to computer-based DSSs because of the manual nature of the analysis, which requires human intervention at every step of the decision-making process [140, 145, 153].

8. *Hybrid DSSs*

A Hybrid Decision Support System combines manual processes with the computational power of software applications to support complex decision-making scenarios. They combine features from multiple types of DSSs, such as data-driven and knowledge-driven systems, using their strengths to address common in industries multifaceted issues like finance and healthcare. Hybrid DSS solutions use tools like spreadsheet analyses to compare options and evaluate what-if scenarios. They can also be implemented using a modular or integrated approach and might involve additional software to help different DSS components work together. Sometimes, human experts analyze and combine the results from each DSS. This model of cooperation ensures comprehensive data extraction

and manipulation, making hybrid DSS valuable for medical professionals, financial decision-makers and researchers [140, 145, 146, 153].

Further researching revealed that because DSSs incorporate advanced technologies like knowledge-based systems, real-time databases and intelligent agents to enhance their predictive and crisis management capabilities, they are therefore utilized across various domains such as pollution control, water resource management, epidemic prevention and of course flood management [154–158].

Summarized Comparison Table

Type of DSSs	***Description***	***Key Features***	***Advantages***	***Limitations***
Data-Driven DSSs	*Use data analysis to support decisions*	*Data storage, querying, visualization*	*Handle large data efficiently*	*Limited to structured data*
Model-Driven DSSs	*Rely on models for decision-making*	*Simulation, optimization, scenario analysis*	*Good for complex, quantitative decisions*	*Model accuracy is critical*
Knowledge-Driven DSSs	*Provide expert knowledge-based recommendations*	*Expert systems, AI-based recommendations*	*Automate expert tasks*	*Complex to maintain and implement*
Document-Driven DSSs	*Use unstructured documents to support decisions*	*Document management, text analysis*	*Useful for text-heavy fields*	*Difficult to analyze unstructured data*
Communication-Driven DSSs	*Facilitate collaboration for decision-making*	*Collaboration tools, communication platforms*	*Enhance team decision-making*	*May lead to groupthink*[6]
Intelligent DSSs	*Use AI and ML to aid decisions*	*AI-based insights, predictive analytics*	*Handle complex problems efficiently*	*Can be unclear and hard to interpret*
Manual DSSs	*Human-centric, with little automation*	*Rely on human analysis, basic tools*	*Flexible and low cost*	*Time-consuming and prone to errors*
Hybrid DSSs	*Combine multiple DSS types for decision support*	*Integrate features of other DSS types*	*Flexible, supports complex decisions*	*Complex to implement and manage*

Table 3: Types of DSSs comparison table [138–158]

[6] "Groupthink is the psychological drive for consensus at any cost that suppresses dissent and appraisal of alternatives in cohesive decision-making groups. [159]"

2.6.3 Types of DSSs in Flood Management

Flood management platforms encompass a range of systems designed to mitigate the impact of floods. These include Flood Information Systems, which centralize extensive flood-related data and display it on integrated dashboards, as well as Flood Early Warning Systems which provide advance warnings based on real-time data in order to facilitate timely responses [160, 161]. Smart Levee Monitoring Systems employ sensors within levees to detect early signs of potential breaches, while Group Decision Support Systems support collaborative decision-making in emergency situations [162, 163]. Likewise, Hybrid Analytics and Forecasting Models utilize a mix of statistical, machine learning and rule-based systems to enhance flood prediction and management [164, 165]. Integrated Smart Levee Systems and Urgent Computing simulate flood scenarios to predict levee failures, supported by advanced computing [164, 166], while Mobile and Web Applications enable both officials and citizens to access and share real-time flood information [162, 165].

Prior research suggests an implementation of a DSS for flood management which is an extensive web-based framework that was tested in Cedar Rapids and Charles City, Iowa, and is designed to optimize road network accessibility and emergency supplies distribution during flood events. It integrates server-side components, using PostgreSQL with PostGIS and FastAPI and client-side components developed with Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript, utilizing the Leaflet library for interactive maps. All these components let users assess road conditions since they can visualize open and closed roads, calculate the shortest path between critical points of interest, manage facility allocation and access multiple service coverage functions. The proposed DSS supports a variety of key analytical methods such as graph theory for road network analysis, bridge functionality evaluation using LiDAR data, routing to critical facilities by employing the Dijkstra algorithm and optimal emergency equipment allocation by integrating the P-median model. High-resolution flood maps and road network data are also included to provide real-time information on road closures, as well as demographic data and thus it enhances community resilience and facilitates informed decision-making and planning during a flood [120].

A similar flood management DSS focused on data-driven decisions was implemented in Jakarta, Indonesia by researchers. The system, which is an integration of technology and analytics, operates in three phases: Sensing, Understanding and Acting, to enhance the

city's flood resilience. In the Sensing phase, IoT sensors are strategically placed across 178 locations to measure crucial parameters such as rainfall, water levels, water currents and temperature. All these sensors stream data in real-time and their values are stored in a centralized yugabyteDB database system. Stored data then undergoes rigorous analysis in the Understanding phase, utilizing the Data, Information, Knowledge and Insights (DIKI) pyramid model. The model facilitates the generation of actionable insights regarding flood risks and predictive analytics. Advanced machine learning techniques, including decision trees, random forests and neural networks, are also applied to enhance the analytical rigor of the system. In the Acting phase, a centralized dashboard provides real-time data and predictive analytics to decision-makers and flood control officers in Jakarta, supporting strategic and operational decision-making. This system also serves as an early warning mechanism, disseminating timely information to the public about impending flood threats [167].

Likewise, for the Province of Leyte in the Philippines a DSS which incorporates an integrated system of hardware sensors and web-based software to enhance flood risk management was proposed. Utilizing Arduino-based sensors equipped with GSM modules, the system collects real-time data on rainfall and water levels from various strategic locations. This data is then transmitted to a central web platform that converts it into comprehensible reports and graphical representations for real-time monitoring. Administrators can manage sensors, categorize alerts and issue warnings via a user-friendly interface that supports immediate data access and visualization. When predefined thresholds are exceeded, the system triggers Short Message Service (SMS) alerts to local government units and emergency responders, so that they can respond in a timely and effective manner to manage flood impacts. Key to the system's effectiveness is its security measures that ensure data integrity and confidentiality, with robust encryption for sensitive data and continuous monitoring of system operations [168].

A massively innovative solution is discussed by some authors, and is about the development of a GDSS for early flood warning in Indonesia, particularly for the Kali Sadar River in Mojokerto Regency, East Java. This solution employs the Analytical Network Process (ANP) and VlseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR) methodologies to manage and synthesize decisions from multiple stakeholders including government and water resource managers. The GDSS model is innovative as it integrates Multiple Criteria Decision-Making (MCDM) to provide a structured approach

to flood hazard early warnings. The system processes data on rainfall, river discharge, water levels, embankment conditions and drainage, to assess flood risk and generate warnings. When different rankings arise from the VIKOR method, the Borda count method is applied to aggregate these diverse opinions into a consensus decision, enhancing the objectivity and reliability of the decision-making process. The performance of the GDSS was evaluated by the use of Spearman Rank Correlation and a Confusion Matrix in order to ensure accuracy and reliability, resulting in positive findings. Spearman correlation showed a high degree of alignment between the system-generated rankings and established criteria, while the Confusion Matrix confirmed the system's high accuracy, precision, recall and F1-score, all at 86.7%. This implementation demonstrates significant potential for improving disaster management and risk reduction through systematic and above all collaborative decision-making [169].

The following implementation of a DSS that utilizes a hierarchical architecture to efficiently allocate resources and tasks during the phase of response in a flood was identified in the literature and was discussed briefly in the previous chapter. It is a system that orchestrates the distribution of resources from top-level authorities downward through multiple levels of Control Centers, which include the Top-Level Control Centre (TCC), Middle-Level Control Centre (MCC) and Low-Level Control Centre (LCC). Resources and tasks are allocated based on real-time data regarding availability and the severity of the disaster scenario. The process begins at the TCC, where resources are initially allocated to MCCs based on their respective demands and criticality of needs. These resources are then further distributed by MCCs to LCCs, which tailor the allocations to the localized requirements of the emergency sites. At the LCC level, tasks are prioritized and assigned to manage various emergency actions such as rescue operations, evacuation and relief distribution. Critical tasks receive priority based on their urgency and the impact on affected populations. Supporting this hierarchical structure, there is a software framework equipped with a Graphical User Interface (GUI) developed in MATLAB, allowing for interactive user input regarding crisis situations. It is an exceptional framework for it enables dynamic adjustments to resource distribution in response to evolving disaster conditions [129].

Building upon the foundational DSS frameworks previously discussed, hereinafter cutting-edge technologies that are redefining the current landscape of decision support are explored. Frameworks that offer transformative possibilities providing deeper insights

and more dynamic solutions to decision-making processes, were located and chosen for mapping through academic libraries.

2.6.4 Types of DSSs in Other Operations

It is apparent that ML and AI are integrated into modern DSSs to enhance decision-making processes through sophisticated computational methods. In this sense, MCDM techniques are employed to elevate DSSs into what is commonly referred to as Intelligent DSSs. Complex algorithms are used to analyze multiple criteria simultaneously, supporting more informed and scaled decision-making that can adjust to different priorities and limitations. This concept is of high importance in environments that require strong and flexible response mechanisms, such as emergency management and strategic planning [170].

In the realm of healthcare, an application of AI within Clinical Decision Support Systems (CDSSs) is elaborated in an eye-catching publication. CNNs, a form of Deep Learning particularly adept at processing and interpreting visual data, are used in CDSSs for medical imaging data analysis so as to assist clinicians in diagnosing diseases, thus illustrating how AI can enhance the accuracy and efficiency of health-related decision-making processes [171].

The utility of ML algorithms in DSSs is further explored by applying supervised learning techniques for binary classification tasks within DSSs. There is a highlighted impact of their use, resulting from the favorable effect in categorizing input data into predefined classes. This is essential for systems that require a binary decision output, such as approving or denying loan applications or identifying risk levels in insurance underwriting [172].

Turning to the logistical aspect of a DSS, there is a performance research study that showcases the utilization of Docker containers as a case study. Containers encapsulate software in a complete file system that contains everything it needs to run: code, runtime environment, system tools, system libraries and anything that can be installed on a server. This provides an extraordinary level of amenity for easier deployment and scalability of DSS applications across different environments, while at the same time they remain reliable and consistent irrespective of the underlying infrastructure. Docker containers' value is realized in disaster management scenarios, where rapid deployment and high reliability of a DSS are critical for effective response [173].

Lastly, the integration of GIS into a DSS, is now becoming mandatory in land planning and management. GIS enhances DSSs by providing spatial analysis tools which aid in visualizing and interpreting data with a geographic component. These additions are further enlightening in a variety of applications, from urban planning and environmental conservation to agricultural management, where understanding the geographical distribution of resources or constraints is of supreme importance [174].

It is evident now that Decision Support Systems are indispensable tools in flood incident management. They improve decision quality, expedite problem-solving and enhance organizational control. The current landscape of DSS platforms includes features which further augment their utility in managing flood incidents. This enhanced level of control facilitates the seamless coordination of every part of the response.

2.6.5 Dashboards in DSSs

An aforementioned structural element of an integrated decision-making system is a Dashboard. Dashboards are strongly required in DSSs, to allow for the real-time monitoring and visualization of data during disaster incidents. They can provide quick snapshots of essential information, integrating data from diverse sources, all of which are possible to be depicted in a creative visual environment. Forming tools that aggregate and present complex data often through charts, graphs and indicators, they are designed to provide to users, especially to managers and executives, a quick and clear overview of current performance and data updates [151, 175].

Dashboards are highly customizable, allowing users to monitor Key Performance Indicators (KPIs), metrics and other relevant data points that are prerequisites for making informed decisions. They have become a popular tool for displaying information in a concise and easy-to-understand manner. In order to effectively utilize dashboards, it is important to first examine their design and implementation in existing applications. This can help identify best practices for creating dashboards that are intuitive, informative and highly useful. Also, understanding how users interact with dashboards can help improve their functionality and ensure that they are meeting the needs of the target audience [148, 176].

There are various types of dashboards that serve different purposes depending on their design and functionality. Below are some common ones, complemented by representative products. While numerous products exist, a complete analysis is beyond the scope of this

study. The listed ones sufficiently demonstrate the range of approaches and performance characteristics relevant to this research, to cover the perspective of choosing the appropriate tool on a case-by-case basis.

1. Operational Dashboards

Operational Dashboards are tools that integrate data from various sources, both structured and unstructured, to provide real-time and historical insights, all of which contribute to a huge extent to decision-making. They use big data analytics to process large volumes of streaming data and transform raw data into actionable intelligence through statistical correlation and other analytical methods. These dashboards are designed to offer insights into various metrics such as order flow health, inventory levels, sales performance and more, thus allowing for better operational decisions and strategic business actions [177].

Representative Products

Tableau and Grafana are indicative tools for Operational Dashboards [178]. Tableau is ideal for businesses seeking powerful visual data representation and advanced analytics with an easy-to-use interface. It is particularly well-suited for business users who need to integrate data from various sources and perform complex analyses. It also provides automated data enrichment enhancing data discovery and understanding [179]. Grafana on the other hand, is better suited for real-time monitoring and IT infrastructure visualization. Grafana dashboards provide insightful evaluation of data gathered from several sources, have dynamic characteristics and can be shared with team members, promoting collaborative data exploration and transparency. They also offer extensive customization and integration with diverse data sources. Its open-source nature makes it cost-effective though it requires more technical expertise to set up and use effectively [180]. The following table summarizes their strengths and weaknesses to provide.

Comparison Table

Feature	***Tableau***	***Grafana***
User-Friendly Interface	*Yes*	*No*
High-Quality Visualizations	*Yes*	*Yes*
Data Integration	*Yes*	*Yes*
Advanced Analytics	*Yes*	*No*
Customization	*Limited*	*Extensive*
Real-Time Monitoring	*Limited*	*Yes*

Feature	***Tableau***	***Grafana***
Plugin Ecosystem	*No*	*Yes*
Cost-Effective	*No*	*Yes*
Strong Community Support	*Yes*	*Yes*
Built-In Alerts	*No*	*Yes*
Open-Source	*No*	*Yes*
Performance with Large Datasets	*No*	*Yes*
Deployment Complexity	*High*	*Moderate*

Table 4: Tableau vs Grafana [178–180]

2. Strategic Dashboards

A Strategic Dashboard is a visual tool designed to help managers and employees monitor, analyze and manage key performance metrics. It enables the identification of trends, patterns and anomalies in a company's performance and supports decision-making in this respect. Strategic Dashboards are characterized by features that enhance their usability and effectiveness, including format flexibility, real-time notifications, scenario analysis and visual elements like high data-to-ink ratios and the use of grids. They are mostly used due to the coherent and interactive representation of critical business processes they provide, which makes strategy execution and employee engagement easier [181].

Representative Products

Microsoft Power BI and Qlik Sense are indicative tools for Strategic Dashboards. Microsoft Power BI is an excellent choice for organizations seeking an affordable, user-friendly BI tool with sturdy data integration, advanced analytics and strong community support. Its natural language querying and seamless integration with other Microsoft products make it particularly appealing for users already in the Microsoft ecosystem [182]. Qlik Sense offers powerful in-memory processing, extensive customization and advanced analytics capabilities, making it suitable for organizations needing highly customizable and fast data processing solutions. Its higher cost and lack of natural language querying are offset by its strengths in self-service BI and interactive visualizations [183]. Their capabilities are summarized in the following table.

Comparison Table

Feature	***Microsoft Power BI***	***Qlik Sense***
User-Friendly Interface	*Yes*	*Yes*

Feature	***Microsoft Power BI***	***Qlik Sense***
High-Quality Visualizations	*Yes*	*Yes*
Data Integration	*Yes*	*Yes*
Advanced Analytics	*Yes*	*Yes*
Customization	*Limited*	*Extensive*
Real-Time Data Access	*Yes*	*Yes*
Self-Service BI Capabilities	*Yes*	*Yes*
Natural Language Querying	*Yes*	*No*
AI and Machine Learning Integration	*Yes*	*Yes*
Cost-Effective	*Yes*	*No*
Strong Community Support	*Yes*	*Yes*
Built-In Data Governance	*Yes*	*Yes*
Mobile Access	*Yes*	*Yes*
On-Premises Deployment	*Yes*	*Yes*
Open-Source	*No*	*No*
Performance with Large Datasets	*Yes*	*Yes*
Data Preparation Tools	*Yes*	*Yes*
Interactive Dashboards	*Yes*	*Yes*
Embedded Analytics	*Yes*	*Yes*
Integration with Microsoft Products	*Yes*	*No*
In-Memory Processing	*No*	*Yes*

Table 5: Microsoft Power BI vs Qlik Sense [181–183]

3. Analytical Dashboards

An Analytical Dashboard is a tool designed to provide deep insights and understanding through the aggregation, analysis and visualization of complex data sets. It focuses on data exploration, trend analysis and the identification of patterns to support strategic decision-making and in-depth analysis. Analytical Dashboards typically incorporate advanced features such as interactive data visualizations, detailed metrics and the ability to drill down into data to uncover underlying trends and correlations. In the context of learning analytics, an Analytical Dashboard would present comprehensive data about

learners, learning processes and learning contexts, enabling stakeholders to perform detailed analyses. This type of dashboards often include features for visualizing patterns, outliers and trends in learning activities, which help in making targeted decisions to enhance educational outcomes [184, 185].

Representative Products

Two indicative dashboards that fall into that category are International Business Machines Corporation (IBM) Cognos Analytics and Apache Superset. IBM Cognos Analytics is a comprehensive BI tool with powerful enterprise-level features, including advanced analytics, strong data governance and AI integration. It is well-suited for large organizations that require extensive customization, high-quality visualizations and strong data integration capabilities, but it comes at a higher cost and with added complexity. On the contrary, Apache Superset is an open-source, cost-effective BI tool with strong visualization capabilities, real-time data access and extensive customization options. It is ideal for organizations looking for a scalable, lightweight solution with strong community support, although it may lack some of the advanced analytics and enterprise-level features found in proprietary solutions like IBM Cognos Analytics. The following table is a comparison of their offered features [186, 187].

Comparison Table

Feature	***IBM Cognos Analytics***	***Apache Superset***
User-Friendly Interface	*Yes*	*No*
High-Quality Visualizations	*Yes*	*Yes*
Data Integration	*Yes*	*Yes*
Advanced Analytics	*Yes*	*No*
Customization	*Extensive*	*Extensive*
Real-Time Data Access	*Yes*	*Yes*
Self-Service BI Capabilities	*Yes*	*Yes*
Natural Language Querying	*Yes*	*No*
AI and Machine Learning Integration	*Yes*	*No*
Cost-Effective	*No*	*Yes*
Strong Community Support	*Yes*	*Yes*
Built-In Data Governance	*Yes*	*Limited*

Feature	***IBM Cognos Analytics***	***Apache Superset***
Mobile Access	*Yes*	*Yes*
On-Premises Deployment	*Yes*	*Yes*
Open-Source	*No*	*Yes*
Performance with Large Datasets	*Yes*	*Yes*
Data Preparation Tools	*Yes*	*Yes*
Interactive Dashboards	*Yes*	*Yes*
Embedded Analytics	*Yes*	*Yes*
Scalability	*Yes*	*Yes*

Table 6: IBM Cognos Analytics vs Apache Superset [186–188]

4. Tactical Dashboards

A Tactical Dashboard is defined as a system used to monitor the managerial activities of a department or project over a medium-term period. It presents detailed information necessary for managers to analyze the results of business processes within a department. Tactical dashboards are distinguished from other types of dashboards by their focus on medium-term objectives and detailed data display, which support managerial decision-making and performance monitoring at the department or project level [189].

Representative Products

Domo and Klipfolio are indicative solutions when it comes to Tactical Dashboards. Domo is a powerful BI tool with a user-friendly interface, high-quality visualizations, extensive data integration and advanced analytics capabilities. It is well-suited for large organizations that require real-time data access, strong collaboration features and scalability, but this also comes at a higher cost and complexity added [190]. Klipfolio is an affordable, user-friendly BI tool with strong data integration and real-time data access capabilities. It is ideal for small and medium-sized businesses looking for a cost-effective solution with high-quality visualizations and extensive customization options [191]. However, it may lack some of the advanced analytics and enterprise-level features found in more robust BI solutions like Domo. Their features are summarized in the following table.

Comparison Table

Feature	***Domo***	***Klipfolio***
User-Friendly Interface	*Yes*	*Yes*
High-Quality Visualizations	*Yes*	*Yes*
Data Integration	*Yes*	*Yes*
Advanced Analytics	*Yes*	*No*
Customization	*Extensive*	*Extensive*
Real-Time Data Access	*Yes*	*Yes*
Self-Service BI Capabilities	*Yes*	*Yes*
Natural Language Querying	*No*	*No*
AI and Machine Learning Integration	*Yes*	*No*
Cost-Effective	*No*	*Yes*
Strong Community Support	*Yes*	*Yes*
Built-In Data Governance	*Yes*	*Limited*
Mobile Access	*Yes*	*Yes*
On-Premises Deployment	*No*	*Yes*
Open-Source	*No*	*No*
Performance with Large Datasets	*Yes*	*Yes*
Data Preparation Tools	*Yes*	*Yes*
Interactive Dashboards	*Yes*	*Yes*
Embedded Analytics	*Yes*	*Yes*
Scalability	*Yes*	*Limited*
Collaboration Features	*Yes*	*Limited*

Table 7: Domo vs Klipfolio [190–192]

All Dashboard Types Comparison

In light of all the research that has been done, Operational Dashboards are best for monitoring real-time operations and immediate response needs, Strategic Dashboards are ideal for tracking high-level metrics and long-term goals by providing insights for executives, Analytical Dashboards are suited for detailed data analysis and identifying trends, proving extremely useful for data analysts and finally, Tactical Dashboards are focused on short-term metrics and tactical decision-making, aiding the middle management. The following table summarizes their overall utilities and features.

Summarized Comparison Table

Feature	*Operational Dashboards*	*Strategic Dashboards*	*Analytical Dashboards*	*Tactical Dashboards*
Real-Time Monitoring	*Yes*	*No*	*Yes*	*Yes*
High-Level Metrics	*No*	*Yes*	*Yes*	*Yes*
In-Depth Analysis	*No*	*Yes*	*Yes*	*No*
Short-Term Metrics	*Yes*	*No*	*Yes*	*Yes*
Long-Term Goals	*No*	*Yes*	*Yes*	*Yes*
Data Visualization	*Yes*	*Yes*	*Yes*	*Yes*
Predictive Analytics	*No*	*Yes*	*Yes*	*No*
User Accessibility (Non-Technical Users)	*Yes*	*Yes*	*Limited*[7]	*Yes*
Integration with Multiple Data Sources	*Yes*	*Yes*	*Yes*	*Yes*
Customization	*Limited*	*Extensive*	*Extensive*	*Moderate*

Table 8: Dashboards Types Comparison Table [177, 181, 184, 185, 189]

2.7 Existing Safety Procedural Protocols and Systems: Analysis of current procedural protocols and systems for flood emergency management.

A procedural protocol is a predetermined sequence of actions or procedures that delineate the proper execution of a specific task or process [193]. It serves as a comprehensive manual to ensure that each stage of the process is carried out accurately and consistently. Inadequate production or poor compliance with protocols can result in confusion, wastage of resources and precious time, inaccurate outcomes and safety issues. Regarding the scope of flood management, a procedural protocol would offer a methodical and organized way to implement and execute the system. This technique enables the standardization of actions, minimizes errors and guarantees adherence to safety and operational requirements. The absence of structured cooperation between stakeholders in disaster management can lead to a lack of coordination, poor communication and information sharing, insufficient resources and a lack of awareness, all of which can

[7] Analytical dashboard ownership usually falls on business analysts/experts.

hinder efforts to reduce losses and build resilience [194, 195]. Therefore, a protocol must be established, particularly for flood management.

Following a thorough examination encompassing scholarly literature, governmental publications and other relevant sources, no documented framework or established protocol pertaining to the management of flood crises was found. This notable absence denotes the lack of a structured, formalized procedure describing requisite actions and strategies in response to flood events. In scholarly terms, the missing of such direction represents a gap in the existing literature, potentially impeding comprehensive and immediate treatment in places with a high risk of flooding events.

Previous efforts, have mainly focused on risk assessment, sustainable urban planning and integrating meteorological and hydrological data to enhance prediction models. Researchers aim to provide solutions that not only reduce the consequences of floods but also improve community preparedness and resilience worldwide.

In the following sub-chapters, some of the procedures that were discovered following a lengthy search, are listed.

2.7.1 Bangladesh: Comprehensive Disaster Management Programme

Bangladesh’s Comprehensive Disaster Management Programme (CDMP) is a highly ambitious initiative aimed at addressing the country’s vulnerability to a wide range of natural hazards, including tropical cyclones, floods and riverbank erosion. The program's proactive approach represents a significant shift from traditional disaster response and relief efforts to comprehensive risk reduction and climate change adaptation strategies.

The CDMP is structured in two phases. Phase I (2004–2009) focused on laying the foundation for long-term disaster risk reduction and climate change adaptation across seven targeted districts. Key achievements during this phase included the development of hazard, risk and vulnerability maps, the establishment of an early warning system and the enhancement of disaster management capacities at various government levels. Phase II (2010–2014) aimed to build upon these foundations, expanding the program's reach and ensuring the institutionalization of risk reduction and climate adaptation across all levels of government. This phase had notable outcomes including the establishment of the Disaster Management Information Centre (DMIC) and the implementation of the Community Risk Assessment (CRA) program, which engaged local communities in disaster risk reduction planning and action.

The DMIC aims to enhance the availability, quality and management of information crucial for emergency disaster response. It operates continuously, monitoring and reporting natural hazards through a web portal and mobile phones, covering 64 districts and 235 high-risk sub-districts. By supporting Bangladesh's Meteorological Department and Flood Forecasting and Warning Centre, it has improved the timeliness and effectiveness of flood warnings. For instance, during the 2010 flooding season, residents of the Sirajganj district received text messages with water levels and river forecasts 72 hours in advance, enhancing preparedness and response for both citizens and the government.

The CRA program employs participatory methods to identify and evaluate community hazards, risks and vulnerabilities. It assists communities in creating Risk Reduction Action Plans (RRAPs) that prioritize disaster risk reduction activities. These local RRAPs are then consolidated into union and district plans, ultimately forming a national action plan to ensure government policies which reflect local needs. Communities can apply for funding from a government and donor-supported fund to implement their RRAPs.

Furthermore, since 2007, over 25,000 officials at various levels have received disaster management training through the CDMP. The programme has established collaborations and training partnerships to enhance the technical capacity of government officials and has engaged 14 national universities in developing disaster management curricula. Local capacity building includes the Livelihood Adaptation to Climate Change (LACC) programme, which helps rural communities adapt to climate risks and is implemented by the Department of Agricultural Extension. The CDMP has also influenced the expansion of Bangladesh's legal framework on disaster risk reduction, including the National Plan for Disaster Management, adherence to the Hyogo Framework for Action and the integration of climate change adaptation and disaster risk reduction into the Poverty Reduction Strategy Papers and Climate Change Action Plan.

Despite its successes, the CDMP has faced several challenges, including natural disasters, political instability and high turnover of high-level officials, which disrupted the program's implementation. Institutionalizing disaster risk reduction and climate change adaptation beyond the Ministry of Food and Disaster Management has been slow and difficult, highlighting the need for broader governmental engagement and capacity building. Another significant challenge is ensuring long-term sustainability and funding. Although Phase I successfully engaged numerous communities and developed robust

disaster management frameworks, the reliance on external funding poses a risk. Increased national funding sources, including contributions from the private sector and local communities, are essential for sustaining the program's initiatives.

The CDMP has been instrumental in shifting Bangladesh's disaster management paradigm from reactive to proactive by focusing on risk reduction and climate change adaptation, thus vividly operating in the pre-flood phase on multiple facets. The program's collaborative networks and high-level government support, have been critical to overcoming initial implementation hurdles and expanding its reach. However, ensuring the program's long-term impact requires continued efforts to institutionalize these practices across all government levels and secure sustainable funding sources. The CDMP's evolution and achievements provide valuable insights and lessons for other developing countries facing similar disaster management challenges [196].

2.7.2 An Expert System for Local Flood Response Coordination and Training

The Local Flood Response System (LFRS) is a proposal that outlines the construction of an expert system that is intended to improve the coordination and training of local flood response initiatives, with the goal of enhancing operations pertaining to flood crisis response. This system is intended to support emergency managers in establishing effective coordination structures and providing education to flood response entities. The LFRS is composed of two primary modules: the Roles Responsibilities and Actions (RRA) module and the Response Operational Structure (ROS) module.

The RRA module is responsible for generating detailed reports that outline the responsibilities and actions required of different roles involved in flood response. Meanwhile, the ROS module is tasked with creating hierarchical operational structures based on the Incident Command System (ICS). These structures are crucial for organizing response efforts effectively during a flood event. The system leverages an inference engine to gather and process the necessary data for both modules.

The LFRS was developed using the Python Knowledge Engine (PyKE), a characteristic that is helpful in developing an expert system with a large knowledge base facilitating the creation of a robust and flexible system [197]. The development process included rigorous verification and validation to ensure the system's accuracy and reliability in real-world scenarios. The document highlights that while the LFRS has proven effective, there is potential for further enhancements. These improvements could enable the system to

manage larger and more complex flood events, thereby increasing its utility and effectiveness in diverse flood response situations [198].

2.7.3 Standard Operating Procedure on Flood Disaster Response

The Standard Operating Procedure (SOP) for the National Disaster Response Force (NDRF) of India, is specifically focused on flood disaster response and addresses the frequent and destructive nature of floods in the country, delineating the extensive damage they inflict on human lives, property and infrastructure. The SOP is intended to establish a structured framework for NDRF's flood response, ensuring comprehensive support to local authorities and the execution of efficient Humanitarian Assistance and Disaster Relief (HADR) operations.

Floods as described in this framework, are characterized as the overflow of water submerging land, predominantly caused by excessive monsoon rains, cyclones, tsunamis, or seismic activities. India, particularly the North-Eastern States, is highly vulnerable to such events. Approximately 40 million hectares, or 10% of India’s landmass, are susceptible to flooding, resulting in significant annual casualties and economic losses.

The primary objective of the SOP is to delineate guidelines for the NDRF battalions to respond effectively to flood emergencies. It aims to ensure the prompt execution of coordinated SAR operations, adopted from the International Search and Rescue Advisory Group (INSARAG) guidelines [199], thereby minimizing reaction times. It specifies roles and responsibilities for responders and commanders to enhance the preparedness and efficacy of disaster management activities.

There are additional roles and responsibilities across various levels within the NDRF, emphasizing the importance of flood preparedness, continuous risk mitigation activities and effective coordination among stakeholders. The decision-making process for deploying NDRF teams is featured, including conditions under which state governments, central authorities, or local commanders can requisition assistance.

In the execution phase, the SOP outlines the establishment of Emergency Operation Centers (EOCs) at NDRF headquarters and battalion levels. These centers are important for maintaining communication with state and district authorities, collecting and disseminating flood emergency information and ensuring operational readiness. Preparatory measures detailed in the SOP include planning and coordination with

forecasting agencies, maintaining personnel and equipment readiness and conducting regular training and mock drills[8].

During the activation stage, the SOP describes the mobilization of SAR teams based on the flood's severity and impact and it ensures the teams' readiness, logistical and operational support and effective deployment. Regular communication and detailed briefings are emphasized as a necessity for coordinated rescue operations.

The operational stage focuses on establishing On-Site Operations Coordination Centers (OSOCC), maintaining NDRF's self-sufficiency to avoid straining local resources and ensuring the safety and security of NDRF personnel. The primary tasks of NDRF during flood response is outlined, including rescuing victims, providing medical assistance and supporting local authorities in relief distribution as well as shelter management.

Finally, the de-activation stage specifies the orderly withdrawal of NDRF teams in the post-disaster response period. It includes the obtaining of necessary clearances, the collection of feedback, the addressing of the physical and mental well-being of rescuers and the implementation of lessons learned for future operations. The procedure stresses the importance of a seamless transition and the incorporation of feedback into preparatory stages for enhanced future response [201].

2.7.4 Netherlands: Delta Works Engineering Project

Delta Works, a monumental flood-control project in the southwestern Netherlands, stands as a testament to innovative engineering and proactive disaster management. This project, devised by Dutch engineer Johan van Veen, aimed to safeguard the region from the destructive forces of the North Sea by closing off the Rhine, Meuse (Maas) and Schelde estuaries with a network of dikes. The urgency of the project was amplified by the catastrophic North Sea floods of 1953, which resulted in the tragic loss of more than 1.800 lives and extensive damage to over 2.000 square kilometers of land. Construction commenced shortly after the disaster and culminated in 1986. Recognized as one of the Seven Wonders of the Modern World by the American Society of Civil Engineers, the Delta Works is a prime example of human ingenuity in the face of natural calamities.

[8] "A mock drill is a simulated exercise or practice run that imitates a real-time emergency, preparing the individuals & organization to evaluate the potential weakness, improving the response time and readiness to handle emergency crises during fires, hazardous chemical release, medical emergencies and natural disasters" [200].

This engineering project comprises a series of 13 dams, including four primary barriers and nine secondary dams, strategically constructed to close off the mouths and inner reaches of long, interconnected inlets. These inlets had historically left the region vulnerable to the North Sea storms. The project significantly shortened the region's exposed coastline by 700 kilometers, thereby reducing the length of dikes vulnerable to the sea's destructive power. As a result, the Rhine and other rivers' freshwater gradually replaced the entrapped saltwater, creating several freshwater lakes free of tidal influences. The total length of these dams extends to approximately 30 kilometers.

The Delta Works not only serves as a flood defense mechanism but also significantly improved regional connectivity. New roadways and bridges constructed over the dams and dikes ended the historical isolation of the area from the rest of the Netherlands, integrating it more fully into the national infrastructure. Between 1995 and 2015, numerous dikes were reinforced or replaced, and plans were made for additional improvements, including higher and wider dikes, to combat the anticipated sea-level rise due to climate change.

Apparently, it is a pre-flood marvel of infrastructure, that exemplifies a successful large-scale engineering project which addresses both immediate disaster response needs and long-term environmental sustainability. Its construction has not only provided critical flood protection but also promoted regional development and ecological balance. Sea levels continue to rise, so the ongoing enhancements to the Delta Works demonstrate a commitment to adapting and fortifying this vital infrastructure against future challenges. The project remains a remarkable achievement in civil engineering and a crucial component of the Netherlands' flood management strategy [202].

Upon further exploration, more extensive crisis response frameworks were discovered, which an increasing number of countries are attempting to implement.

2.7.5 FEMA's National Response Framework (U.S.)

The NRF by **F**ederal **E**mergency **M**anagement **A**gency, is a comprehensive emergency management scheme that outlines how the United States respond to all types of disasters and emergencies. Its scalability, adaptability and flexibility let important tasks and responsibilities be aligned across several tiers of government and sectors. The NRF emphasizes the importance of cooperation among a variety of organizations, including

government agencies, the private sector and non-governmental organizations (NGOs), to ensure a coordinated and effective response to incidents of varying complexity and scale.

The concept of community lifelines is introduced by the NRF and enables the continuous operation of critical government and business functions. Only the most fundamental services that a community depends on are represented by the seven community lifelines: *Safety and Security, Food-Water-Shelter, Health and Medical, Energy (Power & Fuel), Communications, Transportation and Hazardous Materials*. When these services are steady, they allow for all other community activities. After all, during disaster response efforts, the most essential thing is undoubtedly to mitigate threats to public health, safety and the economy.

The framework aligns with the National Incident Management System (NIMS), providing a standardized approach to incident management. NIMS principles facilitate unified command, interoperability and coordination among several jurisdictions and sectors. The NRF identifies core capabilities across five mission areas: *Prevention, Protection, Mitigation, Response and Recovery*. These are necessary to address the full spectrum of threats and hazards and to ensure a resilient nation. Examples of these capabilities include operational coordination, public information and warning and public health services. In addition, the NRF organizes response capabilities into functional areas most frequently needed during incidents, known as Emergency Support Functions (ESFs), which include public works, firefighting and mass care services, amongst others. ESF #14 Cross-Sector Business and Infrastructure Annex for example, is introduced to enhance coordination with the private sector and infrastructure owners/operators.

Recognizing the vital role of the private sector, the NRF fosters better coordination and collaboration with businesses and infrastructure owners to stabilize community lifelines and support incident response. Lifelines are the most fundamental services in a community and when stabilized, allow all other aspects of society to function. Finally, it is directed by a set of guiding principles:

- Engaged Partnership involves all levels of government, the private sector, NGOs and the public in preparedness and response activities
- Tiered Response ensures that incidents are managed at the lowest possible jurisdictional level, with additional support provided as needed
- Scalable, Flexible and Adaptable Operational Capabilities ensure that response efforts can adapt to the changing size, scope and complexity of incidents

- Unity of Effort Through Unified Command guarantees a coordinated and efficient response through a single incident action plan
- Readiness to Act emphasizes the importance of a proactive posture to rapidly respond to incidents

All the principles and concepts the NRF embodies, such as scalable and adaptable response, community lifelines and integrated emergency management, can serve as a model for other countries developing their own national response frameworks [30, 203–206].

2.7.6 INSARAG Operational Field Guide

The INSARAG is a network of countries and organizations focused on Urban Search And Rescue (USAR) and field coordination during disasters. Its primary goals are to set standards and classification for international USAR teams and to develop a methodology for coordinating international responses to earthquakes and structural collapse disasters [207]. However, there have been intense and increased discussions over the last years on adopting ways to extend the deployment of INSARAG teams in various other disasters that does not necessarily involve collapsed buildings [208, 209]. As far as the floods are concerned, in the INSARAG AEME (Africa-Europe-Middle East) Regional Meeting in Doha, Qatar in October 2023, a Search and Rescue – Flood Response Working Group has been assigned to output the INSARAG Flood Response Standard, which is yet to be released [210].

According to INSARAG, each international USAR intervention is defined by an accurately stated response cycle consisting of the following phases [211]:

Preparedness

The Preparedness phase occurs between disaster responses. During this period, USAR teams implement measures to achieve optimal readiness for potential deployments. This includes conducting training sessions and drills, reviewing lessons learned from past experiences, updating SOPs as necessary and strategizing for future responses.

Mobilization

The Mobilization phase takes place immediately after a disaster occurs. During this time, international USAR teams get ready to respond and travel to provide assistance to the affected country.

Operations

Following, is the Operations phase which occurs when international USAR teams are actively engaged in USAR operations within the affected country. This phase begins with the team's arrival at the Reception/Departure Centre (RDC) in the affected country, followed by registration with the OSOCC, reporting to the Local Emergency Management Agency (LEMA) or National Disaster Management Authority (NDMA) and the execution of USAR operations. The phase concludes when the USAR team is instructed to cease operations.

Demobilization

In turn, the Demobilization phase occurs when international USAR teams are instructed to halt their operations. During this phase, their withdrawal begins by coordinating their departure through the OSOCC and subsequently exit the affected country via the RDC.

Post-Mission

The Post-Mission phase begins immediately after a USAR team returns back to its home base. During this phase, the team must complete and submit a post-mission report and conduct a lessons-learned review to enhance the overall effectiveness and efficiency of future disaster responses. This phase seamlessly transitions into the preparedness phase.

Every INSARAG mission consists of a list of actions: Management, Search, Rescue, Medical, Logistics and finally, Safety and Security. In turn, each action has and follows its own response cycle, that contains the phases described above, but each phase has very specific and action related steps and conditions for the USAR to follow.

It is expected that the same, or quite similar approach will be followed in the events of a flooding natural disaster response campaign.

The INSARAG Guidelines delineate a comprehensive framework for SAR procedures, which are essential for managing and coordinating international response efforts during major USAR incidents. In addition to outlining specific procedures for each level of SAR operations, these guidelines also stress the need of structured operations.

SAR procedures within the INSARAG framework are divided into multiple levels, each with specific tasks and objectives. The five defined levels range from initial assessments to full-scale search and recovery operations. The first level involves wide-area assessments to understand the scope of the disaster and identify the most affected areas. This is followed by worksite triage assessments, which prioritize specific locations based

on the likelihood of finding survivors and the resources required. The third level focuses on rapid SAR operations, under the aim of quickly locating and extricating accessible victims. Subsequent levels involve more extensive and intensive efforts, including the full search and rescue of deeply entombed survivors and finally, the total coverage search and recovery, which ensures that all potential victims are accounted for, whether they are alive or deceased.

In addition to defining these operational levels, the guidelines outline the necessary steps for mobilization, coordination and post-mission activities. Mobilization involves preparing and deploying USAR teams with the appropriate equipment and ensuring that all team members are properly briefed regarding their roles and responsibilities. During the mission, coordination is maintained through established communication protocols and information management systems such as the INSARAG Collection and Management System (ICMS), which help in tracking progress and managing resources effectively. Post-mission activities include debriefings, evaluations and the compilation of comprehensive reports to document the operations and lessons learned. As highlighted before, these are important for improving future responses.

Regarding flood-specific guidelines, the INSARAG framework acknowledges the unique challenges posed by hydrometeorological hazards, including floods. The guidelines recommend a detailed risk assessment to understand the potential impact of floods and the necessary response mechanisms. This includes identifying flood-prone areas, preparing for swift water rescues and ensuring that teams are equipped with the appropriate gear for underwater and surface rescues.

To summarize, the INSARAG Guidelines provide a structured approach to SAR operations, ensuring that international rescue teams can operate effectively in various disaster scenarios, including floods. By adhering to these guidelines, teams can enhance their preparedness, resulting in better outcomes for impacted communities [199].

2.7.7 The rescEU Mechanism

In recent years, Europe has faced an array of natural and man-made disasters, ranging from epidemics and flash floods to forest fires, earthquakes and complex human-induced crises. These events have caused significant casualties and extensive damage to infrastructure and the environment, putting immense pressure on the response capabilities of individual countries. To address these challenges and enhance collective disaster

response, the European Commission established the rescEU mechanism, a strategic initiative within the broader EU Civil Protection Mechanism (UCPM).

rescEU was created to strengthen the EU's disaster response capabilities by establishing a European reserve of strategic resources. These resources, collectively known as the rescEU reserve, include firefighting planes and helicopters, medical evacuation planes, stockpiles of medical items and field hospitals, as well as mobile shelters for displaced persons. In addition to that, the EU is developing reserves to respond to chemical, biological, radiological and nuclear incidents. These resources are fully financed by the EU, covering all costs associated with their purchase, operation and maintenance.

One of the primary objectives of rescEU is to ensure a rapid and effective response to disasters that overwhelm the capabilities of individual member states. When a country faces an emergency that exceeds its capacity to respond, it can request assistance through the UCPM. The rescEU reserve can then be deployed swiftly to provide the necessary support. This mechanism promotes cross-border assistance, fostering a spirit of solidarity among EU member states and their people and most importantly ensures a more extensive response to disasters that transcends national borders.

The coordination of rescEU is managed by the European Commission via the Emergency Response Coordination Centre (ERCC) whose role is to channel offers of assistance from the 27 EU member states and the ten additional participating states (Iceland, Norway, Serbia, North Macedonia, Montenegro, Türkiye, Bosnia and Herzegovina, Albania, Moldova and Ukraine). It is thus a complete centralized coordination mechanism that runs under the five objectives of anticipate, prepare, alert, respond and secure to ensure the efficient and effective deployment, minimize the response times and at the same time, maximize the impact of the assistance provided.

The rescEU mechanism is designed to address not only traditional disasters but also emerging threats and complex security concerns. Climate change, for instance, is expected to exacerbate the frequency and severity of natural disasters. Similarly, the recent COVID-19 pandemic and the ongoing war in Ukraine have highlighted the need for strong and adaptable disaster response capabilities. As an answer to these challenges, rescEU has expanded its reserves to include medical equipment such as ventilators, personal protective equipment and other critical supplies, hosted in various locations across the EU, so they can be swiftly delivered to areas in need.

The largest EU civil protection operation, to date, since the creation of the UCPM in 2001 has been the deployment of assistance to Ukraine following the Russian aggression. The EU has utilized its rescEU medical stockpiles to provide essential medical amenities like ventilators, infusion pumps, patient monitors, masks, gowns and oxygen concentrators. This assistance, coordinated through the UCPM, supplements the aid offered by European countries, demonstrating the mechanism's capacity to respond to large-scale crises.

Similarly, during the 2022 forest fire season, rescEU financed the stand-by availability of a firefighting fleet comprising planes and helicopters from Croatia, France, Greece, Italy, Spain and Sweden. This fleet was ready to be deployed to support other member states in case of emergencies, illustrating the mechanism's proactive approach to disaster preparedness.

The EU's commitment to continuously developing and expanding rescEU's capabilities is evident in its ongoing efforts to establish reserves for chemical, biological, radiological, and nuclear emergencies through the assembly of expert teams, detection and decontamination equipment and medical countermeasures such as medicines and vaccines. By anticipating and preparing for a wide range of potential threats, rescEU ensures a robust and adaptable response framework [212–214].

The Role of rescEU in Flood Management

Amongst others, the rescEU mechanism significantly contributes to flood management in Europe by complementing and enhancing the efforts of the European Flood Awareness System (EFAS), the first operational pan-European flood forecasting and monitoring system under the Copernicus Emergency Management Service (CEMS). Its establishment marks a significant advancement in Europe's approach to flood risk management, providing early flood forecasting information based on sophisticated models, satellite data and in-situ measurements.

The inception of EFAS can be traced back to 1999 when the Joint Research Centre (JRC) of the European Commission initiated a research study aimed at creating a European scale flood forecasting system that was partly motivated by the devastating floods of 2002 in the Elbe and Danube rivers. These events exposed the inefficiencies and inconsistencies in flood warning systems across Europe and highlighted the need for a coherent and unified approach to flood forecasting. Consequently, the JRC's early efforts evolved into

the European Flood Awareness System, with its first developments commencing in 2003 in collaboration with several European countries and weather services.

EFAS is integrated into the CEMS, with the JRC overseeing its management, technical implementation and continuous evolution. This ensures that EFAS can provide detailed and timely flood forecasts to national and regional authorities responsible for flood risk management.

The core functionality of EFAS lies in its ability to produce probabilistic river flood forecasts with a lead time of up to ten days and flash flood forecasts with a lead time of up to five days, both updated twice daily. It utilizes the LISFLOOD[9] [216] hydrological model, developed by the JRC, which integrates a range of data sources to simulate and predict flood events accurately. Flood notifications are included, which alert authorities to potential flood risks, enabling them to take preemptive actions.

Since becoming fully operational in 2012, EFAS has made a significant contribution to strengthening flood preparedness and response across Europe, particularly by intervening with the creation of transnational river and lake basins that have become a cornerstone of international cooperation.

Furthermore, it remains vigilant and up to date as it incorporates the latest scientific and technological advancements, new data sources, user feedback and international collaboration. In this direction, EFAS users and partners receive regular training sessions, webinars, and extensive user guides so that they are well-trained to utilize the system's capabilities to their fullest.

As EFAS continues to evolve through the power of collaborative, science-driven approaches, it promises to further enhance European communities' resilience to flood-related disasters, safeguarding lives, property and the environment [217, 218].

A comparison table summarizing the achievements and challenges of each methodology in disaster response, particularly focusing on floods, is provided below.

[9] "LISFLOOD is a model that simulates the full water cycle from rainfall to water in rivers, lakes and groundwater" [215].

System/Methodology	*Achievements*	*Challenges*
Bangladesh: Comprehensive Disaster Management Programme (CDMP) ***(2004 – 2014)***	▪ *Developed a proactive, multi-hazard approach to disaster risk reduction* ▪ *Established Disaster Management Information Centre (DMIC)* ▪ *Created Community Risk Assessment (CRA) program* ▪ *Implemented capacity-building initiatives*	▪ *Slow institutionalization beyond the Ministry of Food and Disaster Management* ▪ *High turnover of key officials* ▪ *Delays due to natural disasters and political unrest* ▪ *Limited impact on reducing vulnerability on the ground*
An Expert System for Local Flood Response Coordination and Training ***(presented on 2017)***	▪ *Developed a computer-based expert system for flood response coordination* ▪ *Improved knowledge management and training for flood response entities* ▪ *Provided scalable and adaptable coordination structures*	▪ *Complex management of emergency response with non-linear dynamics and uncertainty* ▪ *High dependence on accurate and comprehensive data* ▪ *Necessity for continuous updates and validation to maintain system effectiveness*
Standard Operating Procedure on Flood Disaster Response (India) ***(2005 – ongoing)***	▪ *Established clear protocols for flood disaster response* ▪ *Enhanced coordination between various agencies* ▪ *Provided detailed guidelines for flood management at different levels of government*	▪ *Implementation challenges due to the complexity of coordination across different agencies* ▪ *Potential delays in response due to bureaucratic procedures* ▪ *Need for regular updates to keep protocols relevant*

System/Methodology	*Achievements*	*Challenges*
Netherlands: Delta Works Engineering Project ***(1958 – ongoing)***	▪ *World-renowned flood defense system* ▪ *Successfully protected the country from severe flooding* ▪ *Integrated engineering solutions with environmental sustainability*	▪ *High costs of construction and maintenance* ▪ *Potential vulnerability to extreme future climate scenarios* ▪ *Continuous need for technological upgrades and monitoring*
FEMA's National Response Framework (U.S.) ***(2008 – ongoing)***	▪ *Comprehensive framework covering all types of disasters* ▪ *Emphasis on preparedness, response, recovery and mitigation* ▪ *Clear roles and responsibilities across federal, state and local levels*	▪ *Complexity in coordination between multiple levels of government* ▪ *Challenges in maintaining up-to-date training and resources for all responders* ▪ *Issues with communication and data sharing during large-scale disasters*
INSARAG Operational Field Guide ***(2006 – ongoing)***	▪ *Internationally recognized guidelines for urban search and rescue* ▪ *Standardized procedures for effective response* ▪ *Enhanced international cooperation and coordination*	▪ *Need for regular updates to incorporate new best practices* ▪ *Challenges in training and maintaining proficiency among responders* ▪ *Dependence on international cooperation, which can be affected by political and logistical issues*
rescEU Mechanism Enhancing European Disaster Response Capabilities ***(2019 – ongoing)***	▪ *Strengthened European disaster response capacity* ▪ *Created a reserve of response resources* ▪ *Improved coordination among EU member states*	▪ *Dependence on member states' commitment and resources* ▪ *Coordination challenges due to diverse legal and administrative systems across the EU*

System/Methodology	*Achievements*	*Challenges*
		▪ *Necessity for continuous funding and political support to maintain and enhance the mechanism*

Table 9: Systems/Methodologies Comparison Table [30, 196, 198, 199, 201–206, 208, 209, 212–214, 219, 220]

2.8 AllERT: A System Description of the Integrated Flood Management Protocol

Establishing standards is clearly necessary. For that purpose, this thesis introduces the *AllERT* protocol as a procedural protocol for implementing a flood management system using IoT technology with a view to real-time response operations. By employing multiple methods at the same time (IoT sensors, UAVs for an overview of the affected areas, satellite communications and ML methods for the most focused options) this system ensures that decision-makers have immediate access to up-to-the-minute information about the flood situation. Unlike conventional systems that rely on delayed or static information, this approach provides a continuous feedback loop.

The complete system aims to be marked by ease of implementation, so that there are limited to zero chances of failing by losing precious time. All described teams that relate to communication, can perform remotely. This means that the core of stakeholders, will be available immediately.

However, research gaps remain in seamlessly integrating these different technologies into a coherent, easy-to-implement protocol. In particular, there is a need for more comprehensive studies on the interoperability of different systems and the optimization of communication networks in disaster conditions. Addressing these gaps is mandatory to develop a robust, efficient and practical protocol that can significantly enhance the effectiveness of emergency response operations. The approach of this research topic seeks to fill as many gaps as possible, leaving fewer and fewer open-ended concerns in terms of interoperability.

AllERT emphasizes the importance of collaboration and communication among all kinds of stakeholders including government agencies, emergency responders and community organizations. It is a multi-agency approach that enables a more coordinated and effective

response to flood disasters and eventually leading to better outcomes and reduced socio-economic impacts.

The system operates through a well-defined workflow that starts with the activation of the flood emergency protocol. Upon activation, IoT sensors and UAVs collect and transmit real-time data to the control center, where it is processed and analyzed. The DSS then, generates insights and recommendations, which are communicated to field response teams via satellite communication.

