UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Software infrastructure for earthquake detection using smartphone's accelerometer

Diploma Thesis

Sotirios Karamellios

Supervisor: Christos Antonopoulos

Volos 2021

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Software infrastructure for earthquake detection using smartphone's accelerometer

Diploma Thesis

Sotirios Karamellios

Supervisor: Christos Antonopoulos

Volos 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Υποδομή λογισμικού για αναγνώριση σεισμών με χρήση επιταχυνσιόμετρου κινητού τηλεφώνου

Διπλωματική Εργασία

Σωτήριος Καραμέλλιος

Επιβλέπων: Χρήστος Αντωνόπουλους

Βόλος 2021

Approved by the Examination Committee:

Supervisor **Christos Antonopoulos**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Spyridon Lalis**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Panagiota Tsompanopoulou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: 8-7-2021

Acknowledgements

I would like to express my gratitude to my family for encouraging me to pursuit my dream and providing me with unconditional support.

Furthermore, I would like to thank Prof. Christos Antonopoulos for the supervision of this diploma thesis. Moreover, I would like to thank to Prof. Panagiota Tsompanopoulou and Prof. Spyridon Lalis for their evaluation and contribution on my thesis.

Last but not least, I would also like to thank Dr. Vassilis Karastathis, Research Director and Deputy Director of the Institute of Geodynamics of the National Observatory of Athens and Dr. Faidra Gkika, Research Scientist, Institute of Geodynamics, National Observatory of Athens for the idea and the introduction to the problem.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Sotirios Karamellios
8-7-2021

Abstract

Earthquakes are responsible for significant injuries, loss of life, and damages to infrastructure every year. Part of this damage could be mitigated by providing earthquake early warning (EEW). Therefore, this topic attracts the attention of many researchers and engineers. EEW systems exploit two observations: the speed and the destructive power of the waves. Due to the fact that the first waves that arrive in a location, the P or Primary waves, are the fastest and the least destructive ones, correctly identifying them can provide the affected areas with time to prepare for the arrival of the more destructive waves. The warning time derives from the relatively low speed of even the fastest waves, along with the distance the waves have to travel. Also here one should mention that the closer to the epicenter you detect the earthquake, the more time slack for notification you have.

The conventional seismic networks, that are currently in use as input to EEW systems, provide only scarce coverage. However, smartphones, which are far more widespread, provide embedded accelerometers, that can be used to identify seismic events. With the use of mobile phone sensors, EEW coverage can be extended in every populated area. We present a smartphone Android application that detects possible earthquakes and sends accelerometer and GPS (Global Positioning System) data to a computer. The gathered information could then be cross-referenced and utilized to classify the warning as a true earthquake or false alarm, calculate the earthquake's epicenter and issue a warning in the distant areas that will be potentially affected. This Thesis discusses the implementation of the application and evaluates its performance over everyday activities.

Περίληψη

Οι σεισμοί ευθύνονται για πολλούς τραυματισμούς, απώλειες ζωών και ζημιές στις υλικές υποδομές κάθε χρόνο. Μέρος αυτής της ζημιάς θα μπορούσε να αποφευχθεί με την ύπαρξη Έγκαιρης Προειδοποίησης για Σεισμούς(ΕΠΣ). Τα συστήματα ΕΠΣ βασίζονται σε δύο παρατηρήσεις: την ταχύτητα και την καταστροφική δύναμη των σεισμικών κυμάτων. Επειδή τα πρώτα κύματα που φτάνουν σε μία τοποθεσία, τα P ή Πρώτα κύματα, είναι τα πιο γρήγορα και τα λιγότερο καταστροφικά, η σωστή αναγνώρισή τους μπορεί να παρέχει στις περιοχές που θα επηρεαστούν από τον σεισμό χρόνο για να προετοιμαστούν για τα πιο καταστροφικά κύματα. Ο διαθέσιμος χρόνος για την προειδοποίηση προέρχεται από την σχετικά χαμηλή ταχύτητα ακομα και των γρηγοροτερων κυματων, σε συνδυασμό με την αποσταση που αυτά χρειάζεται να ταξιδέψουν. Επιπλέον, πρέπει να αναφέρουμε σε αυτο το σημείο ότι όσο πιο κοντά στο επικεντρο ανιχνευθεί ο σεισμός, τόσο περισσότερο χρονικό περιθώριο για προειδοποίηση υπάρχει.