While the *AllERT* protocol introduces an advanced DSS to enhance flood response, it is essential to recognize the critical roles played by traditional emergency response teams that operate outside this technological framework. Their extensive training, experience and ability to perform under pressure make them indispensable in crisis situations. Emergency management agencies coordinate the overall response efforts, ensuring that resources are allocated efficiently and that different agencies work cohesively. Firefighters and police officers are among the first responders who provide immediate and hands-on assistance during a flood. Firefighters ensure the safety of structures, while police officers ensure to maintain public order, securing affected areas and assisting in evacuation efforts. Medical personnel provide urgent care to the injured and manage medical emergencies.

Civil protection agencies are invaluable in planning, coordinating and executing flood response plans, often serving as the primary link between various governmental and non-governmental organizations. The army provides critical support in logistics, SAR operations and restoring essential services, especially in situations where local resources are depleted. Public health departments monitor and manage health risks, ensuring that water and food supplies remain safe and that potential disease outbreaks are controlled. Public works departments are responsible for restoring critical infrastructure such as roads, bridges and utilities, making sure that affected communities can quickly regain access to essential services. Social services provide support to displaced individuals and families, offering shelter, food and other necessities.

Governmental communication services guarantee that all information is accurately disseminated to the public, keeping citizens informed about safety measures, evacuation routes and assistance availability. Transportation agencies facilitate the movement of resources and personnel to and from affected areas. Environmental agencies assess and mitigate environmental hazards, such as chemical spills or contamination, that can arise

during floods. Finally, local government officials remain important in community outreach, providing leadership and coordination at the municipal level.

Despite not being integrated into the *AllERT* protocol, these traditional responders and governmental services remain an integral part of disaster management. Their efforts are complemented by the technological advancements provided by the *AllERT* system. For instance, while UAVs and IoT sensors can provide real-time data and situational awareness, the actionable insights derived from these technologies require the expertise and physical presence of firefighters, medical teams, police, civil protection, possibly the army and other governmental agencies to be effectively implemented. The integration of *AllERT* with the efforts of these traditional responders ensures a comprehensive and multi-faceted approach to flood management, where advanced technology and human expertise collaborate to save lives and generally ensure the least possible damage.

3. Methodology

The methodology for this thesis involves a combination of research, development and practical implementation. Outlining the technologies and communication strategies to be used, the protocol will be implemented in practical settings, including pilot studies and simulations to evaluate its effectiveness. Data collection, analysis and frequent testing procedures are a requirement.

3.1 Research Design

The research design is a mixed-methods approach, combining both qualitative and quantitative methods to comprehensively evaluate the desired system which will then be tested through a Use Case to produce insights on its performance and effectiveness.

The quantitative component of the design directly addresses the objective of measuring the system's performance, providing data on key metrics such as response times and accuracy of resource allocation. The qualitative component, involves case study analysis that addresses the objective of the system's usability and practical applicability. The case study analysis uses historical data (of recent event) to simulate realistic flood scenario, providing a practical context for observing the operation of the protocol. By incorporating these methods, the research design ensures a thorough review that aligns with the overall goals of the thesis.

3.2 System Description

The system fits within the category of Information System (IS). “An information system is an interconnected set of components used to collect, store, process and transmit data and digital information. At its core, it is a collection of hardware, software, data, people and processes that work together to transform raw data into useful information [221].”

The primary characteristic of this information system is the capability to manage data in real time. Real-time data refers to information that is delivered immediately after collection, without any delay. Its characteristics are the potential of immediate processing which allows for quick reaction to changes and a continuous flow that ensures up-to-date information. The potential it provides to scientists is that it enables quick decision making, critical in dynamic environments where conditions change rapidly [222].

The most essential element within this IoT-based network is the set of sensors and devices used. These are required to be strategically deployed in flood-prone areas as they collect

real-time data on various environmental parameters such as temperature, humidity and flood depth. The system then implements a centralized command and control system, called the “Control Tower”, to facilitate effective real-time coordination and decision-making during the occurrence of a flood. It is considered the central hub for receiving, processing and analyzing all the data collected by the previously mentioned devices and sensors as well as additional data that derive from lists that are part of the preparation phase, such as the human resources catalog.

The main pillar throughout all of the abovementioned, is the fact that everything described in the operation of this project work under and is dictated by a procedural Protocol, which ensures the correct order wherein each process will be implemented. This approach decreases the likelihood of overlooking a critical aspect, therefore minimizing the risk of wasted effort, time and resources.

Basically, five main operational teams are featured and are set up immediately after the Protocol is activated. They consist of experts in each separate field, so that it is confident that the equipment will be handled with care and the analysis tasks will be carried through with professionalism. For the sake of the Protocol establishment, these teams are split into targeted tasks during each step. It goes without saying that all participants are registered as appropriately trained based on strictly defined competencies, for example, unmanned aircraft operators are required to own a UAV pilot certification.

Listed below are all the steps in detail, along with the responsible implementation teams:

Step 1: Alert

Description: Flood Emergency Detected

Responsible Team: Sensor Network Team (located near the affected area)

It is hereby stated that the initial step involves the detection of a flood occurring and all stakeholders are informed through automatic notifications on the devices of their choice.

Once the sensors activate the alarm function, it signals the start of the emergency response protocol. Specifically, the Sensor Network Team is responsible for maintaining and monitoring all installed sensors to ensure they provide accurate, real-time data. When anomalous measurements indicative of a flood are detected, the managers of this department are responsible for confirming that it is not a false detection. Upon their confirmation, the AllERT launch is official.

Step 2: Initiation

Description: Control Center Activation

Responsible Team: Control Tower Crew (located anywhere each remotely)

The next step is the immediate activation of the "Control Tower". The activation includes establishing communication links, installing a virtual center of workstations to connect to the main dashboard that guides the Control Tower's Crew decisions as they have been assigned with overseeing the entire response operation. They are also equipped with satellite phones and backup communication mechanisms TErrestrial Trunked RAdio (TETRA radios) for the entire duration of the coordination procedure as they are also in charge of ensuring that all relevant teams are on standby and ready to respond. It must be made perfectly clear that the Control Tower acts as the central hub for decision-making and coordination.

Shortly following the primary evaluation, the Control Tower defines the areas that should be scanned by the use of the UAVs developed for this scope, in accordance with the first locations detected by the sensor network.

Step 3: Setup

Description: Establish Communication Channels

Responsible Team: Communication Setup Crew (located near the affected area)

Effective communication is of supreme importance in managing an emergency response. The Communication Setup Crew is responsible for establishing the communication channels among all stakeholders. This step involves setting up both wired and wireless communication networks, ensuring interoperability among different agencies and confirming that all teams have access to the necessary communication tools. The goal is to facilitate seamless information flow and coordination.

Step 4: Launch

Description: Deploy UAVs to Scan Area

Responsible Team: UAV Operation Team (located at the edge of the affected area)

In this step, UAVs are deployed to conduct a thorough scan of the affected area. The UAV Operation Team is tasked with operating these vehicles and provide real-time aerial

imagery and data. This will produce information about the extent of the flooding and the affected regions.

The UAVs pilots perform the activity ensuring that all collected data are sent to the Data Analysis Team. The UAVs themselves, provide a fast and reliable method for collecting situational awareness. To this direction, they are equipped with a trapped people identification system and can be used in the case of a flash flood, where people are not capable of finding a single safe place. Extra pilots must be prepared and ready to reinforce the UAV Operation Team, if needed, when the first iteration of the area scan is completed.

Step 5: Processing

Description: Analyze Data in MATLAB

Responsible Team: Data Analysis Team (located anywhere each remotely)

Using real-time aerial data captured by the UAVs, the Data Analysis Team perform complex processes. They use MATLAB and Add-On Toolboxes to perform data analysis such as image processing. The objective is to interpret the raw data so as to extract the affected geo-polygons that will then be used by the GIS Analysis Team in the next step. Especially in the event of a flash flood, they are requested to produce clear information about the precise location and number of people who require immediate help to evacuate their places. The team is responsible for overseeing the entire processing phase, they work closely with other teams, including the GIS Analysis Team and the Control Center Crew, to ensure that the analysis results are integrated into the overall response strategy.

Step 6: Mapping

Description: Path Planning and Route Optimization

Responsible Team: GIS Analysis Team (located anywhere each remotely)

Based on the analyzed data from the previous step, the GIS Analysis Team undertakes the tasks of path planning and route optimization. This involves generating the maps that highlight safe routes for the guiding of the field teams about accessing or for people evacuation accordingly. This is an iterative process that tracks the rapid evolution of the flood event, thus it must be equally prompt.

Step 7: Distribution

Description: DSS Allocates Resources

Responsible Team: DSS Team (located anywhere each remotely)

The actual response commences next. Through the use of ML algorithms and decision-making models, the IoT system calculates the most optimal decisions. This is the Distribution step which is managed by the DSS Team and allocates resources based on the insights from the previous steps. It includes deploying emergency personnel, distributing supplies and ensuring that necessary equipment reaches critical areas for people in need.

Step 8: Dispatch

Description: Communicate with Teams

Responsible Team: Communication Team (located near the affected area)

In the Dispatch step, the Communication Team ensures that all relevant teams are directly informed about their roles and responsibilities. This involves dispatching available teams to their designated areas, providing them with updated information and maintaining continuous communication to address any emerging issues. Effective dispatch ensures that the response is coordinated and that all teams are working in tandem.

Step 9: Response

Description: Execute Response Plan

Responsible Team: Emergency Response Teams (active in the affected area)

The actual execution of the response plan takes place in this step. Emergency Response Teams carry out the planned actions, which may include evacuating residents, providing healthcare assistance, setting up temporary shelters and distribute water and supplies for basic needs. This step requires flexibility, coordination and dedication to the established procedural protocol to make sure that the affected population is safe and sound.

Step 10: Assessment

Description: Continuous Monitoring & Adjustment

Responsible Team: Monitoring and Adjustment Team (composed of members located remotely, near the field and in it, for a complete overview)

The final step involves the continuous monitoring of the situation and repetitious adaptation of the response plan as necessary. The Monitoring and Adjustment Team investigates the effectiveness of the response activities, identifies any gaps or areas for improvement and makes real-time adjustments. This is a much-required iterative process up to Step 4 (Launch), because the response must remain adaptive to changing conditions and the necessity that any unexpected challenges must be swiftly dealt with. Illustrated in Figure 3: The AllERT Procedural Protocol:

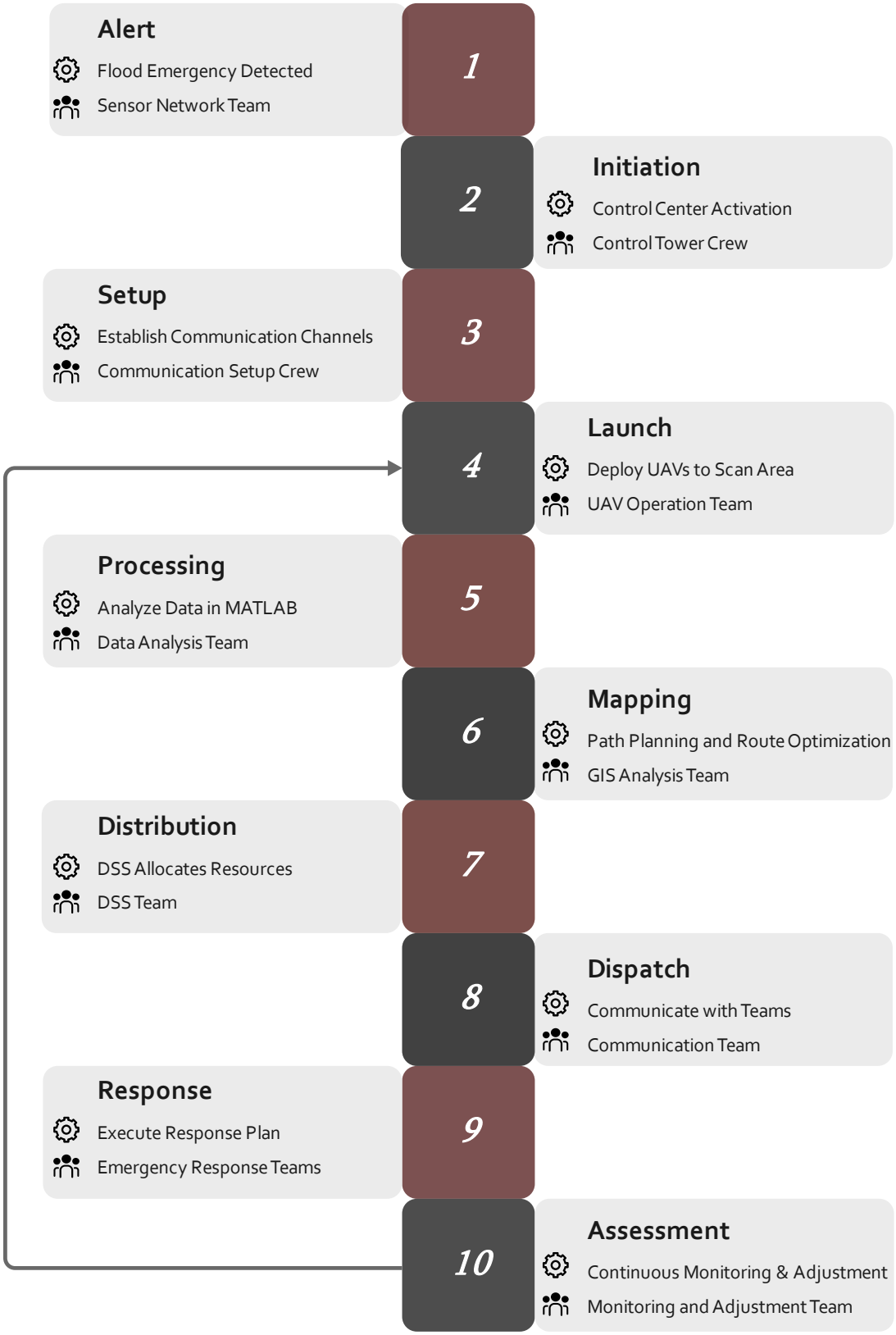


Figure 3: The AllERT Procedural Protocol

3.2.1 Originating from the Prepare Phase

Certain preparatory work must be carried out for a smooth operation of all stages of the response. This involves:

- A morphology study of the terrain of each area
- The installation of the sensors under the spotting of the right locations and the regular checking of sensor functionality. Flood indicators are intended to ensure prompt detection when a flood incidence occurs, so that the activation of *AllERT* can be instant. This process operates continuously. Also, conduction of study for the placement of sufficient LoRaWan network Gateways to ensure that all sensors are within the area of coverage
- An up-to-date database of all available resources should be maintained and checked frequently for any replacement materials based on their expiration date
- Conduction of regular training drills and simulation exercises for all team members, must be scheduled and arranged
- Creation and maintenance of up-to-date contact lists of satellite phones users

Communication Infrastructure Assumption

This study assumes that all cellular networks in the impacted area are inoperable due to major damage caused by the flood or because it was seriously compromised due to network overload. As a result, alternate communication techniques, such as satellite internet and other networks, are being investigated to assure continuous connectivity during response activities.

3.2.2 Software components, platforms and communication protocols

1. **Node-Red** is a low-code development platform for event-driven applications. It is built on Node.js (open-source JavaScript runtime environment) taking full advantage of its event-driven model [223]. This means that systems can remain asynchronous while still sharing information and accomplishing tasks [224]. For this research it is chosen because of its friendly environment and the interconnection it offers between different networks (physical or logical) collaborating seamlessly [225]. It offers browser-based visualization of an application's functionality, something that doesn't require preparation and installation of complex editors and their dependencies.

Node-red is also chosen for it allows data visualization without the need for additional integration with an external dashboard. Although it can run at the edge of the network on low-cost hardware such as the Raspberry Pi or locally on a computer, here it runs in the cloud for it provides autonomy. For this purpose an instance was created on FlowFuse platform [226]. Besides, Node-RED Dashboard 2.0 was formed by **FlowFuse**, as part of efforts to upgrade the original Dashboard which is based on Angular v1, no longer maintained [227].

2. For **MATLAB**, a discussion has already been carried out in the literature review section, comparing it with similar processing and analysis tools. Since it was finally chosen for processing the images received by UAVs in the present system, an introduction and a presentation of the method is given.

   MATLAB provides all the necessary tools for all the UAV aerial footage processing. Short for "MATrix LABoratory", it is a high-performance programming language and numeric computing environment developed by MathWorks. It is designed primarily for engineers and scientists for data analysis, algorithm development and the creation of models and applications. It integrates computation, visualization and programming in a user-friendly environment, allowing users to express problems and solutions in familiar mathematical symbolisms [228, 229].

   For analyzing aerial images, several steps are needed particularly through the Image Processing Toolbox [230] and Mapping Toolbox [231]. The preprocessing step is preceded and this includes filtering, normalization and georeferencing the images to ensure they align correctly with geographical coordinates [232].

   To export the flooded geo-polygons, images before the flood and during it are prerequisites. After the preprocessing procedure the Region Of Interest (ROI) Selection is next. It is about the identification of specific areas within the images that are of interest for flood analysis, meaning that the flooded area is only contained. By analyzing the difference between pre-flood and during-flood images, the identification of changes indicative of flooding is possible. This is done by subtracting the pre-flood image from the post-flood image to create a flood area mask. Then everything except the water is removed from the image by using a threshold. Mapping Toolbox can project the identified flooded areas onto a map, providing spatial context to the analysis. This involves transforming image coordinates into geographic coordinates to represent the flood extent on a real-world

map with accuracy [232]. However, the processed data are exported in shapefile[10] format for GIS analysis and route optimization.

3. **ArcGIS Pro** is a super powerful GIS desktop application, developed by Esri that enables users to visualize, analyze and interpret spatial data. It offers several tools and workflows customized for specific applications and it can integrate data from multiple sources in various formats [234]. It includes a "Find Routes" tool that can be used to identify the best routes for emergency response teams to reach flooded areas. This tool examines factors such as Travel Modes through which users can define the mode of transportation (e.g., vehicle, foot) and the parameters for distance or time to find the most efficient routes and Barrier Modeling that can understand obstacles, such as flooded areas, by modeling how these obstacles affect travel routes [235].

   The results from the above process are exposed as a Representational State Transfer (REST) service in AllERT's Dashboard by publishing the project to ArcGIS Online [236].

4. **InfluxDB** is an open-source Time Series DataBase (TSDB) developed by InfluxData, designed specifically for the efficient storage and retrieval of time series data. It is commonly used in operations as monitoring, application metrics and IoT sensor data. The latest version, InfluxDB 3.0, enhances capabilities for real-time analytics and cost-optimized storage. A time series is a time-stamped set of values where time is an important element of the data. Data are organized into measurements, series and points. Each point consists of a timestamp and several key-value pairs, categorized into field sets and tag sets. This structure facilitates efficient querying and data manipulation [237–239]. InfluxDB uses line protocol to write data points format, in the sense of a single line of text is used to represent one row of data [240, 241].

   For querying and analyzing data within InfluxDB, flux language is used. It is flexible and it enables comprehensive data analysis, which extends what Structured Query Language (SQL) or traditional query languages offer for time series data. It also supports querying, joining and transforming time series, geo location and data from SQL data sources [242].

[10] "A shapefile is a simple, nontopological format for storing the geometric location and attribute information of geographic features. Geographic features in a shapefile can be represented by points, lines, or polygons (areas). The workspace containing shapefiles may also contain dBASE tables, which can store additional attributes that can be joined to a shapefile's features" [233].

Although it can run on-premises or at the edge, here the cloud service was selected. A bucket (logical container) was created to store the measurements of the sensors. Now large volumes of time series data can be handled in a way that high-performance writes and queries is ensured.

5. **MQTT**, is a lightweight publish/subscribe messaging protocol designed primarily for machine-to-machine (M2M) communication and IoT applications. It allows for real-time data transmission between devices, particularly in environments with limited bandwidth or unreliable network connections [243–245].

   MQTT is used in this system for transporting data from the LoRaWAN infrastructure to the bucket in InfluxDB Cloud. HiveMQ Cloud is utilized as the MQTT broker operating in a clustered environment. It is a fully-managed, cloud-native IoT messaging platform that ensures reliable and scalable message distribution, providing high availability and fault tolerance during the transmission process. By this clustered setup it is ensured that even in the event of high data loads or partial node failures, the transmission remains stable, delivering sensor data without interruption [246].

6. **Apache Kafka** is a distributed event streaming platform designed for real-time data ingestion and processing. It is excellent at managing large volumes of data generated by multiple sources, making it suitable for applications that require real-time analytics and data pipelines. Kafka provides three primary functions: publishing and subscribing to streams of records, storing streams of records in the order they were generated and processing these streams in real time [247, 248]. Streaming data through platforms like this, makes it the best practice to bypassing the latency introduced by accessing a cloud storage service. This is particularly important for applications that require immediate action based on data inputs, such as IoT applications [249–251].

   Apache Kafka is selected for the AllERT system for all above reasons, as it is also capable of integrating with MATLAB [252]. This set up allows streaming UAV-captured Synthetic Aperture Radar (SAR) images in real time, enabling direct analysis and processing.

3.2.3 Hardware components

The devices constituting the physical substance of this under-study system contain the following:

1. **Customized UAV**:

   The UAVs are instrumental for AllERT protocol. They must meet all requirements to adjust to it. Generally, they are classified based on various parameters such as their size, the design, the nature of usage and their capabilities. According to previous research and testing, the most suitable ones are the Hybrid Vertical Take-Off and Landing (VTOL) Fixed Wing UAVs. They can operate in confined spaces without the need for a runway and once airborne, they can switch to fixed-wing mode, which is more aerodynamically efficient for long-distance flights. This way it can cover greater distances at higher speeds and with less energy consumption even with heavier payloads, compared to traditional multirotor UAVs [253, 254].

   While off-the-shelf UAVs might be close to the system's requirements, a custom solution from specialized manufacturers [255–257] is a preferable approach to meet the exact payload and weather condition specifications.

   To operate under adverse weather conditions and challenging terrains, the UAVs manufacturers use high-quality, weather-resistant materials like polycarbonate, polyurethane and aluminum alloys for the airframe and other components. They make sure of proper sealing of all joints, openings and access panels to prevent water and dust ingress. Then thorough testing during the design phase is conducted to identify and address potential weak points. This involves ingress protection IP testing as per International Electrotechnical Commission (IEC) 60529 standards for protection against solid objects and liquids and water immersion tests for IP67 and IP68 ratings to verify protection against submersion in water. This way, simulations of heavy rain or cleaning conditions are conducted. As of payload and electronics integration, waterproof and dust-tight connectors are used for all electronic components. Proper sealing and protection of the payload bay and the external sensors or radars are of priority [258–260].

   A UAV like this, is considered already equipped with the basic payloads, such as network interface, spotlights to provide illumination for night-time operations, traditional thermal cameras for heat detection and Real-Time Kinematic (RTK) GPS for accurate positioning data.

Imaging and mission payloads

- **SAR payload**

  The ELM-2054 by Israel Aerospace Industries [261] is an advanced airborne surveillance radar designed for all-weather, day and night operations. It uses Synthetic Aperture Radar (SAR) technology to provide high-resolution images, but what is most exciting is it can penetrate through clouds, fog and rain, making it ideal for a variety of surveillance missions like land monitoring, regardless of light conditions.

  SAR is an active sensor that emits microwave signals and receives the signals returned from the Earth's surface. Active sensors generally emit pulses of energy and detect changes in the returning signal. Most of them operate in the microwave part of the electromagnetic spectrum, which makes them capable of penetrating the atmosphere under most conditions. SAR data requires a different way of thinking from conventional photography, as the signal responds to surface characteristics such as structure and moisture [262–264].

  This module is used to acquire the images that are employed to identify the flooded places enabling the analysis team to determine the most efficient and secure paths for direct access. Figure 4, illustrates the integration on a multirotor UAV.

Figure 4: IAI ELTA's ELM-2054 Synthetic Aperture Radar UAV mounted (Source: [261])

- **FINDER MK4 Generation IV Payload**

  Analyzed already in the literature review section, this device is used in areas where severe damage occurred due to flash floods. In situations like this, individuals are unable to evacuate the impacted region due to insufficient time. Finder MK4 has a weight of 3.6 kg and is fitted within a modified waterproof case that is designed to be attached to the UAV. Particularized UAVs are used independently of the basic ones that perform the function of capturing images [82]. Figures 5 and 6 depict the Finder's software view and an example of a UAV potentially deployed.

Figure 5: FINDER MK4 Generation IV (Software and Human Interaction – Source: [265])

Figure 6: An example of a suggested UAV (Source: [266])

2. **LoRaWAN Network** protocol is used for the collection of the measurements of all sensors, as it is intended for wireless battery-operated Things in a regional, national or global network. In the data link network layer, LoRa (Long Range) is of very low

power consumption (battery life ~10 years for nodes) in unlicensed spectrum with data rate (0.3kbps-50kbps) and range of up to 4.8km (urban) to 16km (rural) [267].

- **LoRa water level industrial sensors:**

  As already mentioned in the Literature Review section, there is a very important collaborative project between New York University (NYU), City University of New York (CUNY), the NYC Mayor's Office of Climate & Environmental Justice and the NYC Office of Technology & Innovation, called the FloodNet network. Its objective is monitoring street-level flooding in New York City in real-time [268]. The sensors developed by the FloodSense project in this same network, are hereby adopted for they are already serving the AllERT's purpose.

  These devices use an ultrasonic-based sensor which was chosen among multiple options due to their low cost and feasibility in deployment and they provide flood depth data (water level). The distance to the surface below the sensor is determined by reflecting ultrasonic wave pulses off of the surface (e.g., sidewalk, road, water, etc.) and calculating the distance based on the time it takes for the pulse to return to the sensor. The distance between the sensor and the surface that the pulses are reflecting off of decreases as flood waters rise and the readings are collected by the sensors every minute in units of millimeters (mm) [269]. A low-power microcontroller is integrated and it has an embedded LoRaWAN radio. It then consists of a 2200 mAh battery, a solar controller, a 1.2W solar panel and a MaxBotix industrial grade range sensor [270].

  The sensor is mounted on top of a pipe resting on the ground, which is used for the installation of an additional humidity and temperature sensor.

  In Figure 7: FLoodNet's NYU/CUNY sensor example is depicted.

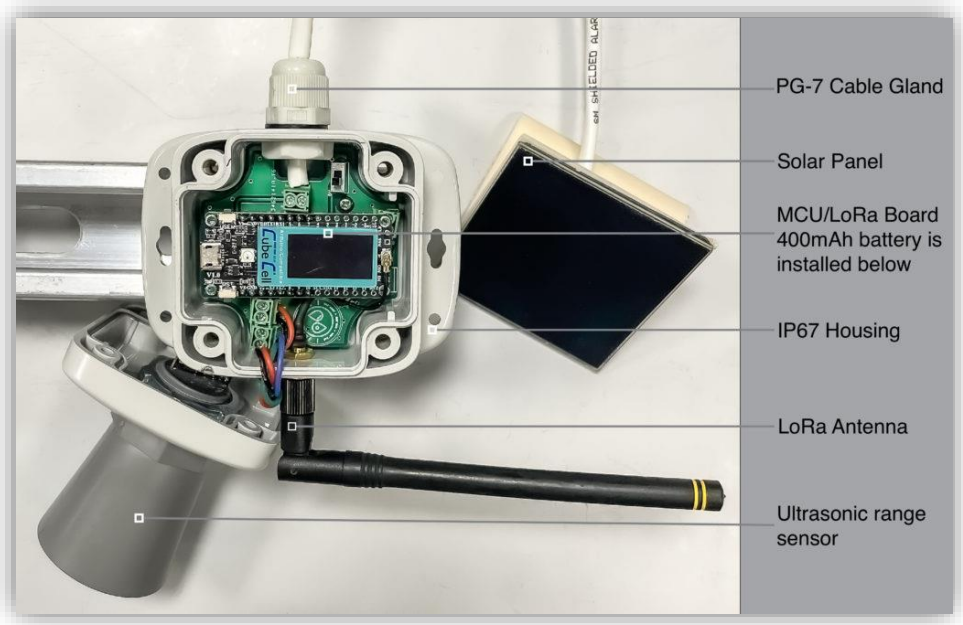


Figure 7: FLoodNet's NYU/CUNY sensor example (Source: [268])

The metadata it provides are its name and the location. Summing up, the sensor implementation provides name, location, flood depth, humidity and temperature.

- **LoRa Gateways** using LoRa RF modulation, serve as concentrators to receive messages from the sensors and forward them through MQTT protocol to the Network Server which runs as a Cloud service. They are mounted on rooftops of very tall buildings or metal pipes using water/dust proof enclosures.

3. Satellite phones - TETRA radios & Base Stations

Satellite phones are used to provide communication channels between emergency teams and authorities, guaranteeing continuous communication even in the most harsh circumstances. Aforementioned in the literature review, their ability to function independently of terrestrial infrastructure can give certain communication links that are able to remain open when traditional networks fail. It is an AllERT protocol requirement that all trained personnel are provided with one, when at the same time traditional responders are considered to already use satellite phones as before.

Suggested devices are the Iridium, Inmarsat or Thuraya all of which are engineered to withstand rugged environments suitable for emergencies [271].

For even satellite telephony sometimes fails due to adverse weather conditions, satellite malfunction or other technical issues, a seamless switching to terrestrial

communication is carried out. This is made possible by the use of TETRA technology.

TETRA is an Analogue Private Mobile Radio (PRM) system developed by European Telecommunications Standards Institute (ETSI) to respond to the requirements of commercial services and emergency services, and to give the possibility for cross-border networks in Europe. It is often used by public safety and other critical organizations [272]. Thus, it can serve as a backup communication method. TETRA devices support full-duplex mode, which means that different frequencies for transmitting and receiving simultaneously are used, allowing both actions at the same time. It can serve one-to-one, one-to-many and many-to-many connection.

Besides TETRA radio devices that already support Direct Mode Operation (walkie-talkie), some TETRA base stations are used for the network establishment. For example, iBS Integrated TETRA Base Station by Hytera is suitable for outdoor and indoor use [273]. It is easy to transport and mount on antenna masts. It is valuable for the quick deployment that reduces networking costs.

Due to the inexistence of dual-mode devices equipped with both satellite and TETRA modules, AllERT uses a dual-mode approach to avoid disrupting ongoing calls. This can be achieved if the communication is monitored by sophisticated software capable of detecting connection issues in real-time. Upon identifying a potential failure in the satellite connection, the system initiates a seamless handover to a pre-established TETRA channel. It is essential not to miss voice data which can be achieved by caching it and managing the session state [238]. The buffered communication data is synchronized with the TETRA channel, ensuring continuous communication without noticeable disruption. It is also mandatory that the switch is transparent to the users, minimizing any disturbance [239].

For that purpose, automatic detection algorithms and session management techniques are used to provide a trusted communication solution. The system continuously monitors the satellite connection in order to detect when the signal strength is weakening and latency is increasing. It then identifies that the satellite connection is likely to fail soon based on predefined thresholds. The satellite module is disconnected and the TETRA module takes over the communication seamlessly. Users receive a brief notification indicating that the communication channel has switched to TETRA.

TETRA networks can be designed to cover large geographic areas with high reliability. In disaster scenarios, where infrastructure might be damaged, TETRA can still function if the base stations are operational. It is already secure with encryption to prevent unauthorized access or interferences. They can be scalable with portable base stations to cover all areas as needed during a flood [240].

4. **Satellite internet**

To support internet connectivity on the LoraWan gateway for the sensors to bridge their signals, and the UAVs for the high-definition images transmission, assuming that parts of traditional communication networks of the local infrastructure are disrupted or completely destroyed (assumption in 3.2.1), the Communication Vans (satellite internet-equipped vans) near the impacted areas are being integrated as part of the system.

SpaceX's Starlink satellite constellation is chosen for it provides services tailored for high-speed, low-latency internet access with global coverage thanks to its extensive network of LEO satellites. Particularly the Land Mobility service plan, is designed for harsh environments and can withstand extreme weather conditions. It offers high-download speeds ranging from 100 Mbps to 350 Mbps and upload speeds between 10 Mbps to 40 Mbps, depending on the location and network conditions. This speed is sufficient for applications that require real-time data transfer. The dish is designed to withstand harsh weather conditions and it automatically switches between satellites to maintain the best possible connection. It is secure since it operates with end-to-end encryption [274].

Although these events are rare, users may find minor slowing down or brief disruptions during storms when the signal strength reduces as it passes through moisture-laden clouds. Still, even with influence, the result is a minor slowdown of internet speed rather than a complete disturbance that would interrupt use [275].

To prevent interference, it's important to ensure that these systems are using non-overlapping frequencies. It is demanding to coordinate with national and local communication authorities to assign distinct frequency bands for each communication system. This is important for TETRA, which often operates on frequencies around 380-430 MHz, while satellite systems might use bands such as L-band (1-2 GHz) or Ku-band (12-18 GHz) [276].

In order to acquire a deeper understanding of the system's operation, **Figure 8: AllERT framework's architecture diagram** captures the design of the entire system including both hardware and software components (as well as data communication protocols), **Figure 9: AllERT framework's Data Flow Diagram** maps out the flow of information for all processes of the system and **Figure 10: AllERT framework's flowchart** depicts its workflow.

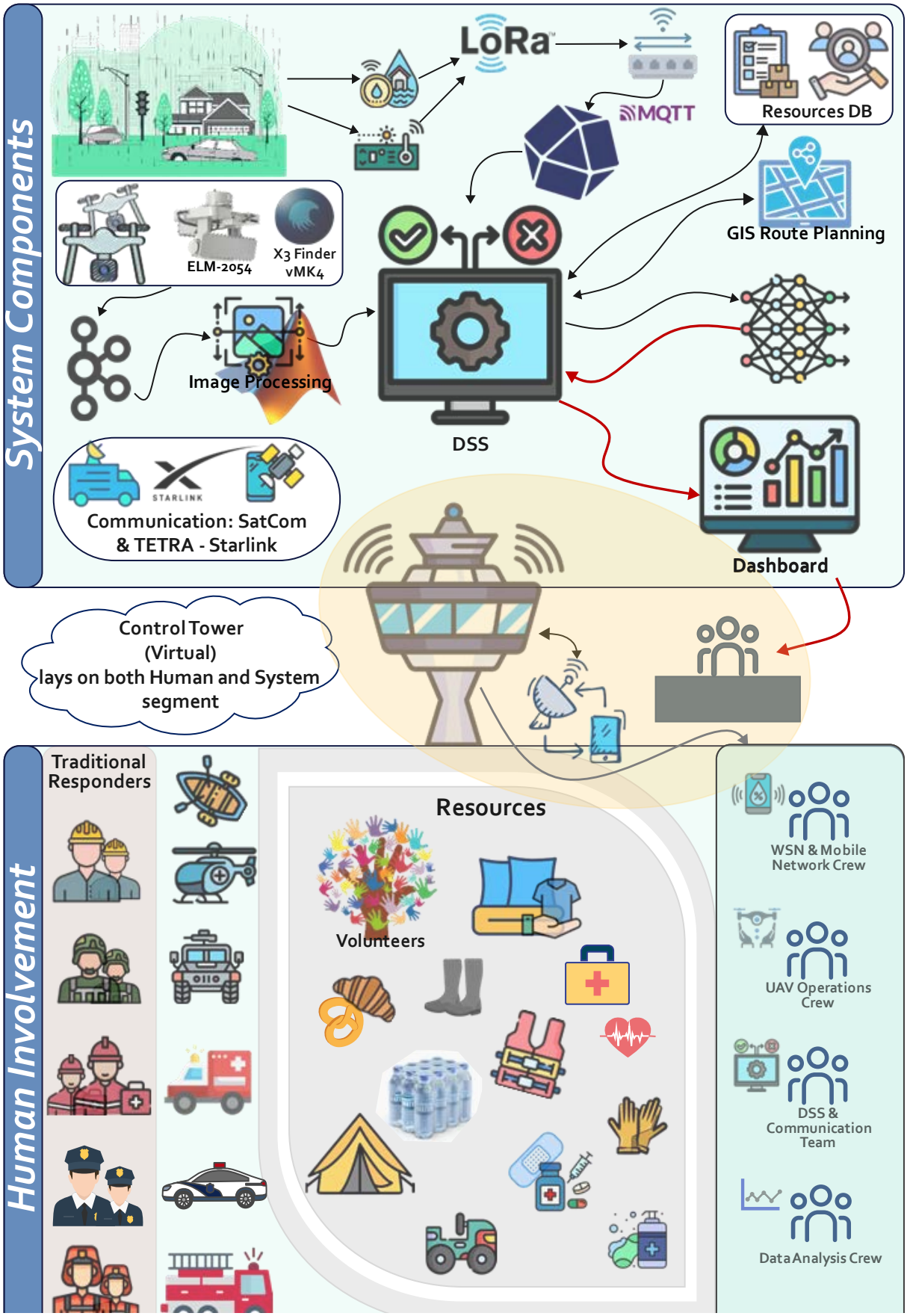


Figure 8: AllERT framework's architecture diagram

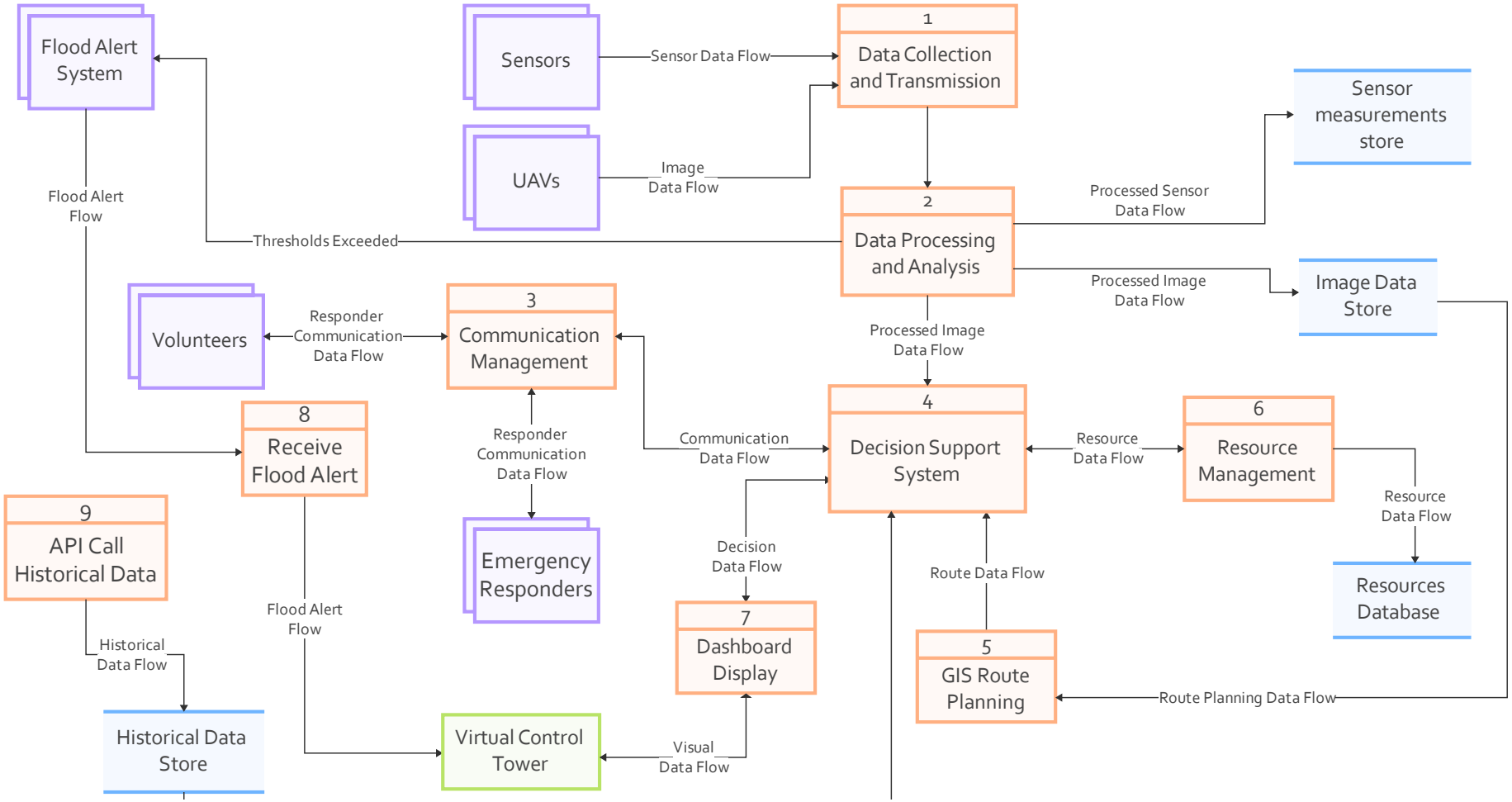


Figure 9: AllERT framework's Data Flow Diagram

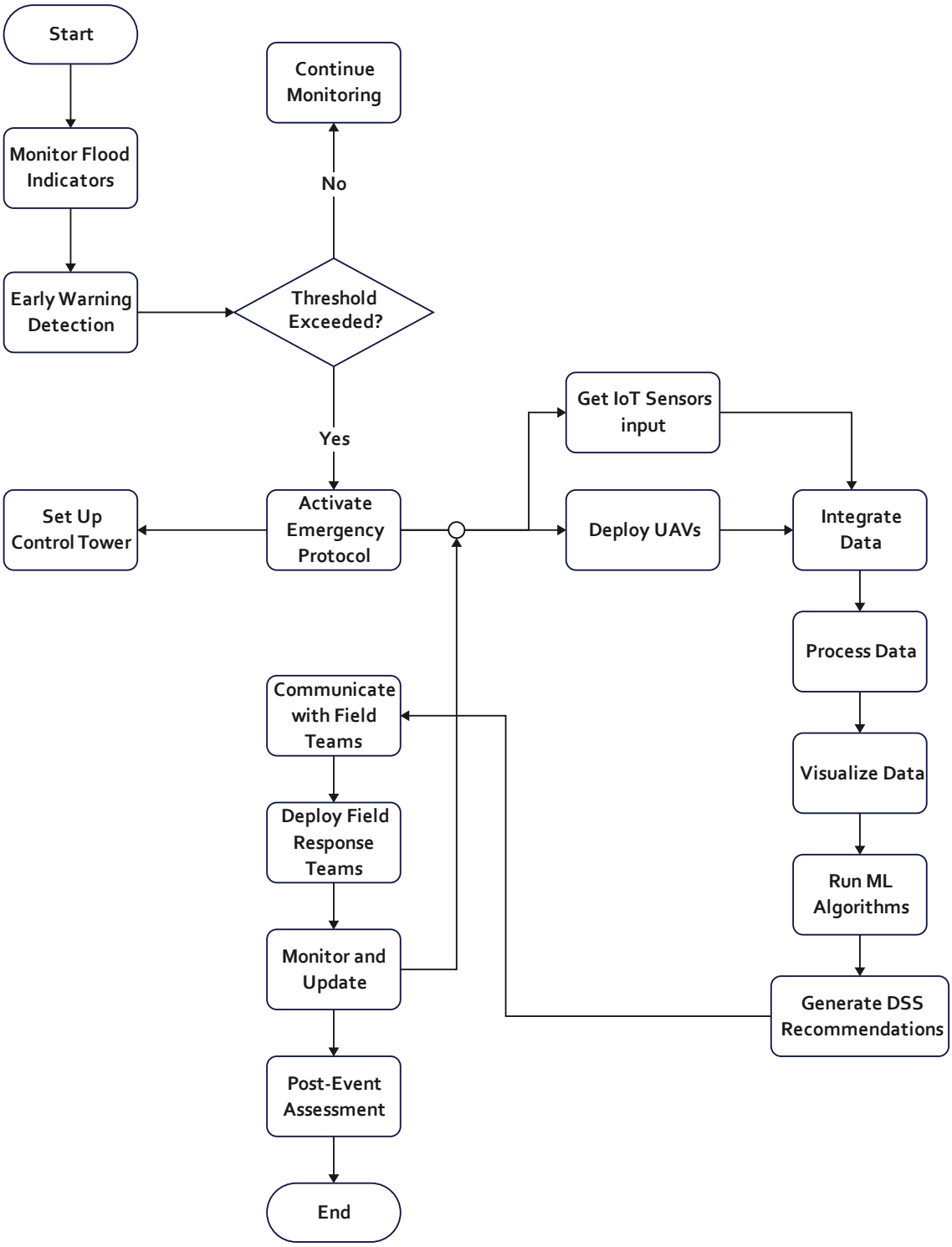


Figure 10: AllERT framework's flowchart

3.3 Limitations of the Study

In assessing the limitations of this study, it's helpful to take a step back and look at the bigger picture of the development and implementation of a DSS designed for flood response. While the integration of IoT technologies, UAVs and real-time data analytics sound like game-changers, several limitations must be acknowledged to provide a balanced perspective on the system's capabilities.

First off, the system's effectiveness is directly reliant to the availability and accuracy of the sensors and UAVs data it relies on. Any gaps in data coverage, sensor malfunctions, or UAV operational challenges could lead to incomplete or delayed information, affecting the quality of the decisions made by the DSS. This dependency on technology underscores a system that needs to be as “bulletproof” as possible to ensure continuity of operations during critical events.

Next, the scope of the study is inherently limited by the specific geographic and environmental contexts in which the system has been tested. While the findings may be relevant to similar urban areas prone to flooding, the generalizability of the results to other types of natural disasters or in totally different regions may be constrained. From this limitation arises the need for further research and adaptation of AllERT to a broader range of disaster scenarios and environments.

Third, while the DSS is designed to support real-time decision-making, its performance is also influenced by the quality and timeliness of the data inputs. The system's algorithms, while powerful, are only as effective as the data they process. Therefore, any delays in data transmission or inaccuracies in sensor readings can reduce the system's ability to provide optimal recommendations in urgent situations.

Lastly, this study does not fully address the long-term implications of deploying such a system on a large scale, particularly regarding the social, economic and ethical considerations. The adoption of advanced technologies in flood disaster management raises questions about the equitable distribution of resources, the potential for over-reliance on automated systems and the broader societal impact of integrating such tools into emergency response frameworks.

Apparently, while the study provides valuable insights into the potential of a DSS for flood response, the above limitations highlight the importance of ongoing research, refinement and broader contextual analysis to fully unlock the system's potential and ensure it performs effectively in different scenarios.

3.4 Ethical Considerations

When developing and deploying a DSS which is the main gear for flood incidents response, ethical considerations are essential to make sure that the system is both effective and aligned with societal values. It is clarified that these parameters are not merely additional but rather integral so that the system operates in a manner that respects human rights, promotes fairness and prioritizes safety and welfare.

1. **Privacy and Data Security**

A key concern in this DSS development is the handling of personal data even if it is not directly linked to individual identities. Privacy and data security must remain at the forefront. All data collected and its use, should be restricted to the specific purposes for which it was collected from the beginning.

Equally important is the implementation of rigorous security measures to protect sensitive information, such as the location of individuals in need or critical infrastructure details. Any breach could have dire consequences.

2. **Fairness in Decision-Making**

Algorithmic bias (a systematic error in predictive computation) forms another ethical challenge. If not carefully managed, the machine learning algorithms used in the DSS, could inadvertently prioritize certain areas or populations, leading to inequitable outcomes. To counteract this, the algorithms must be designed with fairness in mind, so as to ensure that resources are distributed equitably to all affected populations, including marginalized and hard-to-reach communities.

3. **Responsibility Taking**

Establishing clear lines of responsibility is vital when deploying a DSS. Human operators should be well-versed in the system's functionality and capable of overriding its recommendations when necessary. Human oversight ensures that the system's decisions are grounded in practical, real-world considerations. The system must also be transparent in its decision-making processes to foster trust among users and stakeholders.

4. **Human Welfare and Safety**

AllERT's primary goal is to enhance human welfare and safety during flood emergencies. However, avoiding over-reliance on the system is equally crucial. Although the

technology is sophisticated, it is not infallible. There could be a risk that users could place undue trust in the system and this would lead to potential blind spots or errors, especially in high-pressure situations where every moment counts. The system should also be designed to ensure that vulnerable populations, such as the elderly, disabled, or economically disadvantaged, are given special consideration in the response phase.

5. Long-term Implications

The ethical use of data extends beyond the immediate response to an emergency. Considerations must include the duration of the data storing and how it will be utilized in the post-disaster period. Establishing guidelines for the long-term retention and use of data is important to ensure that the system remains beneficial without inadvertently creating dependencies or altering local governance practices.

Ethical considerations in the development and deployment of a DSS for flood response are critical to its success and legitimacy. Addressing all such considerations guarantees that the system operates not only efficiently but also in a way that it is fair, just and respectful of all individuals and the communities involved. If embedding these ethical principles into the fabric of the DSS is ensured, developers and operators can contribute to a disaster response system that prioritizes human dignity and welfare while achieving its operational goals [277].

4. Implementation Use Case*: Application to the flood disaster caused by storm Daniel in Thessaly*

4.1 Introduction

In this use case, the *AllERT* procedural protocol will be carried out based on the defined steps and the actions for each of them will be presented separately. The implementation of the proposed system is tested in the context of a real-world historical event (Storm Daniel), which severely impacted the Thessaly region [27]. It caused widespread flooding and infrastructural damage, making it an ideal scenario for evaluating the performance of the designed framework. This way it does not only serve as a practical test of the system's capabilities but also highlights its relevance in addressing similar natural disasters, showcasing the system's ability to operate effectively in high-pressure, real-time environments.

4.1.1 Step 1 "Alert"

Team: Sensor Network Team

Implementing AllERT, ten sensors are simulated as described in the previous chapter installed in areas such as Larissa and the village of Falani at geographical points that could potentially become the initiator of the flood event, e.g. near the river Pinios to detect rising water levels.

The system was designed for sensors which in theory, would collect and transmit their measurements using LoRaWAN and send them to a LoRaWAN gateway. The gateway would then forward this data using the MQTT protocol. However, it was not possible to fully simulate the LoRaWAN transmission in this study due to limitations in hardware availability. LoRaWAN operates on specific frequency bands and transmission protocols, which may not be easily replicable in a simulated environment without access to actual network components. Despite this, the use of MQTT for data transmission remains viable and is successfully implemented in the system. With proper LoRaWAN infrastructure, the process of collecting and transmitting data could be fully operationalized by those with access to the required hardware. The system's integration with the MQTT protocol via HiveMQ Cloud, demonstrates how data could be processed and transmitted once a full LoRaWAN network is established.

Figures 11, 12 and 13 depict the setup.

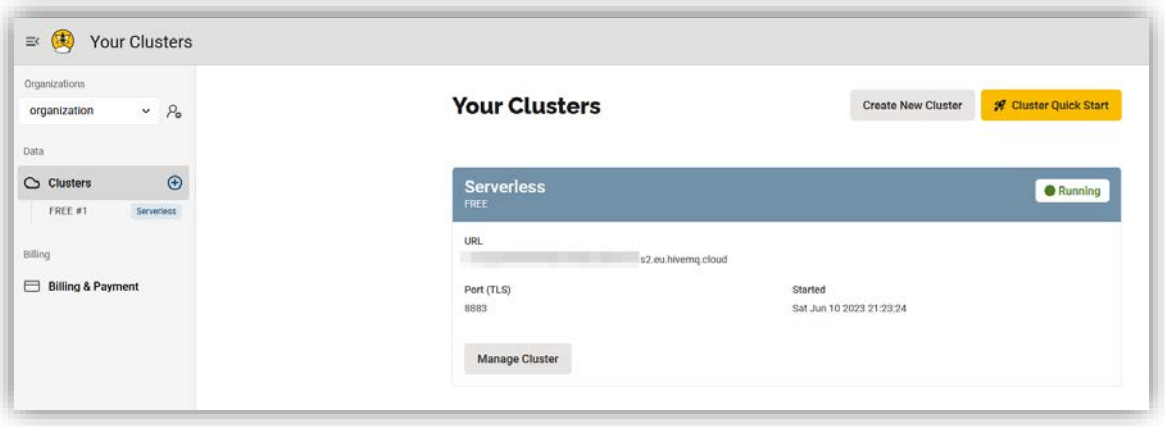


Figure 11: The cluster creation in HiveMQ Cloud platform

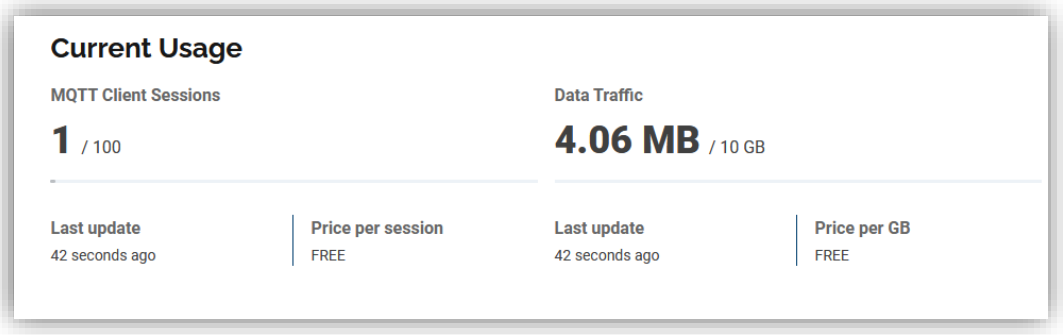


Figure 12: The MQTT Client Sessions

QoS 0 in is selected since low latency is a priority here. It serves faster and more efficiently for real-time purposes. This way it uses less bandwidth and each message is delivered at most once, without acknowledgment from the broker or the receiving client [278]. Because data from sensor readings is updated frequently (per 60s), losing a few messages doesn't have a major impact.