Τα παραδοσιακά σεισμικά δίκτυα, που χρησιμοποιούνται σήμερα ως είσοδος στα συστήματα ΕΠΣ, παρέχουν ελάχιστη μόνο κάλυψη. Παρ' όλ' αυτά, τα κινητά τηλέφωνα, τα οποία είναι πολύ πιο διαδεδομένα, παρέχουν ενσωματωμένα επιταχυνσιόμετρα, τα οποία μπορούν να χρησιμοποιηθούν για να αναγνωριστουν σεισμικά γεγονότα. Με την χρήση των αισθητήρων των κινητών τηλεφώνων, η κάλυψη των συστημάτων ΕΠΣ μπορεί να επεκταθεί σε όλες τις κατοικημένες περιοχές. Παρουσιάζουμε μία εφαρμογή Android για κινητά τηλέφωνα η οποία αναγνωρίζει πιθανούς σεισμούς και στέλνει δεδομένα από το επιταχυνσιόμετρο και το GPS(Παγκόσμιο Σύστημα Στιγματοθέτησης) σε έναν υπολογιστή. Οι συλλεγμένες πληροφορίες μπορούν έπειτα να διασταυρωθούν και να χρησιμοποιηθούν για να κατηγοριοποιηθεί η προειδοποίηση ως πραγματικός σεισμός ή ως λάθος συναγερμός, να υπολογιστεί το επίκεντρο του σεισμού και να σταλεί προειδοποίηση στις περιοχές που πιθανόν να επηρεαστούν. Αυτή η Διπλωματική εργασία διαπραγματεύεται την υλοποίηση της εφαρμογής και αξιολογεί την επίδοσή της σε καθημερινές δραστηριότητες.

Table of contents

Acknowledgements **ix**

Abstract **xi**

Περίληψη **xiii**

Table of contents **xv**

List of figures **xix**

List of tables **xxi**

Abbreviations **xxiii**

1 Introduction **1**

1.1 Earthquake Early Warning 1

1.2 Contribution 3

1.3 Thesis organization 4

2 Background **5**

2.1 Seismic Waves 5

2.1.1 Body Waves 5

2.1.2 Surface Waves 7

2.2 Accelerometers 7

2.3 Global Positioning System 9

2.4 STA/LTA 10

2.4.1 STA window duration 12

2.4.2 LTA window duration 13

2.4.3 STA/LTA event trigger/detrigger parameter 13
2.5 MQTT 13
2.6 Android Interface 15

3 Design and Implementation **19**
3.1 System Architecture 19
3.1.1 Implemented Design 19
3.2 Smartphone Application implementation 20
3.2.1 Data collecting function 20
3.2.2 STA/LTA calculating function 24
3.2.3 MQTT communication 26
3.2.4 Data recording function 28
3.3 MQTT Broker (Mosquitto) 29
3.4 Fog Server Implementation 30

4 Experimental Evaluation **33**
4.1 General Graph Form 33
4.2 Different Everyday Activities 35
4.2.1 Shaking the table (earthquake imitation) 36
4.2.2 Walking 37
4.2.3 Desk bump 38
4.2.4 Smartphone pick up 38
4.2.5 General Conclusions 40
4.3 Fog Server Notification Latency 43
4.3.1 Algorithm's Buffering Delay 43
4.3.2 Communication Delay 43
4.4 Resource Consumption 44