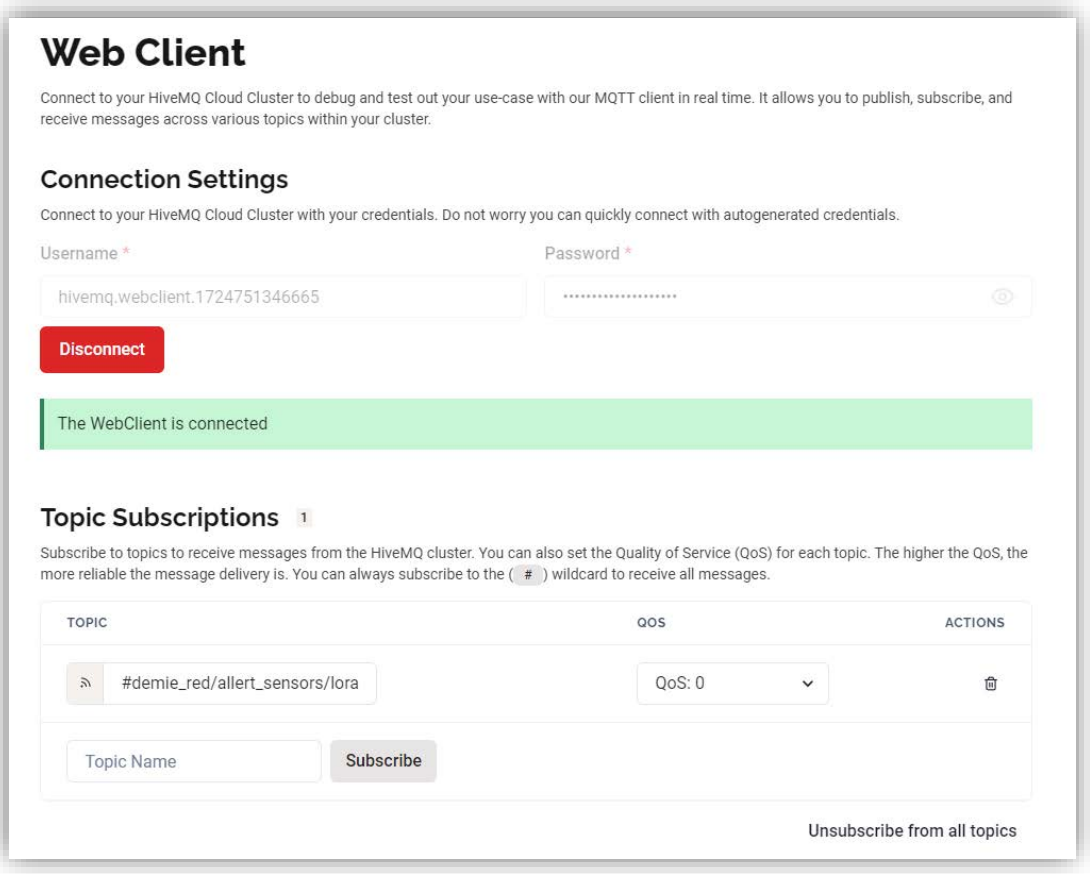


Figure 13: The topic subscription to be used in Node-RED

The Sensor Network Team, review the sensor data and confirm the impending crisis. They undertake the continuous monitoring of the measurements' transmission throughout the entire operation and confirm their network connectivity. This can be achieved through the InfluxDB cloud platform which is part of the IoT system deployed but signing in to the main dashboard is a mandatory requirement.

Figures 14 and 15 are the related snapshots (testing mode using mock data):

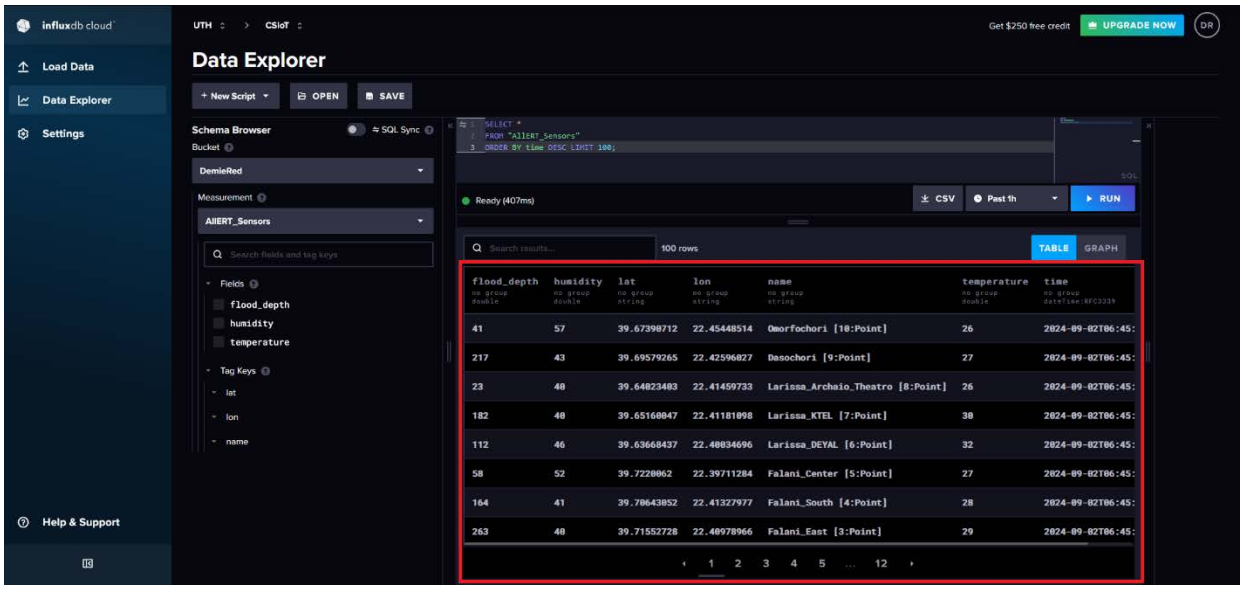


Figure 14: InfluxDB Bucket View

flood_depth no group double	humidity no group double	lat no group string	lon no group string	name no group string	temperature no group double	time no group dateTime:RFC3339
41	57	39.67398712	22.45448514	Omorfochori [10:Point]	26	2024-09-02T06:45:58.298Z
217	43	39.69579265	22.42596027	Dasochori [9:Point]	27	2024-09-02T06:45:58.295Z
23	40	39.64023403	22.41459733	Larissa_Archaio_Theatro [8:Point]	26	2024-09-02T06:45:58.292Z
182	40	39.65160047	22.41181098	Larissa_KTEL [7:Point]	30	2024-09-02T06:45:58.199Z
112	46	39.63668437	22.40034696	Larissa_DEYAL [6:Point]	32	2024-09-02T06:45:58.196Z
58	52	39.7220062	22.39711284	Falani_Center [5:Point]	27	2024-09-02T06:45:58.194Z
164	41	39.70643052	22.41327977	Falani_South [4:Point]	28	2024-09-02T06:45:58.091Z
263	40	39.71552728	22.40978966	Falani_East [3:Point]	29	2024-09-02T06:45:57.994Z

‹ 1 2 3 4 5 ... 12 ›

Figure 15: Sensors' received measurements

Upon detection of rising water levels that exceed pre-defined thresholds, automated push notifications mark the official start of the emergency. Figures 16, 17 and 18 illustrate the implementation of this process.

Figure 16: Push Notification in lock-screen by Pushsafer App

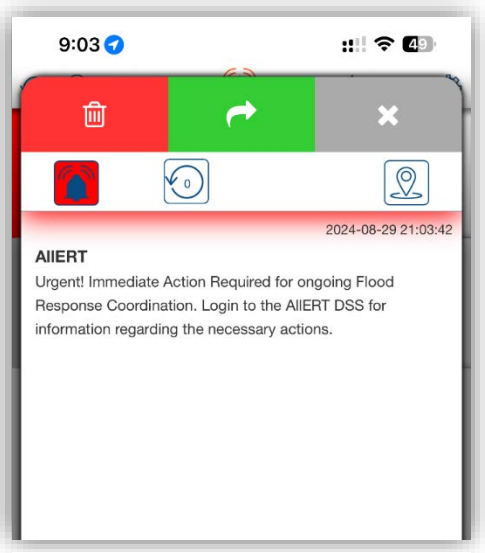


Figure 17: In-App Notification

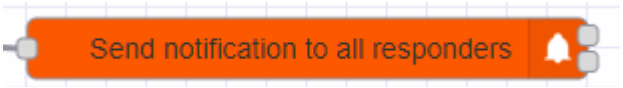


Figure 18: Corresponding node from Node-RED flow

Alerts are typically sent through multiple communication channels (email, SMS, dedicated apps) to ensure that they are received promptly.

4.1.2 Step 2 "Initiation"

Team: Control Tower Crew

Each participant undertakes the role for which they have received particular training, while the Control Tower Crew logs on to the application having all equipment at their disposal. They then begin the process of mobilizing all necessary teams, from the communication setup crew to the UAV operation team. This ensures that all teams are prepared and ready to execute their parts of the protocol. They maintain constant communication with field teams to adjust the response strategy as needed. The crew is

tasked with the responsibility of making real-time decisions, dispatching orders and ensuring that all operations are synchronized.

4.1.3 Step 3 "Setup"

Team: Communication Setup Crew

The setup process involves deploying satellite-equipped vehicles (such as Communication Vans) near the flood-affected area. The Communication Setup Crew begins by positioning these satellite vans in optimal locations, where they can best serve the needs of the field teams. The satellite dishes are aligned and calibrated to ensure a strong and stable connection. The internet connection provided by these satellites is used for sending real-time data, receiving updates from UAVs and communicating with the Control Tower. Once the satellite communication is established, it is tested for stability and bandwidth to ensure that it can handle the data and communication loads required during the operation.

The Communication Setup Crew then ensures that all TETRA devices are operational and correctly configured to switch between different communication channels as needed. They also verify that all field units and response teams are connected through the TETRA network and that the communication lines are clear. As described in the previous sub-chapter, these are dual-mode devices that can switch between satellite and TETRA networks based on availability and signal strength. The setup includes testing of the network for range and connectivity. Test calls and data transmissions are conducted to ensure that all units are within range and that communication is reliable.

4.1.4 Step 4 "Launch"

Team: UAV Operation Team

This is where the UAVs are launched to capture real-time imagery of the affected area. Depending on the severity of the flooding which may be characterized as minor, moderate, severe or catastrophic, two different types of UAVs are deployed as previously detailed. For the two first categories, the type of UAV used is the one deployed with the SAR radar payload, while for the next two severe conditions the UAVs with the Finder radar are added. The UAV Operation Team, responsible for checking the battery levels, calibrating the sensors and ensuring that communication links are stable schedule flight

paths designed to cover the most critical regions first, such as areas where the flood is most severe. They keep monitoring the weather to be prepared to modify flight plans if conditions become hazardous or if the data collection is compromised by high winds or poor visibility. The images are then streamed to the image analysts through Apache Kafka real-time streaming.

Here, the SAR images are downloaded from Copernicus Data Space Ecosystem [279]. As Figures 19 and 20 show, to detect images before and after the flood containing the area of interest, the search criteria and time range were specified.

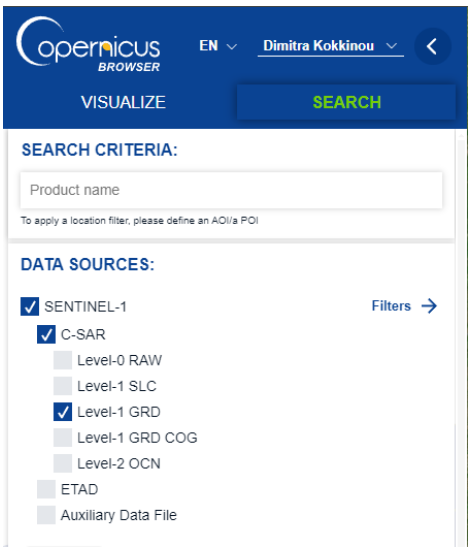


Figure 19: Copernicus browser search criteria

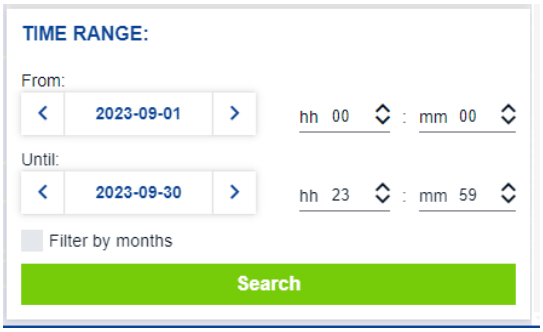


Figure 20: Time range setting

Figure 21 depicts the suitable image that was taken **before** the event (1/9/2023) is entitled as "`S1A_IW_GRDH_1SDV_20230901T043157_20230901T043222_050129_060869_CFB9.SAFE`":

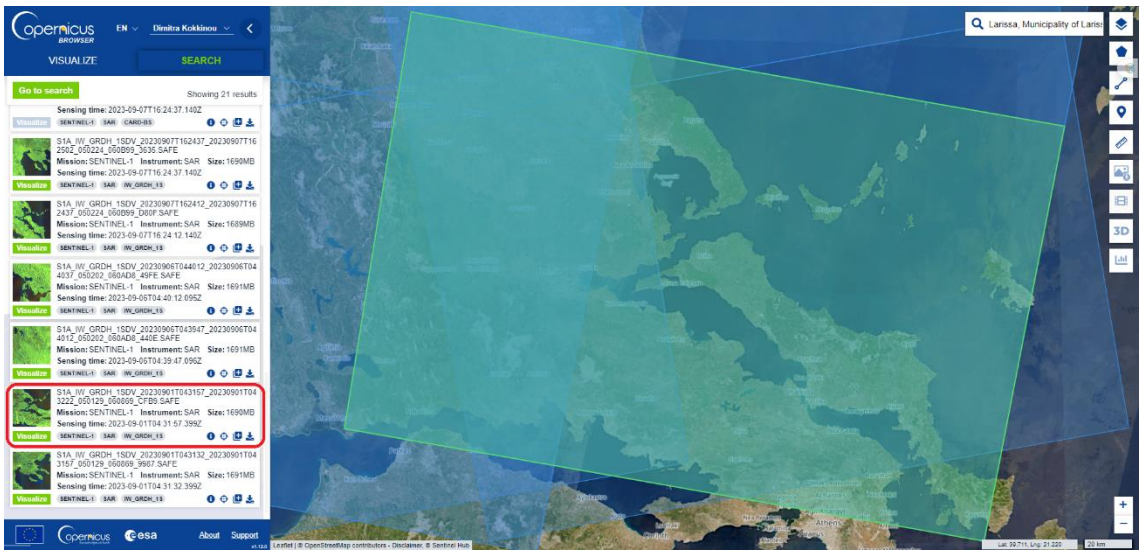


Figure 21: SAR Image on 1st of Sept 2023

Figure 22 depicts the next suitable SAR image that was taken **after** the event (13/9/2023) entitled as:

"S1A_IW_GRDH_1SDV_20230913T043157_20230913T043222_050304_060E62_258F.SAFE"

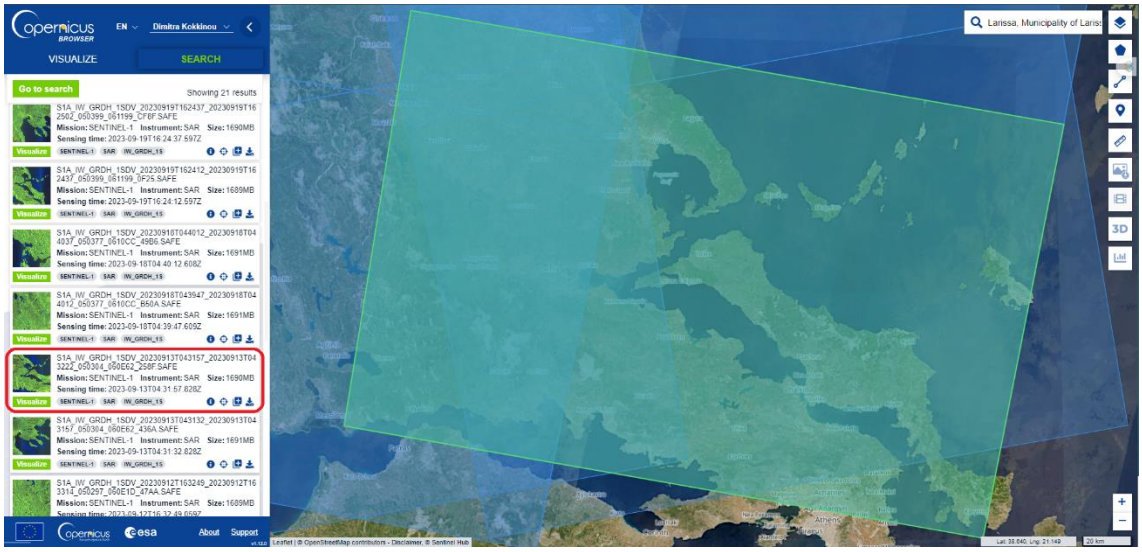


Figure 22: SAR Image on 13th of Sept 2023

Unfortunately, the water was partially absorbed already by that day, but no other suitable image meanwhile that fitted within the area was found.

The size for each image is 1.65GB compressed:

S1A_IW_GRDH_1SDV_20230901T043157_20230901T043222_050129_060869_CFB9.SAFE.zip	18/8/2024 2:07 μμ	WinRAR ZIP archive	1.731.054 KB
S1A_IW_GRDH_1SDV_20230913T043157_20230913T043222_050304_060E62_258F.SAFE.zip	18/8/2024 2:07 μμ	WinRAR ZIP archive	1.731.017 KB

Before starting Kafka, it is essential to first start Zookeeper, as it manages and coordinates the Kafka brokers, ensuring proper synchronization and maintaining distributed system configurations. It is a prerequisite to determine which broker is the leader of the given partition and topic.

Kafka Version here: 2.13-3.8.0

Hence, in the next step Zookeeper is started first as illustrated bellow:

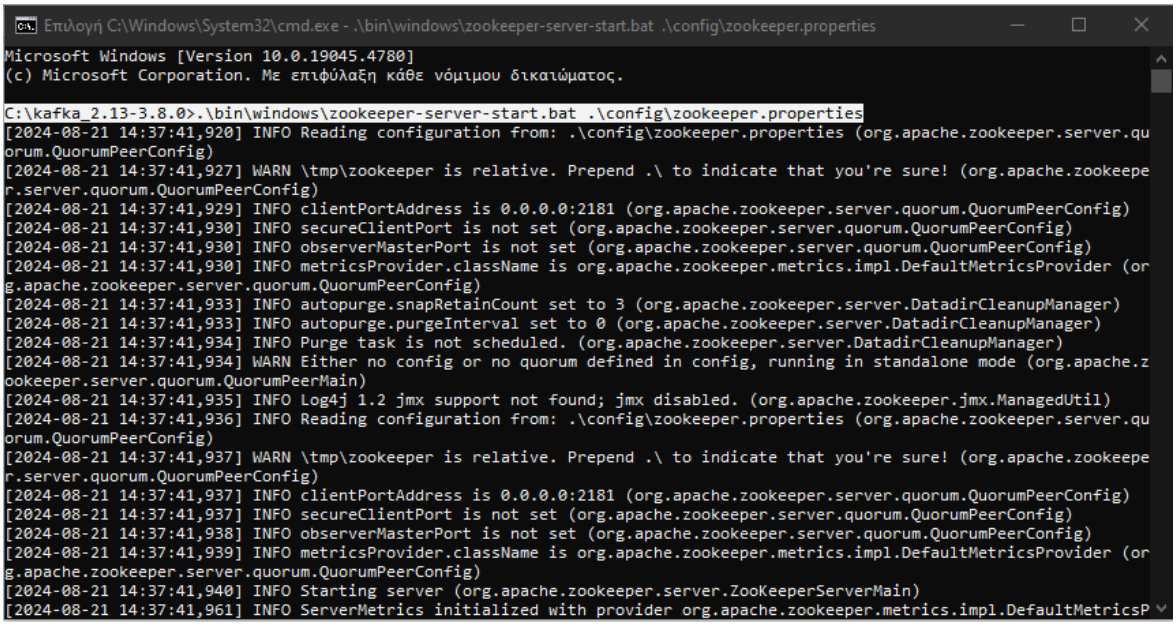


Figure 23: Zookeeper running

Now Kafka broker can be started. Bellow a screenshot of what the user executes:

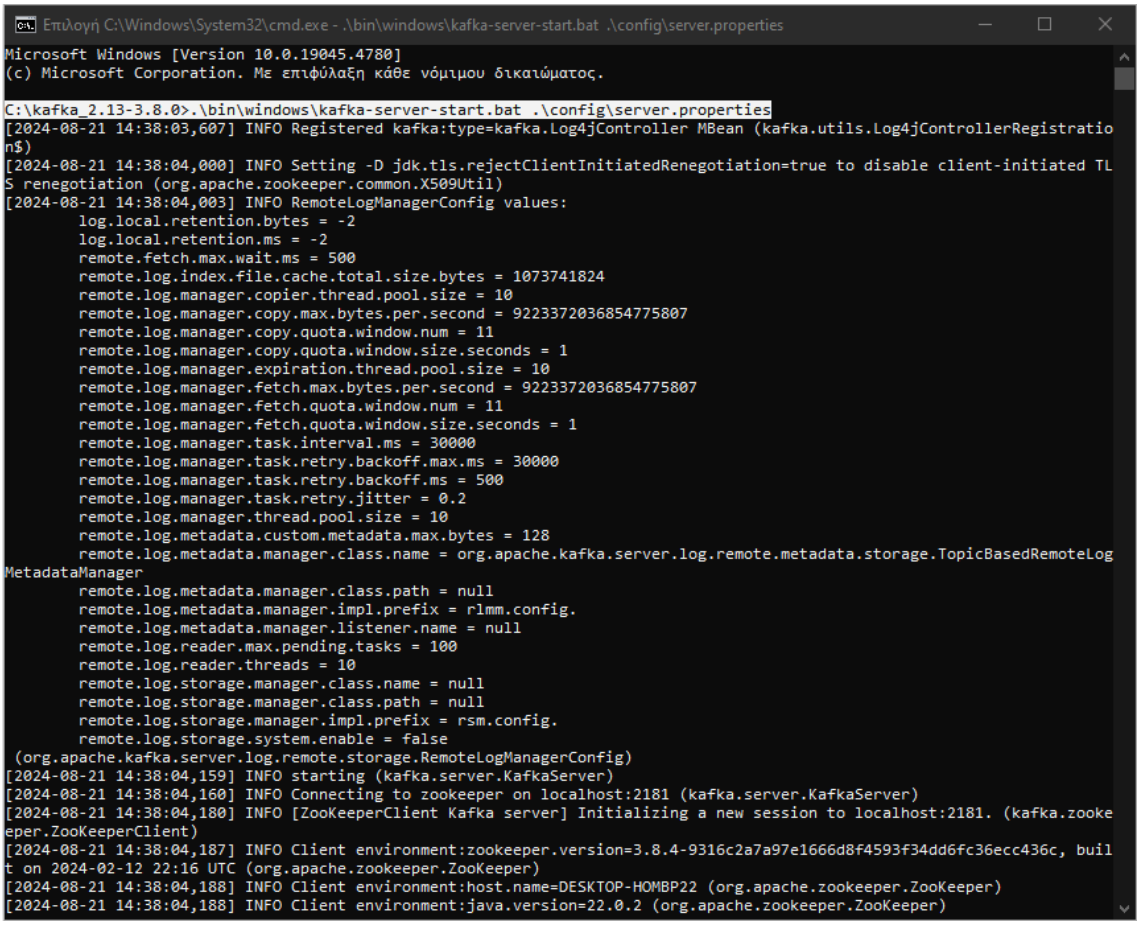


Figure 24: Kafka broker running

Kafka broker service is at this point ready for streaming. In this Use Case the execution is being done locally due to budget constraints (license cost).

The user creates a topic named "*sar-images*" and verifies the existence as shown below:

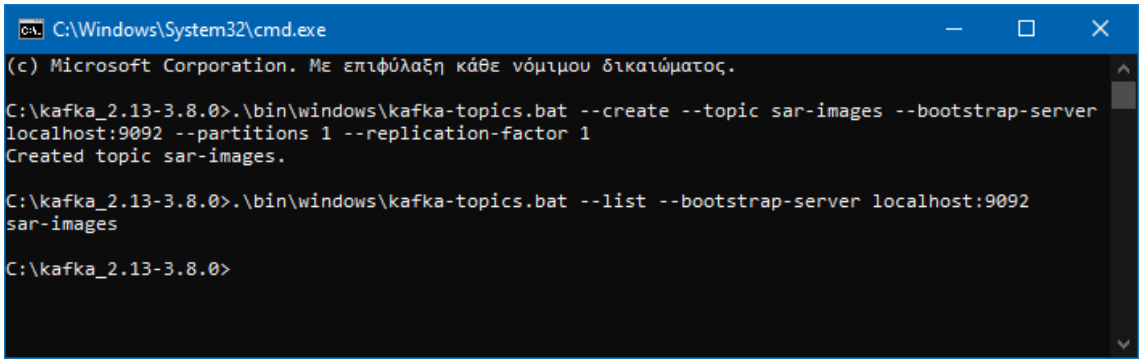
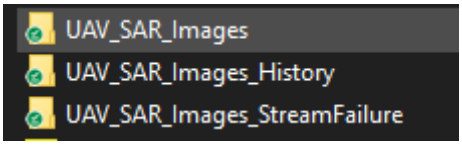


Figure 25: Creating topic

A Python script running on the UAVs' platform is responsible for streaming the real-time SAR images captured during the flight, allowing for immediate processing and analysis. The script is provided in *Appendix C* under the name "*ProducerSARChunks.py*". The process begins by splitting every image file into smaller (50 MB) chunks for easier transmission, as this way their large size is being handled by the broker. It uses three folders created in the storage space of the UAVs. Below an example of the creation:

```
input_folder = 'UAV_SAR_Images'
history_folder = 'UAV_SAR_Images_History'
failure_folder = 'UAV_SAR_Images_StreamFailure'
```



The folder "UAV_SAR_Images" is the area where every captured image is stored for real-time transmission. It is costantly monitored until the flight session is over. When the streaming process is complete and each image is delivered, the folder "UAV_SAR_Images_History" holds the files for historical purposes and permanent storing. In case there's a failure in the procedure, the specific image is stored in the folder "UAV_SAR_Images_StreamFailure" in order for the operators to proceed with the retransmission.

During each flight, all above tasks run continuously until the UAVs' landing.

Image streaming demonstrated below in Figures 26, 27 and 28:

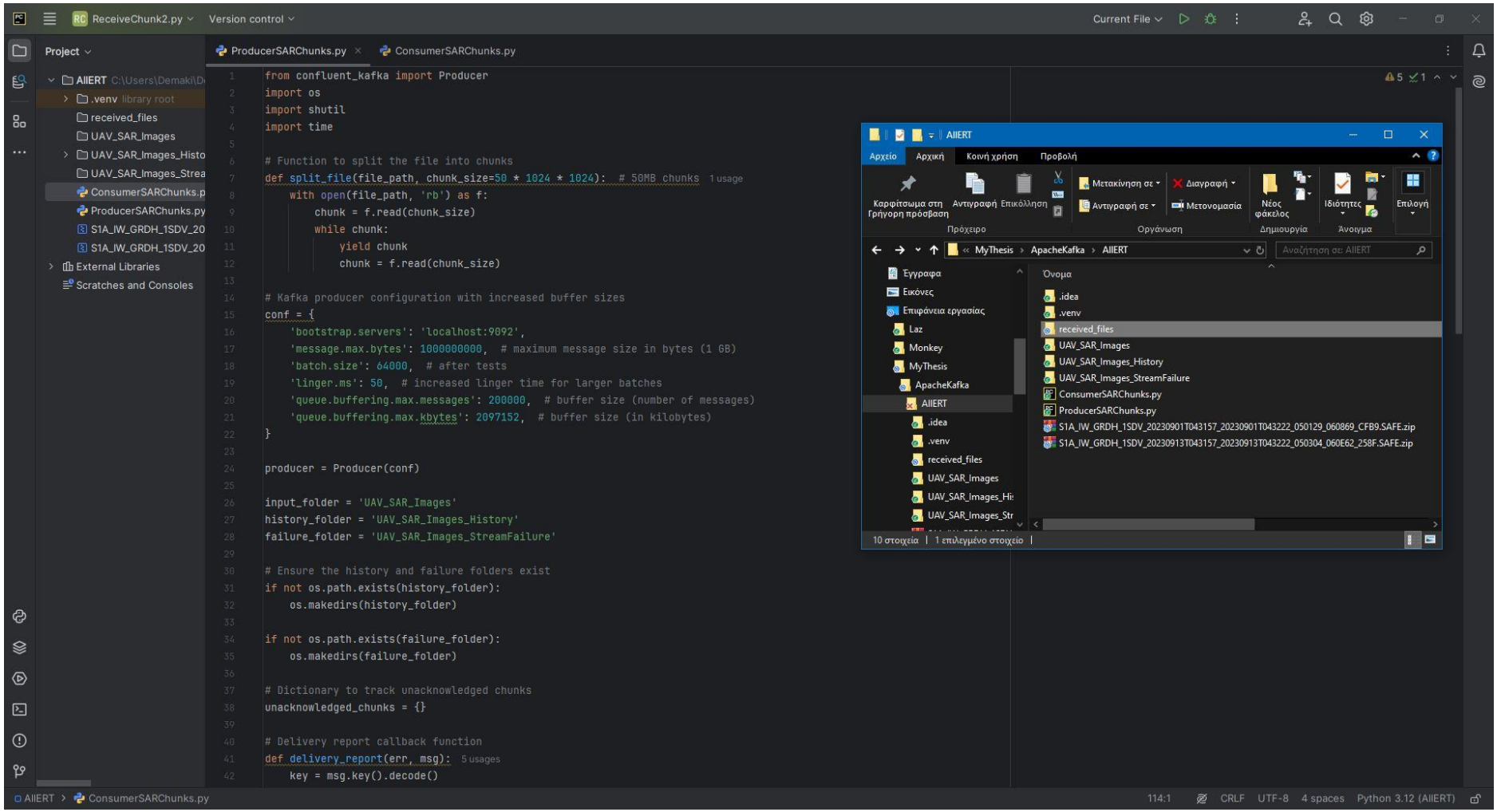


Figure 26: Streaming Procedure (a)

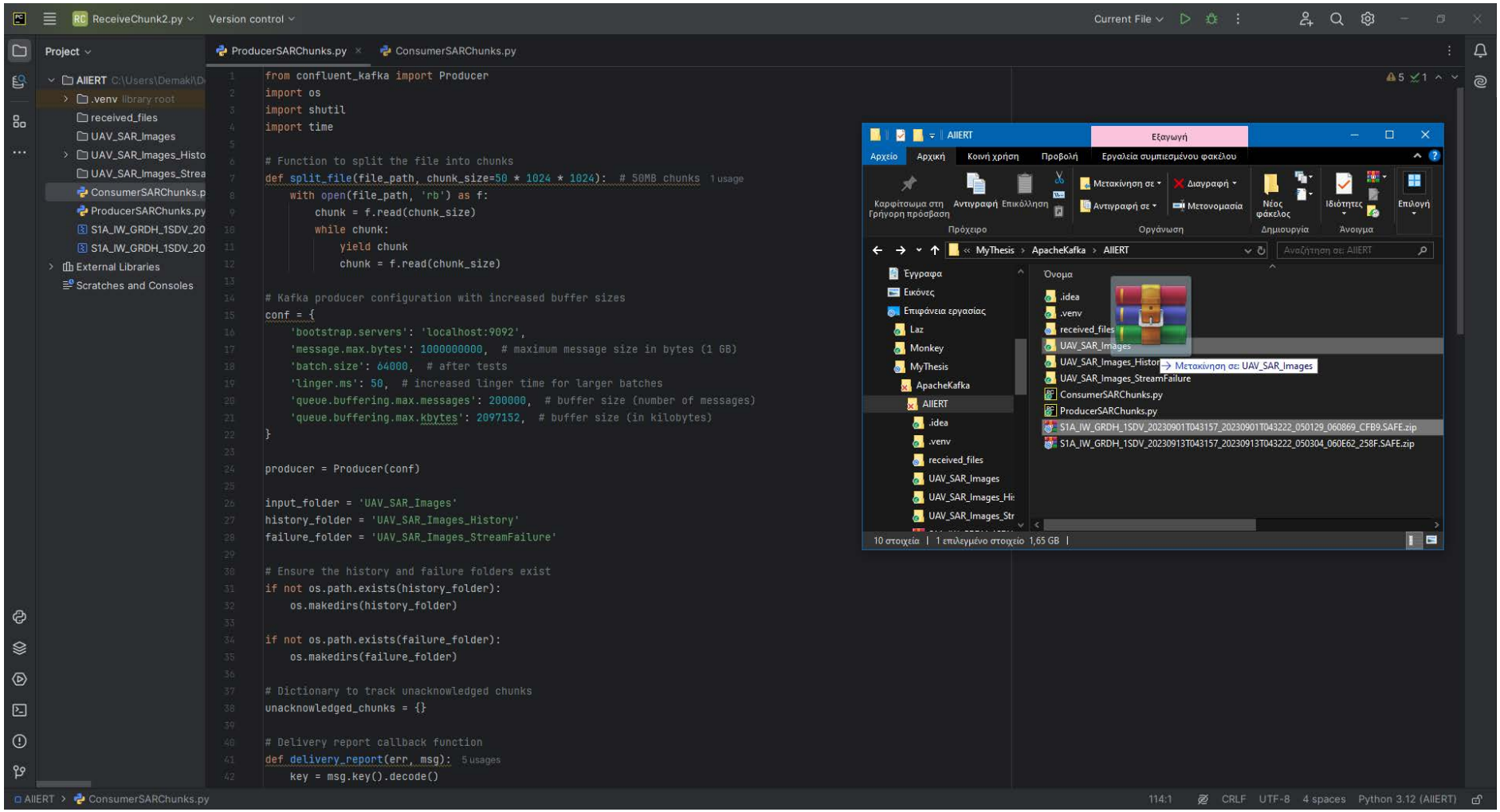


Figure 27: Streaming Procedure (b)

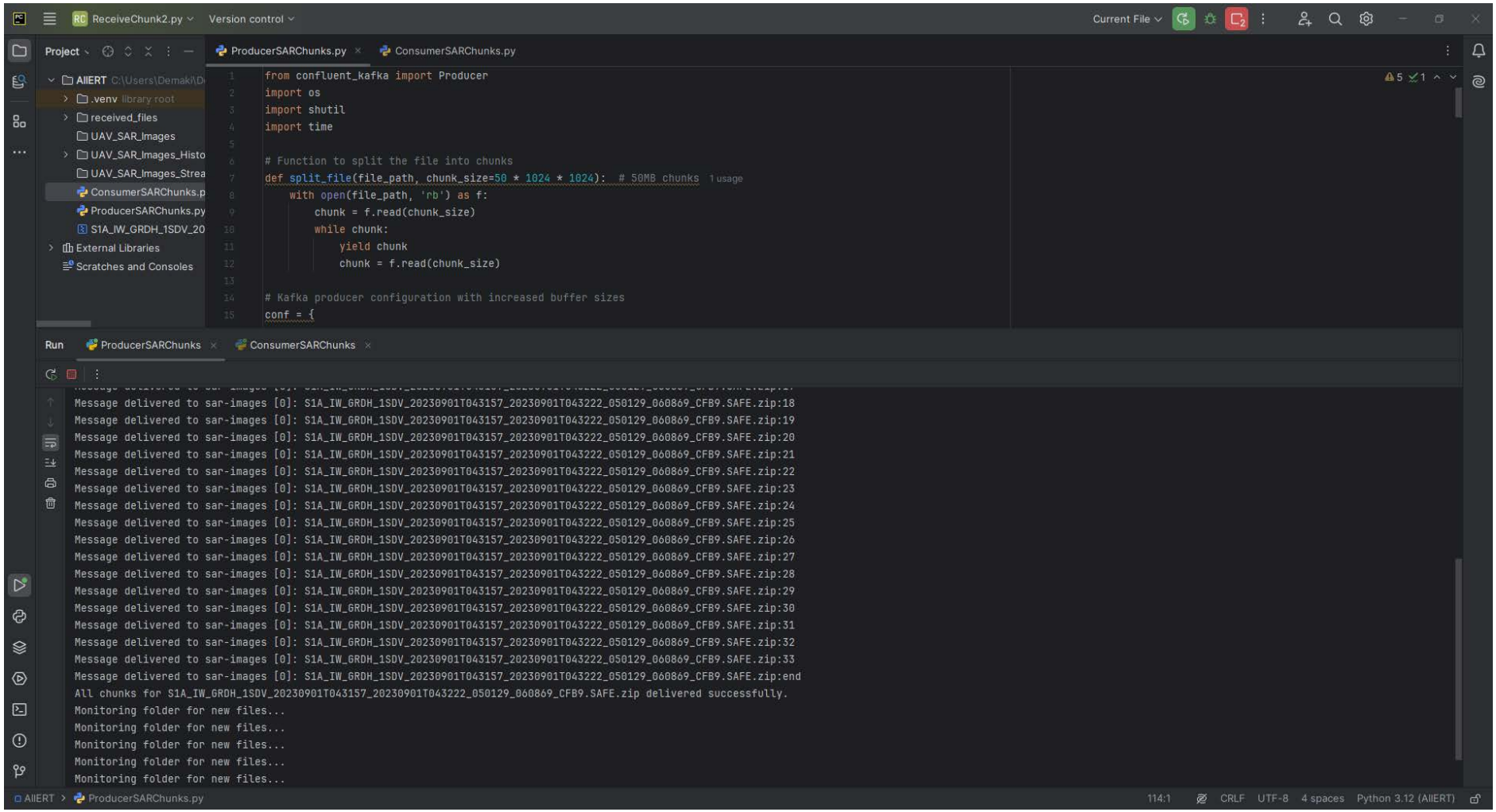


Figure 28: Streaming Procedure (c)

All folders' status after succeeding a transmission is depicted in the following figure:

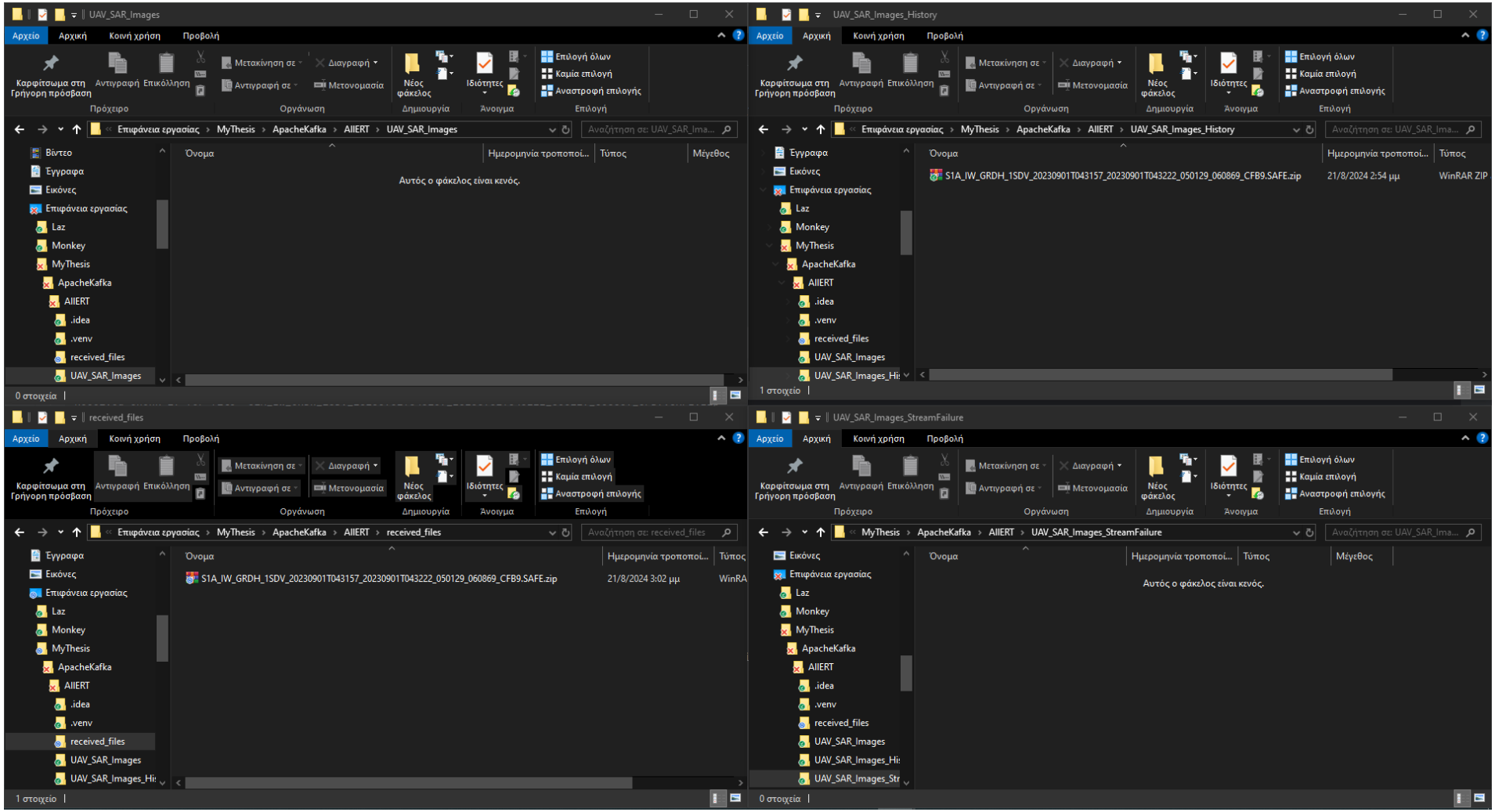


Figure 29: Folder status after Kafka streaming

4.1.5 Step 5 "Processing"

Team: Data Analysis Team

Once the UAVs have completed their data collection flights and the information has been streamed in real-time via Apache Kafka, the next phase is data processing. Before any meaningful analysis can occur, the raw data must be preprocessed. Although they are already geo-referenced, SAR images must also be corrected for any distortion. This involves thermal noise removal, radiometric calibration, despeckling, conversion to decibel (dB) scale and thresholding in dB scale. The methodology suggested here, is the "Map Flood Areas Using Sentinel-1 SAR Imagery" by Mathworks [280], using MATLAB. The example was configured appropriately to match the images extracted in the previous step.

The basic requirement is that Kafka broker service must be running and ready to be used from the consumer's side this time.

The corresponding script designed to handle the consumption of files sent from the UAVs, is also provided in *Appendix C* entitled as "*ConsumerSARChunks.py*". The script receives the chunks from the Kafka topic and reassemble them into complete image files. Each chunk is identified by its file name and chunk index. As the script receives the chunks, it stores them temporarily in memory, organizing them by the chunk index. Once all chunks of a specific image are received and the end signal for that file is detected, the script reassembles the chunks into a single image file and saves it to the specified directory "received_files". It then tracks the number of expected chunks for each image to ensure completeness and handles any errors encountered during message retrieval, allowing the system to reassemble the entire file only after all parts are successfully received.

Running the above python script results in the successful reception of the images as seen in Figure 30:

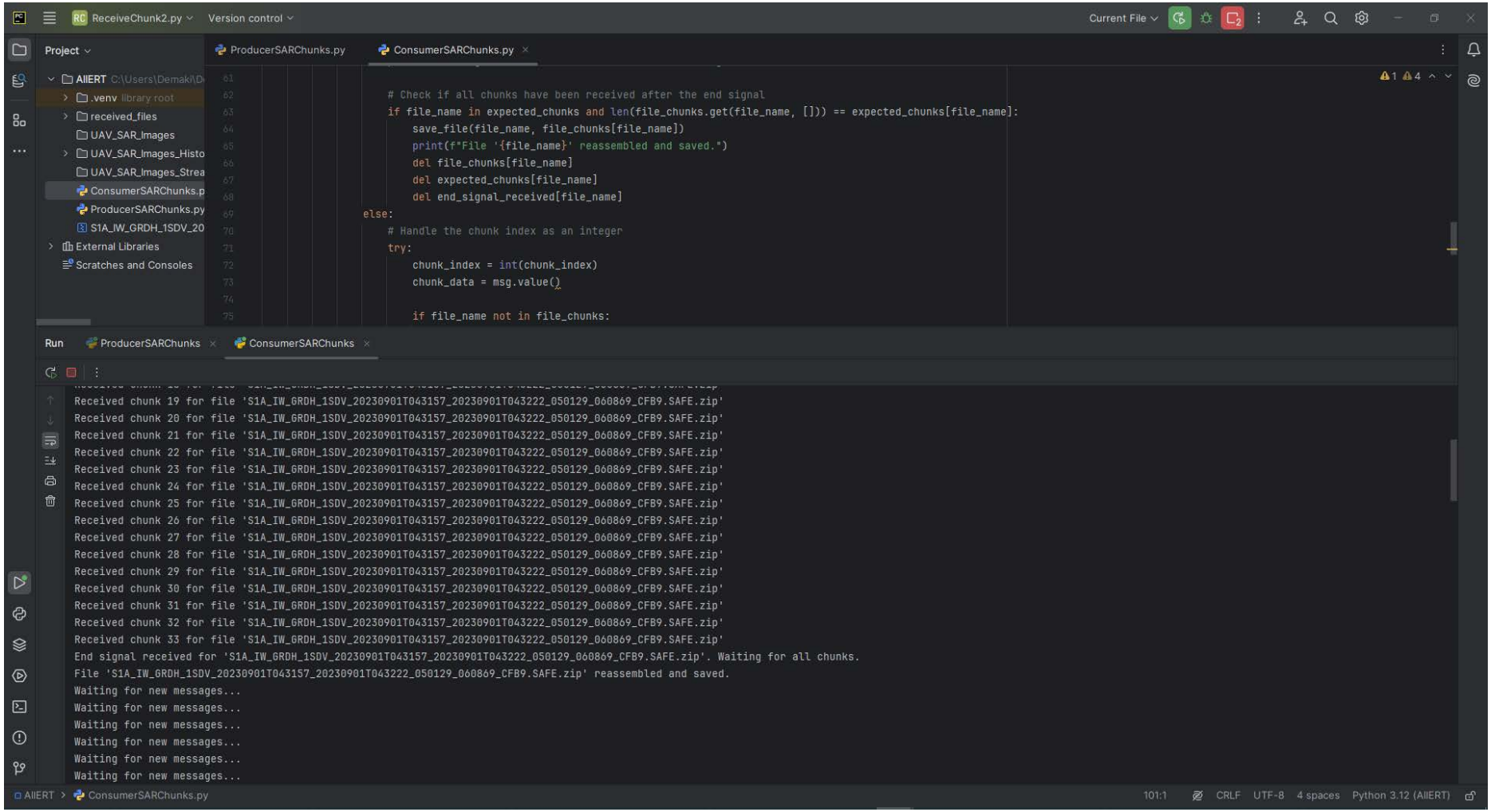


Figure 30: Streaming Procedure (d)

MATLAB

Identification of flooded areas using collected data procedure follows next. The ongoing MATLAB code, describes the procedure of extracting the files that will be used in the next step (***4.1.6 Step 6 "Mapping"***).

Read Sentinel-1 SAR Data from Archive Image

The following code snippet parses metadata from the SAFE file, reads and extracts the VV polarization band from the SAR image (imgVV) and loads associated annotation files to obtain processing-related details.

```
% Read Sentinel-1 SAR Data from Pre-Flood Image (1/9/2023)
dataDir = 'S1A_IW_GRDH_1SDV_20230901T043157_20230901T043222_050129_060869_CFB9.SAFE';
metaInfo = parseSafeFile(dataDir);

imgData = readS1Image(metaInfo);
imgVV = imgData.iw_vv.data;

annFiles = fullfile(metaInfo.rootDir, metaInfo.prodAnnFile);
annMetaInfo = parseAnn(annFiles{1, 1});

processingInfo = annMetaInfo.qualityData.processingInformation;
```

Select ROI of Image

This code clips a ROI from the imgVV by specifying a range of rows and columns (rowIdx and colIdx). The selected ROI is then displayed using the image function while a grayscale colormap is applied.

```
% Select ROI of Image
rowIdx = 19668:22116;
colIdx = 354:1597;
roiImg = imgVV(rowIdx, colIdx);
figure('Name','Pre-Flood','NumberTitle','off');
image(roiImg)
colormap(gray)
xticks([])
yticks([])
axis equal tight
title("ROI: Use Case")
```

Figure 31 is the export from the specified area selected.

Figure 31: Region Of Interest

The **Preprocess Archive (pre-flood) Image** script below, first removes thermal noise, then calibrates the image, applies a Wiener filter for despeckling and finally converts the image to the decibel (dB) scale (Figure 33) before plotting a histogram (Figure 32) of the pixel intensities for the ROI.

```
% Preprocess Pre-Flood Image
noiseFile = fullfile(metaInfo.rootDir,metaInfo.noiseAnnFile{1});
imgTnr = removeThermalNoise(imgVV,noiseFile);

calFile = fullfile(metaInfo.rootDir,metaInfo.calAnnFile{1});
imgTnrCal = calibrateImage(imgTnr,calFile);

imgTnrCalSpk = wiener2(imgTnrCal);

imgTnrCalSpkdB = 10*log10(imgTnrCalSpk);
roiImgTnrCalSpkdB = imgTnrCalSpkdB(rowIdx,colIdx);
figure('Name','Histogram','NumberTitle','off');
histogram(roiImgTnrCalSpkdB)
```

In detail, the thermal noise is removed using a noise annotation file. The removeThermalNoise function, processes the image creating a noise-reduced image (imgTnr). The noise-reduced image (imgTnr) is calibrated using the calibration annotation file (calFile). This corrects the image intensities, compensating for radar system biases and creates a calibrated image (imgTnrCal).

The Wiener filter [281] (wiener2) is applied to the calibrated image (imgTnrCal) to reduce speckle noise (common in SAR images), resulting in a despeckled image (imgTnrCalSpk). The despeckled image is then converted to the dB scale (imgTnrCalSpkdB), which is a common representation for SAR image intensity, making the pixel values easier to interpret. The same specific ROI of the processed image is selected (roiImgTnrCalSpkdB) using rowIdx and colIdx to finally plot a histogram of the pixel intensities in the ROI, showing the distribution of intensity values in the region.