5 Related Work **47**
5.1 Earthquake Network Project 47
5.2 MyShake 48
5.3 Android Earthquake Alerts System 48
5.4 ARIS Project 48

6 Conclusion **51**

6.1 Summary 51

6.2 Future Work 51

Bibliography **53**

List of figures

1.1 Earthquake casualties per year (worldwide)[1] 1

1.2 Earthquake financial destruction per year (worldwide)[2] 2

1.3 Greek Nation Seismograph Network [3] 3

2.1 Comparison of the S-P intervals of a seismic wave in two seismographs. The P-waves are the first to arrive at each station, followed by the S-waves. Because the P-waves travel faster than S-waves, the great the distance between the two of them is, the further away the earthquake's epicenter is.[4] 6

2.2 Smartphone axes [5] 9

2.3 Accelerometer raw data from placing a coffee mug on a table. 11

2.4 Calculated STA and LTA of placing a coffee mug on a table. 12

2.5 Calculated STA/LTA ratio of placing a coffee mug on a table. 12

2.6 MQTT Publish-Subscribe protocol 15

2.7 MQTT command-response protocol 15

2.8 Android System Architecture[6] 16

3.1 Overall design of the full EEW system. 20

3.2 Implemented design of this Thesis. 21

3.3 Block diagram of the implemented application. It shows the main functions and their inputs/outputs. 22

3.4 Flow diagram of the implemented application. 23

3.5 Example of two-windowed STA and six-windowed LTA. 25

3.6 Example of circular array untangling. 28

3.7 User interface for data recording. 29

3.8 Flow diagram of the implemented server-side application. 31

4.1 STA/LTA ratio of walking with the smartphone in a pocket. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows. The red lines indicate when the event starts and when it finishes. . 34
4.2 STA/LTA ratio of walking with the smartphone in a pocket. The legend describes the number of STA windows each graph has. LTA is set on 10 windows. 35
4.3 STA/LTA ratio of shaking the table where the smartphone rests. This is trying to imitate an earthquake event. The legend describes the number of LTA windows each graph corresponds to. STA is set on 1/5 of LTA windows. . . 36
4.4 STA/LTA ratio of walking with a smartphone in a pocket. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows. 38
4.5 STA/LTA ratio of bumping the table where the smartphone is resting. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows. 39
4.6 STA/LTA ratio while picking up a smartphone resting on a table. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows. 40
4.7 Screenshot from the Android Profiler, while running the application on the OnePlus 8. 45

List of tables

4.1 Highest and lowest values from shaking the table where the smartphone rests. These values depict Figure 4.3i. 37
4.2 Highest and lowest values from walking with the smartphone in a pocket. These values depict Figure 4.4i. 39
4.3 Highest and lowest values from bumping the table where the smartphone is resting. These values depict Figure 4.5i. 40
4.4 Highest and lowest values from picking up the smartphone. These values depict Figure 4.6i. 41

Abbreviations

EEW	Earthquake Early Warning
GPS	Global Positioning System
STA	Short-time-average
LTA	Long-time-average
STA/LTA	short-time-average through long-time-average trigger
API	Application programming interface
UI	User interface
OS	Operating system
IMU	Inertial movement unit

Chapter 1

Introduction

1.1 Earthquake Early Warning

Earthquakes are a problem that affects human civilization since the dawn of time, causing destruction in its wake. However, the denser the populated areas become, the greater the fatalities become, ranging up to hundreds of thousands of human casualties in the most severe cases [7]. The earthquake casualties per year are shown in Figure 1.1. However, the earthquake's destruction does not stop in human lives. As we can see in Figure 1.2, many billions of dollars are lost every year. An early warning could potentially save many lives and infrastructure costs by allowing people to get cover, trains and cars to slow down, and much more.