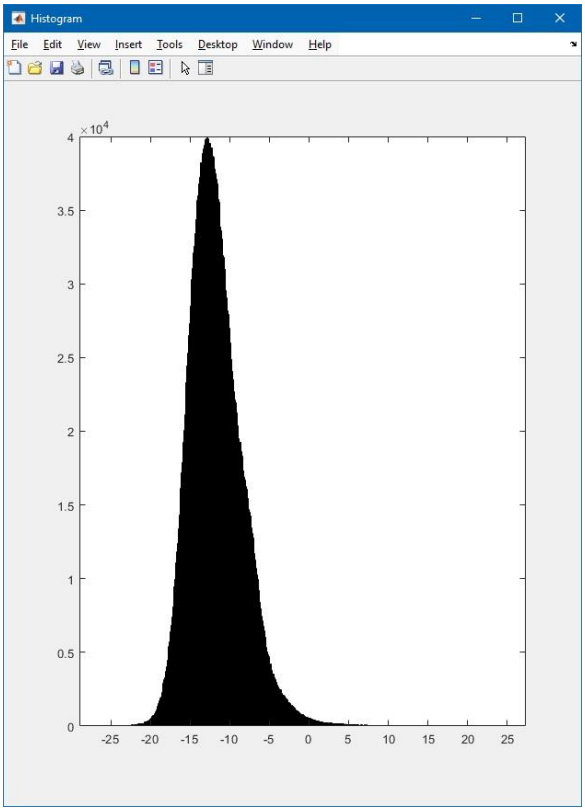


Figure 32: Histogram of archive image

Using the histogram as reference, everything except the water from the image is removed by thresholding the image with a value of -18 to keep only the water bodies in the processed image.

```
imgProc1 = roiImgTnrCalSpkdB;
imgProc1(imgProc1 >= -18) = 1;

figure('Name','Each Stage of Preprocessing','NumberTitle','off');  %Display the ROI of the
pre-flood image after each stage of preprocessing.
tiledlayout(1,4)
nexttile
roiImgTnr = imgTnr(rowIdx,colIdx);
roiImgTnr = rescale(roiImgTnr);
imshow(roiImgTnr,[0 0.1])
title({"Thermal Noise", ...
    "Removal"})
nexttile
roiImgTnrCal = imgTnrCal(rowIdx,colIdx);
```

```
imshow(roiImgTnrCal)
title({"Radiometric", ...
    "Calibration"})
nexttile
roiImgTnrCalSpk = imgTnrCalSpk(rowIdx,colIdx);
imshow(roiImgTnrCalSpk)
title("Despeckling")
nexttile
imshow(imgProc1)
title("Thresholding")
```

The ROI of the archive image after each stage of preprocessing is displayed:

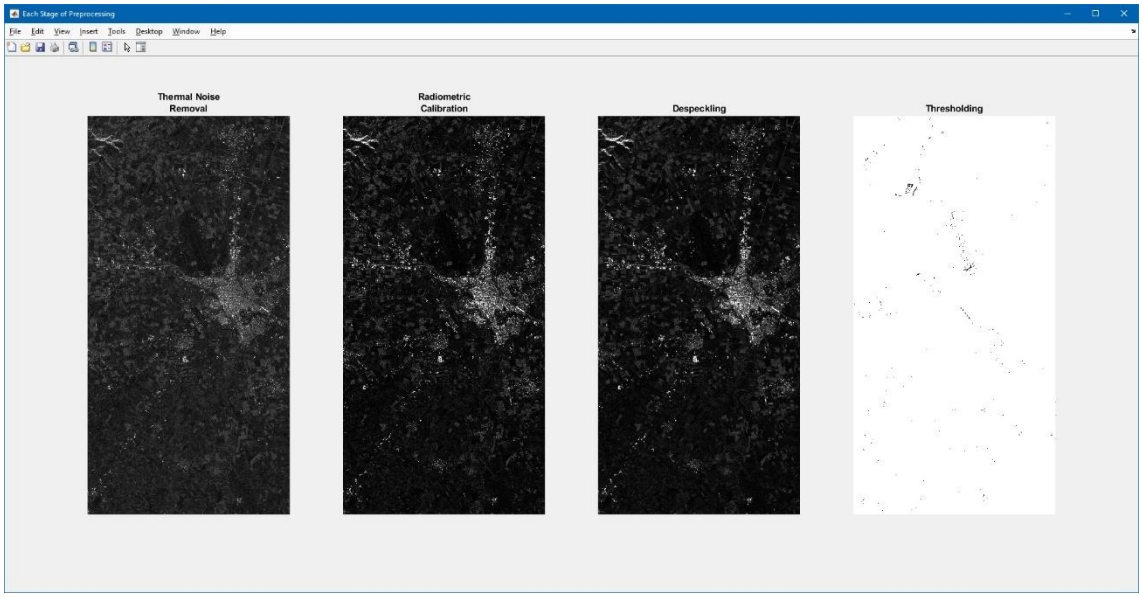


Figure 33: Archive image analysis stages

Same procedure is being carried out for the after-flood image (Figures 34 and 35).

Read Sentinel-1 SAR Data from Crisis Image

```
 % Read Sentinel-1 SAR Data from Crisis Image (13/9/2023)
crisisDataDir =
'S1A_IW_GRDH_1SDV_20230913T043157_20230913T043222_050304_060E62_258F.SAFE';
metaDataCrisis = parseSafeFile(crisisDataDir)

imageDataCrisis= readS1Image(metaDataCrisis);
imgCrisis = imageDataCrisis.iw_vv.data;

annFilesCrisis = fullfile(metaDataCrisis.rootDir,metaDataCrisis.prodAnnFile);
annMetaInfoCrisis = parseAnn(annFilesCrisis{1,1})

processingInfoCrisis = annMetaInfoCrisis.qualityData.processingInformation
```

Select ROI of Image

```
% Select ROI of Image
roiImgCrisis = imgCrisis(rowIdx,colIdx);
figure('Name','Post-Flood','NumberTitle','off');  %Display the ROI of the pre-flood image
after each stage of
image(roiImgCrisis)
```

```
colormap(gray)
xticks([])
yticks([])
axis equal tight
title("ROI of Crisis Image")
```

Figure 34 is the export from the specified area selected.

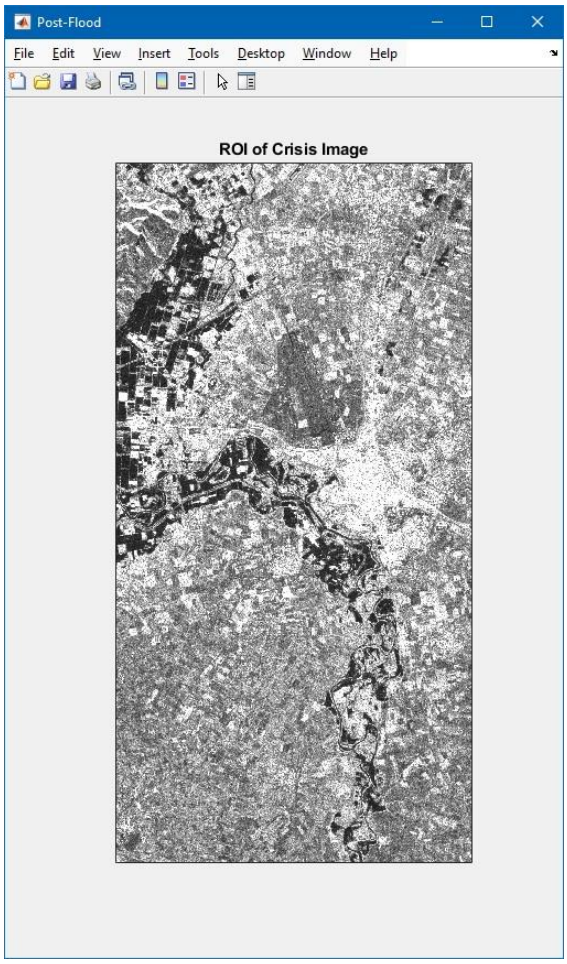


Figure 34: ROI of Crisis Image

Accordingly, the same preprocessing is carried out for the ROI after the flooding event.

Preprocess Crisis Image

```
% Preprocess Crisis Image
noiseFile = fullfile(metaDataCrisis.rootDir,metaDataCrisis.noiseAnnFile{1});
imgTnr = removeThermalNoise(imgCrisis,noiseFile);

calFile = fullfile(metaDataCrisis.rootDir,metaDataCrisis.calAnnFile{1});
imgTnrCal = calibrateImage(imgTnr,calFile);

imgTnrCalSpk = wiener2(imgTnrCal);

imgTnrCalSpkdB = 10*log10(imgTnrCalSpk);
roiImgTnrCalSpkdB = imgTnrCalSpkdB(rowIdx,colIdx);
imgProc2 = roiImgTnrCalSpkdB;
imgProc2(imgProc2 >= -18) = 1;

figure('Name','Each Stage of Preprocessing Crisis Image','NumberTitle','off');  %Display
the ROI of the post-flood image after each stage of
```

```
tiledlayout(1,4)
nexttile
roiImgTnr = imgTnr(rowIdx,colIdx);
roiImgTnr = rescale(roiImgTnr);
imshow(roiImgTnr,[0 0.1])
title({"Thermal Noise", ...
    "Removal"})
nexttile
roiImgTnrCal = imgTnrCal(rowIdx,colIdx);
imshow(roiImgTnrCal)
title({"Radiometric", ...
    "Calibration"})
nexttile
roiImgTnrCalSpk = imgTnrCalSpk(rowIdx,colIdx);
imshow(roiImgTnrCalSpk)
title("Despeckling")
nexttile
imshow(imgProc2)
title("Thresholding")
```

Figure 35, displays the ROI of the crisis image after each stage of preprocessing:

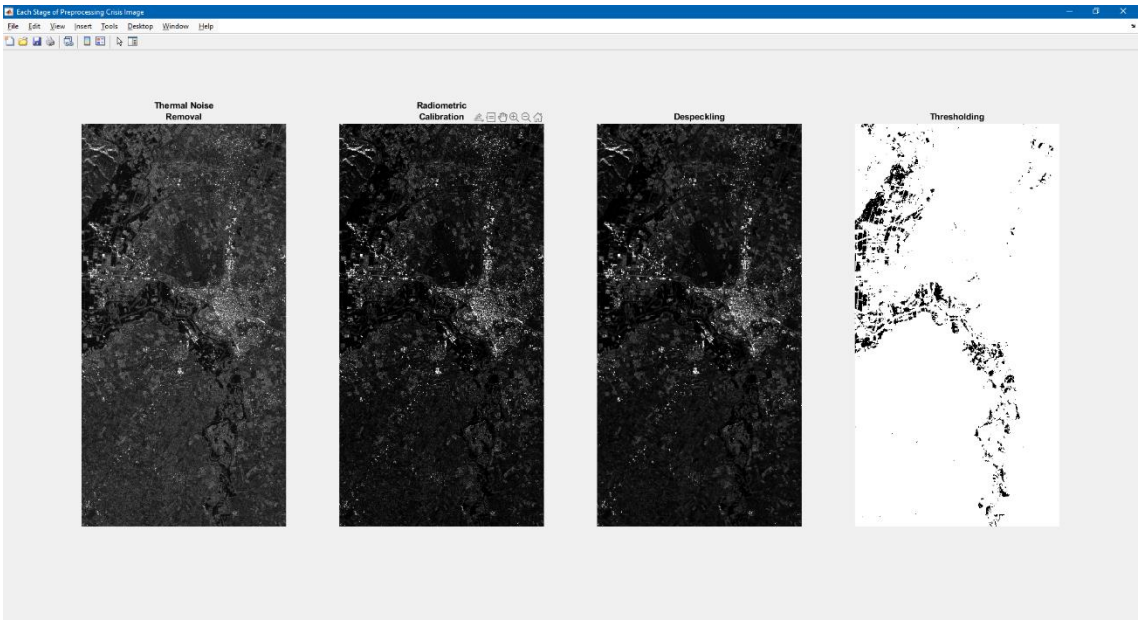


Figure 35: Crisis image analysis stages

The code below, compares pre-flood and post-flood images to detect and highlight flooded areas. The resulting flood mask is inverted and overlaid on the original image to create a visual representation, with flooded regions marked in cyan (Figure 36).

Identify Flooded Areas

```
 % Identify Flooded Areas
floodedAreas = imgProc2 - imgProc1;

floodAreaMask = floodedAreas;
floodAreaMask(floodAreaMask==0) = 1;

roiImg = rescale(roiImg);
floodAreaMaskInv = imcomplement(floodAreaMask);
overlayImg = imoverlay(roiImg,floodAreaMaskInv,"cyan");
```

Display the flood area mask and the flooded areas

```
figure('Name','Flood area mask and the flooded areas','NumberTitle','off');
tiledlayout(1,2)
nexttile
imshow(floodAreaMask)
title("Flood Area Mask")
nexttile
imshow(overlayImg)
title("Flooded Areas")
```

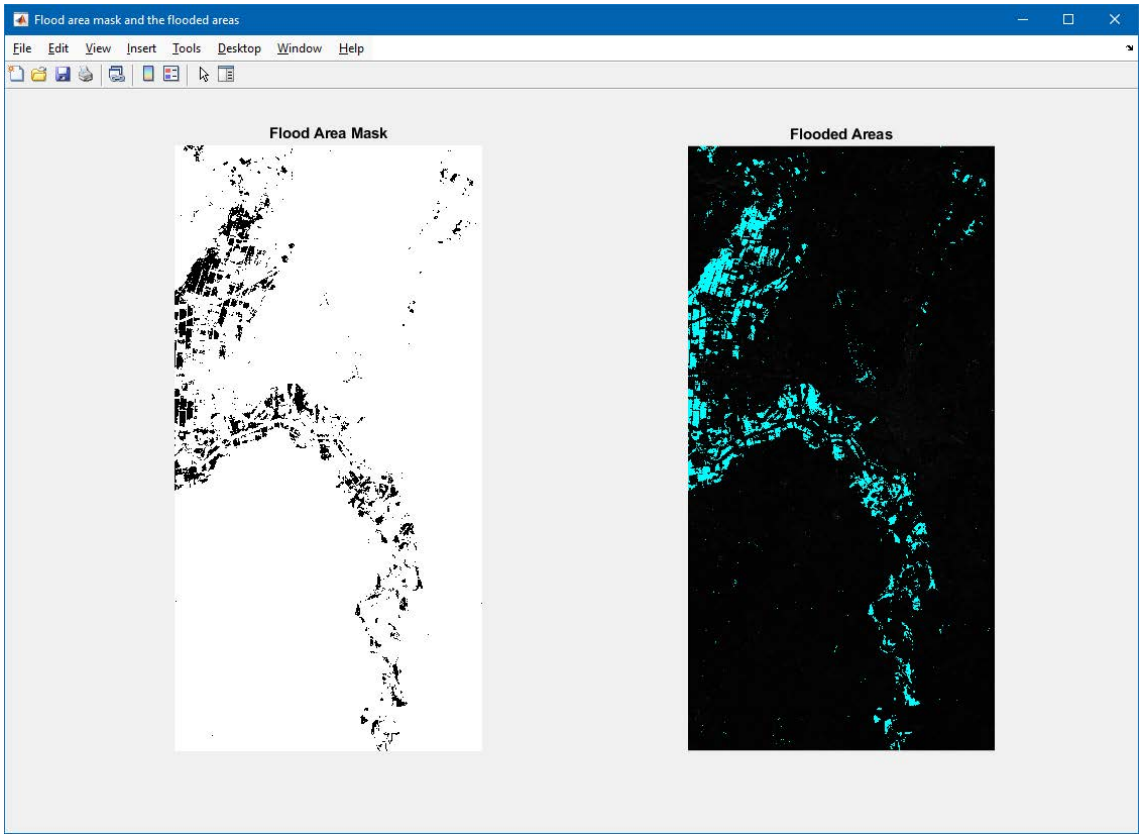


Figure 36: Flooded areas

Next, geolocation data related to the SAR image grid points are projected to a map using a scatter plot. By plotting the latitude and longitude of these points, it helps to visualize the flood area on a geographic map, giving context to the locations affected by the flood.

Project Flood Area Mask on Map

```
% Project Flood Area Mask on Map
gdpoints = annMetaInfo.geolocationGridPoints;
gdpoints = struct2table(gdpoints);
figure('Name','Scatter Plot','NumberTitle','off');
geoscatter(gdpoints,"latitude","longitude")
```

A scatter plot can be an intermediate step before moving to full map projection. The scatter plot displayed here is given below:

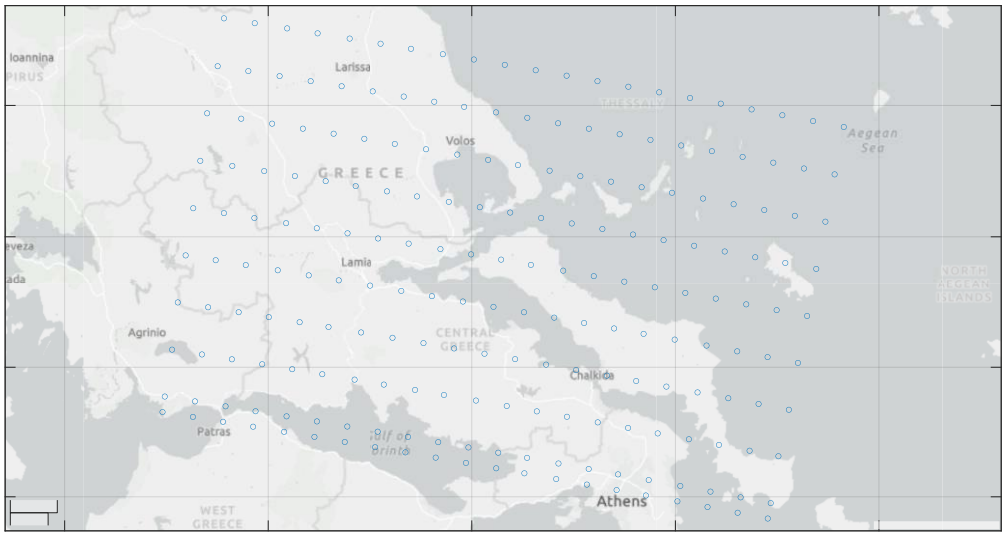


Figure 37: Scatter Plot

The following part of the code takes a flood area mask and projects it onto a map using a defined coordinate reference system. The SAR image's grid points are transformed from latitude and longitude into projected coordinates and a geospatial transformation aligns the flood area mask with the target map region. The flooded areas are then identified and their coordinates are converted into real-world geographic points for visualization.

```
p = projcrs(32634);
[x,y] = projfwd(p,gdpoints.latitude,gdpoints.longitude);
movingPoints = [x y];
line = gdpoints.line;
pixel = gdpoints.pixel;
fixedPoints = [line pixel];
tform = fitgeotform2d(movingPoints,fixedPoints,"similarity");

floodAreaMaskTransformed = imwarp(floodAreaMaskInv,tform);


latROI = [39.735, 39.735, 39.587, 39.587, 39.735];
lonROI = [22.283, 22.541, 22.541, 22.283, 22.283];
[xb,yb] = projfwd(p,latROI,lonROI);
xlimits = [min(xb) max(xb)];
ylimits = [min(yb) max(yb)];
R = maprefcells(xlimits,ylimits,size(floodAreaMaskTransformed),RowsStartFrom="east");
R.ProjectedCRS = p;

dataRegion = mappolyshape(xb,yb);
dataRegion.ProjectedCRS = p;

[rowsVec,colsVec] = find(floodAreaMaskTransformed >20);
[polyX,polyY] = intrinsicToWorld(R,colsVec,rowsVec);

shape = mappointshape(polyX,polyY);
shape.ProjectedCRS = p;
```

Figure 38 displays the flood area in a geographic axes with the data region after running the following commands:

```
figure('Name','Flood area in geographic axes with the data region','NumberTitle','off');
geoplot(dataRegion,LineWidth=2)
hold on
geoplot(shape,"r.")
geobasemap satellite
```

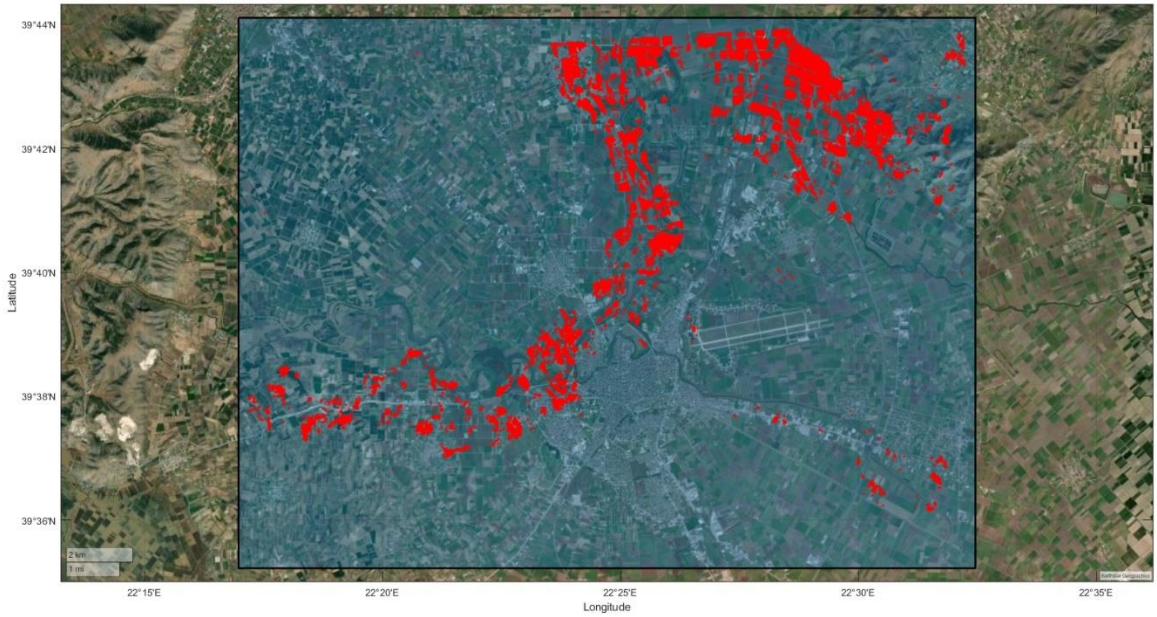


Figure 38: Flooded areas on Map

Using the code below, it is possible to export a file in GeoTIFF format that can be used by GIS if needed.

```
% Export in GeoTIFF
outputFileName = 'flooded_areas.tif';
epsgCode = 32634;  %UTM Zone 34N
geotiffwrite(outputFileName, floodAreaMaskTransformed, R, 'CoordRefSysCode', epsgCode);
```

A shapefile is also exported to be easily processed by ArcGIS Pro in the next phase:

```
% Export flood area to polygons (vector data-SHP)
floodedPolygons = bwboundaries(floodAreaMaskTransformed > 20);
% Initialize arrays to store polygon vertices and attributes
latitudes = [];
longitudes = [];
parts = [];

for k = 1:length(floodedPolygons)
    boundary = floodedPolygons{k};
```

```
    % Convert pixel coordinates to geographic coordinates
    [lat, lon] = intrinsicToWorld(R, boundary(:,2), boundary(:,1));

    % Append the coordinates to arrays
    latitudes = [latitudes; lat; NaN];  % NaN separates polygons
    longitudes = [longitudes; lon; NaN];
    parts = [parts; repmat(k, length(lat)+1, 1)];  % Part numbers
end

% Create a mapshape object
polygonShapes = mapshape(latitudes, longitudes, 'Part', parts);

shapefileName = 'flooded_areas.shp';

% export the polygons as a shapefile
shapewrite(polygonShapes, shapefileName);
```

The above MATLAB code, utilizes functions[11] that are downloaded from the "Map Flood Areas Using Sentinel-1 SAR Imagery" [232] example, configured to run on the images of this specific Use Case. These functions can be found in the *Appendix B* and they are the following:

- *readS1Image.m*

  This function handles the loading of Sentinel-1 radar data. It reads the radar image file and loads the relevant imagery into a MATLAB structure (or array) for further processing. This step prepares the raw satellite data for subsequent analysis.

- *parseSafeFile.m*

  This is a function that parses the SAFE file including not just the radar data but also a variety of metadata files (e.g. geographic information). Its primary role is to extract metadata from the file. It accepts a file path as input and returns metadata in a structured form.

- *parseAnn.m*

  This function processes the annotation file associated with the radar image (accompanying the SAFE file). The annotation file contains metadata about the image acquisition such as orbit parameters, imaging mode and sensor settings. Parsing this file allows for the extraction of technical details that may be necessary for georeferencing, further calibration, or understanding the conditions under which the image was captured. This data can enhance the accuracy of the flood extraction process by providing context to the imagery.

[11] In MATLAB, a function is a block of code that performs a specific task and can be reused multiple times throughout a script or other functions. It is defined with the function keyword, followed by the output variables, the function name, and input arguments [282, 283].

- *calibrateImage.m*

  By this function, radiometric calibration is applied to the radar imagery. Radiometric calibration converts raw radar data into physically meaningful values, ensuring that the image accurately represents the reflectivity of the Earth's surface by converting the raw pixel values to standardized units, such as sigma nought (σ^0) [284]. This is pivotal for quantitative analysis, as it enables comparisons of backscatter values across different regions and time periods. Proper calibration improves the accuracy of flood detection.

- *getXMLnode.m*

  This is a utility function used to extract specific values from XML files. It is employed by parseSafeFile.m and parseAnn.m, to retrieve metadata from the XML documents within the SAFE package. The function takes a node name as input and returns the corresponding data, whether it's a string, number, or array.

- *removeThermalNoise.m*

  This final function processes the SAR image data to remove thermal noise, which is a common practice in radar imagery [285]. Using the noise vectors extracted from the metadata via the parseSafeFile.m function, it adjusts the raw image to enhance the signal-to-noise ratio. A correction is applied by subtracting or compensating the noise levels across the image, ensuring that the data is cleaner and more reliable for detecting surface changes like flood areas.

The following figure shows the MATLAB code execution process (command window), as well as variable and value pairs (workspace pane).

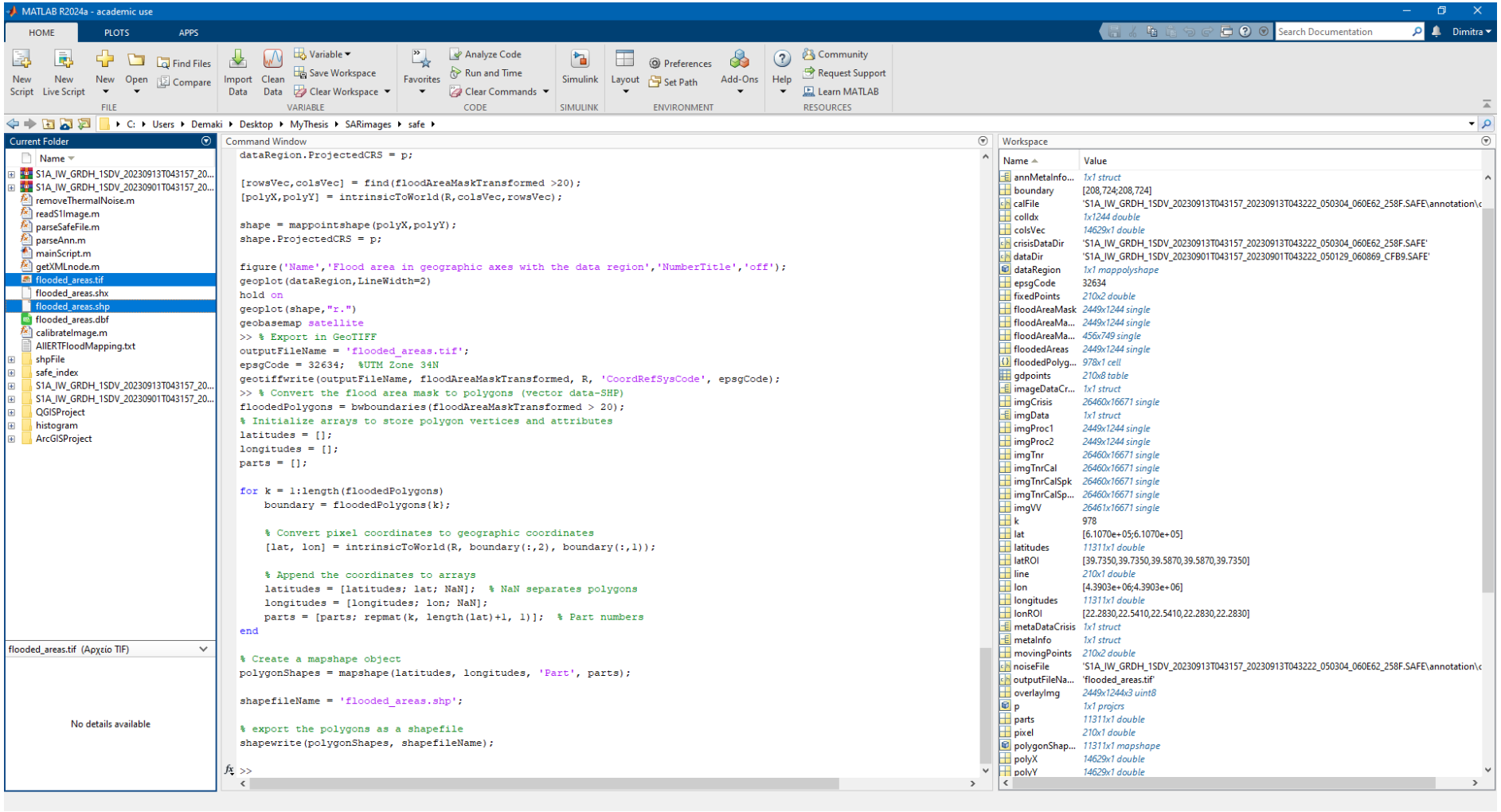


Figure 39: MATLAB Code Execution

An overall view of all MATLAB figures is presented below.

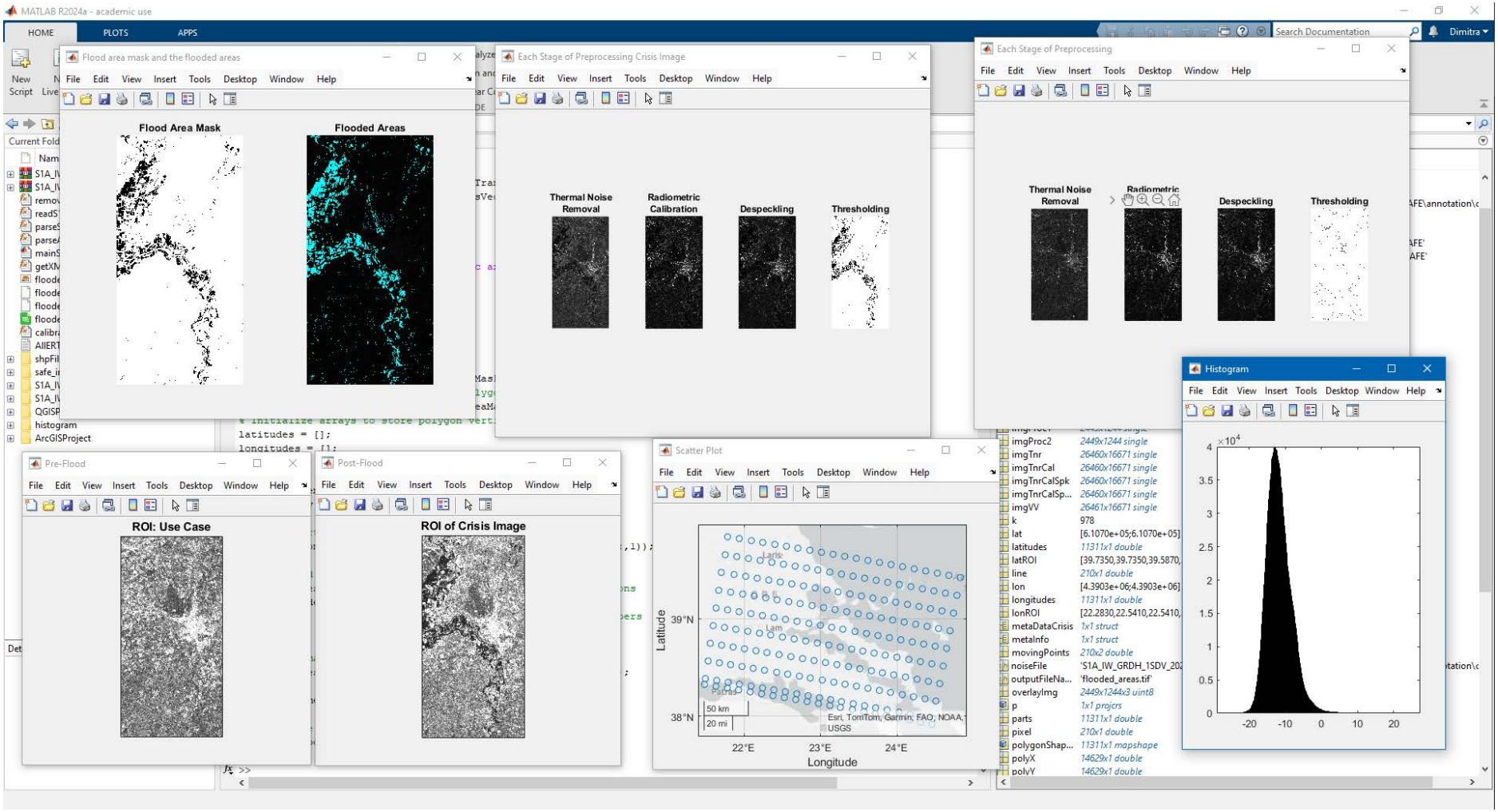


Figure 40: All MATLAB Figures

4.1.6 Step 6 "Mapping"

Team: GIS Analysis Team

The produced shapefile from the previous step is the "flooded_areas.shp" (Figure 41) and it is sent via Apache Kafka same way as the SAR images were delivered to the Data Analysis Team. A new topic is created named "shapefiles" and the corresponding folders are also set accordingly.

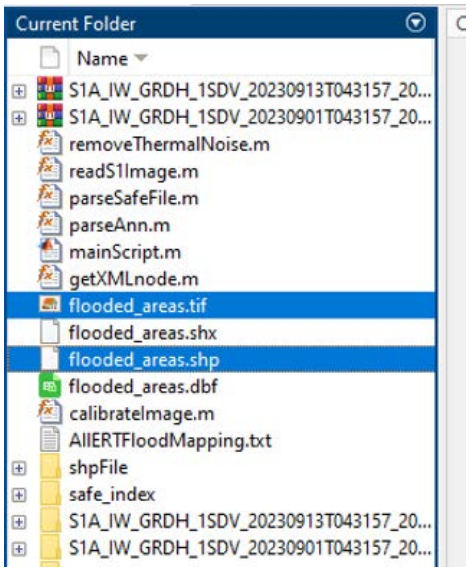


Figure 41: MATLAB produced shapefile to be streamed

It is then pre-processed within **ArcGIS Pro** and two new shapefiles are created. The one is the "Flooded_polygons.shp" and the other is the "Flooded_polylines.shp". This will enable the analysts in this step, to use whichever can best serve the routing procedure.

The methodology uses other datasets as well, necessary to build the network. For example, all the sensors from the Prepare Phase as a point layer, to help combine every possible information they can give by their name and location. This is the "AllERT_Sensors" layer. The roads and municipality of Larissa layers (hotosm_grc_roads_lines_shp & Dimoi_Thessaly.shp) are polyline files downloaded from open data sources [286, 287] and clipped to the area of interest. Figure 42 depicts all the imported layers.

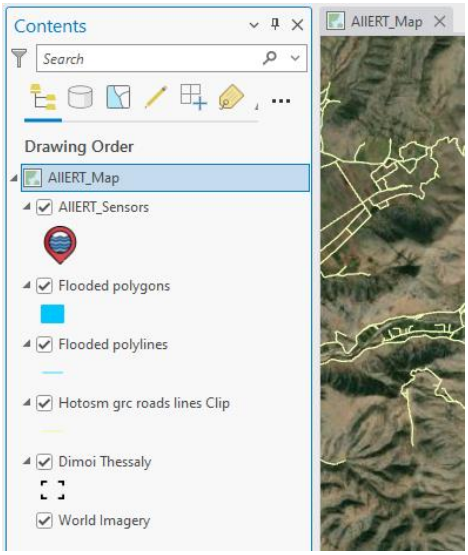


Figure 42: polygons & polylines shapefiles entered in ArcGIS Pro map

Once all the necessary features have been included into the map, a "Share as Web Map" option will be available. This functionality enables the uploading of the specified view to ArcGIS Online, where it can then be integrated into the main dashboard displaying all these spatial information.

Real-time data feeds are also included allowing for up-to-date information and centralizing the management so that users can perform spatial analysis directly within the online environment. It is an interactive map where viewers can zoom, pan and explore different layers.

Figures 43 and 44 illustrate the sharing process.

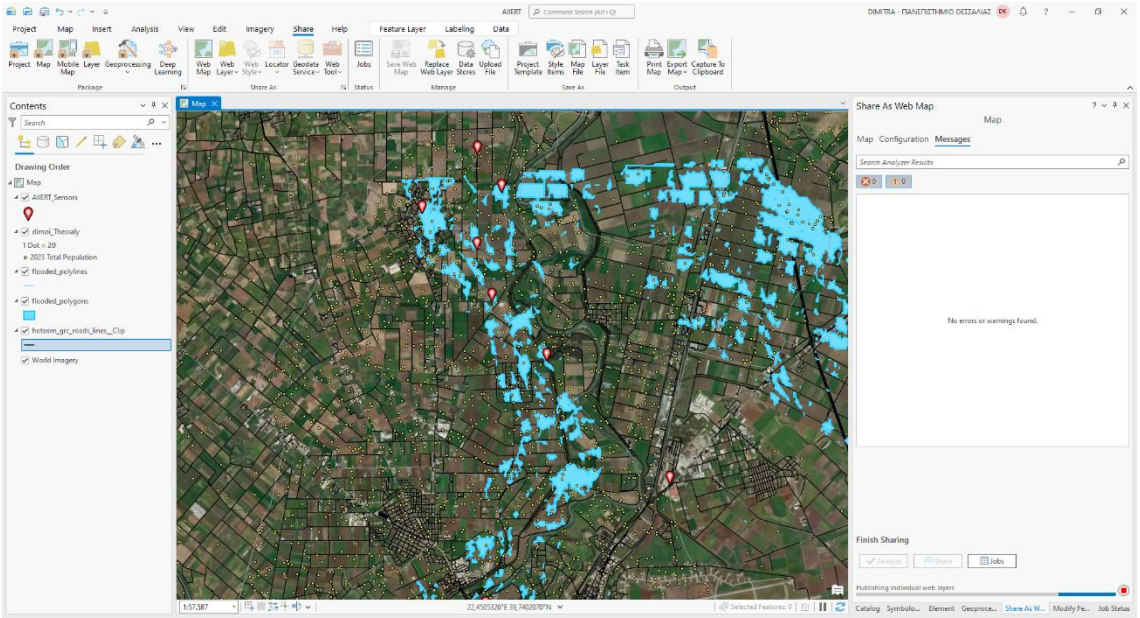


Figure 43: Sharing as a Web Map

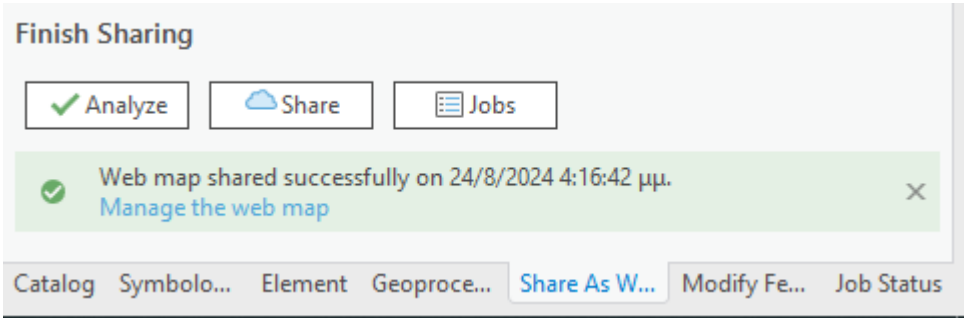


Figure 44: Map successfully shared

Analysis and visualization of the extent of the flooding can subsequently help identifying the most affected areas and determine the safest and most efficient routes for responders to access these areas. The GIS Analysis Team continuously updates these routes as new data becomes available, allowing the emergency response to be as dynamic and responsive as possible to the evolving situation.

During the course of this research, a technical challenge was encountered in the implementation of OAuth 2.0 authentication for accessing the ArcGIS Map Viewer through Node-RED. The authentication process required a redirection to an external identity provider and the management of OAuth 2.0 tokens, which involved complexities beyond the scope of current technical expertise. Specifically, the implementation required interfacing with an OAuth 2.0 authorization server, the existence and configuration of which could not be confirmed.

As a result, the intended automated access to the ArcGIS Map Viewer via Node-RED was not fully realized within the timeframe and technical constraints of this project.

Consequently, ArcGIS Network Analyst, an extension to ArcGIS Pro Desktop is used, which provides various routing algorithms (e.g. shortest path or fastest route) that consider the network dataset's restrictions, including barriers.

Barriers are pre-configured within the network dataset as polyline and polygon barriers that the routing algorithm should avoid. The layers “Flooded_polygons.shp” and “Flooded_polylines.shp” assisted in accomplishing the task. Next, the start and end points are defined and the optimal route is calculated using the Solve function in the Network Analyst toolbar.

Experimenting with various start and end points the following figures (45, 46 and 47) depict an exemplary solution between two points which were phenomenally nearby and after defining a polygon barrier between, a different route was suggested.

Figure 45: Route given without barriers

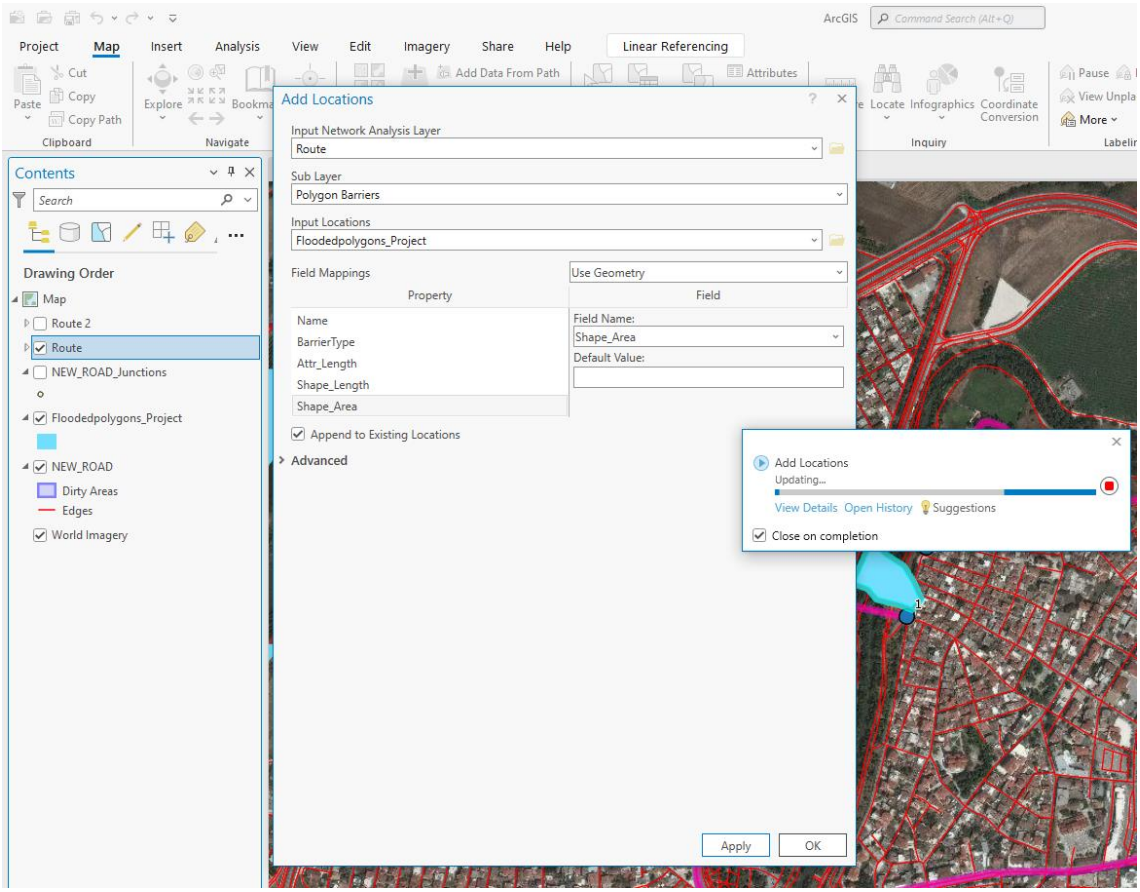


Figure 46: Adding flooding polygons as barriers

Figure 47: Route after barriers configured

The integration of GIS for routing and path planning was an essential element of this study. However, this methodology also encountered multiple challenges in its execution all arising from the quality of the open data used. Most of the attempts were not solved by the Network Analyst Tool as stated previously, generating an error message about no accessible route (Figure 48).

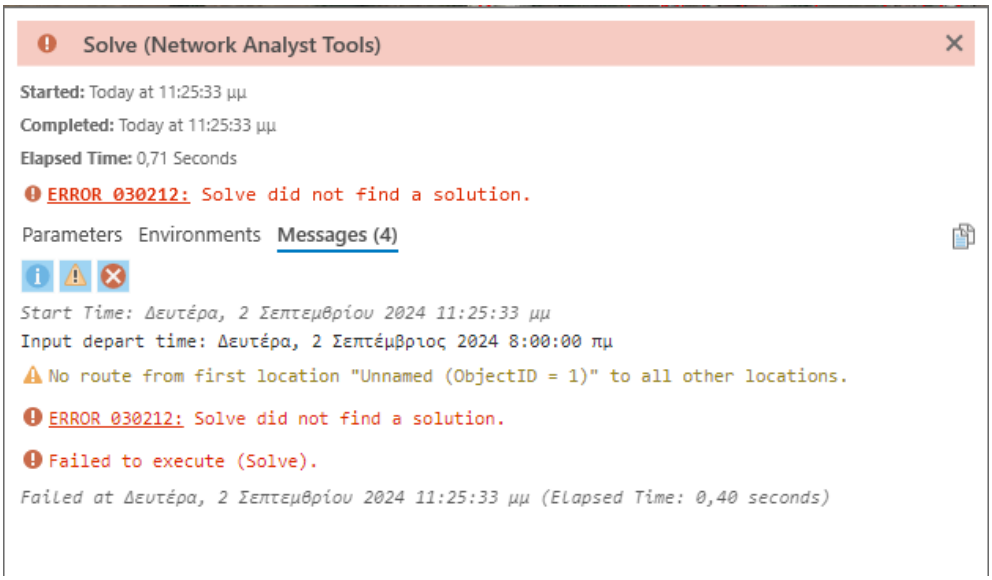


Figure 48: No route found

The assumptions of this issue's investigation are numerous some of which are explained as following:

- Small gaps or disconnections between road segments can prevent the routing algorithm from finding a path between two points and this can lead to incorrect or incomplete routing paths.
- Network attributes such as turn restrictions, one-way streets and barriers, should be reviewed and adjusted, as incorrect configurations can block valid routes.
- Missing attributes can also stop the routing process from operating as intended.
- Misaligned spatial references or projections can cause roads to appear disconnected or incorrectly positioned.
- Topology errors, such as dangles or overlaps can also disrupt routing.

It is important to work with accurate and functional road networks in routing applications as even minor issues in used datasets can impact the results of the routing analysis.

Due to constraints in expertise and available resources, the desired routing functionality using ArcGIS, was not fully implemented. Nevertheless, this functionality is entirely feasible with the involvement of the GIS specialists included in the AllERT protocol already. An expert in GIS would be capable of accurately interpreting the topology and prepare data to create a network dataset for network analysis.

Hence, it is recommended that future work includes the collaboration of GIS professionals and IT departments to fully realize the potential of the system in facilitating real-time decision-making and optimized resource allocation.

4.1.7 Step 7 "Distribution"

Team: DSS Team

This step is the point at which the insights gained from the data analysis are turned into actionable decisions. Having now identified the most critical zones and optimal routes, the DSS uses this information to prioritize and distribute both material and human resources efficiently.

Here, the DSS Team makes decisions based on factors such as flood severity, accessibility, population density and the inventory of resources available, by updating the

system with the decisions they approve in the Dashboard. These decisions are made by consulting the DSS under which the instructions are given and they adjust the missions to be carried out according to the progression of the situation. By efficiently executing this phase, the system helps minimize response times and maximize the efficiency of relief operations. Proper selection of trained personnel with the appropriate skills and the amount of material resources or vehicles, ensures that each team is fully equipped and capable of performing its tasks. Figure 49 illustrates the decision-making environment.

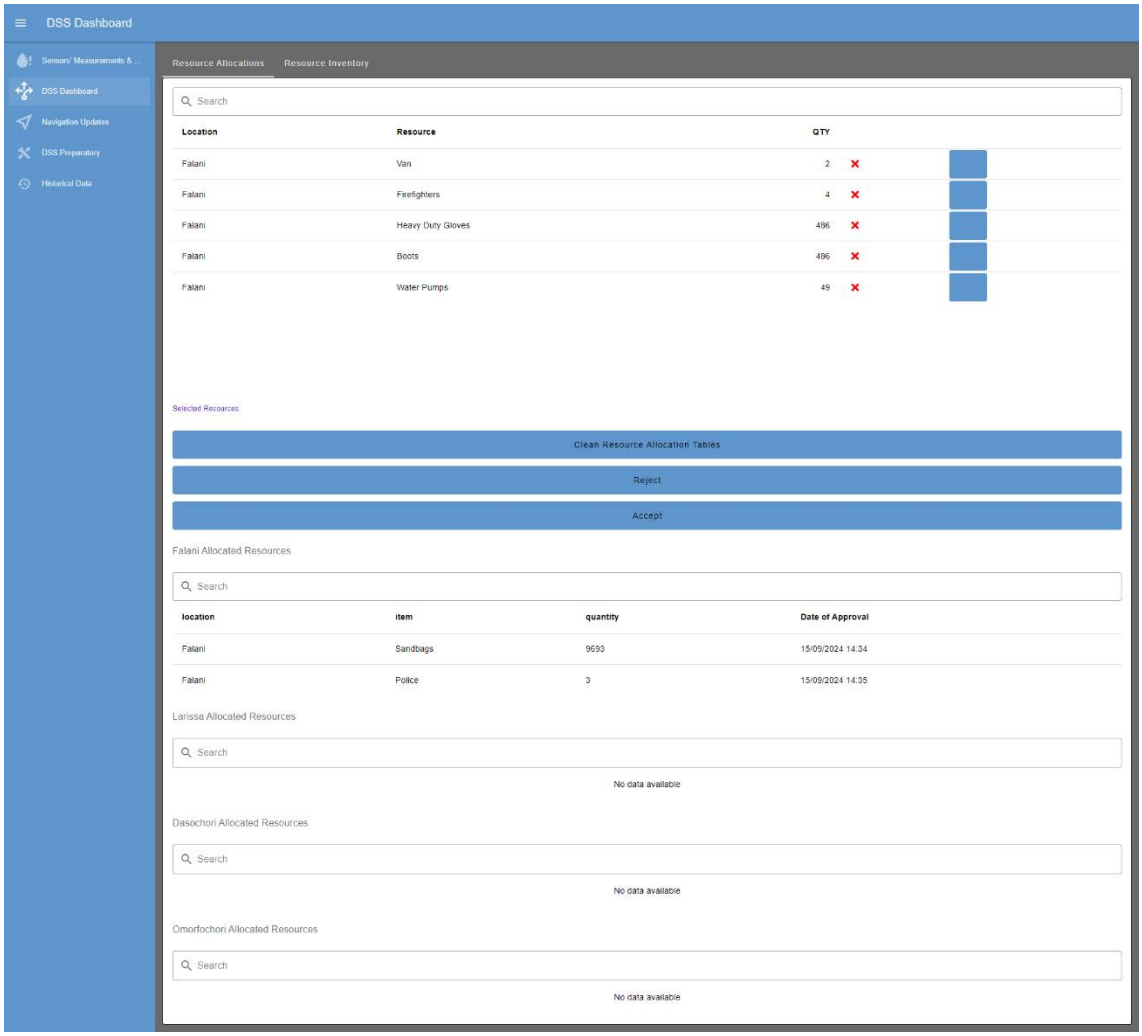


Figure 49: The DSS directions under which the team approves or rejects actions

4.1.8 Step 8 "Dispatch"

Team: Communication Team

This step marks the execution of the response plan developed in the previous steps, where resources and manpower are allocated to the affected areas to provide targeted support. Since the DSS has identified priority areas, necessary supplies and optimal routes, the

dispatch process ensures that instructions are clearly communicated to the respective emergency response teams and that responders are equipped to act promptly.

The Communication Team holds a core role in instructing the field teams, as they are tasked with conveying the execution of the decisions approved by the DSS Team. Responsible for coordinating all dispatch activities, it must be confirmed that they are deployed in an organized and efficient manner. This imposes clear assignment of tasks, real-time updates and dynamic collaboration to ensure that the emergency response is well organized and responsive to the evolving situation. Throughout the whole procedure, communication channels such as satellite phones, TETRA radios, or secure internet links are used. Figure 50 shows the dispatch process supported by the DSS.

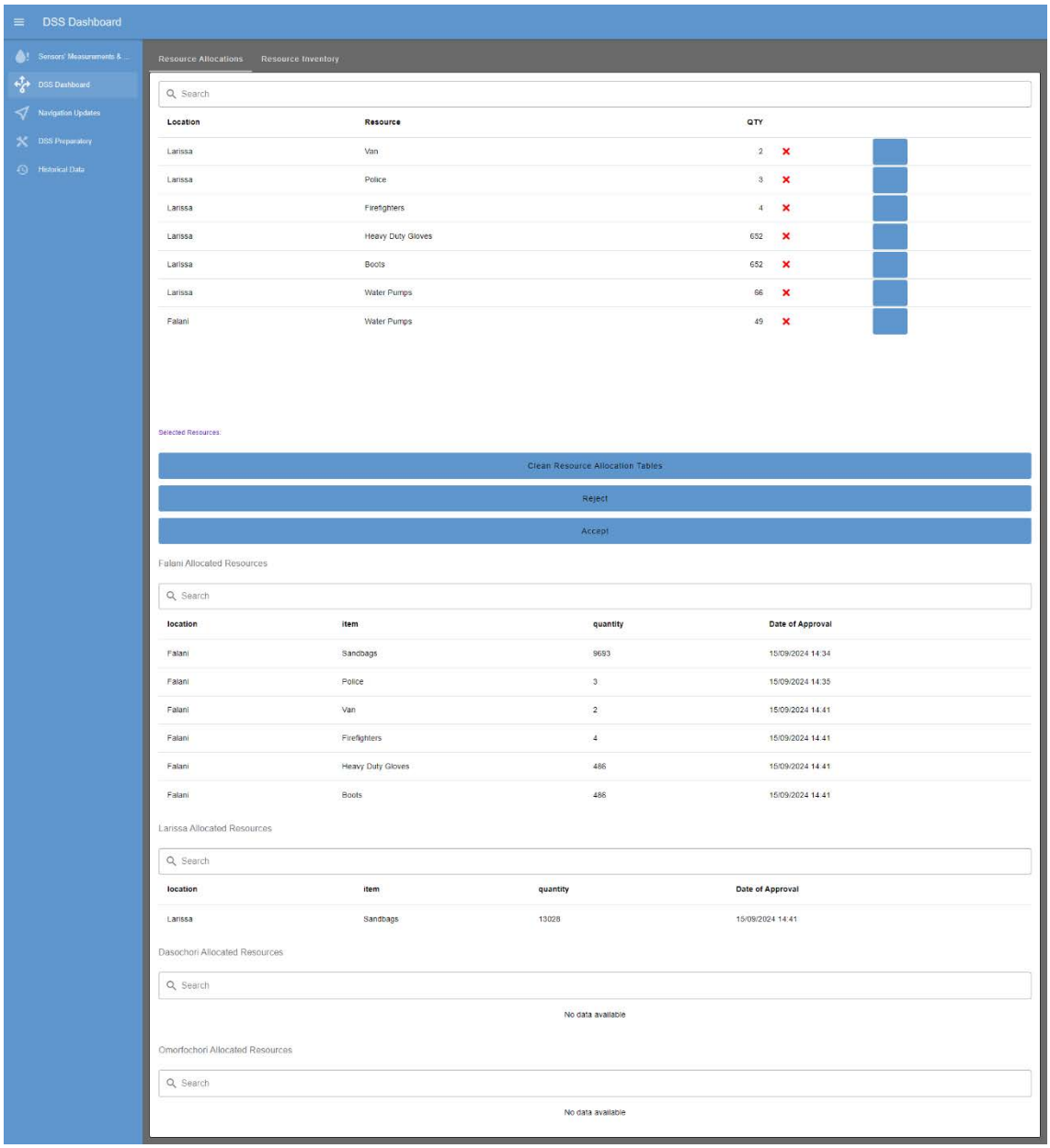


Figure 50: Accepted decisions are communicated to field teams

4.1.9 Step 9 "Response"

Teams: Emergency Response Teams (Firefighters, Medical, Police, Civil Protection, Army and other specialized units)

This is the culmination of all the steps. On arrival at the scene of the incident, response teams carry out evacuation operations, medical assistance, set up temporary shelters and restore basic services according to the defined priorities.

Once a team completes a task, they provide feedback to the Control Tower and the Communication Team. As the situation evolves, responders must remain adaptable, using their training and expertise. Through real-time communication and data analysis they use all available resources to the fullest extent possible.

In the event of flooding with catastrophic consequences (such as in settlements outside this Use Case study where there were even human losses during Daniel storm) the UAV Operation Team uses vehicles with the FINDER payload installed for SAR operations namely the immediate detection of people who are trapped. Figure 51 depicts actual response efforts that took place in the Use Case region.

Figure 51: Response activities during floods in Thessaly due to Daniel storm (Source: [288, 289])

4.1.10 Step 10 "Assessment"

Team: Monitoring and Adjustment Team

The Assessment is a critical concluding step during which, the Monitoring and Adjustment Team analyzes the evolving situation continuously to ensure that all actions taken have achieved the desired outcomes and that the system is ready to adapt to the unpredictable. It is a thorough evaluation of the effectiveness of the response operations, ongoing monitoring of conditions and the identification of areas where further intervention or adjustments are required.

For example, if an area previously marked as critical is now stable, resources may be redirected to newly emerging hotspots. As emergency teams carry out their assigned tasks, the Monitoring and Adjustment Team is responsible for tracking progress through real-time data feeds. Data is continuously collected as well as feedback from field teams. Any gaps in these efforts are identified, and immediate steps are taken to address any outstanding needs. If certain roads are cleared or new hazards arise, the routes and deployment strategies can be altered. In cases where rescue teams encounter new challenges or require more resources, it should be confirmed that the system reallocates them dynamically to support the ongoing response.

Finally, this step includes the compilation of a comprehensive after-action report, which documents the response efforts, highlights successes, and identifies areas where improvements can be made. This report represents a critical learning tool for refining future emergency response, providing valuable insights into what strategies worked and where the system can be optimized. These lessons learned are fed back into the DSS to improve future response strategies.

Through continuous monitoring, appropriate adjustments and post-event analysis, it is certain that the system is ready for future challenges.

4.2 Description of the Dashboard's functionality in Node-RED

AllERT harnesses the power of Node-RED's dashboard module in order to provide a one-stop, fast-responding, user-friendly dashboard where all data is eventually collected, monitored and utilized by the Control Tower Crew and all relevant teams. They are thus empowered by the capabilities of automated alertness, rapid situational assessment, informed decision-making and agile and efficient resource management.

Three are the main pages that constitute the *AllERT* dashboard experience: **"Sensors' Measurements & Map"**, **"Navigation Updates"** and **"DSS Dashboard"** which in turn are complemented by a two more that provide customization and configuration options as well as historical insights (*"DSS Preparatory"* page and *"Historical Data"* page).

Dashboard Group Menu: "Sensors' Measurements & Map"

This page is the eyes of the AllERT dashboard, as it constantly displays the sensors deployed out in the field, together with each location's flooding status and severity as well as the development of the incident in a modular and customizable region map.

At first glance, the Control Tower's Crew members are able to instantly observe the readings (*flood depth*, *temperature*, *humidity*) of all sensors as those are displayed on graphic widgets for easy and accurate interpretation. Moreover, there is a special notification for those sensors whose *flood depth* reading exceeds even the lowest of thresholds (currently set at 250mm). Since sensors are usually spread out in different locations within a region, a specific message updates on the flood severity status for each separate location. These statuses are defined by the highest *flood depth* value amongst sensors for a specific location and are grouped by value ranges as 250mm – 500mm are labeled as Minor Flood, 501mm – 1000mm Moderate Flood, 1001mm – 2000mm Severe Flood and finally 2001mm to 3000mm (which is the sensors' reading limit) as Catastrophic Flood.

In this Use Case, there are sensors placed in four locations in the region of Larissa: the City of Larissa and the villages of Falani, Dasochori and Omorfochori. In order to simulate a more realistic scenario, their temperature readings have been randomly initialized at 28°C to 30°C and have been furtherly constrained to values between 25°C and 32°C as the phenomenon progresses. In a similar fashion, humidity has been initialized at 40% - 60% and can climb as high as 100% (due to the flooding incident or fall back up to 40%). These limitations ensure that in close quarters locations climate readings like temperature and humidity are generally the same without large divergences that could result in an unrealistic representation of a flooding event. Finally, in order to provide a variety of results, two specific sensors have been limited with regard to their flood depth readings to 200mm and 240mm respectively as they reside in places that could prohibit the occurrence of large-scale events due to geomorphological reasons.

In order to simulate the development of a flooding event, several controls have been handed over to the dashboard's operators. As such, a user may initiate a flood (by clicking on the **Initialize Variables & Start Flood Event** button) which causes all sensors to start generating randomly increasing flood depth (and humidity as those two are interconnected) readings, thus gradually raising the location's severity level. Once, three (by convention) or more sensors provide readings that are above the lowest of thresholds (here set as a global variable at 250mm), AllERT is initiated as a flooding event has certainly commenced in the broader region. At the same time, a specific Node-RED module (node) is activated to instantly notify all participants and responders via mobile phone. Those responders possess smartphones with the corresponding application installed, so they receive both an audible and readable alert with the event's details by the AllERT dashboard. For time economy reasons, the *AllERT* initiation may be activated by a dedicated control, called **Force AllERT**, that forces three pre-defined sensors to give over-the-threshold readings. Following the same path, a user may acquire sensor readings on-demand (by pressing the **Get Sensor Values** button) rather than wait for the set periodic time at which sensors send their values -which is set to one minute- to elapse. The option to cease the further upward progress of the flooding event is also provided for simulation purposes. When the corresponding **Stop Flooding Event** button is clicked, sensor flood depth values stop incrementing as the phenomenon begins to discharge. To that end, sensor readings received from that point onward, slowly start to decrease in value until flood depth reaches 0mm and humidity returns to the normal predefined standard of 40%.

Apart from the simulation controls, this dashboard page offers the ability to poll the status of the *AllERT* protocol as well as the number of sensors per location whose flood depth readings currently exceed the lowest of thresholds. This is achieved through the **View AllERT Status** button.

In the same page, a General Sensor Data component displays all sensors' readings in a flood depth to time chart, furtherly enhancing the observation capacity of the incident's development.

Finally, a multi-layered multi-faceted region map enables the operator to view static information such as the sensor's exact location, the region's administrative boundaries and the road network lines. Most importantly of all, however, the flooded areas are also dynamically drawn on the map (as polygons and polylines) to provide the viewers with

an immediate and actual understanding of the whereabouts of the areas at potential risk. This dynamic information is the result of the UAVs data collection that was processed by the Data Analysis Team thus deriving the geolocation of the flooded areas. All of the above are presented as separate features on the map and as such, can be toggled on and off at the user's desire.

Figure 52 illustrates the above implementation in a Node-RED flow, while Figure 53 the corresponding dashboard page.

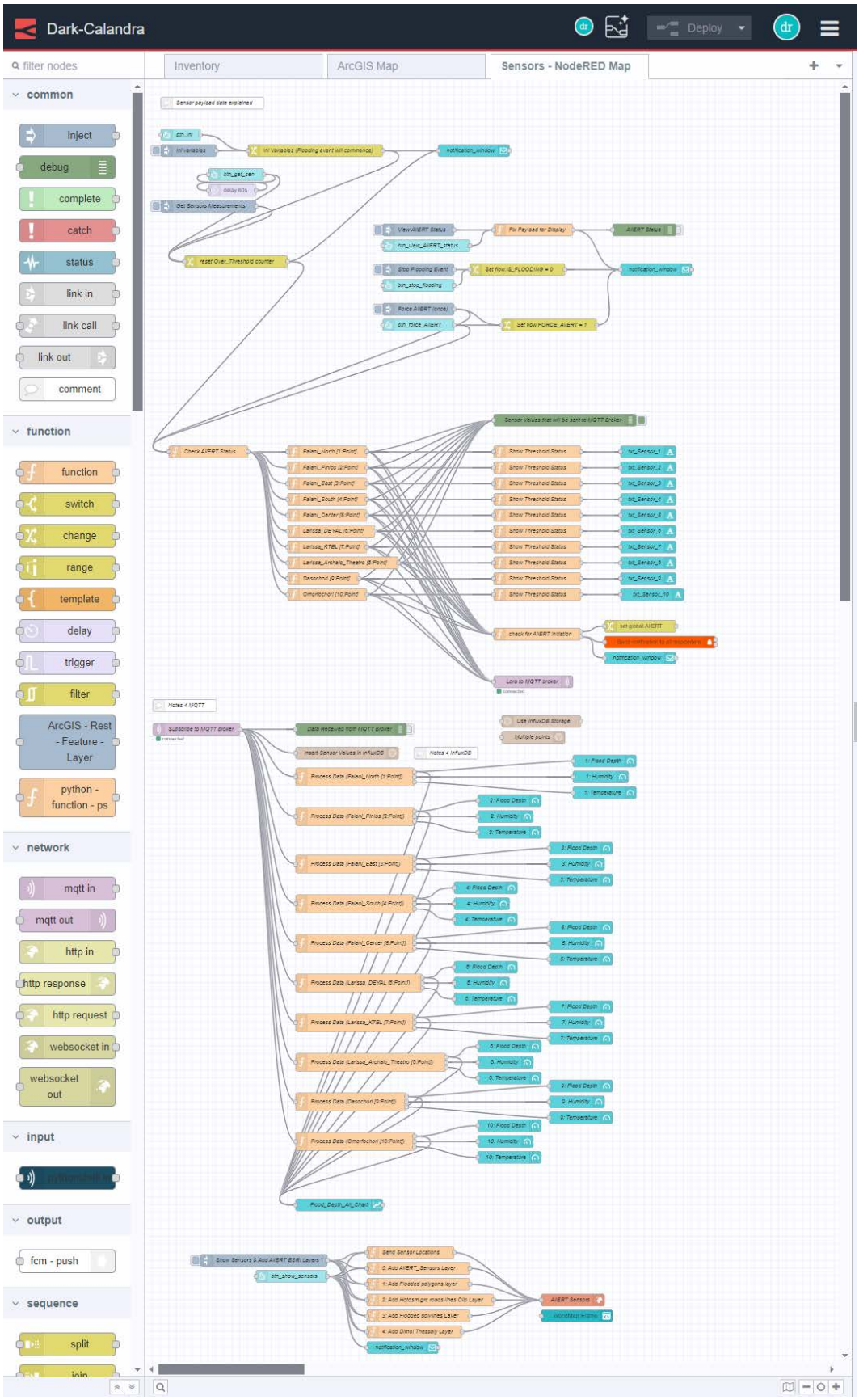


Figure 52: Sensors - Node-RED Map Flow

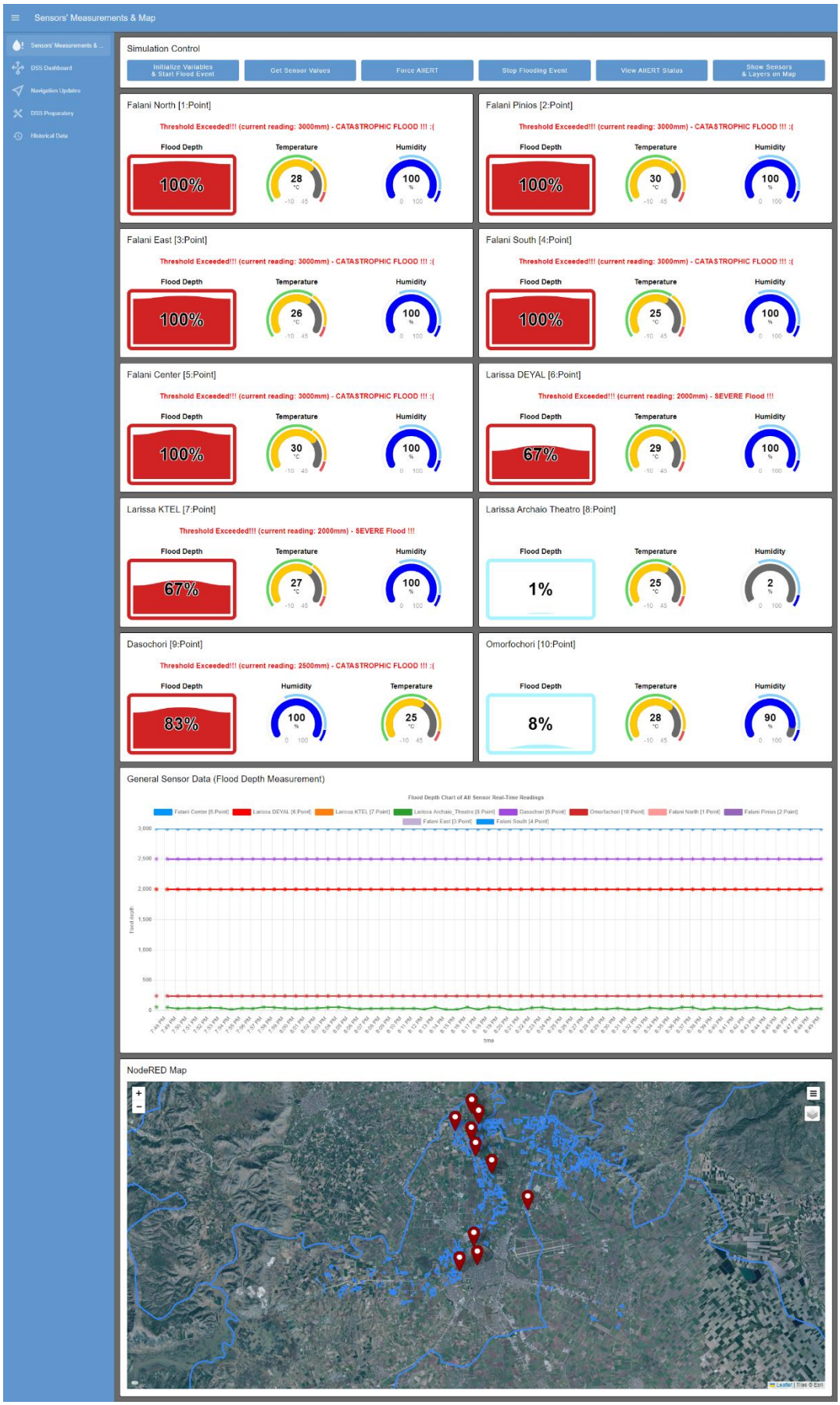


Figure 53: Sensors' Measurements & Map Page in Dashboard (flood event)

Dashboard Group Menu: "DSS Dashboard"

Every time a sensor sends a *flood depth* value above a defined threshold, a further evaluation takes place. Its location is taken into account along with the severity level category in which the collected value falls into. Thus, the location's flood severity is calculated and is compared to the previously known severity level. If a rise is detected, a specially crafted message is forwarded to the mind of the AllERT DSS system, the DSS Dashboard page.

This is when a Machine Learning model takes over and decides the type and quantities of amenities plus the response personnel that should be allocated to the said location. This ML model has undergone rigorous training and by utilizing the Random Forest Regression supervised learning algorithm, has the capacity to properly and almost instantly predict and decide on the number of resources needed to be dispatched to the affected location. Since resources are generally limited, sending the ideal number of them is of utmost importance as through the use of AI the risk of over or underspending is greatly minimized. The DSS Team has at its disposal the predicted list of resources and can accept or reject each and every one of them. Those accepted are automatically transferred to the location's table of resources which is monitored by the Communication Team and at the same time its values are subtracted from the Resource Inventory.

The Resource Inventory is a fully detailed table that contains every single resource, be it material or personnel, along with their initial and current quantity balance, that has been characterized as available for immediate allocation within the boundaries of the region. This inventory may be uploaded and imported to the DSS Dashboard through a comma separated values (.csv) file. In the table below, the content of the "inventory_initial.csv" file used here:

Resource	***Initial Quantity***	***Remaining Quantity***
Sandbags	*100000*	*100000*
Water Pumps	*500*	*500*
Food Supplies	*30000*	*30000*
Medical Kits	*10000*	*10000*
Life Vests	*5000*	*5000*
Clothing	*8000*	*8000*
Boots	*15000*	*15000*

Drinking Water	*50000*	*50000*
Living Tents	*2000*	*2000*
Heavy Duty Gloves	*10000*	*10000*
Antiseptics	*8000*	*8000*
Medicines & Antibiotics	*5000*	*5000*
Firefighters	*300*	*300*
Paramedics	*150*	*150*
Police	*500*	*500*
Rescue Divers	*100*	*100*
Engineers	*200*	*200*
Search & Rescue	*1000*	*1000*
Army	*500*	*500*
Truck	*50*	*50*
Ambulance	*10*	*10*
Helicopter	*6*	*6*
Rescue Boat	*500*	*500*
Van	*30*	*30*
Backhoe Loader	*20*	*20*
Crawler Loader	*20*	*20*

Table 10: The CSV file uploaded to the DSS as the Inventory of Resources

During the flood event, approved decisions update the resource table with the remaining quantities of available resources. Figure 54 is a snapshot of this functionality.

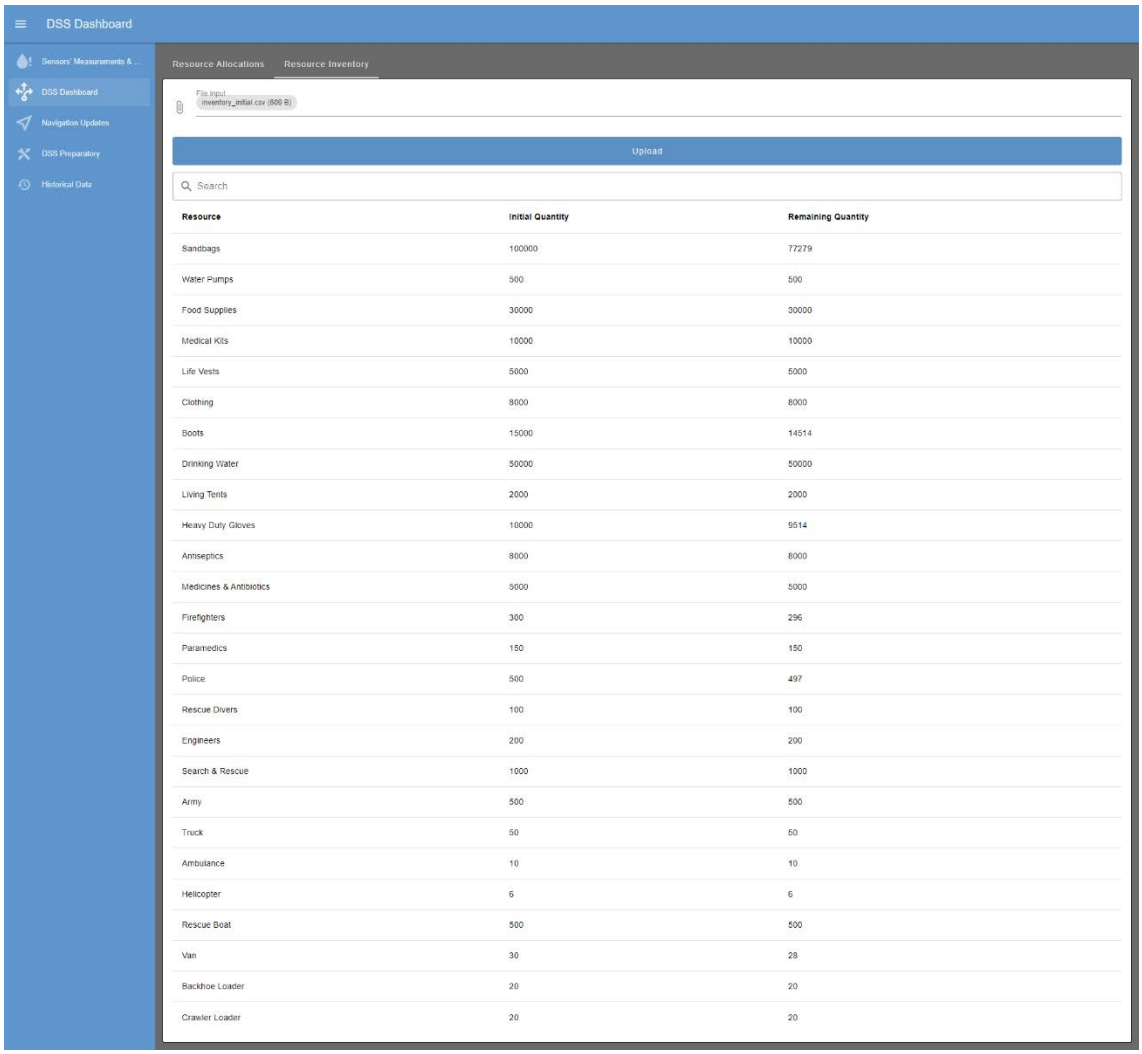


Figure 54: Values are subtracted from the Resource Inventory

The DSS dashboard presented earlier encompasses several functions within. In particular, it includes the Resource Inventory management based on which the allocation of available resources is carried out according to the decisions approved for implementation. The ML model responsible for predicting the necessary resource requirements per affected area, is trained and utilized in the background. This is carried out using appropriate Python scripts. Figure 55 illustrates the Node-RED flow that implements these capabilities.

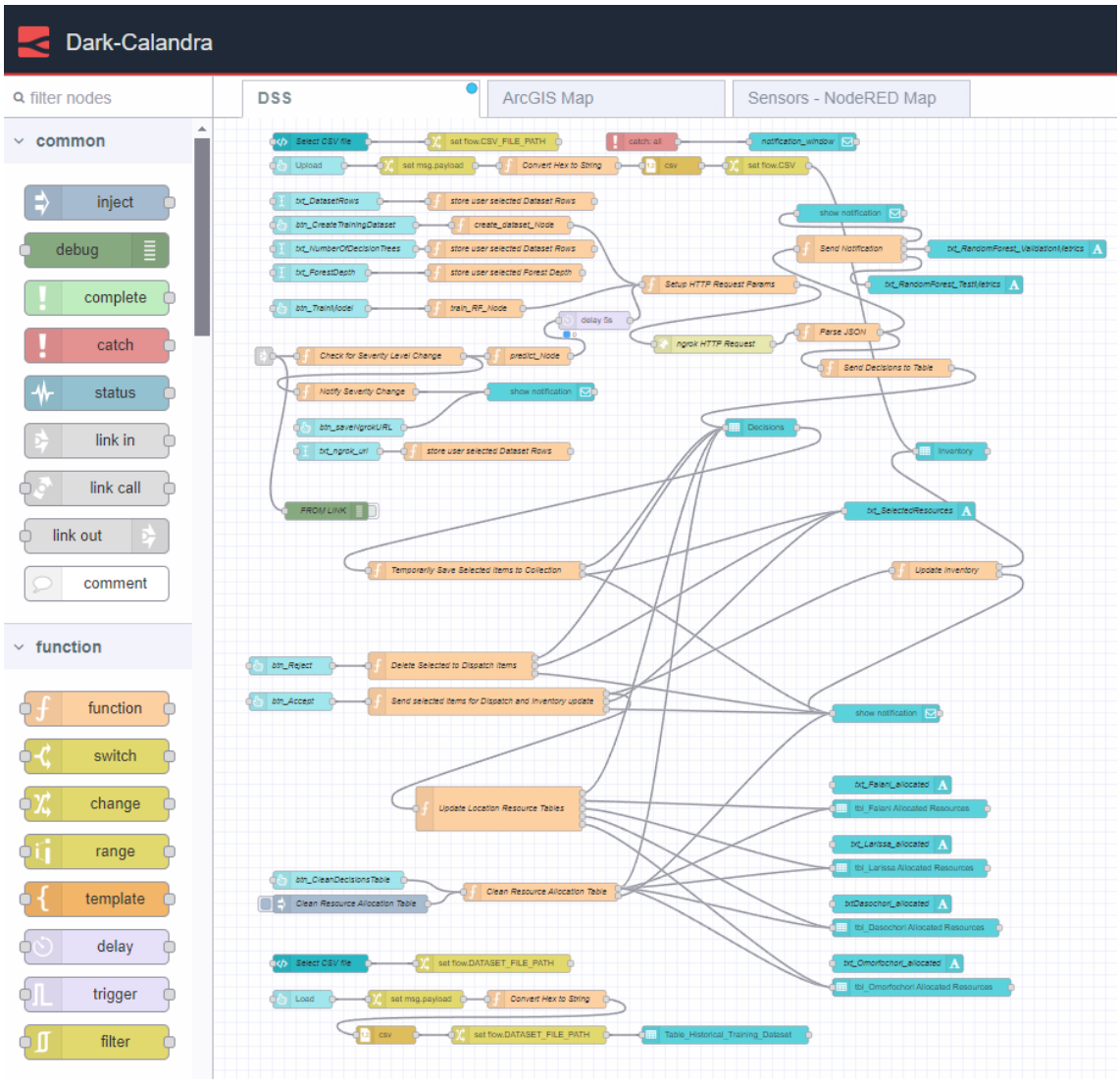


Figure 55: DSS Flow

Dashboard Group Menu: "Navigation Updates"

Once the resources for allocation are accepted on the previous page by the DSS Team, a proper, effective (fast) and above all accessible route must be selected for those amenities and responders to be dispatched to their destination. This is the purpose of the *Navigation Updates* page.

A detailed map is present here, similar to the one in the *Sensors' Measurements & Map* page as it too depicts the sensors, region boundaries, road network and current flooded areas. However, this map directly derives from ArcGIS Online thus offering route calculation capabilities that avoid those flooded areas or suggest a safe path as close as possible to the target location. Two screenshots (Route before flood & Route after flood) are also included in the page, taken from a use case scenario in order to demonstrate this

calculation's effectiveness. This is of crucial importance as resources reach their destination in both a rapid and safe manner thus completing an extremely effective and efficient circle of response operation from the moment of flood detection up to the point of resources reaching the affected area and population. Figure 56 shows the corresponding Node-RED flow.

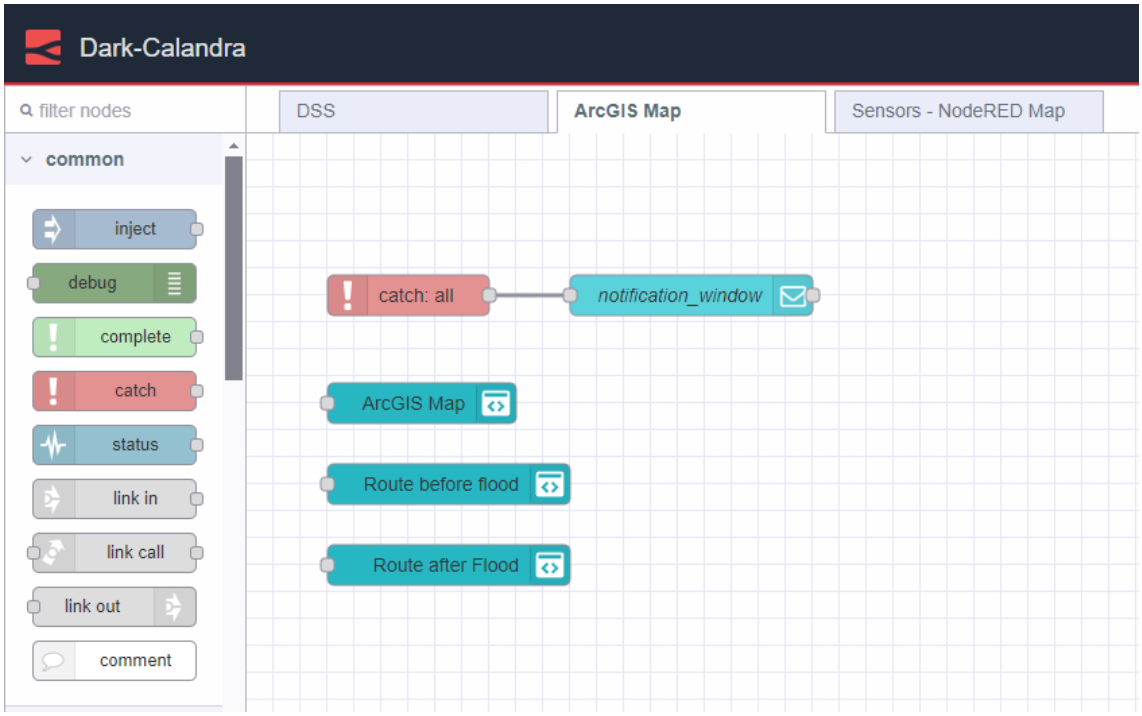


Figure 56: ArcGIS Map Flow

Figures 57, 58, and 59 represent the dashboard's view of the above flow.

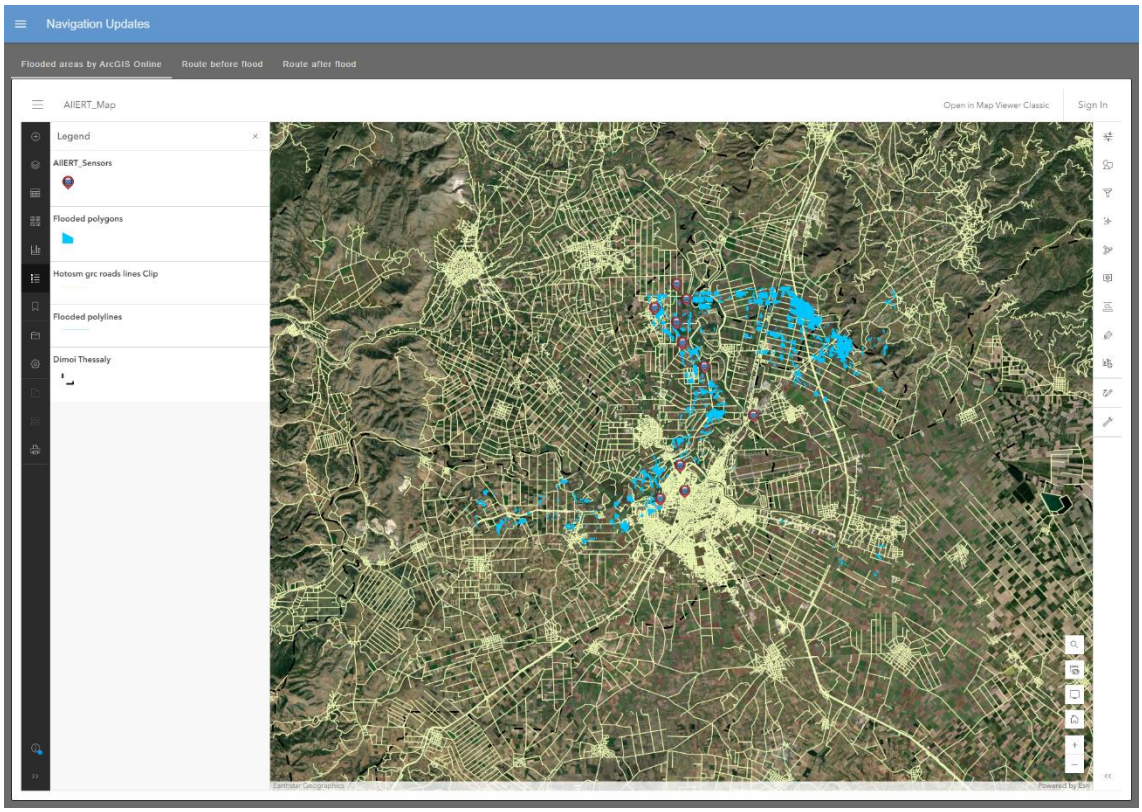


Figure 57: Tab 1 - The Map embedded directly from ArcGIS Online

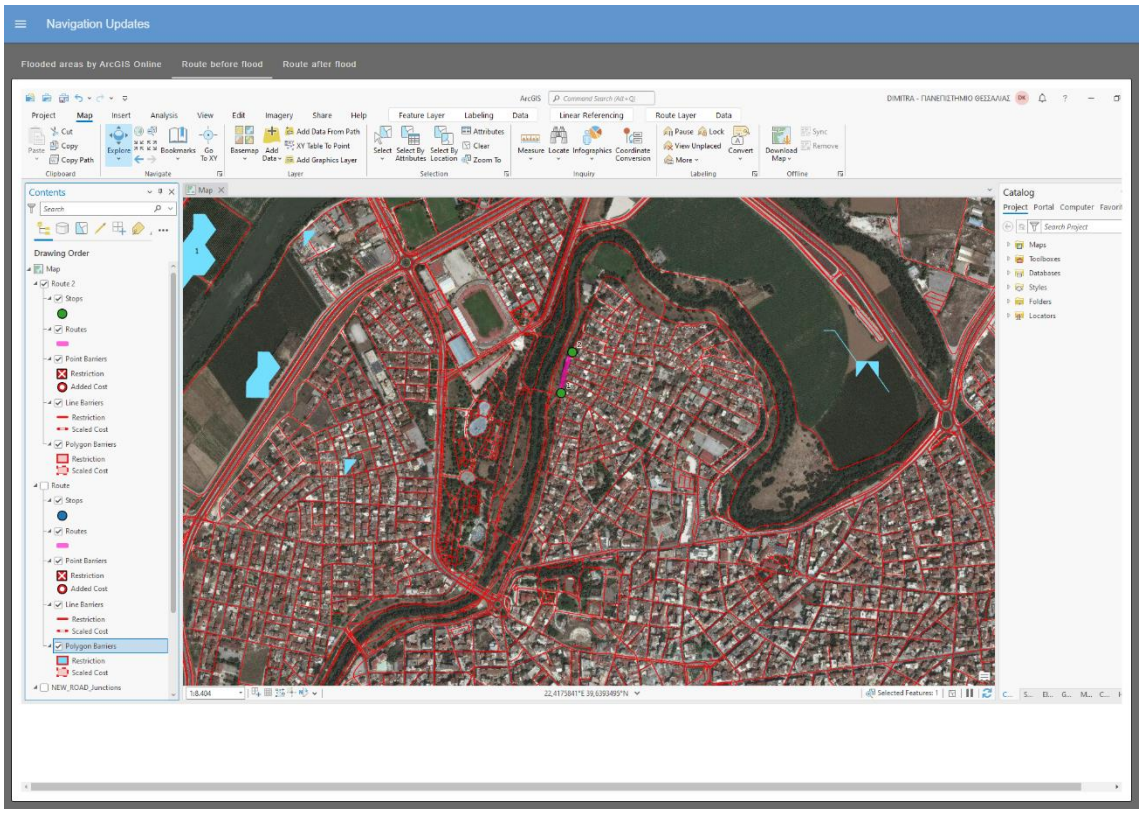


Figure 58: Route before flood in Tab 2

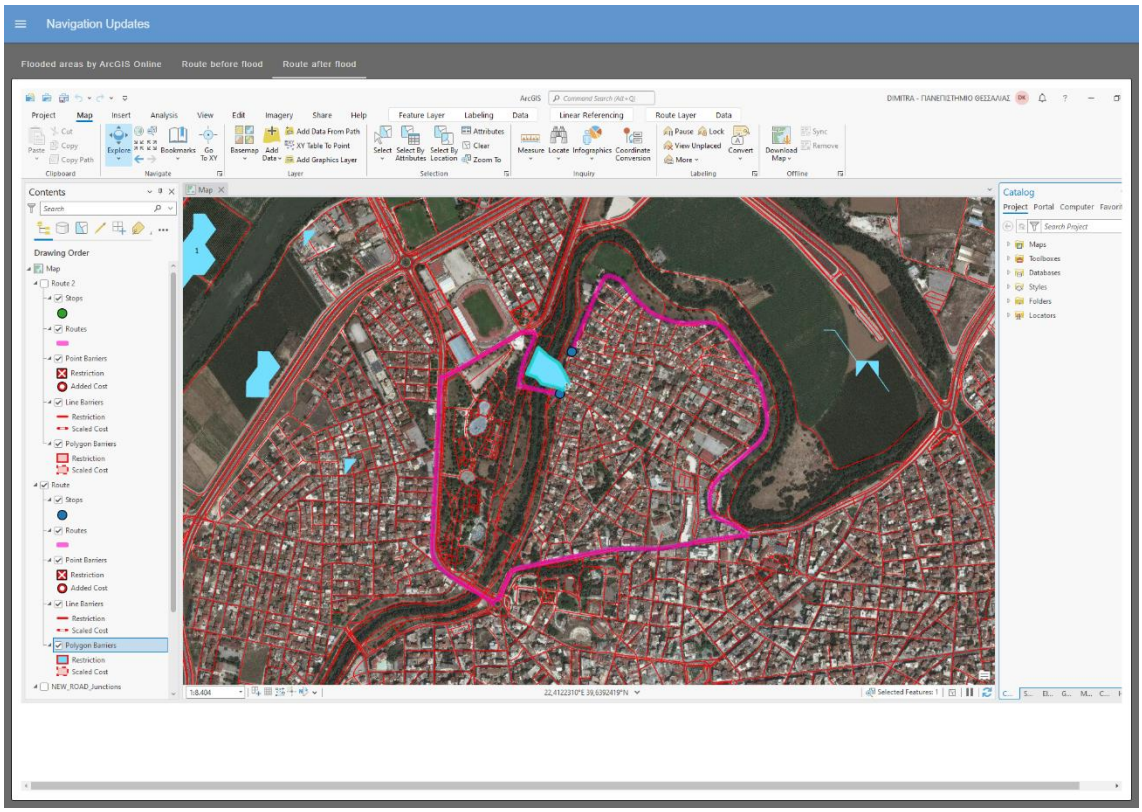


Figure 59: Route after flood in Tab 3

Dashboard Group Menus: "*DSS Preparatory*" & "*Historical Data*"

The *AllERT* Dashboard is complemented by two more pages. The *DSS Preparatory* page (Figure 61) offers configuration capabilities such as the ngrok (described in *4.3 The DSS implementation*) forwarding Uniform Resource Locator (URL) as well as parameters for the ML model training procedure. Even though the Regression model has already been fine-tuned, an AI-specialist operator is able to decide on the number of training data rows, Random Forest decision trees and overall forest depth in an attempt to re-train the model. The training evaluation metrics are also presented on this page; a proof of the AI model's effectiveness. Finally, a table of historical flood data offers the user an insight into the area's flooding past which may offer hints over the anticipated event's development.

Similar data are also available in per-location charts on the final page of the *AllERT* Dashboard, named *Historical Data* (Figure 60). These charts illustrate the sensors past *flood depth* values in a more historical approach.

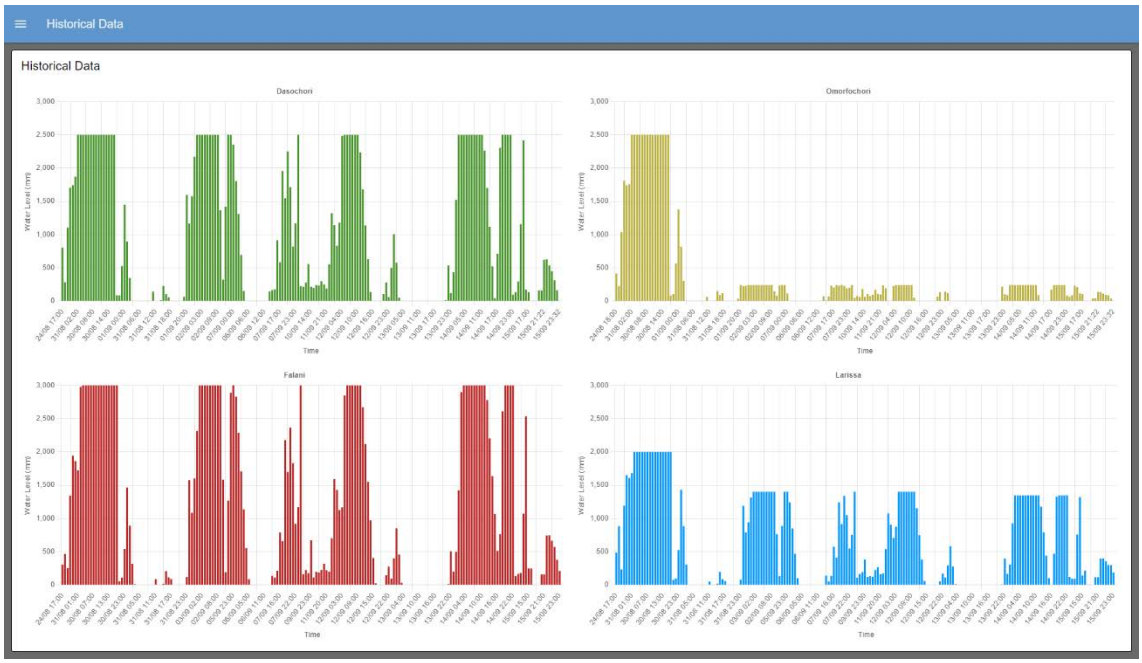


Figure 60: Historical Data imported from InfluxDB

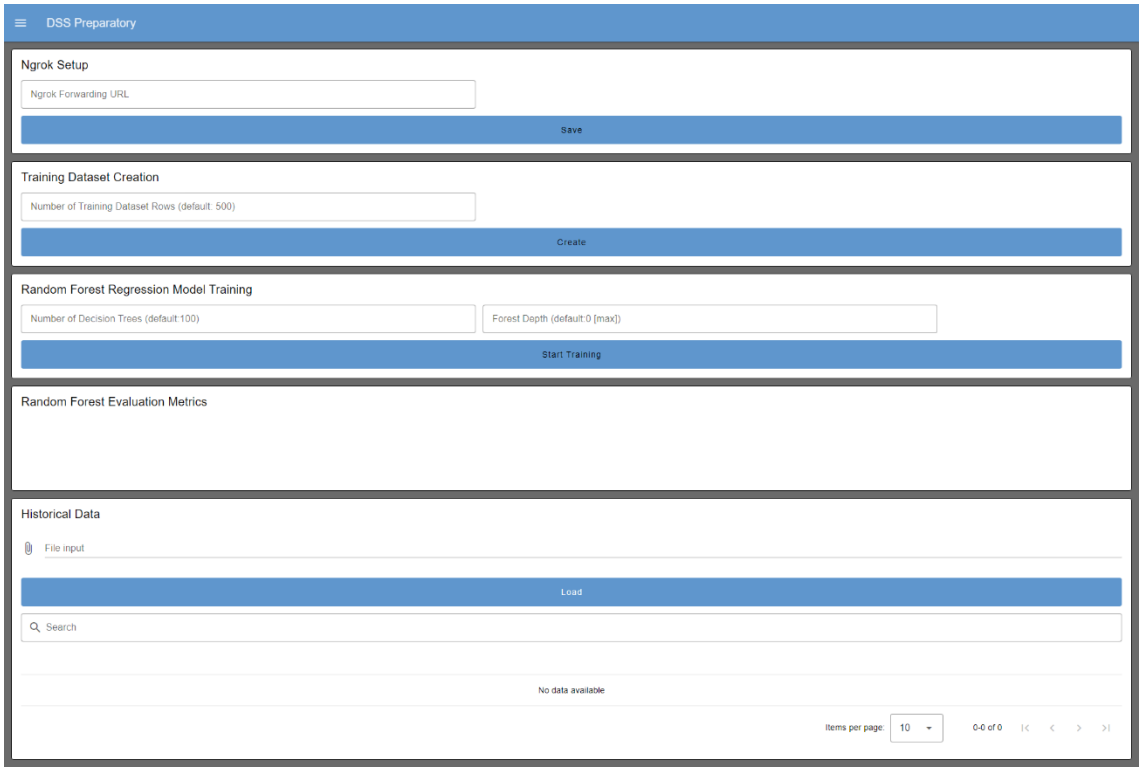


Figure 61: DSS Preparatory page

The complete flow configuration is provided in compact JavaScript Object Notation (JSON) format, available for download in the accompanying GitHub repository. The flow can be imported directly into Node-RED for replication of the system.

4.3 The DSS implementation

Particular research was carried out to select the appropriate machine learning algorithm based on which resource allocation decisions are given to support the affected population. First, the problem had to be identified regarding its type.

4.3.1 ML Approach

The relationship between flood severity and the required resources is complex and likely to be non-linear. Small increases in water level may not require many additional resources, but once a certain threshold is reached, the need for resources could increase exponentially. For example, a water level above three meters may require many more resources for people evacuating from higher-risk areas. Other factors such as the population density, interact in non-linear ways to influence the resource needs. Accordingly, this is a non-linear problem and cannot be captured by a simple linear model. Given the nature of the above relationship between input variables and the predicted variables as well as that the outputs (labels) are continuous, applying a non-linear regression algorithm serves best. The model, must learn this relationship from the labeled training data and then use that trained model to make predictions on new, unseen data [172, 290, 291].

Supervised learning is designed for cases where the objective is to map input features to output labels based on past examples. It will be effective in learning patterns from historical flood data and the trained model can continue to be improved as new labeled data becomes available. In this context, supervised learning was selected for this project because it involves predicting specific outcomes (the type and quantity of resources) based on known input variables (flood severity and location) [292, 293].

The Random Forest Regression algorithm was applied since it is ideal when the data contains multiple features or inputs and there is a need for accurate predictions that are more resistant to overfitting (too specific to the training data). Unlike linear models, Random Forest can handle large datasets and does not assume a linear relationship between the variables, allowing it to model complex interactions between multiple features. After the training process, the model is evaluated for its performance measuring the accuracy and precision on new data points, by calculating relevant metrics [294–297].

Having established the logic that led to the selection of a Random Forest Regressor model over other machine learning algorithms, it is now essential to detail the specific parameters and methodologies applied to the Use Case. Following are the key variables involved (target and independent), the dataset characteristics, the step-by-step approach used for training the model, and finally, the chosen evaluation metrics that are used to assess and validate the model's performance.

4.3.2 Random Forest Implementation

Dependent Variables (Targets)

By definition, the dependent variables (targets) are the columns that contain the quantities the model is trained to predict. Since the current model is focused on forecasting the number of resources (materials, personnel and vehicles) that are needed based on the input features of location and flood severity, the dependent variables are all the resource types (e.g. Sandbags, Water Pumps, Food Supplies, Firefighters, Paramedics, Engineers, Ambulance, Helicopter, Rescue Boat, etc.) as described in the Resource Inventory. Out of all the variable types, the aforementioned target (dependent) variables are continuous. This is because the model predicts quantities, team sizes and vehicle count, which are numerical values that can take any value within a certain range.

Independent Variables (Features)

Therefore, the independent variables are Location (e.g. Falani, Larissa, etc.) and Flood Severity (Minor, Moderate, Severe, Catastrophic). These variables are independent because they are the factors that influence the outcomes (the dependent variables), but they themselves are not affected by the predictions and thus the model uses them to determine how much of each resource is needed.