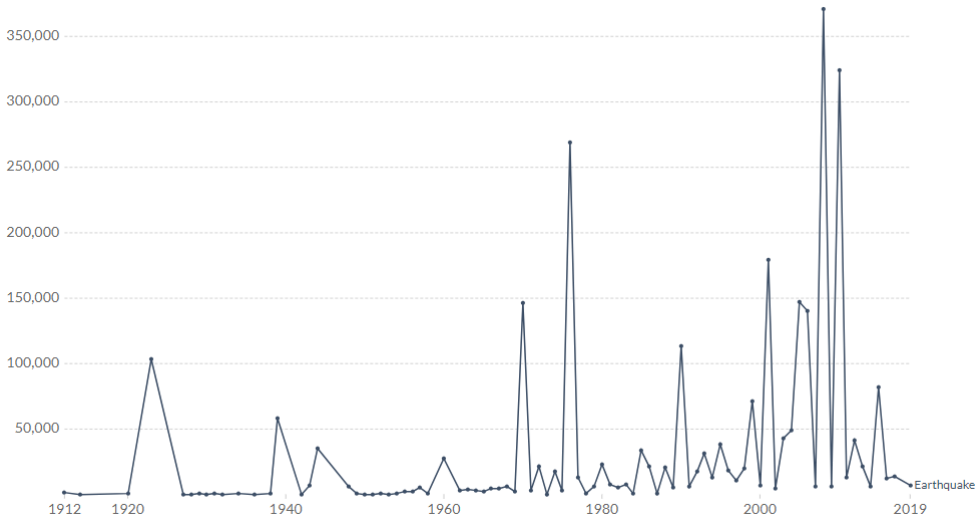


Figure 1.1: Earthquake casualties per year (worldwide)[1]

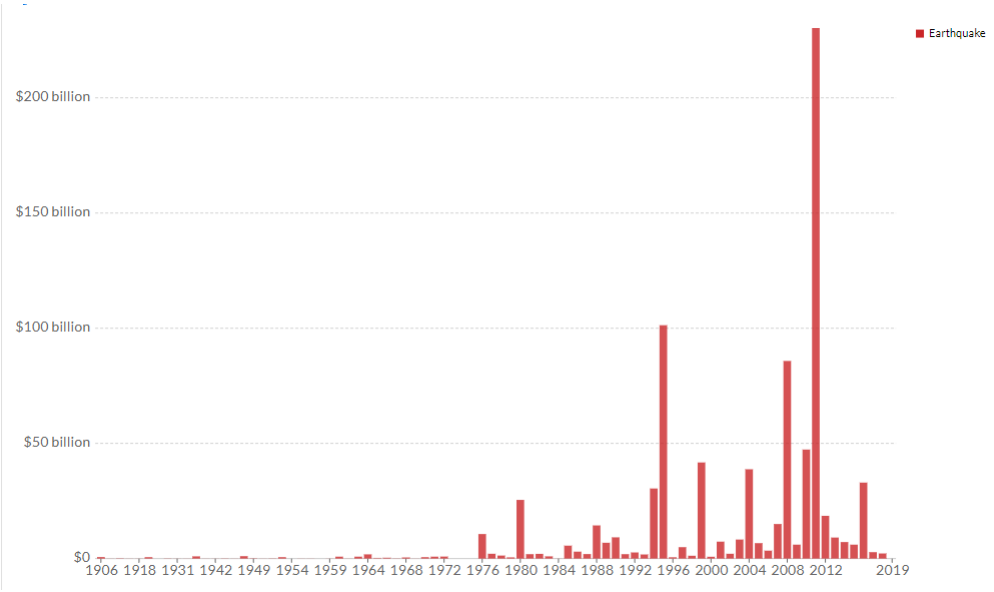


Figure 1.2: Earthquake financial destruction per year (worldwide)[2]

EEW systems have been developed with one purpose: to detect the earthquake near its epicenter and issue warnings in the areas that will be afflicted before the damaging waves arrive. This is possible due to the difference in the propagation speed of the various waves an earthquake emits, and also the fact that these speeds are relatively low. Correctly detecting the faster waves near the epicenter can be utilized to warn areas further away from the epicenter, that will be afflicted, providing them with some precious seconds to prepare.