Dataset

Since this Use Case is a simulation of a flood scenario, and no actual historical data is provided in the form needed, the training dataset was created using a python script. When creating the training dataset, it was decided to classify the water level into four classes (Minor, Moderate, Severe, Catastrophic) in order to assign a reasonable number of resources corresponding to the severity of the flood. Otherwise, it had to be manually

constructed, an extremely time-consuming process with low diversity and insufficient number of records to properly train the model. With that in mind, the dataset used in the model training procedure is the CSV file "flood_resource_allocation_single_row.csv", produced by the *create_dataset_Node.py* (*Appendix C*) script and contains all the historical data with regards to past flooding events. Specifically, this dataset contains information about locations, flood severity levels and the corresponding resource quantities (materials, personnel and vehicles) that were required for each event. An indicative snapshot of such a generated file is shown in the following figure (only rows and columns that fit in screen):

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Location	Flood Severity	Water Level (mm)	Sandbags Quantity	Water Pumps Quantity	Food Supplies Quantity	Medical Kits Quantity	Life Vests Quantity	Clothing Quantity	Boots Quantity
2	2023-12-31	Volos	Catastrophic	2324	0	0	6450	1290	3870	2580	2580
3	2023-12-01	Almyros	Minor	338	10626	54	0	0	0	0	532
4	2023-11-01	Palamas	Severe	1673	0	0	937	469	937	937	937
5	2023-10-02	Larissa	Catastrophic	2741	0	0	6514	1303	3909	2606	2606
6	2023-09-02	Tyrnavos	Minor	273	10570	53	0	0	0	0	529
7	2023-08-03	Domokos	Minor	378	9161	46	0	0	0	0	459
8	2023-07-04	Dasochori	Moderate	953	0	69	0	137	0	0	0
9	2023-06-04	Agia	Severe	1210	0	0	929	465	929	929	929
10	2023-05-05	Elassona	Moderate	830	0	115	0	230	0	0	0
11	2023-04-05	Trikala	Minor	461	12278	62	0	0	0	0	614
12	2023-03-06	Falani	Catastrophic	2952	0	0	4549	910	2729	1820	1820
13	2023-02-04	Mouzaki	Catastrophic	2814	0	0	4729	946	2837	1892	1892
14	2023-01-05	Palamas	Moderate	882	0	94	0	188	0	0	0
15	2022-12-06	Falani	Moderate	733	0	91	0	182	0	0	0
16	2022-11-06	Larissa	Catastrophic	2642	0	0	6514	1303	3909	2606	2606
17	2022-10-07	Agia	Moderate	694	0	93	0	186	0	0	0
18	2022-09-07	Domokos	Minor	250	9161	46	0	0	0	0	459
19	2022-08-08	Trikala	Catastrophic	2215	0	0	6139	1228	3684	2456	2456
20	2022-07-09	Volos	Minor	351	12899	65	0	0	0	0	645

Figure 62: Historical Data CSV generated

Training

The model's training is accomplished through a properly crafted Python script using suitable libraries. The procedure starts by loading the CSV file containing the dataset with the historical data, using the *pandas* library. *Pandas* is also responsible for defining the dependent (target) and independent (categorical) variables as well as dropping the dataset columns that will not take part in the training operation (here the 'Date" column).

Using the *sklearn* library, and in order to check for and avoid the overfitting of the trained model and to be able to evaluate its performance over new unknown data, the training dataset is split into three parts: a Training set which consists of the 60% portion of the initial dataset volume and is used for the training of the model, a Validation set (20%) which allows for validating its predictions during its training phase, and finally a Test set (the rest 20%) which evaluates its performance over completely unseen data.

Sklearn eventually trains the Random Forest Regression model using the Training dataset part as well as a number of defined variables:

- *Number of trees (n_estimators)*
  As its name suggests, this is the number of Decision Trees in the Forest. The higher the number of trees the better the accuracy, as the model averages over more trees, but this can also come at the cost of computation time [298, 299].
- *Maximum depth of trees (max_depth)*
  This parameter dictates the number of branches each tree may have, in short how deep it can grow. Finetuning it, is vital to the model's performance, as limiting the depth prevents trees from becoming too complex and overfitting the training data while too shallow trees may underfit [300, 301].
- *Minimum samples per leaf (min_samples_leaf)*
  A parameter that sets the minimum number of samples required to form a leaf (the end of a branch in a decision tree). By increasing this value, the model can be prevented from creating leaves that fit the training data too closely, thus helping to prevent overfitting [300, 301].
- *Number of features considered for splitting (max_features)*
  This determines the number of features the model considers when splitting a node in a tree. Limiting its value can reduce overfitting and improve the model's performance, as it uses only a subset of the available features at each split [298, 300, 301].

It is noteworthy that the *n_estimators* and *max_depth* parameters can be fine-tuned by the user, whereas *min_samples_leaf* and *max_features* were left at their default values because they are generally optimized for a wide range of use cases. More specifically, the default setting for *min_samples_leaf* allows each leaf node to contain at least one sample which offers flexibility without causing significant overfitting, especially given the model's ensemble nature. As for *max_features* is concerned, the default value (square root of the number of features) introduces randomness while maintaining sufficient predictive power, a combination that prevents overfitting and ensures that the trees in the forest remain diverse.

Finally, the model's training performance is measured by proper evaluation and testing metrics, as follows.

Evaluation Metrics

Due to the nature of the Random Forest Regression as a supervised machine learning algorithm, four key metrics were considered: Mean Absolute Error (MAE), Mean Squared Error (MSE), R^2 Score and Symmetric Mean Absolute Percentage Error (SMAPE). Each one of those metrics plays a key role in assessing the model's predictive capabilities from a different point of view and when used together, they offer a well-rounded, comprehensive view of both the accuracy and reliability of the model [290, 302].

- *Mean Absolute Error (MAE)*

  MAE offers a simple, straightforward measure of how far the model's predictions are from the actual values on average. One of the reasons MAE is preferred is its linear penalization of errors, where each error contributes equally to the overall score. It is also robust to outliers since it does not square the errors, making it less sensitive to extreme values in the data. Finally, prediction errors are measured in the same units as the target variable which makes it easy and simple for their magnitude to be realized in relation to the actual values.
- *Mean Squared Error (MSE)*

  On the other hand, the MSE, which is widely used as a well-established standard in regression models, is the average of the squared differences between predicted and actual values. Due to this squaring, it penalizes larger errors which makes it an ideal metric for evaluating the model's performance when large deviations in particular are undesirable, such as in resource allocation scenarios where large prediction errors can result in significant misallocations and the over or sub supply of flooded areas in peril.
- *R^2 Score*

  The R^2 Score, also called the coefficient of determination, shows which percentage of the changes in what is been predicted are explained by the model. Its value ranges from 0 to 1, where a higher value signifies the model is better at capturing the changes in the data, thus making it better at predicting a result based on the input information. R^2 role is significant as it helps evaluate how well the model fits the data as well as how strong it is in explaining the relationship its different variables.

- *Symmetric Mean Absolute Percentage Error (SMAPE)*
  SMAPE is a percentage-based error metric that provides a relative measure of error between the predicted and actual values. Its main feature is that it normalizes the error relative to the magnitude of the actual values, making it especially useful for datasets with wide-ranging target values. Additionally, SMAPE is symmetric, meaning that it treats overestimations and underestimations equally, thus preventing bias in one direction. The latter makes it ideal for understanding how well the model's predictions perform relative to the actual values, especially when working with data of different scales.

As a result, the above metrics work in tandem in order to provide a collective evaluation of the trained model's performance by assessing its overall prediction accuracy (MAE), its sensitivity to large errors (MSE), its ability to capture variance and overall fit (R^2), and finally its scale-independent relative accuracy (SMAPE). It is important to note that the aforementioned metrics were explicitly chosen over others, since Root Mean Squared Error (RMSE) is a variation of MSE that does not offer significantly different insights and accuracy or classification metrics like F1-score do not apply, as they are suited for classification problems.

In order to conclude and deem the training of the model as successful, the metrics from the Validation and Test set are compared. So, if the model performs well on the training and validation data but the test set metrics are significantly worse this could be a sign that the model is overfitting the training data and needs fine-tuning using the appropriate parameters.

Overcoming an Unforeseen Challenge

An unforeseen functional problem that arose while trying to implement the Random Forest algorithm using python language, is that cloud-based platforms like FlowFuse, which run Node-RED as a service, face limitations in executing scripts that have dependencies on specific libraries. Because FlowFuse does not support this resolution at the time this thesis is being developed, it was decided that computational resources on a local machine should be used. Another common challenge arose when a Node-RED instance hosted in the cloud needed to trigger the execution of Python scripts on this local machine, which is not accessible through standard networking methods.

A workaround that integrates a local Python environment was selected, with the cloud-based Node-RED instance using Flask, a lightweight web framework [303] and Ngrok, a tunneling service that enables secure, temporary public access to local servers [304].

Flask procedure

The steps to create a simple web server that listens for incoming http post requests and executes Python scripts based on the request payload are listed below:

1. Installation of Flask on the local machine through command prompt:

```
pip install flask
```

2. Creating the Python file, "*server.py*" (provided in *Appendix C*). The script is a simple Flask-based server application for executing Python scripts securely within a specified directory. The server accepts a JSON payload from the client (here a function node within Node-RED flow DSS) containing the script name and relevant parameters. It supports running the three specific scripts described in ***4.3.3 Random Forest Python scripts description***(*create_dataset_Node.py*, *train_RF_Node.py* and *predict_Node.py*), each with its own set of required parameters. Specifically, *predict_Node.py* requires a location, severity and flood depth, *create_dataset_Node.py* needs a dataset size and *train_RF_Node.py* expects the number of decision trees and the forest depth. The server validates the input, builds the appropriate command and then runs the script. It captures and returns the output or error message, depending on the execution result. If the script or required parameters are missing, the server returns corresponding error responses.

3. Running the Flask server:

```
python server.py
```

Now Flask server listens on port 5000 and executes Python scripts based on the script_name received in the post request.

Ngrok procedure

Finally, Ngrok provides a public URL that allows the cloud-based Node-RED instance to access the local Flask server. The procedure is described below:

1. Downloading and installation through command prompt:

```
choco install ngrok
```

2. Installation of the authtoken and connection of the ngrok agent to the account:

```
ngrok config add-authtoken <TOKEN>
```

3. Starting of Ngrok to tunnel traffic to the Flask server:

```
ngrok http 5000
```

Ngrok now provides the URL that forwards Hypertext Transfer Protocol (HTTP) requests to the local Flask server on port 5000. A permanent URL is also available by the use of a static domain and can be claimed by the dashboard of Ngrok platform. Following this case for the specific created account, the command to start Ngrok tunelling becomes:

```
ngrok http --domain=emerging-living-chimp.ngrok-free.app 5000
```

In the FlowFuse Node-RED instance, an “http request” node is configured to send a post request to the public URL generated by Ngrok and triggers the execution of Python scripts on the local machine.

Once the flow is deployed in Node-RED, users can trigger it via the inject node. Each execution will send a request to the Flask server via Ngrok, passing the script name in the payload. The Flask server will validate the request and execute the specified Python script locally.

This solution allows the cloud-hosted Node-RED instance to trigger Python scripts on a local machine using Flask (Figure 63) and Ngrok (Figure 64), bridging the gap between cloud and local resources. This method is only suitable for this use case in which local processing or resource-intensive tasks must be offloaded to a machine which is not directly accessible from the cloud. While effective for demonstration and testing purposes, more secure solutions would be implemented in a real-world environment to ensure the safety of the system.

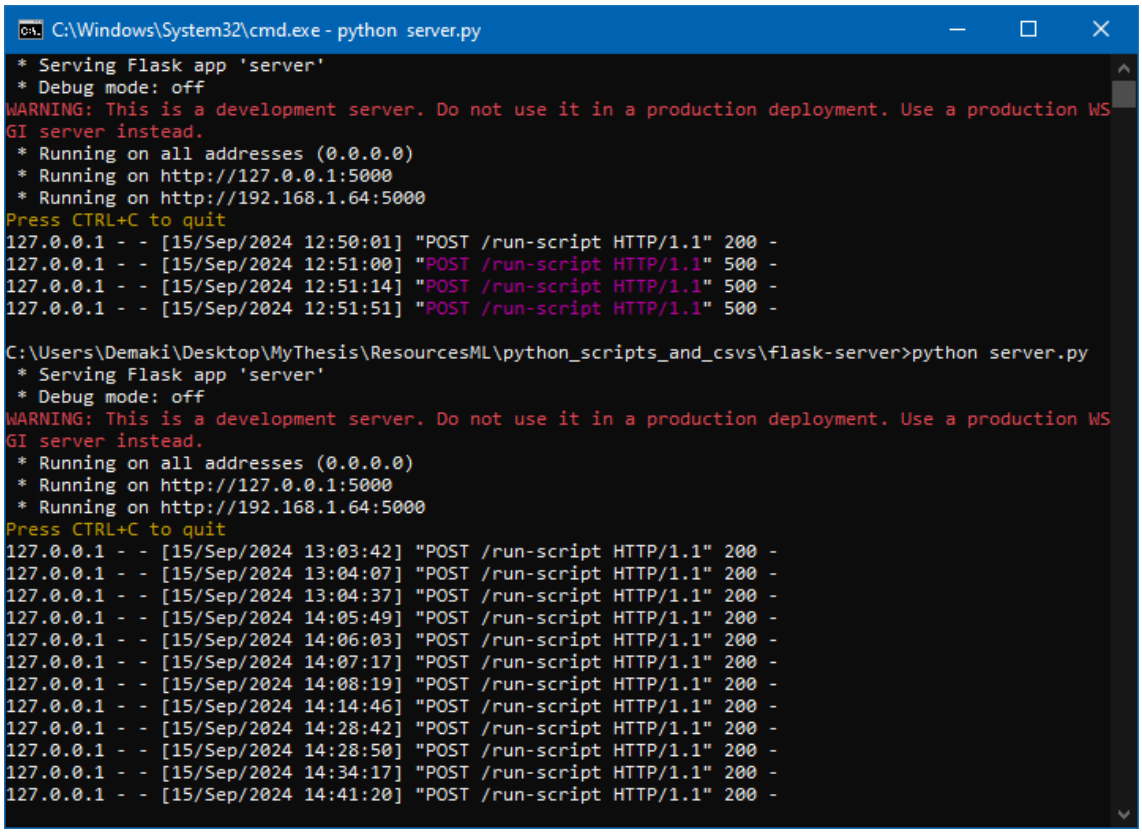


Figure 63: Flask Server Running

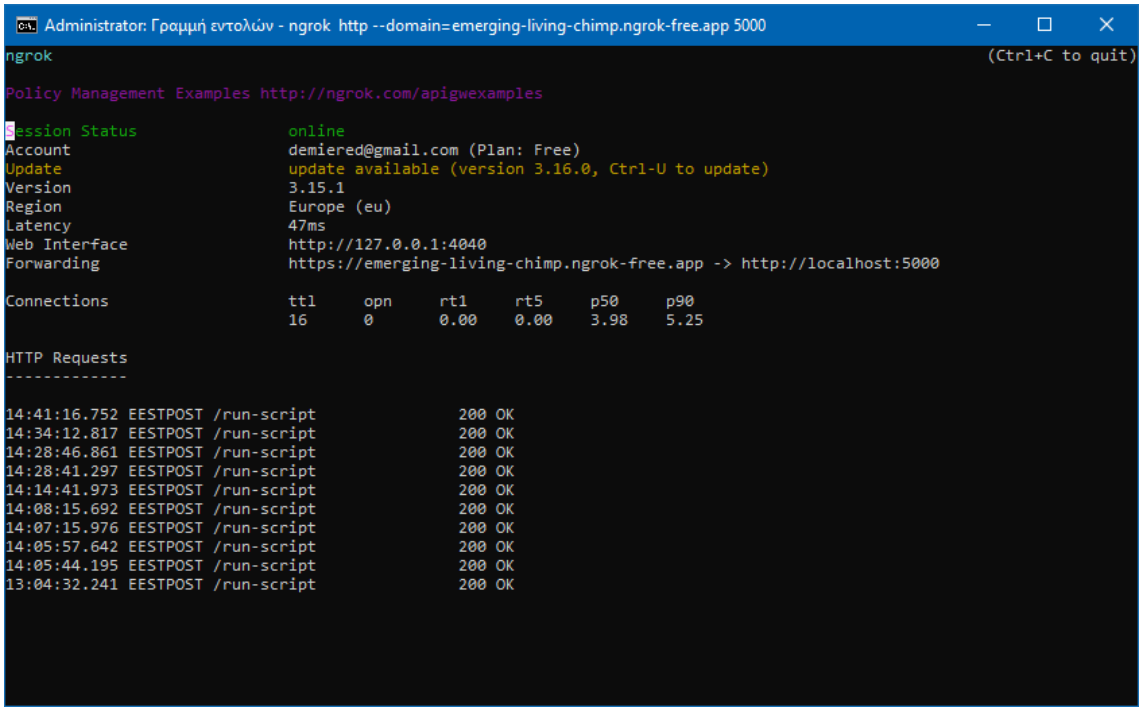


Figure 64: Ngrok Tunelling

4.3.3 Random Forest Python scripts description

- *create_dataset_Node.py*

  This script generates a synthetic dataset for flood resource allocation based on various flood severity levels, locations and required resources. It begins by defining a list of locations with their respective populations, severity levels of flood, material and human resources and vehicles that might be essential. For each severity level, it specifies the type and quantity of resources needed, scaling these based on the population of the affected location. The script uses command-line arguments to determine how many rows of data to generate. It simulates flood events by selecting a past date, location and severity, calculates the necessary resources and outputs the dataset in CSV format. The final dataset includes resource allocation details such as water levels, quantities of materials, team sizes for human resources and vehicle count. The resulting CSV file is stored in the same directory as the script and the path to the file is returned as a JSON object intended for the corresponding function node in Node-RED.

- *train_RF_Node.py*

  The train_RF_Node.py script is designed to train a Random Forest Regressor model for predicting flood resource allocation. It begins by accepting command-line arguments that specify the number of decision trees and the maximum depth of the trees in the Random Forest model. The script then loads the dataset of flood events from the pre-mentioned CSV file. After encoding the categorical variables (location and flood severity) and splitting the data into Training, Validation and Test sets, it trains the model using the specified parameters. It then evaluates the model's performance on both the Validation and Test sets using the metrics already discussed (MAE, MSE, R^2 and SMAPE). Finally, the trained model is saved to a .pkl[12] file (random_forest_flood_resource_allocation_model.pkl) and the evaluation metrics are output as a JSON object for the prediction process.

- *predict_Node.py*

  This is the essence of the *AllERT* DSS, designed to predict the necessary resources during the Use Case flood event based on a location, flood severity level and flood

[12] A .pkl file provides an efficient way to save and load Python objects, especially in the context of large and complex objects like machine learning models [305].

depth. It loads the previously trained Random Forest model from the .pkl file, and uses the input parameters (location, severity and flood depth) to generate predictions. The input data is formatted into a DataFrame, reindexed to match the model's feature structure, and passed to the trained model. The model then predicts quantities of various resources such as sandbags, water pumps, human teams and vehicles, and only outputs resources where the predicted quantity is greater than zero. The results are finally returned as a JSON object containing the location, predicted resources and their respective quantities.

4.4 Results

Analyzing the heart of the system that dictated the results of the entire synthesis, it seems that machine learning is the solution to such a complex system on the basis of which critical decisions can be made.

After multiple fine-tuning efforts the Random Forest Regression model was successfully trained, and the following metrics as shown in Figure 65 were extracted by its training and evaluation procedure.

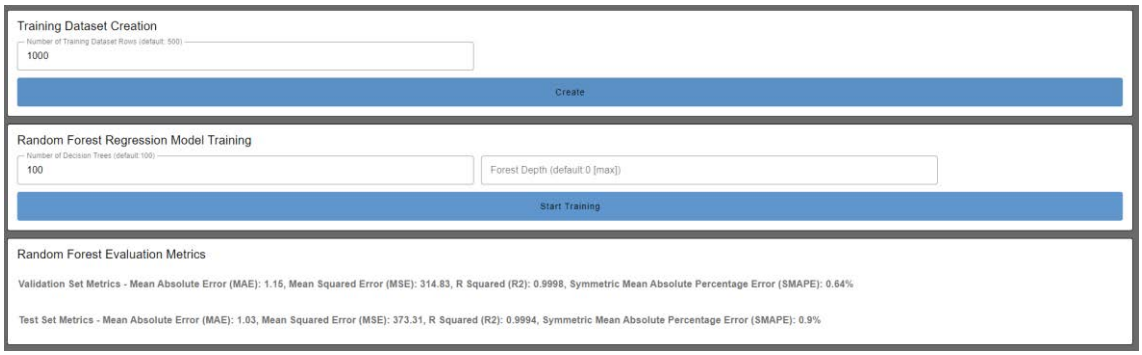


Figure 65: The model training process

Validation Set Metrics

Mean Absolute Error (MAE): 1,15

The model's predictions are off target by only 1,15 units which is extremely low and indicates high predictive accuracy.

Mean Squared Error (MSE): 314,83

Squared errors are also small, showcasing that the model avoids large prediction mistakes.

R Squared (R2): 0,9998

Variance in validation data is explained by 99,98% which means that the model's predictions are very close to the actual values, and the outcomes can be predicted with high accuracy based on the input data.

Symmetric Mean Absolute Percentage Error (SMAPE): 0,64%

A low percentage error denotes a fine performance across different ranges of values.

Test Set Metrics

Mean Absolute Error (MAE): 1,03

A slightly lower than the validation's set value, showing that the model generalizes well to new data.

Mean Squared Error (MSE): 373,31

Even though it is just a tad higher than the one on the validation set, it is still a small error value.

R Squared (R2): 0,9994

A strong fit is also demonstrated in the test data as the model is capable of accurately explaining 99,94% of new, unseen data.

Symmetric Mean Absolute Percentage Error (SMAPE): 0,9%

Still a low percentage, confirming good performance across different scales.

The consistency across both the validation and test sets metrics suggests high predictive accuracy and no signs of overfitting. The low errors and near-perfect R^2 indicate that the model is not only reliable but also effective in predicting the target values, traits that ultimately makes it a strong candidate for deployment in real-world scenarios.

For a configuration of a dataset containing 1000 rows, 100 decision trees and no depth limit, the model showed strong performance, although to further refine the model, experimenting with different settings might prove even better results.

4.5 System Performance

This section evaluates the system's performance in relation to the research questions and objectives set out in the study. The framework was designed to optimize real-time decision-making during flood emergencies, particularly focusing on resource allocation, communication and response coordination. The performance was measured based on

how accurately and efficiently the system predicted the necessary resources (personnel, vehicles, medical supplies etc.) based on flood severity data. The system effectively processed real-time sensor and UAV data, integrated with machine learning model training (Random Forest Regression), reducing the response time.

In terms of deployment, the system was able to be operational within a very short time of the emergency alert, and the response setup was quick. This was necessary to ensure timely decision-making during this critical event. Once operational, the system's decision-making process was streamlined, producing actionable outputs promptly after receiving corresponding input data from sensors. This speed was of paramount importance in this high-pressure situation, where every minute impacted the outcome of the flood response. The accuracy of predictions, measured through appropriate evaluation metrics, demonstrated that the system successfully met the study's goals.

The system proved to be not only highly responsive but also scalable, with the potential to be adapted across different geographic regions.

Decision-making is simplified and follows a well-defined sequence of actions, a direction that characterizes and differentiates it from more fixed solutions, making it a flexible and future-proof approach to disaster management.

4.6 Contribution

The research presented in this thesis offers several significant contributions to the field of flood disaster management. It is a practical solution that enhances the ability to respond quickly and efficiently to flood disasters, while also offering a methodological framework that can be adapted to other disaster types or further expanded in future research. Therefore, contributions are categorized as both practical and methodological [306], addressing the challenges of integrating real-time data for effective emergency response.

Practical Contributions

- Rapid assessment of the extent of the disaster through advanced image analysis tools and simultaneous map-based imaging combined with sensor data to enable a targeted approach.
- Integration of an ML model trained through historical data, which helps the DSS accurately predict the type and quantity of resources (e.g. human, material, vehicle etc.) needed based on flood severity data.

- The system is built using modular components that allow for scalability and adaptability across different types of disasters, offering potential for future use in other emergency scenarios beyond flooding.
- Uninterrupted communication and interoperability between all different field responders by utilizing alternative methods such as satellite internet, satellite phones and TETRA radios, ensure that all stakeholders remain connected and can operate efficiently, even in the most challenging conditions.

Methodological Contributions

Valuable methodological advancements in flood disaster management are presented, particularly through the integration of IoT real-time data under the structure of a procedural protocol that eliminates the risk of delayed responses. Unlike many existing frameworks, the system designed here leverages several data sources to provide immediate, actionable insights, ensuring that no critical time is lost in disaster response operations. Specifically:

- One of the primary methodological contributions is the real-time processing of IoT data across the entire system. Many existing disaster response frameworks collect data from sensors but lack the capability to process and act on that data immediately. This research introduces a system that continuously ingests and processes data from several sources, feeding a DSS that employs a ML model for on-the-fly decision-making. Such an integration ensures that decisions regarding resource allocation, responder deployment and safe routes are based on up-to-the-minute information, offering a significant improvement over systems that rely on static or delayed data processing. This approach represents a novel use of technology where the entire response chain (from data collection to decision-making) operates in real-time, enabling faster and more accurate reactions to dynamic flood conditions.
- A key methodological contribution is the design of a procedural protocol under which no option of missing precious time is allowed. The protocol follows a structured step-by-step approach that automates the initiation of disaster response activities as soon as a flood alert is triggered. This ensures that the system moves through predefined, efficient workflows without human-induced delays. The protocol's design guarantees that each phase of the response from data gathering, route planning, resource allocation and team dispatch operates seamlessly, leaving no room for manual delays or miscommunication. In contrast to traditional disaster response frameworks, where

decision-making may be subject to delays due to manual data interpretation or communication bottlenecks, here, instantaneous actions based on real-time insights are ensured. The method contributes to disaster response by preventing response lags and making sure that emergency teams are immediately dispatched to the highest-priority areas, guided by continuously updated data.

Table 11 summarizes the aforementioned contributions along with the corresponding tools used.

Table of Contributions

Parameter	***Contribution***	***Contribution Type***	***Tools/Technologies Used***
Rapid Disaster Assessment	*Rapid disaster assessment using image analysis and sensor data for targeted responses*	*Practical*	*MATLAB, UAVs, IoT sensors, Radars, ArcGIS, GIS platforms*
ML-Based Resource Allocation	*ML model predicts the type and quantity of resources needed based on flood severity*	*Practical*	*Python, Random Forest Regression, CSV-based resource data*
Scalable and Modular System	*System built with modular components for scalability and adaptability in different disaster scenarios*	*Practical*	*Node-RED, Python, Modular IoT devices*
Uninterrupted Communication and Interoperability	*Continuous communication ensuring all responders remain connected*	*Practical*	*TETRA radios, Satellite internet, Satellite phones*
Real-Time Data Integration & End-to-End Response Chain	*Real-time data processing to provide immediate, actionable insights and real-time response chain from data collection to decision-making and deployment*	*Methodological*	*LoRaWAN, MQTT, InfluxDB, Apache Kafka, UAV systems, IoT sensors, Node-RED, Machine Learning*

Parameter	***Contribution***	***Contribution Type***	***Tools/Technologies Used***
Procedural Protocol for Time-Sensitive Response	*Automated protocol ensuring no delays in disaster response workflows and deployment*	*Methodological*	*Process/Workflow Modeling*

Table 11: Table of Contributions

5. Future Work and the Path to a Better Tomorrow Paved by Volunteers

5.1 Future Work

While the implemented system successfully meets the objectives specified in this study, significant progress and enhancement still remain ahead for further research.

- Future work could explore the integration of more advanced ML algorithms, to improve the accuracy of resource allocation predictions. To push the boundaries of what this system can achieve, further exploration of AI integration is also recommended. Advanced AI models can enhance the system's predictive and decision-making capabilities, allowing it to offer proactive solutions rather than merely reactive measures.
- Expanding the system to include a wider range of sensors, such as water quality sensors or weather stations, would provide a more comprehensive view of flood conditions. This way, more accurate assessments of the situation would be provided and potentially earlier flood warnings would be allowed. Not only that, but weather stations and similar sensors could directly be connected to the National Meteorological Center so that AllERT could benefit from heavy flood-prone storms that are imminent and thus raise a yellow alert and increase the overall readiness level.
- The current system is optimized for flood management, but future work could involve expanding it to handle other types of natural disasters, such as earthquakes or wildfires, by adjusting the data inputs and decision-making algorithms.
- Future development could also explore the use of edge computing to process data closer to the source (e.g. IoT devices and UAVs), reducing latency and improving the system's responsiveness in real-time critical scenarios.
- Although this study used simulations to evaluate the system, future work should include extensive real-world field testing to validate the system's performance in actual flood conditions. This will provide more reliable data on its effectiveness and areas for improvement.
- Developing APIs and protocols that allow the system to integrate with existing emergency management systems across different regions or agencies could improve collaboration and resource sharing during large-scale disasters.

- Even though the current *AllERT* version not only operates via satellite communications, with the valid assumption that cellular networks will be non-operational under a flooding event, but also pre-emptively supports a backup communications network via TETRA, it could also benefit from other alternative backup solutions. One such solution would be the rapid on-site deployment of properly equipped drones in order to set up an ad-hoc local area network. Another, more long-term solution would be a beacon-like network architecture where each municipality would have large-distance Wi-Fi antennas installed that send or receive internet connections from its nearest neighbors and thus provide it to cover the affected area.
- It is a no-brainer that UAVs are the go-to solution especially in flood-stricken areas since they provide immediacy, agility and reduce the risk factor of the further endangerment of humans when aid is to be delivered. With that in mind, Tether Management System [307] could be utilized in order to deliver materials (e.g. life vests, food, medicines) in emergency situations until the SAR responders reach the site.
- For simulation purposes, the *AllERT* protocol is initiated when three or more water level sensors send flood depth readings above the global threshold set at 250mm. It goes without saying that in real-world scenarios the warning threshold should be defined per sensor by taking into account ground elevation that could derive from further harnessing the power of ArcGIS. Furthermore, the *AllERT* activation should be determined by a more dynamically set number of sensors that send over the threshold reading with dependance to their proximity with one another.
- Integrating real-time data from a wider array of traffic and road sensors and improving the accuracy of GIS layers, will allow for more reliable and efficient routing during flood events. Also testing these improvements in real-world disaster scenarios will be required to ensure their effectiveness in live emergency situations.
- It is undoubtful that such a critical system that deals with and makes vital decisions that can directly affect human lives absolutely must be bulletproof with regard to its information system's security. Under this scope, each and every communication channel needs to be encrypted and moreover its participating parties verified via the use of proper certificates so that data integrity and

confidentiality is ensured. The use of blockchain technology could potentially be integrated in order to further enhance data immutability.

- Sending push notifications and guidelines from the *AllERT* DSS to the mobile devices of flood-affected citizens will be nearly impossible with the cellular network out of order, so a different approach could be adopted. Given the fact that the region will already be under the umbrella coverage of the LoRaWAN network, this could be further utilized besides the sensor data communication. Municipalities could provide each household with a LoRa pager device [308, 309] specifically for emergency purposes like in the event of a flood. This can easily and swiftly substitute the role of mobile phones in the effort of informing residents and guiding them out of harm's way thus greatly enhancing coordination and SAR missions.
- In the era of the IoT and the spike of extreme weather phenomena due to climate change, such a critical system could (and should) be integrated into a Smart City's architectural plan and furthermore enhance the capabilities of its Digital Twin [310, 311].

5.2 Volunteers deserve special honor

Volunteers are a very special part of crisis response, as they show an attitude that reflects the positive spirit of a community. Devoting their time, selfless dedication and willingness to be present when others are in need, provides vital support and inspires hope and solidarity. Whether helping in search and rescue operations, distributing supplies or giving emotional support, they are the backbone of a response movement, ensuring that help reaches those who need it the most. Their input is invaluable, reminding everyone that in the face of adversity, the strength of humanity rises through collective action and compassion. During the flood caused by Storm Daniel in Thessaly, volunteers resembled angels coming from heaven. Is this not a source of hope for a better future?

> *"For a community to be whole and healthy, it must be based on people's love and concern for each other." -Millard Fuller* [312]

6. Conclusions and Insights

This system's ability to offer rapid, data-driven decisions during flood events sets it apart from traditional approaches. However, the journey does not end here. To maximize the potential of such a system, it is important to consider several strategic recommendations that could promote its effectiveness even further.

To refine and expand this system, fostering interdisciplinary collaboration is essential. Bringing together experts from fields such as engineering, data science, emergency management and psychology, the system can evolve to become more powerful, adaptive and user-friendly. This way it will be guaranteed that the system meets the technical requirements and also addresses the human factors that are critical in emergency situations.

Beyond its technical capabilities, the success of *AllERT* hinges on the engagement and preparedness of the communities it serves. It is recommended that local communities should be actively involved in understanding and interacting with the system through regular training sessions, workshops and simulations. This initiative will ensure that its development involves the participation of well-informed and proactive users, thereby reinforcing its usefulness during real flood events.

With great power comes great responsibility, particularly in terms of data collection and privacy. It is imperative to establish strict guidelines and policies to ensure that all data collected by the system is used responsibly and transparently. Protecting the rights and privacy of individuals must be a top priority, and ethical considerations should guide the system's ongoing development and deployment.

Adding to the above, the effectiveness of this system should be continuously evaluated and enhanced through feedback loops. Regular assessing its performance and obtaining perspectives from users, responders and developers, can make it adapt to new challenges and technologies. It is a dynamic approach that will ensure that *AllERT* remains at the forefront of emergency management, evolving in real-time to meet the needs of those it serves.

All these insights aim to transform *AllERT* protocol from a powerful tool into a transformative force in disaster management, bridging the gap between cutting-edge technology and human resilience.

References

1. Mohamed Shaluf, I.: Disaster types. Disaster Prev. Manag. Int. J. 16, 704–717 (2007). https://doi.org/10.1108/09653560710837019
2. Powers, M., Monson, M.J.E., Zimmerman, F.S., Einav, S., Dries, D.J.: Anthropogenic Disasters. Crit. Care Clin. 35, 647–658 (2019). https://doi.org/10.1016/j.ccc.2019.06.002
3. Jha, M.K.: Natural and Anthropogenic Disasters: An Overview. In: Jha, M.K. (ed.) Natural and Anthropogenic Disasters. pp. 1–16. Springer Netherlands, Dordrecht (2010)
4. Sholihah, Q., Kuncoro, W., Wahyuni, S., Suwandi, S.P., Feditasari, E.D.: The analysis of the causes of flood disasters and their impacts in the perspective of environmental law. IOP Conf. Ser. Earth Environ. Sci. 437, 012056 (2020). https://doi.org/10.1088/1755-1315/437/1/012056
5. Bradshaw, C.J.A., Sodhi, N.S., Peh, K.S.-H., Brook, B.W.: Global evidence that deforestation amplifies flood risk and severity in the developing world. Glob. Change Biol. 13, 2379–2395 (2007). https://doi.org/10.1111/j.1365-2486.2007.01446.x
6. Mason, V., Andrews, H., Upton, D.: The psychological impact of exposure to floods. Psychol. Health Med. 15, 61–73 (2010). https://doi.org/10.1080/13548500903483478
7. Aldardasawi, A.F.M., Eren, B.: Floods and Their Impact on the Environment. Acad. Perspect. Procedia. 4, 42–49 (2021). https://doi.org/10.33793/acperpro.04.02.24
8. Yen, B.C.: Hydraulics And Effectiveness Of Levees For Flood Control. (1995)
9. Perera, D., Seidou, O., Agnihotri, J., Mohamed Rasmy, A.W., Smakhtin, V., Coulibaly, P., Mehmood, H.: Flood Early Warning Systems: A Review Of Benefits, Challenges And Prospects. (2019)
10. Khan, S.M., Shafi, I., Butt, W.H., Diez, I. de la T., Flores, M.A.L., Galán, J.C., Ashraf, I.: A Systematic Review of Disaster Management Systems: Approaches, Challenges, and Future Directions. Land. 12, 1514 (2023). https://doi.org/10.3390/land12081514
11. Khan, S.M., Shafi, I., Butt, W.H., Diez, I. de la T., Flores, M.A.L., Galán, J.C., Ashraf, I.: A Systematic Review of Disaster Management Systems: Approaches, Challenges, and Future Directions. Land. 12, 1514 (2023). https://doi.org/10.3390/land12081514
12. Sawalha, I.H.: Evolution of modern disaster management. foresight. 25, 808–820 (2023). https://doi.org/10.1108/FS-08-2022-0093
13. Europe is not prepared for rapidly growing climate risks, https://www.eea.europa.eu/en/newsroom/news/europe-is-not-prepared-for
14. Pradhan, A.R., Laefer, D.F., Rasdorf, W.J.: Infrastructure Management Information System Framework Requirements for Disasters. J. Comput. Civ. Eng. 21, 90–101 (2007). https://doi.org/10.1061/(ASCE)0887-3801(2007)21:2(90)

15. Shah, S.A., Seker, D.Z., Hameed, S., Draheim, D.: The Rising Role of Big Data Analytics and IoT in Disaster Management: Recent Advances, Taxonomy and Prospects. IEEE Access. 7, 54595–54614 (2019). https://doi.org/10.1109/ACCESS.2019.2913340

16. Damaševičius, R., Bacanin, N., Misra, S.: From Sensors to Safety: Internet of Emergency Services (IoES) for Emergency Response and Disaster Management. J. Sens. Actuator Netw. 12, 41 (2023). https://doi.org/10.3390/jsan12030041

17. Stolpe, M.: The Internet of Things: Opportunities and Challenges for Distributed Data Analysis. ACM SIGKDD Explor. Newsl. 18, 15–34 (2016). https://doi.org/10.1145/2980765.2980768

18. Ray, P.P., Mukherjee, M., Shu, L.: Internet of Things for Disaster Management: State-of-the-Art and Prospects. IEEE Access. 5, 18818–18835 (2017). https://doi.org/10.1109/ACCESS.2017.2752174

19. Bail, R. de F., Kovaleski, J.L., da Silva, V.L., Pagani, R.N., Chiroli, D.M. de G.: Internet of things in disaster management: technologies and uses. Environ. Hazards. 20, 493–513 (2021). https://doi.org/10.1080/17477891.2020.1867493

20. Ghasemi, P., Karimian, N.: A Qualitative Study of Various Aspects of the Application of IoT in Disaster Management. In: 2020 6th International Conference on Web Research (ICWR). pp. 77–83 (2020)

21. Dube, K., Chapungu, L., Fitchett, J.M.: Meteorological and Climatic Aspects of Cyclone Idai and Kenneth. In: Nhamo, G. and Dube, K. (eds.) Cyclones in Southern Africa: Volume 2: Foundational and Fundamental Topics. pp. 19–36. Springer International Publishing, Cham (2021)

22. Koks, E.E., van Ginkel, K.C.H., van Marle, M.J.E., Lemnitzer, A.: Brief communication: Critical infrastructure impacts of the 2021 mid-July western European flood event. Nat. Hazards Earth Syst. Sci. 22, 3831–3838 (2022). https://doi.org/10.5194/nhess-22-3831-2022

23. LeComte, D.: International Weather Highlights 2022: Historic Heat in Europe and China, Catastrophic Flooding in Pakistan. Weatherwise. 76, 23–29 (2023). https://doi.org/10.1080/00431672.2023.2180977

24. Ajumobi, V., Womboh, S., Ezem, S.: Impacts of the 2022 Flooding on the Residents of Yenagoa, Bayelsa State, Nigeria. 1–6 (2023)

25. Ngcamu, B.S.: Application of the disaster management cycle and climate change: Studying flood disasters in South Africa. Soc. Sci. Humanit. Open. 8, 100657 (2023). https://doi.org/10.1016/j.ssaho.2023.100657

26. He, K., Yang, Q., Shen, X., Dimitriou, E., Mentzafou, A., Papadaki, C., Stoumboudi, M., Anagnostou, E.N.: Brief communication: Storm Daniel flood impact in Greece in 2023: mapping crop and livestock exposure from synthetic-aperture radar (SAR). Nat. Hazards Earth Syst. Sci. 24, 2375–2382 (2024). https://doi.org/10.5194/nhess-24-2375-2024

27. Storm Daniel, https://en.wikipedia.org/w/index.php?title=Storm_Daniel&oldid=1225490475, (2024)

28. Jesus, T.C., Portugal, P., Costa, D.G., Vasques, F.: Reliability and Detectability of Emergency Management Systems in Smart Cities under Common Cause Failures. Sensors. 24, 2955 (2024). https://doi.org/10.3390/s24092955

29. Ray, P.P., Mukherjee, M., Shu, L.: Internet of Things for Disaster Management: State-of-the-Art and Prospects. IEEE Access. 5, 18818–18835 (2017). https://doi.org/10.1109/ACCESS.2017.2752174

30. National Response Framework | FEMA.gov, https://www.fema.gov/emergency-managers/national-preparedness/frameworks/response

31. Adams, J.H., Southard, M., Forbes, S., Leinweber, B., Nix, D.: Effective Training Improves Disaster Response. Opflow. 48, 10–14 (2022). https://doi.org/10.1002/opfl.1632

32. Sinha, A., Kumar, P., Rana, N.P., Islam, R., Dwivedi, Y.K.: Impact of internet of things (IoT) in disaster management: a task-technology fit perspective. Ann. Oper. Res. 283, 759–794 (2019). https://doi.org/10.1007/s10479-017-2658-1

33. Eugene H. Kopp: Why Not Satellite Phones for Disaster Recovery - A Challenge to the Satellite Community. (2014). https://doi.org/DOI: 10.2514/6.2014-4244

34. Choksi, M., Zaveri, M.A., Kumar, J.S., Pandey, S.K.: Cloud-Based Real Time Data Acquisition in IoT Environment for Post Disaster Management. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–6 (2018)

35. Van Ackere, Verbeurgt, Sloover, Gautama, De Wulf, De Maeyer: A Review of the Internet of Floods: Near Real-Time Detection of a Flood Event and Its Impact. Water. 11, 2275 (2019). https://doi.org/10.3390/w11112275

36. Binti Zahir, S., Ehkan, P., Sabapathy, T., Jusoh, M., Nasrun Osman, M., Najib Yasin, M., Abdul Wahab, Y., Hambali, N.A.M., Ali, N., Bakhit, A.S., Husin, F., Kamil, M.K.Md., Jamaludin, R.: Smart IoT Flood Monitoring System. J. Phys. Conf. Ser. 1339, 012043 (2019). https://doi.org/10.1088/1742-6596/1339/1/012043

37. Bhatt, H., G, P., Shankar, S., Haralikar, S.: Wireless Sensor Networks for Optimisation of Search and Rescue Management in Floods. In: 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). pp. 1–6 (2021)

38. Singh, M., Sharma, K.: Wireless Sensor Networks for Natural Disaster Management - A Survey. In: 2023 International Conference on IoT, Communication and Automation Technology (ICICAT). pp. 1–7. IEEE, Gorakhpur, India (2023)

39. Mehmood, U., Mansoor, U., Hwang, D.Y., Kim, K.-H., Lee, T., Yoo, S.W.: Wireless Sensor Networks for integrated search and rescue efforts for disaster hit areas. In: 2012 Fourth International Conference on Ubiquitous and Future Networks (ICUFN). pp. 306–309 (2012)

40. Lohokare, J., Dani, R.: An Intelligent cloud ecosystem for disaster response and management leveraging opportunistic IoT mesh networks. In: 2021 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM). pp. 125–133 (2021)

41. Mitra, P., Ray, R., Chatterjee, R., Basu, R., Saha, P., Raha, S., Barman, R., Patra, S., Biswas, S.S., Saha, S.: Flood forecasting using Internet of things and artificial neural networks. In: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). pp. 1–5 (2016)

42. Hasan, Md.M., Rahman, Md.A., Sedigh, A., Khasanah, A.U., Taufiq Asyhari, A., Tao, H., Bakar, S.A.: Search and rescue operation in flooded areas: A survey on emerging sensor networking-enabled IoT-oriented technologies and applications. Cogn. Syst. Res. 67, 104–123 (2021). https://doi.org/10.1016/j.cogsys.2020.12.008

43. Byukusenge, P., Zhang, Y.: Life Detection Based on UAVs - Thermal Images in Search and Rescue Operation. In: 2022 IEEE 22nd International Conference on Communication Technology (ICCT). pp. 1728–1731. IEEE, Nanjing, China (2022)

44. Ingabire, W., Larijani, H., Gibson, R.M., Qureshi, A.-U.-H.: Outdoor Node Localization Using Random Neural Networks for Large-Scale Urban IoT LoRa Networks. Algorithms. 14, 307 (2021). https://doi.org/10.3390/a14110307

45. Adeel, A., Gogate, M., Farooq, S., Ieracitano, C., Dashtipour, K., Larijani, H., Hussain, A.: A Survey on the Role of Wireless Sensor Networks and IoT in Disaster Management. In: Durrani, T.S., Wang, W., and Forbes, S.M. (eds.) Geological Disaster Monitoring Based on Sensor Networks. pp. 57–66. Springer, Singapore (2019)

46. Satyanarayanan, M.: The Emergence of Edge Computing. Computer. 50, 30–39 (2017). https://doi.org/10.1109/MC.2017.9

47. Samikwa, E., Voigt, T., Eriksson, J.: Flood Prediction Using IoT and Artificial Neural Networks with Edge Computing. In: 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics). pp. 234–240. IEEE, Rhodes, Greece (2020)

48. Bemposta Rosende, S., Ghisler, S., Fernández-Andrés, J., Sánchez-Soriano, J.: Implementation of an Edge-Computing Vision System on Reduced-Board Computers Embedded in UAVs for Intelligent Traffic Management. Drones. 7, 682 (2023). https://doi.org/10.3390/drones7110682

49. Mansour, M., Gamal, A., Ahmed, A.I., Said, L.A., Elbaz, A., Herencsar, N., Soltan, A.: Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions. Energies. 16, 3465 (2023). https://doi.org/10.3390/en16083465

50. Halbgewachs, M., Angermann, L., Wieland, M., Kippnich, U., Lechner, K.: Using UAV Data to Improve the Situational Awareness for First Responders in Disaster Management: The Example of Flooding in the AHR Valley, Germany. In: IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium. pp. 934–937 (2023)

51. Sharevski, F.: Leveraging Cellular Intemet-of-Things for Resilient and Robust Disaster Management. In: 2018 IEEE Global Communications Conference (GLOBECOM). pp. 1–5 (2018)

52. Ilukkumbure, S.P.M.K.W., Samarasiri, V.Y., Mohamed, M.F., Selvaratnam, V., Samantha Rajapaksha, U.U.: Early Warning for Pre and Post Flood Risk Management by Using IoT and Machine Learning. In: 2021 3rd International Conference on Advancements in Computing (ICAC). pp. 252–257 (2021)

53. Pragna, N.V., Srinivasan, J., M, Greeshma., Sri Vishnu, A.J., Polavarapu, R., Mohanty, A.: Flood Surveillance using FPV drones. In: 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT). pp. 771–776 (2023)

54. Pseudorange, https://en.wikipedia.org/w/index.php?title=Pseudorange&oldid=1120472750, (2022)

55. Albanese, A., Sciancalepore, V., Costa-Pérez, X.: SARDO: An Automated Search-and-Rescue Drone-based Solution for Victims Localization. IEEE Trans. Mob. Comput. 21, 3312–3325 (2022). https://doi.org/10.1109/TMC.2021.3051273

56. Illahi, A.A.C., Kitane, C.S., Lee, I.C., Minguez, E.R., Señires, J.P.: A Development of an autonomous Unmanned Aerial Vehicle (UAV) that can locate and map people in flooded areas using image processing and GPS technology. In: 2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM). pp. 1–5 (2022)

57. Basir, R., Qaisar, S., Ali, M., Ahmad Chughtai, N., Ali Imran, M., Hashmi, A.: Performance Analysis of UAV-Enabled Disaster Recovery Networks. In: Autonomous Airborne Wireless Networks. pp. 157–194. IEEE (2021)

58. Alsamhi, S.H., Almalki, F.A., AL-Dois, H., Shvetsov, A.V., Ansari, M.S., Hawbani, A., Gupta, S.K., Lee, B.: Multi-Drone Edge Intelligence and SAR Smart Wearable Devices for Emergency Communication. Wirel. Commun. Mob. Comput. 2021, 1–12 (2021). https://doi.org/10.1155/2021/6710074

59. Stampa, M., Sutorma, A., Jahn, U., Willich, F., Pratzler-Wanczura, S., Thiem, J., Rohrig, C., Wolff, C.: A Scenario for a Multi-UAV Mapping and Surveillance System in Emergency Response Applications. In: 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS). pp. 1–6. IEEE, Dortmund, Germany (2020)

60. Peng, G., Xia, Y., Zhang, X., Bai, L.: UAV-Aided Networks for Emergency Communications in Areas with Unevenly Distributed Users. J. Commun. Inf. Netw. 3, 23–32 (2018). https://doi.org/10.1007/s41650-018-0034-1

61. Micheletto, M., Petrucci, V., Santos, R., Orozco, J., Mosse, D., Ochoa, S.F., Meseguer, R.: Flying Real-Time Network to Coordinate Disaster Relief Activities in Urban Areas. Sensors. 18, 1662 (2018). https://doi.org/10.3390/s18051662

62. Jayalath, K., Munasinghe, S.R.: Drone-based Autonomous Human Identification for Search and Rescue Missions in Real-time. 2021 10th Int. Conf. Inf. Autom. Sustain. ICIAfS. 518–523 (2021). https://doi.org/10.1109/ICIAfS52090.2021.9606048

63. Iyer, A., Narayan, S., M, N., Rajagopal, M. kumar: Autonomous Systems: Indoor Drone Navigation, http://arxiv.org/abs/2304.08893, (2023)

64. Mohd Daud, S.M.S., Mohd Yusof, M.Y.P., Heo, C.C., Khoo, L.S., Chainchel Singh, M.K., Mahmood, M.S., Nawawi, H.: Applications of drone in disaster management: A scoping review. Sci. Justice. 62, 30–42 (2022). https://doi.org/10.1016/j.scijus.2021.11.002

65. Telli, K., Kraa, O., Himeur, Y., Ouamane, A., Boumehraz, M., Atalla, S., Mansoor, W.: A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs). Systems. 11, 400 (2023). https://doi.org/10.3390/systems11080400

66. Wakeling, J.F., Dobbie, W.H.: Satellite Access Services. BT Technol. J. 16, 112–121 (1998). https://doi.org/10.1023/A:1009699814474

67. Pavur, J., Martinovic, I.: Building a launchpad for satellite cyber-security research: lessons from 60 years of spaceflight. J. Cybersecurity. 8, tyac008 (2022). https://doi.org/10.1093/cybsec/tyac008

68. ruge.axessnet: Satellite phone: know the 5 sectors that use them the most, https://axessnet.com/en/satellite-phone-know-the-5-sectors-that-use-them-the-most/, (2018)

69. Leo T. Danaher, M.S., Erik Wood, M.S., Tim Frazier, P.: Rescue coordination common operating picture: Enhancement through satellite technology. J. Emerg. Manag. 20, 73–90 (2022). https://doi.org/10.5055/jem.0682

70. Burguillos, C.: Emergency Communications Network for Disaster Management, DOI: http://dx.doi.org/10.5772/intechopen.85872, (2019)

71. Carreras-Coch, A., Navarro, J., Sans, C., Zaballos, A.: Communication Technologies in Emergency Situations. Electronics. 11, 1155 (2022). https://doi.org/10.3390/electronics11071155

72. Delmonteil, F.-X., Rancourt, M.-È.: The role of satellite technologies in relief logistics. J. Humanit. Logist. Supply Chain Manag. 7, 57–78 (2017). https://doi.org/10.1108/JHLSCM-07-2016-0031

73. Völk, F., Schwarz, R.T., Lorenz, M., Knopp, A.: Emergency 5G Communication on-the-Move: Concept and field trial of a mobile satellite backhaul for public protection and disaster relief. Int. J. Satell. Commun. Netw. 39, 417–430 (2021). https://doi.org/10.1002/sat.1377

74. Casoni, M., Grazia, C.A., Klapez, M., Patriciello, N., Amditis, A., Sdongos, E.: Integration of satellite and LTE for disaster recovery. IEEE Commun. Mag. 53, 47–53 (2015). https://doi.org/10.1109/MCOM.2015.7060481

75. Hu, Z., Wen, F., Yong, J., Fan, F., Wu, B., Qiu, K.: Delay performance comparison across seven Low-Earth-Orbit (LEO) satellite constellations. In: Ninth Symposium on Novel Photoelectronic Detection Technology and Applications. pp. 515–524. SPIE (2023)

76. Domb Alon, M.M., Leshem, G.: Satellite to Ground Station, Attenuation Prediction for 2.4–72 GHz Using LTSM, an Artificial Recurrent Neural Network Technology. Electronics. 11, 541 (2022). https://doi.org/10.3390/electronics11040541

77. Tirmizi, S.B.R., Chen, Y., Lakshminarayana, S., Feng, W., Khuwaja, A.A.: Hybrid Satellite–Terrestrial Networks toward 6G: Key Technologies and Open Issues. Sensors. 22, 8544 (2022). https://doi.org/10.3390/s22218544

78. Aboelala, O., Lee, I.E., Chung, G.C.: A Survey of Hybrid Free Space Optics (FSO) Communication Networks to Achieve 5G Connectivity for Backhauling. Entropy. 24, 1573 (2022). https://doi.org/10.3390/e24111573

79. R, S., Sharma, S., Vishwakarma, N., Madhukumar, A.S.: HAPS-Based Relaying for Integrated Space–Air–Ground Networks With Hybrid FSO/RF Communication: A Performance Analysis. IEEE Trans. Aerosp. Electron. Syst. 57, 1581–1599 (2021). https://doi.org/10.1109/TAES.2021.3050663

80. Kapovits, A., Corici, M.-I., Gheorghe-Pop, I.-D., Gavras, A., Burkhardt, F., Schlichter, T., Covaci, S.: Satellite communications integration with terrestrial networks. China Commun. 15, 22–38 (2018). https://doi.org/10.1109/CC.2018.8438271

81. FINDER_MK4_V4Mi.pdf

82. Finder, https://www.specopsgroup.com/finder

83. FINDER Finds Its Way into Rescuers' Toolkits | NASA Spinoff, https://spinoff.nasa.gov/FINDER-Finds-Its-Way-into-Rescuers-Toolkits

84. Spec Ops Cyclops Mark II Life Detection | Beechwood Equipment, https://www.beechwoodequipment.com/cyclops-mark-ii/, (2020)

85. Wouters, L., Couasnon, A., de Ruiter, M.C., van den Homberg, M.J.C., Teklesadik, A., de Moel, H.: Improving flood damage assessments in data-scarce areas by retrieval of building characteristics through UAV image segmentation and machine learning – a case study of the 2019 floods in southern Malawi. Nat. Hazards Earth Syst. Sci. 21, 3199–3218 (2021). https://doi.org/10.5194/nhess-21-3199-2021

86. Dhongade, A., Thorat, A., Alone, D., Sawant, S., Joshi, A.: Flood Damage Detection Using Satellite Images. In: Sugumaran, V., Upadhyay, D., and Sharma, S. (eds.) Advancements in Interdisciplinary Research. pp. 362–374. Springer Nature Switzerland, Cham (2022)

87. Crognale, M., De Iuliis, M., Rinaldi, C., Gattulli, V.: Damage detection with image processing: a comparative study. Earthq. Eng. Eng. Vib. 22, 333–345 (2023). https://doi.org/10.1007/s11803-023-2172-1

88. Shubhasree, A.V., Sankaran, P., Raghu, C.V.: UAV Image Analysis of Flooded Area Using Convolutional Neural Networks. In: 2022 International Conference on Connected Systems & Intelligence (CSI). pp. 1–7 (2022)

89. Webb, S.A., Goode, S.R.: An Astronomers Guide to Machine Learning, http://arxiv.org/abs/2304.00512, (2023)

90. Munawar, H.S., Hammad, A.W.A., Waller, S.T., Thaheem, M.J., Shrestha, A.: An Integrated Approach for Post-Disaster Flood Management Via the Use of Cutting-Edge Technologies and UAVs: A Review. Sustainability. 13, 7925 (2021). https://doi.org/10.3390/su13147925

91. Widodo, T.N., Zubair, H., Padjung, R.: Land use change study and the increased risk of floods disaster in Jeneberang watershed at Gowa Regency, South

Sulawesi, Indonesia. IOP Conf. Ser. Earth Environ. Sci. 824, 012045 (2021). https://doi.org/10.1088/1755-1315/824/1/012045

92. Unsupervised Classification - NAARM, https://naarm.org.in/VirtualLearning/vlc/erdas13.htm

93. Bhatt, C.M., Rao, G.S., Manjusree, P., Bhanumurthy, V.: REMOTE SENSING AND GIS TECHNOLOGY FOR FLOOD MONITORING AND MANAGEMENT.

94. Pech-May, F., Sanchez-Hernández, J.V., López-Gómez, L.A., Magaña-Govea, J., Mil-Chontal, E.M.: Flooded Areas Detection through SAR Images and U-NET Deep Learning Model. Comput. Sist. 27, (2023). https://doi.org/10.13053/cys-27-2-4624

95. Kapilaratne, R.G.C.J., Kaneta, S.: TOWARDS AN AUTOMATED FLOOD AREA EXTRACTION FROM HIGH RESOLUTION SATELLITE IMAGES. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. XLIII-B3-2020, 97–104 (2020). https://doi.org/10.5194/isprs-archives-XLIII-B3-2020-97-2020

96. Shinde, S., Pande, C.B., Barai, V.N., Gorantiwar, S.D., Atre, A.A.: Flood Impact and Damage Assessment Based on the Sentitnel-1 SAR Data Using Google Earth Engine. In: Pande, C.B., Moharir, K.N., Singh, S.K., Pham, Q.B., and Elbeltagi, A. (eds.) Climate Change Impacts on Natural Resources, Ecosystems and Agricultural Systems. pp. 483–502. Springer International Publishing, Cham (2023)

97. Farhang, S.H., Rezaifar, O., Sharbatdar, M.K., Ahmady Fard, A.: Evaluation of Different Methods of Machine Vision in Health Monitoring and Damage Detection of Structures. J. Rehabil. Civ. Eng. 9, 93–132 (2021). https://doi.org/10.22075/jrce.2021.21760.1452

98. Rahnemoonfar, M., Chowdhury, T., Sarkar, A., Varshney, D., Yari, M., Murphy, R.R.: FloodNet: A High Resolution Aerial Imagery Dataset for Post Flood Scene Understanding. IEEE Access. 9, 89644–89654 (2021). https://doi.org/10.1109/ACCESS.2021.3090981

99. Safavi, F., Rahnemoonfar, M.: Comparative Study of Real-Time Semantic Segmentation Networks in Aerial Images During Flooding Events. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 16, 4–20 (2023). https://doi.org/10.1109/JSTARS.2022.3219724

100. Jaccard index, https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=1196092673, (2024)

101. Intersection over Union (IoU), https://wiki.cloudfactory.com/docs/mp-wiki/metrics/iou-intersection-over-union

102. Whitehurst, D., Joshi, K., Kochersberger, K., Weeks, J.: Post-Flood Analysis for Damage and Restoration Assessment Using Drone Imagery. Remote Sens. 14, 4952 (2022). https://doi.org/10.3390/rs14194952

103. Rudner, T.G.J., Rußwurm, M., Fil, J., Pelich, R., Bischke, B., Kopačková, V., Biliński, P.: Multi3Net: Segmenting Flooded Buildings via Fusion of Multiresolution, Multisensor, and Multitemporal Satellite Imagery. Proc. AAAI

Conf. Artif. Intell. 33, 702–709 (2019). https://doi.org/10.1609/aaai.v33i01.3301702

104. Qiao, W., Ma, B., Liu, Q., Wu, X., Li, G.: Computer Vision-Based Bridge Damage Detection Using Deep Convolutional Networks with Expectation Maximum Attention Module. Sensors. 21, 824 (2021). https://doi.org/10.3390/s21030824
105. Johary, R., Révillion, C., Catry, T., Alexandre, C., Mouquet, P., Rakotoniaina, S., Pennober, G., Rakotondraompiana, S.: Detection of Large-Scale Floods Using Google Earth Engine and Google Colab. Remote Sens. 15, 5368 (2023). https://doi.org/10.3390/rs15225368
106. Hänsch, R., Arndt, J., Lunga, D., Gibb, M., Pedelose, T., Boedihardjo, A., Petrie, D., Bacastow, T.M.: SpaceNet 8 - The Detection of Flooded Roads and Buildings. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1471–1479 (2022)
107. Moel, H., Aerts, J.: Effect of uncertainty in land use, damage models and inundation depth on flood damage estimates. (2018)
108. Hocini, N., Payrastre, O., Bourgin, F., Gaume, E., Davy, P., Lague, D., Poinsignon, L., Pons, F.: Performance of automated methods for flash flood inundation mapping: a comparison of a digital terrain model (DTM) filling and two hydrodynamic methods. Hydrol. Earth Syst. Sci. 25, 2979–2995 (2021). https://doi.org/10.5194/hess-25-2979-2021
109. Parallel Computing Toolbox, https://www.mathworks.com/products/parallel-computing.html
110. Manhui, L.: Bobholamovic/ChangeDetectionToolbox, https://github.com/Bobholamovic/ChangeDetectionToolbox, (2024)
111. Statistics and Machine Learning Toolbox, https://www.mathworks.com/products/statistics.html
112. Deep Learning Toolbox, https://www.mathworks.com/products/deep-learning.html
113. Image Processing and Computer Vision - MATLAB & Simulink, https://www.mathworks.com/help/overview/image-processing-and-computer-vision.html
114. Tensor Processing Units (TPUs), https://cloud.google.com/tpu
115. What is BSD licenses? | Definition from TechTarget, https://www.techtarget.com/whatis/definition/BSD-licenses
116. Malitesta, D., Gassi, G., Pomo, C., Di Noia, T.: Ducho: A Unified Framework for the Extraction of Multimodal Features in Recommendation, https://arxiv.org/abs/2306.17125v2
117. Aakanksha Toutam, Amey Zade, Sanika Gongale, Kaushal Kamde, Prof. Anand Donald: Video Summarization using Python. Int. J. Adv. Res. Sci. Commun. Technol. 187–191 (2024). https://doi.org/10.48175/IJARSCT-18324
118. Hanbay, K., Golgiyaz, S., Talu, M.F.: Real time fabric defect detection system on Matlab and C++/Opencv platforms. 2017 Int. Artif. Intell. Data Process. Symp. IDAP. 1–8 (2017). https://doi.org/10.1109/IDAP.2017.8090180

119. Khvan, R.V.: Comparative Analysis of the Performance of Artificial Neural Networks in Assessing the Technical Condition of Steel Ropes. Saf. Technog. Nat. Syst. 68–77 (2024). https://doi.org/10.23947/2541-9129-2024-8-2-68-77

120. Alabbad, Y., Mount, J., Campbell, A.M., Demir, I.: A web-based decision support framework for optimizing road network accessibility and emergency facility allocation during flooding. Urban Inform. 3, 10 (2024). https://doi.org/10.1007/s44212-024-00040-0

121. Panakkal, P., Wyderka, A.M., Padgett, J.E., Bedient, P.B.: Safer this way: Identifying flooded roads for facilitating mobility during floods. J. Hydrol. 625, 130100 (2023). https://doi.org/10.1016/j.jhydrol.2023.130100

122. Baykal, T., Terzi, S., Taylan, E.D.: Examination of safe routes for emergency responders and people during urban flood: a case study of Isparta, Türkiye. Nat. Hazards. 119, 1379–1397 (2023). https://doi.org/10.1007/s11069-023-06171-y

123. Jeon, S., Jung, K., Kim, J., Jung, H.: Map API-Based Evacuation Route Guidance System for Floods. Appl. Sci. 13, 9141 (2023). https://doi.org/10.3390/app13169141

124. Wang, F., Xie, Z., Liu, H., Pei, Z., Liu, D.: Multiobjective Emergency Resource Allocation under the Natural Disaster Chain with Path Planning. Int. J. Environ. Res. Public. Health. 19, 7876 (2022). https://doi.org/10.3390/ijerph19137876

125. Ma, R., Meng, F., Du, H.: Research on Intelligent Emergency Resource Allocation Mechanism for Public Health Emergencies: A Case Study on the Prevention and Control of COVID-19 in China. Systems. 11, 300 (2023). https://doi.org/10.3390/systems11060300

126. Dijkstra's Algorithm Definition | GIS Dictionary, https://support.esri.com/en-us/gis-dictionary/dijkstra-s-algorithm

127. Gigović, L., Pamučar, D., Bajić, Z., Drobnjak, S.: Application of GIS-Interval Rough AHP Methodology for Flood Hazard Mapping in Urban Areas. Water. 9, 360 (2017). https://doi.org/10.3390/w9060360

128. Breen, J.J., Parrish, D.R.: GIS in Emergency Management Cultures: An Empirical Approach to Understanding Inter- and Intra-agency Communication During Emergencies. J. Homel. Secur. Emerg. Manag. 10, 477–495 (2013). https://doi.org/10.1515/jhsem-2013-0014

129. Jana, S., Majumder, R., Menon, P.P., Ghose, D.: Decision Support System (DSS) for Hierarchical Allocation of Resources and Tasks for Disaster Management. Oper. Res. Forum. 3, 37 (2022). https://doi.org/10.1007/s43069-022-00148-6

130. Majumder, R., Warier, R.R., Ghose, D.: Game-Theoretic Model Based Resource Allocation During Floods, http://arxiv.org/abs/2112.01439, (2021)

131. Gooding, K., Bertone, M.P., Loffreda, G., Witter, S.: How can we strengthen partnership and coordination for health system emergency preparedness and response? Findings from a synthesis of experience across countries facing shocks. BMC Health Serv. Res. 22, 1441 (2022). https://doi.org/10.1186/s12913-022-08859-6

132. Decision support systems: Drive better decision-making with data, https://www.cio.com/article/193521/decision-support-systems-sifting-data-for-better-business-decisions.html

133. Decision Support System (DSS): What It Is and How Businesses Use Them, https://www.investopedia.com/terms/d/decision-support-system.asp

134. Decision Support System (DDS): Definition, Benefits and Types, https://www.indeed.com/career-advice/career-development/decision-support-system

135. MES Certification Repository: Decision Support System & Data Warehouse, https://cmsgov.github.io/CMCS-DSG-DSS-Certification/Outcomes%20and%20Metrics/Decision%20Support%20System%20&%20Data%20Warehouse/

136. DSS Web Tour, https://www.dssresources.com/tour/index.html

137. Decision support system, https://en.wikipedia.org/w/index.php?title=Decision_support_system&oldid=1216075157, (2024)

138. What is a decision support system (DSS)?, https://www.techtarget.com/searchcio/definition/decision-support-system

139. A Brief History of Decision Support Systems, https://dssresources.com/history/dsshistoryv28.html

140. Decision Support System (DSS): Definition & Best Practices, https://www.qlik.com/us/business-intelligence/decision-support-system

141. Mehta, K.: Decision Support System (DSS) - What Is It, Example, Components, https://www.wallstreetmojo.com/decision-support-system/

142. Components of Decision Support Systems, http://dsssystem.blogspot.com/2010/01/components-of-decision-support-systems.html

143. Decision Support System (DSS) - Glossary, https://www.engati.com/glossary/decision-support-system

144. Decision Support System (DSS), https://corporatefinanceinstitute.com/resources/management/decision-support-system-dss/

145. What Is a Decision Support System? (With Components), https://ca.indeed.com/career-advice/career-development/decision-support-system

146. What are the different types of decision support systems?, https://www.linkedin.com/advice/3/what-different-types-decision-support-systems-skills-research-ogqde

147. Decision support systems (DSS), https://www.euvic.com/decision-support-systems-dss/

148. Fernando, J.: Decision Support System: Overview, Different Types and Elements. TechnoareteTransactions Intell. Data Min. Knowl. Discov. 2, (2022). https://doi.org/10.36647/TTIDMKD/02.02.A003

149. Agbehadji, I.E., Schütte, S., Masinde, M., Botai, J., Mabhaudhi, T.: Climate Risks Resilience Development: A Bibliometric Analysis of Climate-Related Early Warning Systems in Southern Africa. 3 (2024). https://doi.org/10.3390/cli12010003

150. Types of Decision Support Systems (DSS), https://www.gdrc.org/decision/dss-types.html

151. Dařena, F.: Decision support for customers in electronic environments. Acta Univ. Agric. Silvic. Mendel. Brun. 59, 51–58 (2011). https://doi.org/10.11118/actaun201159020051

152. Intelligent decision support system, https://en.wikipedia.org/w/index.php?title=Intelligent_decision_support_system&oldid=1219171411, (2024)

153. Logic, R.: Five Decision Support System Examples You Need to Know, https://download.riverlogic.com/blog/five-decision-support-system-examples

154. Ma, H., Zhu, M., Qi, D., Yang, J., Pan, R.: A Cloud-Native-Oriented AI Epidemic Four-Color Situation Decision Support System. In: 2022 4th International Symposium on Smart and Healthy Cities (ISHC). pp. 141–147 (2022)

155. Wardropper, C., Brookfield, A.: Decision-support systems for water management. J. Hydrol. 610, 127928 (2022). https://doi.org/10.1016/j.jhydrol.2022.127928

156. Izady, A., Joodavi, A., Ansarian, M., Shafiei, M., Majidi, M., Davary, K., Ziaei, A.N., Ansari, H., Nikoo, M.R., Al-Maktoumi, A., Chen, M., Abdalla, O.: A scenario-based coupled SWAT-MODFLOW decision support system for advanced water resource management. J. Hydroinformatics. 24, 56–77 (2021). https://doi.org/10.2166/hydro.2021.081

157. Govardhan, G., Ghude, S.D., Kumar, R., Sharma, S., Gunwani, P., Jena, C., Yadav, P., Ingle, S., Debnath, S., Pawar, P., Acharja, P., Jat, R., Kalita, G., Ambulkar, R., Kulkarni, S., Kaginalkar, A., Soni, V.K., Nanjundiah, R.S., Rajeevan, M.: Decision Support System version 1.0 (DSS v1.0) for air quality management in Delhi, India. Geosci. Model Dev. 17, 2617–2640 (2024). https://doi.org/10.5194/gmd-17-2617-2024

158. Dinar, A., Quinn, N.W.T.: Developing a Decision Support System for Regional Agricultural Nonpoint Salinity Pollution Management: Application to the San Joaquin River, California. Water. 14, 2384 (2022). https://doi.org/10.3390/w14152384

159. Janis, I.L.: Victims of Groupthink: A Psychological Study of Foreign-Policy Decisions and Fiascoes. (1972)

160. Shi, H., Du, E., Liu, S., Chau, K.-W.: Advances in Flood Early Warning: Ensemble Forecast, Information Dissemination and Decision-Support Systems. Hydrology. 7, 56 (2020). https://doi.org/10.3390/hydrology7030056

161. Fernández-Nóvoa, D., González-Cao, J., García-Feal, O.: Enhancing Flood Risk Management: A Comprehensive Review on Flood Early Warning Systems with Emphasis on Numerical Modeling. Water. 16, 1408 (2024). https://doi.org/10.3390/w16101408

162. Josipovic, N., Viergutz, K.: Smart Solutions for Municipal Flood Management: Overview of Literature, Trends, and Applications in German Cities. Smart Cities. 6, 944–964 (2023). https://doi.org/10.3390/smartcities6020046

163. Ni, X., Dong, Z., Xie, W., Jia, W., Duan, C., Yao, H.: Research on the Multi-Objective Cooperative Competition Mechanism of Jinsha River Downstream

Cascade Reservoirs during the Flood Season Based on Optimized NSGA-III. Water. 11, 849 (2019). https://doi.org/10.3390/w11040849

164. Garcia, M., Juan, A., Bedient, P.: Integrating Reservoir Operations and Flood Modeling with HEC-RAS 2D. Water. 12, 2259 (2020). https://doi.org/10.3390/w12082259

165. de Bruijn, K.M., Maran, C., Zygnerski, M., Jurado, J., Burzel, A., Jeuken, C., Obeysekera, J.: Flood Resilience of Critical Infrastructure: Approach and Method Applied to Fort Lauderdale, Florida. Water. 11, 517 (2019). https://doi.org/10.3390/w11030517

166. Song, Y., Park, Y., Lee, J., Park, M., Song, Y.: Flood Forecasting and Warning System Structures: Procedure and Application to a Small Urban Stream in South Korea. Water. 11, 1571 (2019). https://doi.org/10.3390/w11081571

167. Sulasikin, A., Nugrahat, Y., Aminanto, M.E., Nasution, B.I., Kanggrawan, J.I.: Developing a knowledge management system for supporting flood decision-making. In: 2022 IEEE International Smart Cities Conference (ISC2). pp. 1–4 (2022)

168. Bentoso, L.D., Juan, E.O., Brosas, D.G., Paragas, J.R., Nuevas, L.K., Velarde, Ma.W.C.: Web-Based Solution for Flood Warning Decision Support in the Province of Leyte, Philippines. In: 2021 3rd International Conference on Research and Academic Community Services (ICRACOS). pp. 185–190 (2021)

169. Soebroto, A.A., Limantara, L.M., Suhartanto, E., Sholichin, M.: Modelling of Flood Hazard Early Warning Group Decision Support System. Civ. Eng. J. 10, 614–627 (2024). https://doi.org/10.28991/CEJ-2024-010-02-018

170. Ali, R., Hussain, A., Nazir, S., Khan, S., Khan, H.U.: Intelligent Decision Support Systems—An Analysis of Machine Learning and Multicriteria Decision-Making Methods. Appl. Sci. 13, 12426 (2023). https://doi.org/10.3390/app132212426

171. Jhade, S., Gangavarapu, S.P., Channabasamma, C., Rozhdestvenskiy, O.I.: Smart Medicine: Exploring the Landscape of AI-Enhanced Clinical Decision Support Systems. MATEC Web Conf. 392, 01083 (2024). https://doi.org/10.1051/matecconf/202439201083

172. Silva, H., Bernardino, J.: Machine Learning Algorithms: An Experimental Evaluation for Decision Support Systems. Algorithms. 15, 130 (2022). https://doi.org/10.3390/a15040130

173. Widodo, D., Kristalina, P., Hadi, Moch.Z.S., Kurniawati, A.D.: Performance Evaluation of Docker Containers for Disaster Management Dashboard Web Application. In: 2023 International Electronics Symposium (IES). pp. 551–556. IEEE, Denpasar, Indonesia (2023)

174. Terribile, F., Acutis, M., Agrillo, A., Anzalone, E., Azam-Ali, S., Bancheri, M., Baumann, P., Birli, B., Bonfante, A., Botta, M., Cavaliere, F., Colandrea, M., D'Antonio, A., De Mascellis, R., De Michele, C., De Paoli, G., Monica, C.D., Di Leginio, M., Ferlan, M., Ferraro, G., Florea, A., Hermann, T., Hoenig, H., Jahanshiri, E., Jevšenak, J., Kárpáti, V., Langella, G., Le, Q.B., Lezzi, D., Loishandl, H., Loudin, S., Manna, P., Marano, G., Marotta, L., Merticariu, V., Mileti, F.A., Minieri, L., Misev, D., Montanarella, L., Munafò, M., Neuwirth, M., Orefice, N., Pácsonyi, I., Panagos, P., Perego, A., Huu, B.P., Pinto, F., Prebeck,

K., Puig, A., Pump, J., Schillaci, C., Simončič, P., Skudnik, M., Stankovics, P., Tóth, G., Tramberend, P., Vingiani, S., Vuolo, F., Zucca, C., Basile, A.: The LANDSUPPORT geospatial decision support system (S-DSS) vision: Operational tools to implement sustainability policies in land planning and management. Land Degrad. Dev. 35, 813–834 (2024). https://doi.org/10.1002/ldr.4954

175. Yassin, M.I.H.M., Wan, A.T., Newaz, S.H.S., Omar, S.: LoRa Based Real-time Flood Detection and Monitoring System: A Brunei Darussalam Based Study. In: 2021 International Conference on Electronics, Communications and Information Technology (ICECIT). pp. 1–5 (2021)
176. HydroNET Decision Support System for IWRM - HydroNET.com, https://www.hydronet.com/longreads/hydronet-dss/
177. Yesudas, M., Menon, G., Ramamurthy, V.: Intelligent operational dashboards for smarter commerce using big data. IBM J. Res. Dev. 58, 13:1-13:10 (2014). https://doi.org/10.1147/JRD.2014.2346131
178. MetricFire: Grafana vs. Tableau, https://www.metricfire.com/blog/grafana-vs-tableau/
179. Tableau, https://www.tableau.com/products/tableau
180. Grafana | Query, visualize, alerting observability platform, https://grafana.com/grafana/
181. Vasnier, J.-M., Maranzana, N., Messaadia, M., Aoussat, A.: Preliminary Design and Evaluation Of Strategic Dashboards Through The Technology Acceptance Model. Proc. Des. Soc. Des. Conf. 1, 777–786 (2020). https://doi.org/10.1017/dsd.2020.18
182. Power BI - Data Visualization | Microsoft Power Platform, https://www.microsoft.com/en-us/power-platform/products/power-bi
183. Qlik Sense | Modern Analytics, https://www.qlik.com/us/products/qlik-sense
184. Verbert, K., Ochoa, X., De Croon, R., Dourado, R.A., De Laet, T.: Learning analytics dashboards: the past, the present and the future. In: Proceedings of the Tenth International Conference on Learning Analytics & Knowledge. pp. 35–40. Association for Computing Machinery, New York, NY, USA (2020)
185. KU Leuven, Belgium, Klerkx, J., Verbert, K., KU Leuven, Belgium, Duval, E., KU Leuven, Belgium: Learning Analytics Dashboards. In: Columbia University, USA, Lang, C., Siemens, G., University of Texas at Arlington, USA, Wise, A., New York University, USA, Gasevic, D., and University of Edinburgh, UK (eds.) Handbook of Learning Analytics. pp. 143–150. Society for Learning Analytics Research (SoLAR) (2017)
186. IBM Cognos Analytics, https://www.ibm.com/products/cognos-analytics
187. Welcome | Superset, https://superset.apache.org/
188. Compare Apache Superset vs IBM Cognos Dashboard, https://www.peerspot.com/product_comparisons/34601-37529
189. Piantari, E., Megasari, R., Hidayat, K.: Tactical Dashboard Design for Study Program in University. Presented at the Proceedings of the 7th Mathematics,

Science, and Computer Science Education International Seminar, MSCEIS 2019, 12 October 2019, Bandung, West Java, Indonesia July 30 (2020)

190. Discover the Domo Data Experience Platform | Domo, https://www.domo.com/
191. Make Data Everybody's Business | Klipfolio, https://www.klipfolio.com/
192. Compare Domo vs Klipfolio 2024, https://www.capterra.com/compare/119119-130237/Domo-vs-Klipfolio-Dashboard
193. Manaher, S.: Protocol vs Procedure: Unraveling Commonly Confused Terms, https://thecontentauthority.com/blog/protocol-vs-procedure, (2023)
194. Lee, K.J., Malinen, S.K., Nilakant, V.: The dynamics of cross-sector collaboration in disasters. Disaster Prev. Manag. Int. J. 32, 337–351 (2023). https://doi.org/10.1108/DPM-09-2022-0184
195. Bharosa, N., Lee, J., Janssen, M.: Challenges and obstacles in sharing and coordinating information during multi-agency disaster response: Propositions from field exercises. Inf. Syst. Front. 12, 49–65 (2010). https://doi.org/10.1007/s10796-009-9174-z
196. Bangladesh's Comprehensive Disaster Management Programme, https://cdkn.org/sites/default/files/files/Bangladesh-InsideStory_5pp_pr4F_LR1.pdf
197. Welcome to Pyke, https://pyke.sourceforge.net/
198. Zhang, X., Moynihan, G., Ernest, A., Gutenson, J.: An Expert System for Local Flood Response Coordination and Training. Eng. Manag. Res. 6, p1 (2017). https://doi.org/10.5539/emr.v6n2p1
199. INSARAG GUIDELINES 2020 – INSARAG, https://www.insarag.org/methodology/insarag-guidelines/
200. admin: Importance of Mock drill in Workplace Safety, https://www.greenwgroup.com/importance-of-mock-drill-in-workplace/, (2023)
201. National Disaster Response Force Government of India: Standard Operating Procedure on Flood Disaster Response, https://ndrf.gov.in/sites/default/files/FLOOD.pdf
202. Delta Works | History, Flood, Dams, & Facts | Britannica, https://www.britannica.com/event/Delta-Works
203. National Disaster Recovery Framework | FEMA.gov, https://www.fema.gov/emergency-managers/national-preparedness/frameworks/recovery
204. National Planning Frameworks | FEMA.gov, https://www.fema.gov/emergency-managers/national-preparedness/frameworks
205. How FEMA Works | FEMA.gov, https://www.fema.gov/about/how-fema-works
206. National Incident Management System | FEMA.gov, https://www.fema.gov/emergency-managers/nims
207. BACKGROUND – INSARAG, https://www.insarag.org/about/background/
208. Flexible Response Working Group – INSARAG, https://www.insarag.org/global-structures/working-groups/flexible-response-working-group/, (2024)

209. United Nations Office for the Coordination of Humanitarian Affairs: INSARAG Strategic Plan 2021-2026 Endorsed 27-Jan-2021 V2, https://www.insarag.org/wp-content/uploads/2021/07/INSARAG-Strategic-Plan-2021-2026_Endorsed-27-Jan-2021-V2-.pdf, (2024)

210. INSARAG AEME Regional Meeting.

211. INSARAG-Guidelines-V2-Manual-B-Operations.pdf, https://preparecenter.org/wp-content/uploads/2021/03/INSARAG-Guidelines-V2-Manual-B-Operations.pdf

212. rescEU - European Commission, https://civil-protection-humanitarian-aid.ec.europa.eu/what/civil-protection/resceu_en

213. Emergency Response Coordination Centre (ERCC) - European Commission, https://civil-protection-humanitarian-aid.ec.europa.eu/what/civil-protection/emergency-response-coordination-centre-ercc_en

214. EU and World Bank create new facility to strengthen capacity of civil protection in Europe, https://ec.europa.eu/commission/presscorner/detail/en/IP_24_1184

215. model LISFLOOD - LISFLOOD hydrological model | Modelling Inventory and Knowledge Management System of the European Commission (MIDAS), https://web.jrc.ec.europa.eu/policy-model-inventory/explore/models/model-lisflood/

216. Open Source Lisflood, https://ec-jrc.github.io/lisflood/

217. European Flood Awareness System | Copernicus, https://www.copernicus.eu/en/european-flood-awareness-system

218. European Flood Awareness System – EFAS | Copernicus EMS - European Flood Awareness System, https://european-flood.emergency.copernicus.eu/en/european-flood-awareness-system-efas

219. INSARAG AEME Regional Meeting.

220. United Nations Office for the Coordination of Humanitarian Affairs: INSARAG Guidelines V2 Manual B - Operations, https://preparecenter.org/wp-content/uploads/2021/03/INSARAG-Guidelines-V2-Manual-B-Operations.pdf, (2024)

221. What is information systems (IS)? | Definition from TechTarget, https://www.techtarget.com/whatis/definition/IS-information-system-or-information-services

222. What is Real-Time Data? Definition & Best Practices, https://www.qlik.com/us/streaming-data/real-time-data

223. Node-RED, https://nodered.org/

224. What is event-driven architecture?, https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture

225. What is interconnection: Examples and definition, https://www.flexential.com/resources/blog/interconnection-examples

226. FlowFuse • Node-RED for professionals, https://flowfuse.com/

227. Getting Started | Node-RED Dashboard 2.0, https://dashboard.flowfuse.com/getting-started.html

228. What Is MATLAB?, https://www.mathworks.com/discovery/what-is-matlab.html

229. MATLAB, https://en.wikipedia.org/w/index.php?title=MATLAB&oldid=1238254003, (2024)

230. Image Processing Toolbox Documentation, https://www.mathworks.com/help/images/

231. Mapping Toolbox Documentation, https://www.mathworks.com/help/map/

232. Map Flood Areas Using Sentinel-1 SAR Imagery - MATLAB & Simulink, https://www.mathworks.com/help/images/map-flood-areas-using-sentinel-1-sar-imagery.html

233. What is a shapefile?—ArcMap | Documentation, https://desktop.arcgis.com/en/arcmap/latest/manage-data/shapefiles/what-is-a-shapefile.htm

234. Desktop GIS Software | Mapping Analytics | ArcGIS Pro, https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview

235. Find Routes (Ready To Use)—ArcGIS Pro | Documentation, https://pro.arcgis.com/en/pro-app/latest/tool-reference/ready-to-use/itemdesc-findroutes.htm

236. An introduction to ArcGIS Online—ArcGIS Online Help | Documentation, https://doc.arcgis.com/en/arcgis-online/get-started/what-is-agol.htm

237. InfluxDB | Real-time insights at any scale, https://www.influxdata.com/home/

238. Introduction to InfluxDB: A time-series database, https://wearecommunity.io/communities/india-java-user-group/articles/891

239. InfluxDB, https://en.wikipedia.org/w/index.php?title=InfluxDB&oldid=1236639088, (2024)

240. InfluxDB Line Protocol | TDengine, https://docs.tdengine.com/develop/insert-data/influxdb-line/

241. Line protocol | InfluxDB OSS v2 Documentation, https://docs.influxdata.com/influxdb/v2/reference/syntax/line-protocol/

242. Master Using Time Series Data with InfluxDB and Flux, https://www.influxdata.com/products/flux/

243. MQTT - The Standard for IoT Messaging, https://mqtt.org/

244. What is MQTT? Definition and Details, https://www.paessler.com/it-explained/mqtt

245. Team, E.: What Is the MQTT Protocol: A Beginner's Guide, https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt

246. HiveMQ – The Most Trusted MQTT platform to Transform Your Business, https://www.hivemq.com/

247. Apache Kafka, https://kafka.apache.org/quickstart

248. What is Apache Kafka®? | Everything you need to know for Kafka, https://aiven.io/developer/what-is-apache-kafka

249. Spark vs Kafka - Difference Between Data Processing Engines - AWS, https://aws.amazon.com/compare/the-difference-between-kafka-and-spark/

250. Kafka and Cloud Storage: A Guide to Seamless Integration - Alibaba Cloud, https://www.alibabacloud.com/tech-news/a/kafka/gtv2q097hw-kafka-and-cloud-storage-a-guide-to-seamless-integration

251. Apache Kafka vs Spark: Real-time data streaming showdown, https://double.cloud/blog/posts/2023/06/kafka-vs-spark/

252. mathworks-ref-arch/matlab-apache-kafka, https://github.com/mathworks-ref-arch/matlab-apache-kafka, (2024)

253. Chaurasia, R., Mohindru, V.: Unmanned Aerial Vehicle (UAV): A Comprehensive Survey. In: Unmanned Aerial Vehicles for Internet of Things (IoT). pp. 1–27. John Wiley & Sons, Ltd (2021)

254. Figliozzi, M.: Analyzing the Impact of Technological Improvements on the Performance of Delivery Drones. Transp. Res. Procedia. 79, 68–75 (2024). https://doi.org/10.1016/j.trpro.2024.03.011

255. Custom Drones for Aerial Surveying Mapping & Inspection | Customized Drones, https://c-drones.com/shop/uav-mapping-drones/

256. Long Distance Drones | Customizable Drones with Long Range | Twin-wing, https://www.unmannedsystemstechnology.com/company/cannon-dynamics/

257. First Response - Law Enforcement - DJI Enterprise, https://enterprise.dji.com/public-safety/photo

258. IP Ratings and Drones, https://www.kdedirect.com/blogs/news/ip-ratings-drones

259. Shardlow, R.: What is an IP Rating? A Guide to Understanding Drone IP Ratings, https://coptrz.com/blog/what-is-an-ip-rating-a-guide-to-understanding-drone-ip-ratings/

260. Ingress Protection (IP) ratings, https://www.iec.ch/ip-ratings

261. GMTI Radar - IAI and ELTA's ELM-2054 and Lightweight SAR Payload, https://www.iai.co.il/p/elm-2054

262. Earth Science Data Systems, N.: Active Sensors | Earthdata, https://www.earthdata.nasa.gov/learn/backgrounders/active-sensors

263. Facility 20230618, A.S.: What is SAR?, https://asf.alaska.edu/information/sar-information/what-is-sar/

264. Earth Science Data Systems, N.: What is Synthetic Aperture Radar? | Earthdata, https://www.earthdata.nasa.gov/learn/backgrounders/what-is-sar

265. FINDER_MK4_V4Mi.pdf, https://www.specopsgroup.com/_files/ugd/2d6b76_c92484642e27439b8ab71de69bccac68.pdf

266. Vector, https://quantum-systems.com/vector/

267. Thanasis Papaioannou: Introduction to IoT Systems and Technologies MSc on Modern Communication Systems and Internet of Things University of Thessaly. lecture7-8. , Larissa (Fall ’22)
268. FloodNet Methodology, https://www.floodnet.nyc/methodology/
269. FloodNet Methodology & Data Sources V3 (21 December 2022), https://docs.google.com/document/d/1dUQtK9ef3vo3G_AIh-viKEoSyc7hCrW7jI6X29lGxyg/edit?usp=embed_facebook
270. MaxBotix Collections, https://maxbotix.com/collections
271. Our Top Devices Orbital Satcom, https://osat.com/collections/devices
272. Mobile and Private Mobile Radio, https://www.etsi.org/technologies/mobile-radio
273. iBS Integrated TETRA Base Station - Hytera - Hytera, https://www.hytera.com/en/product-new/digital-radio/tetra-system/ibs.html
274. Starlink Business | Land Mobility, https://www.starlink.com/business/mobility
275. Spacelink: Does Cloud Cover Affect Starlink? The Truth about Bad Weather & Starlink, https://www.spacelink-installations.co.uk/blog/does-cloud-cover-affect-starlink/, (2024)
276. TETRA Antenna 400MHz High Gain TETRA Fiberglass Collinear Antennas 400MHz High Gain Tetra Yagi Antenna 400MHz TETRA Circular Polarized Helical Helix Antenna, https://www.antennaexperts.in/tetra_antennas_400_MHz.asp
277. The Ethics of AI Addressing Bias, Privacy, and Accountability in Machine Learning, https://www.cloudthat.com/resources/blog/cybersecurity-in-the-modern-world/
278. What is MQTT Quality of Service (QoS) 0,1, & 2? – MQTT Essentials: Part 6, https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/
279. Copernicus Browser, https://browser.dataspace.copernicus.eu/
280. Map Flood Areas Using Sentinel-1 SAR Imagery - MATLAB & Simulink, https://www.mathworks.com/help/images/map-flood-areas-using-sentinel-1-sar-imagery.html
281. wiener2 - 2-D adaptive noise-removal filtering - MATLAB, https://www.mathworks.com/help/images/ref/wiener2.html
282. MATLAB - Functions, https://www.tutorialspoint.com/matlab/matlab_functions.htm
283. Functions - MATLAB & Simulink, https://www.mathworks.com/help/matlab/functions.html?s_tid=CRUX_lftnav
284. S1-Radiometric-Calibration-V1.0.pdf, https://sentinel.esa.int/documents/247904/685163/S1-Radiometric-Calibration-V1.0.pdf
285. Level-1 Post-Processing Algorithms - Sentinel-1 SAR Technical Guide - Sentinel Online, https://copernicus.eu/level-1-post-processing-algorithms

286. Greece Roads (OpenStreetMap Export) - Humanitarian Data Exchange, https://data.humdata.org/dataset/hotosm_grc_roads?

287. Regions of Greece - Περιφέρειες Ελλάδας - GEODATA.gov.gr, https://geodata.gov.gr/en/dataset/periphereies-elladas/resource/7c80a2c1-93b7-4814-9fc4-245e775acaa6

288. Κακοκαιρία Daniel: Όλη τη νύχτα έβγαζαν εγκλωβισμένους στη Θεσσαλία - Μάχη με τον χρόνο δίνουν οι διασώστες, https://www.protothema.gr/greece/article/1410163/kakokairia-daneil-oli-ti-nuhta-evgazan-eglovismenous-sti-thessalia-mahi-me-ton-hrono-dinoun-oi-diasostes/

289. Παύλου, Α.: Κακοκαιρία Daniel: Στους 12 οι νεκροί, https://www.ot.gr/2023/09/10/epikairothta/kakokairia-daniel-stous-12-oi-nekroi/

290. Zhou, W., Yan, Z., Zhang, L.: A comparative study of 11 non-linear regression models highlighting autoencoder, DBN, and SVR, enhanced by SHAP importance analysis in soybean branching prediction. Sci. Rep. 14, 5905 (2024). https://doi.org/10.1038/s41598-024-55243-x

291. Regression in machine learning, https://www.geeksforgeeks.org/regression-in-machine-learning/

292. What Is Supervised Learning? | IBM, https://www.ibm.com/topics/supervised-learning

293. Nikolopoulou, K.: Supervised vs. Unsupervised Learning: Key Differences, https://www.scribbr.com/ai-tools/supervised-vs-unsupervised-learning/

294. Random Forest Regression in Python, https://www.geeksforgeeks.org/random-forest-regression-in-python/

295. Wei, D.: Demystifying Machine Learning Models: Random Forest, https://medium.com/@weidagang/demystifying-machine-learning-models-random-forest-f992dc50b427, (2024)

296. AnalytixLabs: Random Forest Regression — How it Helps in Predictive Analytics?, https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-in-predictive-analytics-01c31897c1d4, (2023)

297. Machine Learning: Linearity vs Nonlinearity, https://www.linkedin.com/pulse/machine-learning-linearity-vs-nonlinearity-reday-zarra

298. Huang, F., Chen, J., Lin, Z., Kang, P., Yang, Z.: Random Forest Exploiting Post-related and User-related Features for Social Media Popularity Prediction. In: Proceedings of the 26th ACM international conference on Multimedia. pp. 2013–2017. Association for Computing Machinery, New York, NY, USA (2018)

299. Probst, P., Boulesteix, A.-L., Wright, M.: Hyperparameters and Tuning Strategies for Random Forest. (2018)

300. Saxena, S.: A Beginner's Guide to Random Forest Hyperparameter Tuning, https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/, (2020)

301. Contreras, P., Orellana-Alvear, J., Muñoz, P., Bendix, J., Célleri, R.: Influence of Random Forest Hyperparameterization on Short-Term Runoff Forecasting in an

Andean Mountain Catchment. Atmosphere. 12, 238 (2021). https://doi.org/10.3390/atmos12020238

302. SMAPE - Symmetric Mean Absolute Percentage Error — Permetrics 2.0.0 documentation, https://permetrics.readthedocs.io/en/latest/pages/regression/SMAPE.html
303. Flask: A simple framework for building complex web applications.
304. Install ngrok, https://ngrok.com/download
305. Academy, E.: What is the concept of “pickling” in machine learning and how does it help in the prediction process?, https://eitca.org/artificial-intelligence/eitc-ai-mlp-machine-learning-with-python/regression/regression-forecasting-and-predicting/examination-review-regression-forecasting-and-predicting/what-is-the-concept-of-pickling-in-machine-learning-and-how-does-it-help-in-the-prediction-process/, (2023)
306. Song, D.F.: PRACTICAL CONTRIBUTIONS.
307. Talke, K.A., Stroumtsos, N.C.: Tether management system for a tethered UAV, https://patents.google.com/patent/US11440680B2/en, (2022)
308. Build a Two-Way Pager With LoRa - IEEE Spectrum, https://spectrum.ieee.org/build-a-twoway-pager-with-lora
309. sqij: LoRa QWERTY Pager, https://www.instructables.com/LoRa-QWERTY-Pager/
310. Halúsková, B.: Digital Twin in Smart City. Transp. Res. Procedia. 74, 1471–1478 (2023). https://doi.org/10.1016/j.trpro.2023.11.308
311. What Is a Digital Twin? | IBM, https://www.ibm.com/topics/what-is-a-digital-twin
312. Millard Fuller, https://en.wikipedia.org/w/index.php?title=Millard_Fuller&oldid=1230836912, (2024)

Tables Index

Figures Index

Appendices

Appendix A

List of Abbreviations

Abbreviation	*Definition*
AllERT©	*All Emergency in Real Time*
AEME	*Africa-Europe-Middle East*
AI	*Artificial Intelligence*
ANP	*Analytical Network Process*
API	*Application Programming Interface*
BSD	*Berkeley Source Distribution*
CDMP	*Comprehensive Disaster Management Programme*
CDSS	*Clinical Decision Support Systems*
CEMS	*Copernicus Emergency Management Service*
CNN	*Convolutional Neural Network*
CRA	*Community Risk Assessment*
CSS	*Cascading Style Sheets*
CUNY	*City University of New York*
DAM	*Data Asset Management*
DHS	*Department of Homeland Security*
DIKI	*Data, Information, Knowledge and Insights*
DMIC	*Disaster Management Information Centre*
DSS	*Decision Support System*
EFAS	*European Flood Awareness System*
EIS	*Executive Information Systems*
ERCC	*Emergency Response Coordination Centre*
ESF	*Emergency Support Functions*
ESRI	*Environmental Systems Research Institute*
ETSI	*European Telecommunications Standards Institute*
FEMA	*Federal Emergency Management Agency*
FINDER	*Finding Individuals for Disaster Emergency Response*
FMS	*Flexible Manufacturing Systems*
GDSS	*Group Decision Support System*
GIS	*Geographic Information Systems*
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
GSM	*Global System for Mobile Communications*
GUI	*Graphical User Interface*
HADR	*Humanitarian Assistance and Disaster Relief*
HALE	*High-Altitude Long-Endurance*
HTML	*Hypertext Markup Language*
HTS	*High-Throughput Satellites*
HTTP	*Hypertext Transfer Protocol*
IBM	*International Business Machines Corporation*
ICMS	*INSARAG Collection and Management System*

ICS	*Incident Command System*
IDE	*Integrated Development Environment*
IDSS	*Intelligent Decision Support System*
IEC	*International Electrotechnical Commission*
INSARAG	*International Search and Rescue Advisory Group*
IoT	*Internet of Things*
JRC	*Joint Research Centre*
JSON	*JavaScript Object Notation*
LACC	*Livelihood Adaptation to Climate Change*
LCC	*Low-Level Control Centre*
LEMA	*Local Emergency Management Agency*
LEO	*Low Earth Orbit*
LFRS	*Local Flood Response System*
MALE	*Medium-Altitude Long-Endurance*
MATLAB	*MATrix LABoratory*
MCC	*Middle-Level Control Centre*
MCDM	*Multiple Criteria Decision-Making*
ML	*Machine Learning*
MLC	*Maximum Likelihood Classification*
MQTT	*Message Queuing Telemetry Transport*
NASA	*National Aeronautics and Space Administration*
NDMA	*National Disaster Management Authority*
NDRF	*National Disaster Response Force*
NIMS	*National Incident Management System*
NRF	*National Response Framework*
NYU	*New York University*
OBIA	*Object-Based Image Analysis*
ODSS	*Organizational Decision Support Systems*
OLAP	*Online Analytical Processing*
OSOCC	*On-Site Operations Coordination Centers*
PMR	*Analogue Private Mobile Radio*
RDC	*Reception/Departure Centre*
REST	*Representational State Transfer*
RGB	*Red, Green and Blue*
ROI	*Region of Interest*
ROS	*Response Operational Structure*
RRA	*Roles Responsibilities and Actions*
RTK	*Real-Time Kinematic*
SAFE	*Standard Archive Format for Europe*
SAR	*Search And Rescue*
SAR	*Synthetic Aperture Radar*
SARDO	*Search-And-Rescue DrOne-based solution*
SHP	*Shapefile*
SMS	*Short Message Service*
SOP	*Standard Operating Procedures*
SQL	*Structured Query Language*
SVM	*Support Vector Machines*

SWOT	*Strengths, Weaknesses, Opportunities and Threats analysis*
TCC	*Top-Level Control Centre*
TETRA	*TErrestrial Trunked RAdio*
TIFF	*Tag Image File Format*
TPU	*Tensor Processing Unit*
TSDB	*Time Series DataBase*
UAV	*Unmanned Aerial Vehicles*
UCPM	*EU Civil Protection Mechanism*
URL	*Uniform Resource Locator*
USAR	*Urban Search and Rescue*
VIKOR	*VlseKriterijumska Optimizacija I Kompromisno Resenje*
VQA	*Visual Question Answering*
VTOL	*Vertical Take-Off and Landing*

Appendix B

Functions Used by MATLAB

readS1Image.m

```
function imgData = readS1Image(metaData)
    % The function READS1IMAGE takes metadata as input and returns the image
    % data structure as output.

    % Full path to TIFF files
    tiffFiles = fullfile(metaData.rootDir, metaData.dataFile);
    % Read TIFF image file
    [r, c] = size(tiffFiles);

    % Output Structure
    imgData = struct;
    for iRow = 1:r
        polIm = struct;
        for jCol = 1:c
            file = tiffFiles{iRow, jCol};
            prodType = metaData.productType;
            [data, info] = readTiffImage(file, prodType);
            fName = metaData.dataFile{iRow, jCol};
            % Add details to sub-structure
            polIm.file = fName;
            polIm.data = data;
            polIm.info = info;

            % Construct field name for output structure
            fieldName = string(extractBetween(fName, 17, 19)) +
string(extractBetween(fName, 24, 25));
            fieldName = replace(fieldName, '-', '_');
            imgData.(fieldName) = polIm;
        end
    end
end

function [data, fileInfo] = readTiffImage(inpFile, prodType)
    % The function READTIFFIMAGE takes the Sentinel-1 TIFF file and product
    % type as input and returns the image data and image information as output.

    % Open the input file as little-endian
    fid = fopen(inpFile, 'r', 'ieee-le');

    % Get image information from input TIFF data file
    fileInfo = imfinfo(inpFile);
    fileInfo.dataType = 'int16';

    % Check the product type
    if strcmp(prodType, 'GRD')
        fileInfo.complexFlag = 0;
    elseif strcmp(prodType, 'SLC')
        fileInfo.complexFlag = 1;
    end

    % Number of range samples and azimuth lines in the data file
    Nrg = fileInfo.Width;
    Naz = fileInfo.Height;

    % Calculate bytes offsets for the ROI
    roiBytesOffset = fileInfo.StripOffsets;
    fseek(fid, roiBytesOffset(1), -1);

    % Dimensions of the data array to read
    dataDim = [Nrg * (1 + fileInfo.complexFlag), Naz];
```

```
    % Data type (save as single)
    dtype = fileInfo.dataType;
    if ~strcmpi(fileInfo.dataType, 'single')
        dtype = [dtype, '=>single'];
    end
    data = fread(fid, dataDim, dtype);

    % Convert samples to complex data if needed
    if fileInfo.complexFlag
        data = complex(data(1:2:end, :), data(2:2:end, :));
    end
    fclose(fid);
end
```

parseSafeFile.m

```
function metaInfo = parseSafeFile(rootDir)
    % The function PARSESAFEFILE takes the root directory name as input and
    % extracts information from the SAFE manifest file of the Sentinel-1 L-1 dataset

    import matlab.io.xml.dom.*

    % Read the input manifest file
    maniFile = fullfile(rootDir, 'manifest.safe');
    try
        xdoc = parseFile(Parser, maniFile);
    catch err
        rethrow(err);
    end

    % Root node of the tree
    xroot = xdoc.getDocumentElement;
    % Check if the input file is indeed a manifest file
    if ~strcmp(xroot.getNodeName, 'xfdu:XFDU')
        error('The input file is not a SAFE manifest file.');
    end

    % Read metadataObject section
    meta = xroot.getElementsByTagName('metadataSection').item(0);
    % Get satellite platform name
    platform = lower(getXMLnode('safe:familyName', meta, 'str'));
    % Capitalize first letter
    platform = regexprep(platform, '(\<\w)', '${upper($1)}');
    % Make sure the platform is indeed Sentinel-1
    assert(strcmpi(platform, 'sentinel-1'), ['The platform in the manifest file is not
supported: %s'], platform);
    % Add platform number to the name
    platform = strcat(platform, getXMLnode('safe:number', meta, 'str'));
    % Get orbit pass
    pass = lower(getXMLnode('s1:pass', meta, 'str'));
    % Get product type
    prodType = getXMLnode('s1sarl1:productType', meta, 'str');
    % Make sure the product is SLC or GRD
    prodTest = any(strcmpi(prodType, {'SLC', 'GRD'})); % Test for product type
    assert(prodTest, ['The product type in the manifest file is not supported: %s'],
prodType);
    % Get acquisition mode
    acqMode = getXMLnode('s1sarl1:mode', meta, 'str');
    % Get number of subswaths and IDs
    node = meta.getElementsByTagName('s1sarl1:swath');
    numSwath = node.getLength;
    swathID = cell(1, numSwath);
    % Get IDs for all subswaths
    for cnt = 1:numSwath
        swathID{cnt} = char(node.item(cnt-1).getTextContent);
    end

    % Get polarizations
```

```
node = meta.getElementsByTagName('s1sarl1:transmitterReceiverPolarisation');
numPol = node.getLength;
pol = cell(1, numPol);
% Get polarization names
for cnt = 1:numPol
    pol{cnt} = char(node.item(cnt-1).getTextContent);
end

% Number of data channels
if ~strcmpi(acqMode, 'WV')
    % IW, EW or SM mode
    numChan = length(pol) * length(swathID);
else
    % WV mode: number of vignettes
    FPnode = meta.getElementsByTagName('gml:coordinates'); % Vignettes coords
    numChan = FPnode.getLength;
end

% Get acquisition start and stop times
acqT0 = getXMLnode('safe:startTime', meta, 'str');
acqTE = getXMLnode('safe:stopTime', meta, 'str');

% Geographic coordinates of swath
if ~strcmpi(acqMode, 'WV')
    % IW, EW or SM mode
    FPnode = meta.getElementsByTagName('gml:coordinates');
    swathCoords = str2num(FPnode.item(0).getTextContent);
    swathCoords = reshape(swathCoords, 2, []).';
else
    % WV mode
    swathCoords = cell(1, numChan);
    for nc = 1:numChan
        swathCoords{nc} = str2num(FPnode.item(nc-1).getTextContent);
        swathCoords{nc} = reshape(swathCoords{nc}, 2, []).';
    end
end

% Read dataObject section
data = xroot.getElementsByTagName('dataObjectSection').item(0);

% Initialize storage for different files
dataFile = cell(1, numChan);
prodAnnFile = cell(1, numChan);
calAnnFile = cell(1, numChan);
noiseAnnFile = cell(1, numChan);
cntData = 1;
cntProd = 1;
cntCal = 1;
cntNoise = 1;
% Get data and annotations files names (attributes)
for cnt = 1:data.getLength
    % Only process current item if it is not empty
    if data.item(cnt-1).hasAttributes
        % repID attribute name
        repID = char(data.item(cnt-1).getAttribute('repID'));
        % Get fileLocation node
        loc = data.item(cnt-1).getElementsByTagName('fileLocation').item(0);
        % Determine the type of file
        if strcmp(repID, 's1Level1MeasurementSchema')
            % Measurement dataset
            str = char(loc.getAttribute('href'));
            dataFile{cntData} = str(3:end); % Remove './'
            cntData = cntData + 1;
        elseif strcmp(repID, 's1Level1ProductSchema')
            % Product annotation
            str = char(loc.getAttribute('href'));
            prodAnnFile{cntProd} = str(3:end); % Remove './'
            cntProd = cntProd + 1;
        elseif strcmp(repID, 's1Level1CalibrationSchema')
            % Calibration annotation
```

```
                str = char(loc.getAttribute('href'));
                calAnnFile{cntCal} = str(3:end); % Remove './'
                cntCal = cntCal + 1;
            elseif strcmp(repID, 's1Level1NoiseSchema')
                % Noise annotation
                str = char(loc.getAttribute('href'));
                noiseAnnFile{cntNoise} = str(3:end); % Remove './'
                cntNoise = cntNoise + 1;
            end
        end
    end
    % Number of vignettes
    numV = numChan / numSwath / numPol;

    % Sort data and annotation files
    dataFileS = cell(numSwath, numPol, numV);
    prodAnnS = cell(numSwath, numPol, numV);
    calAnnS = cell(numSwath, numPol, numV);
    noiseAnnS = cell(numSwath, numPol, numV);
    for ns = 1:numSwath
        sflag = contains(dataFile, swathID{ns}, 'IgnoreCase', true);
        assert(sum(sflag) == numPol * numV, 'Missing data file(s) for swath %s.',
swathID{ns});
        for np = 1:numPol
            pflag = contains(dataFile, pol{np}, 'IgnoreCase', true);
            assert(sum(pflag) == numSwath * numV, 'Missing data file(s) for polarization
%s.', pol{np});
            dataFileS(ns, np, :) = dataFile(sflag & pflag);
            prodAnnS(ns, np, :) = prodAnnFile(sflag & pflag);
            calAnnS(ns, np, :) = calAnnFile(sflag & pflag);
            noiseAnnS(ns, np, :) = noiseAnnFile(sflag & pflag);
        end
    end

    % Construct the manifest header structure
    metaInfo.rootDir = fileparts(maniFile); % Root directory
    metaInfo.platform = platform; % Satellite platform
    metaInfo.orbitPass = pass; % Orbit pass
    metaInfo.productType = prodType; % Product type
    metaInfo.acqMode = acqMode; % Acquisition mode
    metaInfo.numSwath = numSwath; % Number of subswaths
    metaInfo.swathID = swathID; % Subswaths IDs
    metaInfo.numPolarization = numPol; % Number of polarization channels
    metaInfo.polarization = pol; % Polarization channels
    if strcmpi(acqMode, 'WV')
        % WV mode: number of vignettes
        metaInfo.Nvignette = numV;
    end
    metaInfo.acqStartTime = acqT0; % Acquisition start time
    metaInfo.acqStopTime = acqTE; % Acquisition stop time
    metaInfo.swathCoords = swathCoords; % Geographic coordinates of swath's corners
    metaInfo.dataFile = dataFileS; % Data files names
    metaInfo.prodAnnFile = prodAnnS; % Product annotation files names
    metaInfo.calAnnFile = calAnnS; % Calibration annotation files names
    metaInfo.noiseAnnFile = noiseAnnS; % Noise annotation files names
end
```

parseAnn.m

```
function annData = parseAnn(xmlFile)
    % The function PARSEANN takes the annotation XML file as input and returns
    % the extracted meta information as output.

    import matlab.io.xml.dom.*

    % Read the input XML file
    try
        xdoc = parseFile(Parser, xmlFile);
```

```
    catch
        error('Failed to read the input XML file: %s', xmlFile);
    end

    % Root node of the XML file
    xroot = xdoc.getDocumentElement;

    % Extracting adsHeader
    adsHead = xroot.getElementsByTagName('adsHeader').item(0);
    adsHeader = struct;
    adsHeader.missionId = getXMLnode('missionId', adsHead, 'str');
    adsHeader.productType = getXMLnode('productType', adsHead, 'str');
    adsHeader.polarisation = getXMLnode('polarisation', adsHead, 'str');
    adsHeader.mode = getXMLnode('mode', adsHead, 'str');
    adsHeader.swath = getXMLnode('swath', adsHead, 'str');
    adsHeader.startTime = getXMLnode('startTime', adsHead, 'str');
    adsHeader.stopTime = getXMLnode('stopTime', adsHead, 'str');
    adsHeader.absoluteOrbitNumber = getXMLnode('absoluteOrbitNumber', adsHead, 'int');
    adsHeader.missionDataTakeId = getXMLnode('missionDataTakeId', adsHead, 'str');
    adsHeader.imageNumber = getXMLnode('imageNumber', adsHead, 'int');

    % Extracting geolocationGridPointList
    ggpListHead = xroot.getElementsByTagName('geolocationGridPointList').item(0);
    ggpListHeader = struct;
    ggpListCount = str2double(ggpListHead.getAttribute('count'));
    ggpListHeader.count = ggpListCount;
    for cnt = 1:ggpListCount
        gridPoint = ggpListHead.getElementsByTagName('geolocationGridPoint').item(cnt-1);
        ggpListHeader(cnt).line = getXMLnode('line', gridPoint, 'dbl');
        ggpListHeader(cnt).pixel = getXMLnode('pixel', gridPoint, 'dbl');
        ggpListHeader(cnt).latitude = getXMLnode('latitude', gridPoint, 'dbl');
        ggpListHeader(cnt).longitude = getXMLnode('longitude', gridPoint, 'dbl');
        ggpListHeader(cnt).height = getXMLnode('height', gridPoint, 'dbl');
        ggpListHeader(cnt).incidenceAngle = getXMLnode('incidenceAngle', gridPoint,
'dbl');
        ggpListHeader(cnt).elevationAngle = getXMLnode('elevationAngle', gridPoint,
'dbl');
    end

    % Extracting qualityInformation
    qInfo = xroot.getElementsByTagName('qualityInformation').item(0);
    qualityData = struct;
    % substruct 1: downlinkQuality
    downlinkQuality = struct;
    downlinkQuality.iInputDataMean = getXMLnode('iInputDataMean', qInfo, 'dbl');
    downlinkQuality.qInputDataMean = getXMLnode('qInputDataMean', qInfo, 'dbl');
    downlinkQuality.iInputDataStdDev = getXMLnode('iInputDataStdDev', qInfo, 'dbl');
    downlinkQuality.qInputDataStdDev = getXMLnode('qInputDataStdDev', qInfo, 'dbl');
    % substruct 2: rawDataAnalysisQuality
    rawDataAnalysisQuality = struct;
    rawDataAnalysisQuality.iBias = getXMLnode('iBias', qInfo, 'dbl');
    rawDataAnalysisQuality.qBias = getXMLnode('qBias', qInfo, 'dbl');
    rawDataAnalysisQuality.iqGainImbalance = getXMLnode('iqGainImbalance', qInfo, 'dbl');
    rawDataAnalysisQuality.iqQuadratureDeparture = getXMLnode('iqQuadratureDeparture',
qInfo, 'dbl');
    % substruct 3: imageStatistics
    imageStatistics = struct;
    % substruct 3.1: outputDataMean
    imageStatistics.outputDataMean = struct;
    opMeanData = xroot.getElementsByTagName('outputDataMean').item(0);
    imageStatistics.outputDataMean.re = getXMLnode('re', opMeanData, 'dbl');
    imageStatistics.outputDataMean.im = getXMLnode('im', opMeanData, 'dbl');
    % substruct 3.2: outputDataStdDev
    imageStatistics.outputDataStdDev = struct;
    opStdDev = xroot.getElementsByTagName('outputDataStdDev').item(0);
    imageStatistics.outputDataStdDev.re = getXMLnode('re', opStdDev, 'dbl');
    imageStatistics.outputDataStdDev.im = getXMLnode('im', opStdDev, 'dbl');

    % Extracting image processing information
    processingInformation = struct;
```

```
    processingInfo = xroot.getElementsByTagName('processingInformation').item(0);
    processingInformation.rawDataAnalysisUsed = getXMLnode('rawDataAnalysisUsed',
processingInfo, 'str');
    processingInformation.orbitDataFileUsed = getXMLnode('orbitDataFileUsed',
processingInfo, 'str');
    processingInformation.attitudeDataFileUsed = getXMLnode('attitudeDataFileUsed',
processingInfo, 'str');
    processingInformation.rxVariationCorrectionApplied =
getXMLnode('rxVariationCorrectionApplied', processingInfo, 'str');
    processingInformation.antennaElevationPatternApplied =
getXMLnode('antennaElevationPatternApplied', processingInfo, 'str');
    processingInformation.antennaAzimuthPatternApplied =
getXMLnode('antennaAzimuthPatternApplied', processingInfo, 'str');
    processingInformation.antennaAzimuthElementPatternApplied =
getXMLnode('antennaAzimuthElementPatternApplied', processingInfo, 'str');
    processingInformation.dcMethod = getXMLnode('dcMethod', processingInfo, 'str');
    processingInformation.dcInputData = getXMLnode('dcInputData', processingInfo, 'str');
    processingInformation.rangeSpreadingLossCompensationApplied =
getXMLnode('rangeSpreadingLossCompensationApplied', processingInfo, 'str');
    processingInformation.srgrConversionApplied = getXMLnode('srgrConversionApplied',
processingInfo, 'str');
    processingInformation.detectionPerformed = getXMLnode('detectionPerformed',
processingInfo, 'str');
    processingInformation.thermalNoiseCorrectionPerformed =
getXMLnode('thermalNoiseCorrectionPerformed', processingInfo, 'str');

    % Adding to qualityData structure
    qualityData.downlinkQuality = downlinkQuality;
    qualityData.rawDataAnalysisQuality = rawDataAnalysisQuality;
    qualityData.imageStatistics = imageStatistics;
    qualityData.processingInformation = processingInformation;

    % Extracting generalAnnotation
    genAnn = xroot.getElementsByTagName('generalAnnotation').item(0);
    generalAnnotation = struct;
    generalAnnotation.pass = getXMLnode('pass', genAnn, 'str');
    generalAnnotation.timelinessCategory = getXMLnode('timelinessCategory', genAnn,
'str');
    generalAnnotation.platformHeading = getXMLnode('platformHeading', genAnn, 'dbl');
    generalAnnotation.projection = getXMLnode('projection', genAnn, 'str');
    generalAnnotation.rangeSamplingRate = getXMLnode('rangeSamplingRate', genAnn, 'dbl');
    generalAnnotation.radarFrequency = getXMLnode('radarFrequency', genAnn, 'dbl');
    generalAnnotation.azimuthSteeringRate = getXMLnode('azimuthSteeringRate', genAnn,
'dbl');

    % Extracting imageAnnotation
    imgAnn = xroot.getElementsByTagName('imageAnnotation').item(0);
    imageAnnotation = struct;
    imageAnnotation.productFirstLineUtcTime = getXMLnode('productFirstLineUtcTime',
imgAnn, 'dateStr');
    imageAnnotation.productLastLineUtcTime = getXMLnode('productLastLineUtcTime', imgAnn,
'dateStr');
    imageAnnotation.slantRangeTime = getXMLnode('slantRangeTime', imgAnn, 'dbl');
    imageAnnotation.pixelValue = getXMLnode('pixelValue', imgAnn, 'str');
    imageAnnotation.outputPixels = getXMLnode('outputPixels', imgAnn, 'str');
    imageAnnotation.rangePixelSpacing = getXMLnode('rangePixelSpacing', imgAnn, 'dbl');
    imageAnnotation.azimuthPixelSpacing = getXMLnode('azimuthPixelSpacing', imgAnn,
'dbl');
    imageAnnotation.azimuthTimeInterval = getXMLnode('azimuthTimeInterval', imgAnn,
'dbl');
    imageAnnotation.azimuthFrequency = getXMLnode('azimuthFrequency', imgAnn, 'dbl');
    imageAnnotation.numberOfSamples = getXMLnode('numberOfSamples', imgAnn, 'dbl');
    imageAnnotation.numberOfLines = getXMLnode('numberOfLines', imgAnn, 'dbl');
    imageAnnotation.zeroDopMinusAcqTime = getXMLnode('zeroDopMinusAcqTime', imgAnn,
'dbl');
    imageAnnotation.incidenceAngleMidSwath = getXMLnode('incidenceAngleMidSwath', imgAnn,
'dbl');

    % Adding to annData structure
    annData.adsHeader = adsHeader;
```

```
    annData.qualityData = qualityData;
    annData.generalAnnotation = generalAnnotation;
    annData.imageAnnotation = imageAnnotation;
    annData.geolocationGridPoints = ggpListHeader;
end
```

calibrateImage.m

```
function imgCal = calibrateImage(img, calFile)
    % The function CALIBRATE performs radiometric calibration on the input image
    % and returns the sigma-nought calibrated image as output.
    % Reference: https://sentinel.esa.int/documents/247904/685163/S1-Radiometric-
Calibration-V1.0.pdf

    % Image size
    [Nrg, Naz] = size(img);

    % Get the calibration meta information
    calAnnData = parseCal(calFile);

    % Get calibration range pixel data and azimuthal lines
    calRg = calAnnData(1).pixel;
    NrgCal = length(calRg);
    calAz = [calAnnData.line];

    % LUT Array table
    calLutArr = [calAnnData.sigmaNought];
    calLutArr = reshape(calLutArr, NrgCal, []);

    % Interpolate LUT data
    % Create a mesh grid of calibration range pixels and azimuth lines
    [calAzMeshGrid, calRgMeshGrid] = meshgrid(calAz, calRg);

    % Create a mesh grid of interpolation range pixels and azimuth lines
    [imgAz, imgRg] = meshgrid((0:Naz-1), (0:Nrg-1));

    % Perform a bilinear interpolation of the desired LUT matrix
    calLutArrInp = interp2(calAzMeshGrid, calRgMeshGrid, calLutArr, imgAz, imgRg, ...
        'linear', nan);
    calLutArrInp = reshape(calLutArrInp, Nrg, Naz);

    % Apply LUT to input image
    img = abs(img).^2;  % Calculate the magnitude squared of the image pixel values
    calLutArrInp = calLutArrInp.^2;

    % Apply calibration LUT to image
    imgCal = img ./ calLutArrInp;
end


function calAnnData = parseCal(xmlFile)
    % The function PARSECAL reads the input calibration annotation file and
    % returns the retrieved meta information structure.

    import matlab.io.xml.dom.*;

    % Read the input XML file
    try
        xdoc = parseFile(Parser, xmlFile);
    catch
        error('Failed to read the input XML file: %s', xmlFile);
    end
    xroot = xdoc.getDocumentElement; % Root node of the XML file

    % Form the output struct
    calAnnData = struct;

    % Reading the sigma nought LUT present in calibration vectors
```

```
    calVecList = xroot.getElementsByTagName('calibrationVectorList').item(0);
    count = str2double(calVecList.getAttribute('count'));

    for cnt = 1:count
        calVec = calVecList.getElementsByTagName('calibrationVector').item(cnt-1);
        calAnnData(cnt).azimuthTime = getXMLnode('azimuthTime', calVec, 'str');
        calAnnData(cnt).line = getXMLnode('line', calVec, 'dblArr');
        calAnnData(cnt).pixel = getXMLnode('pixel', calVec, 'dblArr');
        calAnnData(cnt).sigmaNought = getXMLnode('sigmaNought', calVec, 'dblArr');
    end
end
```

getXMLnode.m

```
function data = getXMLnode( tag, pnode, varargin )
% The function GETXMLNODE extracts the specified tag from a node in a XML tree.
% validate the number of input parameters
narginchk( 2, 5 );

% set default values for optional inputs
cnt = 0;
type = 'str';
dateFmt = '';

% list of supported data types
dtypeList = {'str', 'dbl', 'dblArr', 'int', 'intArr', 'uint', 'uintArr', ...
    'dateStr', 'dateVec'};

% list of supported fields in the format string for a date vector
dateVecList = {'year', 'mon', 'day', 'hour', 'min', 'sec', 'msec', 'usec'};

% check the optional inputs
if nargin > 2
    % optional inputs provided, assign each value to the correct variable
    for idx = 1:nargin-2
        if isnumeric( varargin{idx} )
            % current input is the occurence number for the tag
            cnt = varargin{idx};
        elseif any( strcmpi( varargin{idx}, dtypeList ) )
            % current input is the data type for the tag
                type = varargin{idx};
        else
            % current input is the date format string, or is invalid
            dateFmt = varargin{idx};
        end
    end
end

% try to extract the element
try
    ndata = pnode.getElementsByTagName(tag).item(cnt).getTextContent;
catch
    error( 'Non-existent node with name "%s"', tag );
end

% convert data to the specified type
if strcmpi( type, 'str' )
    % string
    data = char( ndata );
elseif strncmpi( type, 'date', 4 )
    % date vector or string

    % convert tag data according to the provided type
    if strcmpi( type, 'dateStr' )
        % date string type, convert to string
        ndata = char( ndata );
    else
        % date vector type, convert data vector to doubles
```

```
        ndata = str2num( ndata );
    end

    if isempty( dateFmt )
        % date format string not provided, return converted data directly
        data = ndata;
    else
        % date format string provided, use it to correctly convert data
        if strcmpi( type, 'dateStr' )
            % date string type, process string

            % find sub-second portion of the date string, if any
            subSecIdx = strfind( dateFmt, 'F' ); % sub-second part of the string

            % convert string to date vector
            if isempty( subSecIdx )
                % no sub-second part in date string, use it directly
                data = datevec( ndata, dateFmt );
            else
                % sub-second part present in date string, handle it separately
                data = datevec( ndata(1:subSecIdx(1)-1), ...
                    dateFmt(1:subSecIdx(1)-1) );
                data(6) = data(6) + str2double( ndata(subSecIdx) ) / ...
                    10^(length(subSecIdx));
            end
        else
            % date vector type, process data vector

            % split the format string at the commas
            dFields = strsplit( dateFmt, ',' );

            % construct the date vector by looping over the elements
            data = zeros( 1, 6 ); % initialize the date vector
            for nel = 1:6
                % search the current element in the format string
                elIdx = strcmp( dFields, dateVecList{nel} );

                % if it exists, copy the current element in the date vector
                if ~isempty( elIdx )
                    if nel < 6
                        % for the first five elements, round the data
                        data(nel) = round( ndata(elIdx) );
                    else
                        % for the last element, also check for sub-seconds data
                        msecIdx = strcmp( dFields, 'msec' ); % milliseconds
                        usecIdx = strcmp( dFields, 'usec' ); % microseconds
                        if ~isempty( usecIdx )
                            % microsecond element present in date vector
                            data(6) = round( ndata(elIdx) ) + ...
                                ndata(usecIdx) / 1e6;
                        elseif ~isempty( msecIdx )
                            % millisecond element present in date vector
                            data(6) = round( ndata(elIdx) ) + ...
                                ndata(msecIdx) / 1e3;
                        else
                            % no sub-second component in date vector
                            data(6) = ndata(elIdx);
                        end
                    end
                end
            end
        end
    end
else
    % numeric value or array

    % check if an array type is specified
    if ~strcmpi( type(end-2:end), 'Arr' )
        % single value, convert to double for now
        tmpDbl = str2double( ndata );
```

```
    else
        % array, convert to an array of doubles for now
        tmpDbl = str2num( ndata );
    end

    % convert double value(s) to another numeric type, if necessary
    if strncmpi( type, 'uint', 4 )
        % unsigned integer(s)
        data = uint64( tmpDbl );
    elseif strncmpi( type, 'int', 3 )
        % signed integer(s)
        data = int64( tmpDbl );
    else
        % double(s)
        data = tmpDbl;
    end
end
```

removeThermalNoise.m

```
function imgTnr = removeThermalNoise(img, noiseFile)
    % The function REMOVETHERMALNOISE removes the thermal noise from the input
    % image and returns the noise-corrected image as output.

    % Image size
    [Nrg, Naz] = size(img);
    % Extract relevant information from the noise annotation file
    [noiseRangeVectorList, noiseAzimuthVectorList] = parseNoise(noiseFile);

    % Get the pixel and lines for Noise Range Vector List
    pixels = noiseRangeVectorList(1).pixel;
    lenPixels = length(pixels);
    lineRange = [noiseRangeVectorList.line];
    % LUT Array table
    rangeLut = [noiseRangeVectorList.noiseRangeLut];
    rangeLut = reshape(rangeLut, lenPixels, []);
    [rangeAzMeshGrid, rangeRgMeshGrid] = meshgrid(lineRange, pixels);

    % Create a mesh grid of interpolation range pixels and azimuth lines
    [imgAz, imgRg] = meshgrid((0:Naz-1), (0:Nrg-1));

    % Perform a bilinear interpolation of the desired LUT matrix
    rangeLUTInp2 = interp2(rangeAzMeshGrid, rangeRgMeshGrid, rangeLut, imgAz, imgRg, ...
        'linear', NaN);
    rangeMatrix = reshape(rangeLUTInp2, Nrg, Naz);

    % Initialize to store the azimuth matrix
    azimuthMatrix = zeros(Nrg, Naz);
    numFlag = 0;
    for b = 1:length(noiseAzimuthVectorList)
        noiseAzimuthVector = noiseAzimuthVectorList(b);
        numOfSamples = noiseAzimuthVector.lastRangeSample -
noiseAzimuthVector.firstRangeSample + 1;
        numOfLines = noiseAzimuthVector.lastAzimuthLine -
noiseAzimuthVector.firstAzimuthLine + 1;
        sampleIndex = linspace(noiseAzimuthVector.firstRangeSample,
noiseAzimuthVector.lastRangeSample, numOfSamples);
        lineIndex = linspace(noiseAzimuthVector.firstAzimuthLine,
noiseAzimuthVector.lastAzimuthLine, numOfLines);

        nsLines = noiseAzimuthVector.line;
        nsLUTs = noiseAzimuthVector.noiseAzimuthLut;

        [nsLines, ~, idx] = unique(nsLines, 'stable');
        nsLUTs = accumarray(idx, nsLUTs)';

        subAzimuthMat = interp1(nsLines, nsLUTs, lineIndex, 'linear');
        subAzimuthMat = repmat(subAzimuthMat, numOfSamples, 1);
```

```
        azimuthMatrix(1+numFlag:numFlag+numOfSamples, 1:numOfLines) = subAzimuthMat;
        numFlag = numFlag + numOfSamples;
    end

    % Create the noise correction matrix
    noiseLUTArr = rangeMatrix .* azimuthMatrix;
    % Calculate the magnitude squared of the image pixel values
    img = abs(img) .^2;
    % Apply the noise LUT array on the image data
    img = max((img - noiseLUTArr), 0);
    imgTnr = sqrt(img);
end


function [noiseRangeVectorList, noiseAzimuthVectorList] = parseNoise(xmlFile)
    % This function extracts the meta information from the noise annotation
    % file and returns the range and azimuth noise vector lists.

    import matlab.io.xml.dom.*;

    % Read the input XML file
    try
        xdoc = parseFile(Parser, xmlFile);
    catch
        error('Failed to read the input XML file: %s', xmlFile);
    end
    xroot = xdoc.getDocumentElement; % Root node of the XML file

    % Form the output struct for range vectors
    noiseRangeVectorList = struct;

    % Read the range denoising LUT and other info
    noiseRgVecList = xroot.getElementsByTagName('noiseRangeVectorList').item(0);
    rgCount = str2double(noiseRgVecList.getAttribute('count'));

    for cnt = 1:rgCount
        noiseRgVec = noiseRgVecList.getElementsByTagName('noiseRangeVector').item(cnt-1);
        noiseRangeVectorList(cnt).azimuthTime = getXMLnode('azimuthTime', noiseRgVec,
'str');
        noiseRangeVectorList(cnt).line = getXMLnode('line', noiseRgVec, 'dblArr');
        noiseRangeVectorList(cnt).pixel = getXMLnode('pixel', noiseRgVec, 'dblArr');
        noiseRangeVectorList(cnt).noiseRangeLut = getXMLnode('noiseRangeLut', noiseRgVec,
'dblArr');
    end

    % Form the output struct for azimuth vectors
    noiseAzimuthVectorList = struct;

    % Reading the azimuth denoising LUT and other info
    noiseAzimuthVecList = xroot.getElementsByTagName('noiseAzimuthVectorList').item(0);
    azCount = str2double(noiseAzimuthVecList.getAttribute('count'));

    for cnt = 1:azCount
        noiseAzimuthVec =
noiseAzimuthVecList.getElementsByTagName('noiseAzimuthVector').item(cnt-1);
        noiseAzimuthVectorList(cnt).swath = getXMLnode('swath', noiseAzimuthVec, 'str');
        noiseAzimuthVectorList(cnt).firstAzimuthLine = getXMLnode('firstAzimuthLine',
noiseAzimuthVec, 'dblArr');
        noiseAzimuthVectorList(cnt).firstRangeSample = getXMLnode('firstRangeSample',
noiseAzimuthVec, 'dblArr');
        noiseAzimuthVectorList(cnt).lastAzimuthLine = getXMLnode('lastAzimuthLine',
noiseAzimuthVec, 'dblArr');
        noiseAzimuthVectorList(cnt).lastRangeSample = getXMLnode('lastRangeSample',
noiseAzimuthVec, 'dblArr');
        noiseAzimuthVectorList(cnt).line = getXMLnode('line', noiseAzimuthVec, 'dblArr');
        noiseAzimuthVectorList(cnt).noiseAzimuthLut = getXMLnode('noiseAzimuthLut',
noiseAzimuthVec, 'dblArr');
    end
end
```

Appendix C

1. Python scripts used for SAR images streaming

ProducerSARChunks.py

```
from confluent_kafka import Producer
import os
import shutil
import time

# Function to split the file into chunks
def split_file(file_path, chunk_size=50 * 1024 * 1024):  # 50MB chunks
    with open(file_path, 'rb') as f:
        chunk = f.read(chunk_size)
        while chunk:
            yield chunk
            chunk = f.read(chunk_size)

# Kafka producer configuration with increased buffer sizes
conf = {
    'bootstrap.servers': 'localhost:9092',
    'message.max.bytes': 1000000000,  # maximum message size in bytes (1 GB)
    'batch.size': 64000,  # after tests
    'linger.ms': 50,  # increased linger time for larger batches
    'queue.buffering.max.messages': 200000,  # buffer size (number of messages)
    'queue.buffering.max.kbytes': 2097152,  # buffer size (in kilobytes)
}

producer = Producer(conf)

input_folder = 'UAV_SAR_Images'
history_folder = 'UAV_SAR_Images_History'
failure_folder = 'UAV_SAR_Images_StreamFailure'

# Ensure the history and failure folders exist
if not os.path.exists(history_folder):
    os.makedirs(history_folder)

if not os.path.exists(failure_folder):
    os.makedirs(failure_folder)

# Dictionary to track unacknowledged chunks
unacknowledged_chunks = {}

# Delivery report callback function
def delivery_report(err, msg):
    key = msg.key().decode()
    if err is not None:
        print(f"Message delivery failed for {key}: {err}")
        unacknowledged_chunks[key] = msg.value()  # Store the chunk for retry
    else:
        print(f"Message delivered to {msg.topic()} [{msg.partition()}]: {key}")
        if key in unacknowledged_chunks:
            del unacknowledged_chunks[key]  # Remove from the retry list if
successful

# Function to process files in the input folder
def process_files():
    for file_name in os.listdir(input_folder):
        file_path = os.path.join(input_folder, file_name)

        # Skip directories
        if os.path.isdir(file_path):
            continue
```

```
        print(f"Processing file: {file_name}")

        # Produce each chunk to the 'sar-images' topic
        for i, chunk in enumerate(split_file(file_path)):
            key = f"{file_name}:{i}"
            if chunk is not None:
                try:
                    producer.produce('sar-images', key=key.encode(), value=chunk,
callback=delivery_report)
                    producer.poll(0.1)  # Poll frequently to handle delivery reports
and free up queue space
                except BufferError:
                    print(f"Queue is full. Waiting before retrying...")
                    producer.poll(1)  # Wait to free up space in the queue
                    producer.produce('sar-images', key=key.encode(), value=chunk,
callback=delivery_report)
            else:
                print(f"Skipping null chunk for {key}")

        # Signal the end of the file transmission
        producer.produce('sar-images', key=f"{file_name}:end".encode(), value=b'',
callback=delivery_report)
        producer.flush()  # Ensure all messages are sent

        # Retry unacknowledged chunks
        retry_limit = 5  # Max number of retries
        retry_count = 0
        while unacknowledged_chunks and retry_count < retry_limit:
            print(f"Retrying {len(unacknowledged_chunks)} unacknowledged chunks...")
            for key, chunk in list(unacknowledged_chunks.items()):
                if chunk is not None:
                    try:
                        producer.produce('sar-images', key=key.encode(), value=chunk,
callback=delivery_report)
                        producer.poll(0.1)  # Poll frequently during retries
                    except BufferError:
                        print(f"Queue is full during retry. Waiting before
retrying...")
                        producer.poll(1)
                        producer.produce('sar-images', key=key.encode(), value=chunk,
callback=delivery_report)
                else:
                    print(f"Skipping null chunk for {key}")
            producer.flush()
            retry_count += 1
            time.sleep(2)  # Wait before retrying

        if not unacknowledged_chunks:
            print(f"All chunks for {file_name} delivered successfully.")
            # Move the file to the history folder
            shutil.move(file_path, os.path.join(history_folder, file_name))
        else:
            print(f"Failed to deliver some chunks for {file_name} after {retry_limit}
retries.")
            # Move the file to the failure folder
            shutil.move(file_path, os.path.join(failure_folder, file_name))

# Continuously monitor the input folder for new files
while True:
    process_files()
    print("Monitoring folder for new files...")
    time.sleep(10)  # Check every 10 seconds
```

ConsumerSARChunks.py

```
from confluent_kafka import Consumer, KafkaError
import os

# Kafka consumer configuration
conf = {
    'bootstrap.servers': 'localhost:9092',
    'group.id': 'file-receiver-group',
    'auto.offset.reset': 'earliest',
    'receive.message.max.bytes': 300000000,  # Set to 300MB to accommodate larger
messages
    'api.version.request': False,  # Disable ApiVersionRequest for older brokers
    'socket.timeout.ms': 30000,  # Increase the socket timeout to 30 seconds
    'session.timeout.ms': 60000,  # Increase the session timeout to 60 seconds
}

consumer = Consumer(conf)
consumer.subscribe(['sar-images'])

output_directory = 'received_files'
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Dictionary to hold the file chunks and track the end signal
file_chunks = {}
expected_chunks = {}
end_signal_received = {}

def save_file(file_name, chunks):
    """Reassemble and save the file from its chunks."""
    with open(os.path.join(output_directory, file_name), 'wb') as f_out:
        for chunk in sorted(chunks, key=lambda x: int(x[0])):
            f_out.write(chunk[1])

try:
    while True:
        msg = consumer.poll(1.0)  # Poll for messages with a 1-second timeout

        if msg is None:
            print("Waiting for new messages...")
            continue

        if msg.error():
            if msg.error().code() == KafkaError._PARTITION_EOF:
                continue
            else:
                print(f"Consumer error: {msg.error()}")
                break

        # Extract file name and chunk index from the message key
        key = msg.key()
        if key is not None:
            key = key.decode()

            # Ensure the key has the expected format with a colon separator
            if ":" in key:
                file_name, chunk_index = key.split(":")

                if chunk_index == "end":
                    # Mark that the end signal has been received
                    end_signal_received[file_name] = True
                    print(f"End signal received for '{file_name}'. Waiting for all
chunks.")

                    # Check if all chunks have been received after the end signal
```

```
                    if file_name in expected_chunks and
len(file_chunks.get(file_name, [])) == expected_chunks[file_name]:
                        save_file(file_name, file_chunks[file_name])
                        print(f"File '{file_name}' reassembled and saved.")
                        del file_chunks[file_name]
                        del expected_chunks[file_name]
                        del end_signal_received[file_name]
                else:
                    # Handle the chunk index as an integer
                    try:
                        chunk_index = int(chunk_index)
                        chunk_data = msg.value()

                        if file_name not in file_chunks:
                            file_chunks[file_name] = []
                            expected_chunks[file_name] = 0

                        # Store the chunk data with its index
                        file_chunks[file_name].append((chunk_index, chunk_data))
                        expected_chunks[file_name] = max(expected_chunks[file_name],
chunk_index + 1)
                        print(f"Received chunk {chunk_index} for file '{file_name}'")

                        # After receiving a chunk, check if we've received all
expected chunks and the end signal
                        if file_name in end_signal_received and
len(file_chunks[file_name]) == expected_chunks[file_name]:
                            save_file(file_name, file_chunks[file_name])
                            print(f"File '{file_name}' reassembled and saved.")
                            del file_chunks[file_name]
                            del expected_chunks[file_name]
                            del end_signal_received[file_name]

                    except ValueError:
                        print(f"Invalid chunk index received: {chunk_index}")

            else:
                print(f"Unexpected key format: {key}")

finally:
    # Close down consumer to commit final offsets
    consumer.close()
```

2. Python Scripts Used by the DSS to deploy Random Forest Regression model

server.py

```
from flask import Flask, request, jsonify
import subprocess
import os

app = Flask(__name__)

# Define a safe directory where scripts are allowed to run
SAFE_SCRIPTS_DIRECTORY = ''

@app.route('/run-script', methods=['POST'])
def run_script():
    # Get script name and parameters from the POST request's JSON payload
    data = request.get_json()
    script_name = data.get('script_name')
```

```
    # Extract the parameters based on the script
    predict_for_location = data.get('location', None)
    predict_for_severity = data.get('severity', None)
    predict_for_flood_depth = data.get('flood_depth', None)
    dataset_rows = data.get('dataset_rows', None)
    decision_trees = data.get('decision_trees', None)
    forest_depth = data.get('forest_depth', None)

    # Validate the script name
    if script_name and script_name.endswith('.py'):
        # Build the full script path (ensure it's within the safe directory)
        script_path = os.path.join(SAFE_SCRIPTS_DIRECTORY, script_name)

        if os.path.exists(script_path):
            try:
                # Initialize command with Python executable and script path
                command = ['python', script_path]

                # Conditional logic for each script type
                if script_name == 'predict_Node.py':
                    # Check if required arguments are provided
                    if predict_for_location is None or predict_for_severity is None
or predict_for_flood_depth is None:
                        return jsonify({"error": "Missing 'location' or 'severity' or
'flood_depth' for predict_Node.py."}), 400
                    # Append location, severity and flood_depth (severity,
flood_depth are converted to string)
                    command.extend([predict_for_location, str(predict_for_severity),
str(predict_for_flood_depth)])

                elif script_name == 'create_dataset_Node.py':
                    # Check if dataset_rows is provided
                    if dataset_rows is None:
                        return jsonify({"error": "Missing 'dataset_rows' for
create_dataset_Node.py."}), 400
                    # Append dataset_rows as an argument
                    command.append(str(dataset_rows))

                elif script_name == 'train_RF_Node.py':
                    # Check if decision_trees and forest_depth are provided
                    if decision_trees is None or forest_depth is None:
                        return jsonify({"error": "Missing 'decision_trees' or
'forest_depth' for train_RF_Node.py."}), 400
                    # Append decision_trees and forest_depth as arguments
                    command.extend([str(decision_trees), str(forest_depth)])

                else:
                    return jsonify({"error": "Unsupported script name."}), 400

                # Execute the script using subprocess
                result = subprocess.run(command, capture_output=True, text=True)

                # Check the result and return appropriate response
                if result.returncode == 0:
                    return jsonify(
                        {"message": f"Script {script_name} executed successfully.",
"output": result.stdout}), 200
                else:
                    return jsonify({"error": f"Script {script_name} execution
failed.", "stderr": result.stderr}), 500
            except Exception as e:
                return jsonify({"error": str(e)}), 500
        else:
            return jsonify({"error": "Script not found."}), 404
    else:
        return jsonify({"error": "Invalid script name."}), 400
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

create_dataset_Node.py

```
import os
import pandas as pd
import random
from datetime import datetime, timedelta
import math
import json
import argparse

# Define Location: Population values
locations = {
    "Larissa": 162591, "Volos": 144420, "Trikala": 81455, "Karditsa": 56148,
    "Elassona": 38114, "Farsala": 11127, "Kalambaka": 21763, "Tyrnavos": 16900,
    "Almyros": 17795, "Falani": 4350, "Mouzaki": 6060, "Palamas": 5561,
    "Makrychori": 1892, "Agia": 5169, "Domokos": 4616, "Dasochori": 547,
"Omorfochori": 362
}

flood_severity = ['Minor', 'Moderate', 'Severe', 'Catastrophic']

material_resources = ['Sandbags', 'Water Pumps', 'Food Supplies', 'Medical Kits',
'Life Vests',
                      'Clothing', 'Boots', 'Drinking Water', 'Living Tents', 'Heavy
Duty Gloves',
                      'Antiseptics', 'Medicines & Antibiotics']

human_resources = ['Firefighters', 'Paramedics', 'Police', 'Rescue Divers',
'Engineers', 'Search & Rescue', 'Army']

vehicles = ['Truck', 'Ambulance', 'Helicopter', 'Rescue Boat', 'Van', 'Backhoe
Loader', 'Crawler Loader']

resource_rules = {
    'Minor': {
        'Material Resource': ['Sandbags', 'Water Pumps', 'Heavy Duty Gloves',
'Boots'],
        'Quantity': [10000, 50, 500, 500],
        'Human Resource': ['Police', 'Firefighters'],
        'Team Size': [2, 3],
        'Vehicle': ['Van'],
        'Vehicle Count': [1]
    },
    'Moderate': {
        'Material Resource': ['Water Pumps', 'Medical Kits', 'Antiseptics', 'Drinking
Water'],
        'Quantity': [100, 200, 500, 500],
        'Human Resource': ['Firefighters', 'Search & Rescue'],
        'Team Size': [10, 15],
        'Vehicle': ['Truck', 'Ambulance', 'Backhoe Loader', 'Rescue Boat'],
        'Vehicle Count': [1, 2, 2, 10]
    },
    'Severe': {
        'Material Resource': ['Life Vests', 'Food Supplies', 'Medical Kits',
'Clothing', 'Boots', 'Drinking Water', 'Living Tents', 'Antiseptics', 'Medicines &
Antibiotics'],
        'Quantity': [1000, 1000, 500, 1000, 1000, 2000, 500, 4000, 2000],
        'Human Resource': ['Paramedics', 'Engineers', 'Search & Rescue',
'Firefighters'],
        'Team Size': [5, 20, 30, 50],
        'Vehicle': ['Rescue Boat', 'Crawler Loader'],
```

```
        'Vehicle Count': [30, 5]
    },
    'Catastrophic': {
        'Material Resource': ['Life Vests', 'Food Supplies', 'Medical Kits',
'Clothing', 'Boots', 'Drinking Water', 'Living Tents', 'Antiseptics', 'Medicines &
Antibiotics'],
        'Quantity': [3000, 5000, 1000, 2000, 2000, 10000, 2000, 6000, 7000],
        'Human Resource': ['Paramedics', 'Rescue Divers', 'Search & Rescue',
'Firefighters', 'Army'],
        'Team Size': [25, 30, 200, 100, 100],
        'Vehicle': ['Helicopter', 'Rescue Boat'],
        'Vehicle Count': [3, 100]
    }
}

# Generate data
data = []
start_date = datetime(2023, 12, 31)

def get_random_water_level(severity):
    if severity == 'Minor':
        return random.randint(250, 500)
    elif severity == 'Moderate':
        return random.randint(501, 1000)
    elif severity == 'Severe':
        return random.randint(1001, 2000)
    else:  # Catastrophic
        return random.randint(2001, 3000)

def initialize_resource_dict():
    resource_dict = {
        'Date': None, 'Location': None, 'Flood Severity': None, 'Water Level (mm)':
None
    }
    for material in material_resources:
        resource_dict[f"{material} Quantity"] = 0
    for human in human_resources:
        resource_dict[f"{human} Team Size"] = 0
    for vehicle in vehicles:
        resource_dict[f"{vehicle} Count"] = 0
    return resource_dict

# Add argparse to handle command-line arguments
parser = argparse.ArgumentParser(description="Generate flood resource allocation
dataset.")
parser.add_argument("dataset_rows", type=int, help="Number of dataset rows to
generate")

args = parser.parse_args()

# Use the dataset_rows argument to define the loop range
for i in range(args.dataset_rows):
    # Subtract days from the start_date
    date = start_date - timedelta(days=i*30)
    location = random.choice(list(locations.keys()))
    severity = random.choice(flood_severity)
    water_level = get_random_water_level(severity)
    row = initialize_resource_dict()
    row['Date'] = date.strftime("%Y-%m-%d")
    row['Location'] = location
    row['Flood Severity'] = severity
    row['Water Level (mm)'] = water_level
    resources = resource_rules[severity]
    population_factor = math.log1p(locations[location]) / math.log1p(10000)
    for j in range(len(resources['Material Resource'])):
        material = resources['Material Resource'][j]
```

```
        quantity = resources['Quantity'][j] * population_factor
        row[f"{material} Quantity"] += math.ceil(quantity)
    for j in range(len(resources['Human Resource'])):
        human = resources['Human Resource'][j]
        team_size = resources['Team Size'][j] * population_factor
        row[f"{human} Team Size"] += math.ceil(team_size)
    for j in range(len(resources['Vehicle'])):
        vehicle = resources['Vehicle'][j]
        vehicle_count = resources['Vehicle Count'][j] * population_factor
        row[f"{vehicle} Count"] += math.ceil(vehicle_count)
    data.append(row)

df = pd.DataFrame(data)

# Save the CSV file in the same directory as the Python script
script_directory = os.path.dirname(os.path.abspath(__file__))
csv_file_path = os.path.join(script_directory,
'flood_resource_allocation_single_row.csv')

df.to_csv(csv_file_path, index=False)

# Create the returned JSON object
output = {
    "script": "create dataset",
    "csv_path": csv_file_path
}

# Output the JSON string
print(json.dumps(output))
```

train_RF_Node.py

```
import os
import pandas as pd
import argparse
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import joblib
import json
import numpy as np

# Add argparse to handle command-line arguments
parser = argparse.ArgumentParser(description="Train Random Forest model for flood
resource allocation.")
parser.add_argument("decision_trees", type=int, help="Number of decision trees to use
in the Random Forest.")
parser.add_argument("forest_depth", type=int, help="Maximum depth of the trees in the
Random Forest. Set to 0 for no max depth.")
args = parser.parse_args()

# Get the directory where the script is located
script_directory = os.path.dirname(os.path.abspath(__file__))

# Load the dataset from the same directory as the script
csv_file_path = os.path.join(script_directory,
'flood_resource_allocation_single_row.csv')
df = pd.read_csv(csv_file_path)

# Encode categorical variables (Location, Flood Severity)
df_encoded = pd.get_dummies(df, columns=['Location', 'Flood Severity'])

# Drop the 'Date' column (not needed for training)
df_encoded = df_encoded.drop(columns=['Date'])
```

```
# Define features (input) and targets (output)
features = df_encoded.drop(columns=[col for col in df_encoded.columns if 'Quantity'
in col or 'Team Size' in col or 'Count' in col])
targets = df_encoded[[col for col in df_encoded.columns if 'Quantity' in col or 'Team
Size' in col or 'Count' in col]]

# Split data into training (60%), validation (20%), and testing (20%) sets
X_train, X_temp, y_train, y_temp = train_test_split(features, targets, test_size=0.4)
# 40% of the data in temp
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5)  #
Split temp into validation and test sets

# Set up the Random Forest Regressor based on command-line arguments
if args.forest_depth == 0:
    model = RandomForestRegressor(n_estimators=args.decision_trees)  # No
random_state
else:
    model = RandomForestRegressor(n_estimators=args.decision_trees,
max_depth=args.forest_depth)  # No random_state

# Train the Random Forest Regressor
model.fit(X_train, y_train)

# Evaluate on validation set
y_val_pred = model.predict(X_val)
mae_val = round(mean_absolute_error(y_val, y_val_pred), 2)
mse_val = round(mean_squared_error(y_val, y_val_pred), 2)
r2_val = round(r2_score(y_val, y_val_pred), 4)

# Avoid division by zero by adding a small epsilon
epsilon = np.finfo(np.float64).eps
# Calculate Symmetric Mean Absolute Percentage Error (SMAPE) on the validation set
smape_val = round(np.mean(2 * np.abs(y_val - y_val_pred) / (np.abs(y_val) +
np.abs(y_val_pred) + epsilon)) * 100, 2)

# Evaluate on test set (after training, for final performance and overfitting check)
y_test_pred = model.predict(X_test)
mae_test = round(mean_absolute_error(y_test, y_test_pred), 2)
mse_test = round(mean_squared_error(y_test, y_test_pred), 2)
r2_test = round(r2_score(y_test, y_test_pred), 4)

# Calculate Symmetric Mean Absolute Percentage Error (SMAPE) on the test set
smape_test = round(np.mean(2 * np.abs(y_test - y_test_pred) / (np.abs(y_test) +
np.abs(y_test_pred) + epsilon)) * 100, 2)

# Define the model filename (save it in the same directory as the script)
model_filename = os.path.join(script_directory,
'random_forest_flood_resource_allocation_model.pkl')

# Save the trained model to a file
joblib.dump(model, model_filename)

# Output the evaluation metrics for both validation and test sets
evaluation_results = {
    "script": "train model",
    "validation_set_metrics": {
        "mean_absolute_error": mae_val,
        "mean_squared_error": mse_val,
        "r_squared": r2_val,
        "smape": smape_val
    },
    "test_set_metrics": {
        "mean_absolute_error": mae_test,
        "mean_squared_error": mse_test,
        "r_squared": r2_test,
```

```
        "smape": smape_test
    },
    "trained_model_path": model_filename
}

# Print the JSON string to output
evaluation_json = json.dumps(evaluation_results)
print(evaluation_json)
```

predict_Node.py

```
import random
import sys
import pandas as pd
import joblib
import math
import argparse
import json


# Load the trained model
model_filename = 'random_forest_flood_resource_allocation_model.pkl'
model = joblib.load(model_filename)

# Retrieve the trained feature names
if hasattr(model, 'feature_names_in_'):
    feature_columns = model.feature_names_in_
else:
    feature_columns = [
        'Water Level (mm)', 'Location_Agia', 'Location_Almyros', 'Location_Domokos',
        'Location_Elassona', 'Location_Falani', 'Location_Farsala',
'Location_Kalambaka',
        'Location_Karditsa', 'Location_Larissa', 'Location_Makrychori',
'Location_Mouzaki',
        'Location_Palamas', 'Location_Trikala', 'Location_Tyrnavos',
'Location_Volos',
        'Location_Dasochori', 'Location_Omorfochori',
        'Flood Severity_Minor', 'Flood Severity_Moderate', 'Flood Severity_Severe',
        'Flood Severity_Catastrophic'
    ]

# Map locations to encoded columns
location_map = {
    'Larissa': 'Location_Larissa', 'Karditsa': 'Location_Karditsa', 'Farsala':
'Location_Farsala',
    'Agia': 'Location_Agia', 'Almyros': 'Location_Almyros', 'Domokos':
'Location_Domokos',
    'Elassona': 'Location_Elassona', 'Falani': 'Location_Falani', 'Kalambaka':
'Location_Kalambaka',
    'Makrychori': 'Location_Makrychori', 'Mouzaki': 'Location_Mouzaki', 'Palamas':
'Location_Palamas',
    'Trikala': 'Location_Trikala', 'Tyrnavos': 'Location_Tyrnavos', 'Volos':
'Location_Volos',
    'Dasochori': 'Location_Dasochori', 'Omorfochori': 'Location_Omorfochori'
}

# Map severity level input (1, 2, 3, 4) to one-hot encoded columns
severity_map = {
    1: 'Flood Severity_Minor',
    2: 'Flood Severity_Moderate',
    3: 'Flood Severity_Severe',
    4: 'Flood Severity_Catastrophic'
}
```

```
# Parse command-line arguments
parser = argparse.ArgumentParser(description='Flood Resource Allocation Predictor')
parser.add_argument('location', type=str, help='Location for prediction (e.g.,
Larissa)')
parser.add_argument('severity_level', type=int, help='Flood severity level (1, 2, 3,
or 4)')
parser.add_argument('flood_depth', type=int, help='Flood depth (numeric over 250mm)')
args = parser.parse_args()

location_name = args.location
severity_level = args.severity_level
flood_depth = args.flood_depth

# Initialize the new input data with zero values for all encoded locations and
severities
new_data = {col: [0] for col in feature_columns if col.startswith('Location') or
col.startswith('Flood Severity') or col.startswith('Flood Depth')}

# Set the appropriate encoded value for the selected location and severity
if location_name in location_map:
    new_data[location_map[location_name]] = [1]
else:
    sys.stdout.write(json.dumps({"error": f"Location '{location_name}' is not
recognized."}))
    exit()

if severity_level in severity_map:
    new_data[severity_map[severity_level]] = [1]  # Set the one-hot encoded column
for severity to 1
else:
    sys.stdout.write(json.dumps({"error": f"Severity level '{severity_level}' is not
valid. Must be 1, 2, 3, or 4."}))
    exit()

new_data['Water Level (mm)'] = [flood_depth]

# Convert the new data to a DataFrame
new_data_df = pd.DataFrame(new_data)

# Reindex the new data to ensure the columns match the order of the training data
new_data_df = new_data_df.reindex(columns=feature_columns, fill_value=0)

# Make predictions
predictions = model.predict(new_data_df)

# Define the resource columns to map to predictions
resource_columns = [
    'Sandbags Quantity', 'Water Pumps Quantity', 'Food Supplies Quantity', 'Medical
Kits Quantity', 'Life Vests Quantity',
    'Clothing Quantity', 'Boots Quantity', 'Drinking Water Quantity', 'Living Tents
Quantity', 'Heavy Duty Gloves Quantity',
    'Antiseptics Quantity', 'Medicines & Antibiotics Quantity', 'Firefighters Team
Size', 'Paramedics Team Size',
    'Police Team Size', 'Rescue Divers Team Size', 'Engineers Team Size', 'Search &
Rescue Team Size',
    'Army Team Size', 'Truck Count', 'Ambulance Count', 'Helicopter Count', 'Rescue
Boat Count',
    'Van Count', 'Backhoe Loader Count', 'Crawler Loader Count'
]

# Create a list to store the modified json objects
resource_allocation = []

# Function to clean up key names by removing 'Quantity', 'Team Size', 'Count'
def clean_key_name(key):
```

```
    return key.replace(" Quantity", "").replace(" Team Size", "").replace(" Count",
"").strip()

# Add resources with a predicted value greater than zero, rounded up, to the array
for i, resource in enumerate(resource_columns):
    if predictions[0][i] > 0:  # Show only resources where predicted value is greater
than zero
        rounded_value = math.ceil(predictions[0][i])  # Round up the predicted value
        resource_allocation.append({
            "location": location_name,
            "item": clean_key_name(resource),  # Clean the key name
            "quantity": rounded_value,
            "selected": False  # Fixed key-value pair
        })

# Create the final output with the 'script' key and 'decisions' key
output = {
    "script": "predict",
    "decisions": resource_allocation
}

# Output the final JSON object
sys.stdout.write(json.dumps(output, indent=4))
```