It is crucial that EEW is not confused for earthquake prediction. In opposition to earthquake prediction that would offer us knowledge about the earthquake before it has even begun, EEW offers us only a few precious seconds between the arrival of the faster-moving waves in our detection sensor and the arrival of the slower destructive ones in the near areas. This means that it is crucial to detect the earthquake near its epicenter in order to provide as much time as possible and for this to be achieved sensors should be spread everywhere. However, modern seismic and geodetic networks exist in a handful of countries and even then they offer scarce coverage.

In Figure 1.3 we can see that even in a country that is experiencing multiple earthquakes every year, there are only $\approx$ 100 earthquake stations, making the coverage of the country inadequate. Moreover, the fact that the older-traditional seismographs are not built to work in real-time makes the coverage even scarcer. This scarce coverage of sensors can be solved by utilizing sensors that are able to detect earthquake waves and are very widespread: the accelerometers in smartphones. According to statistics [8], there are approximately 4 billion

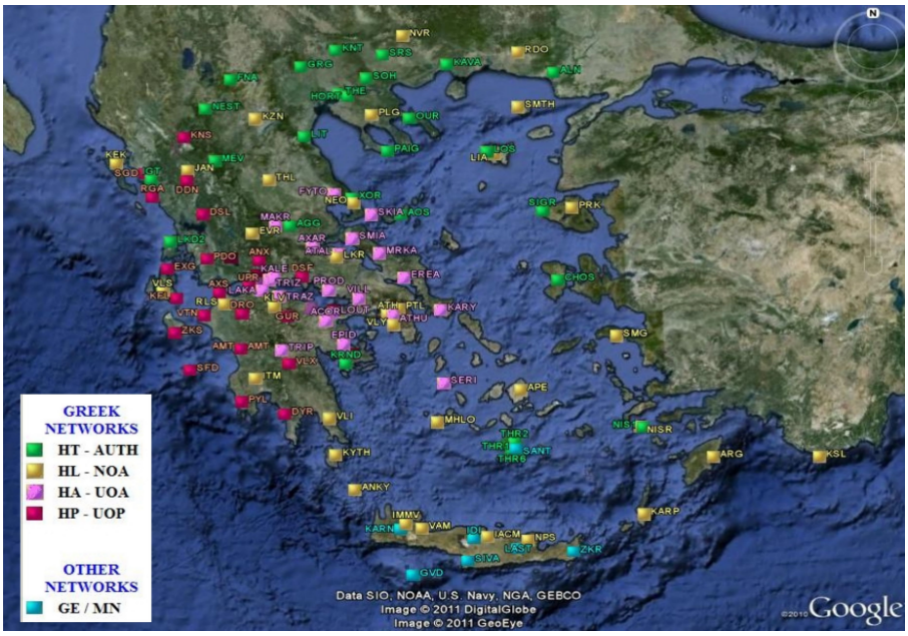


Figure 1.3: Greek Nation Seismograph Network [3]

smartphone users in the world, which could help by allowing their smartphone to work like an earthquake detector. This would solve the problem of the coverage in populated areas. Due to the fact that smartphones are not stationary sensors but are moving throughout the day due to human interaction, the data from smartphones will not be always accurate. Even though the smartphones' accelerometer will have more noise than the usual sensors used for earthquake detection, this idea is enabled by the large scale. By having thousands of phones scanning for an earthquake in a city it is safe to assume that an earthquake is happening when a significant number of smartphones report an event simultaneously.

1.2 Contribution

The goal of this diploma Thesis is the proposal of a solution to the Earthquake Early Warning problem. The main contribution of this Thesis can be summarized below:

1. Smartphone application. For the purpose of harnessing the data of the smartphones' accelerometers, an Android application was developed. This application monitors the accelerometer, detects possible events with the use of the STA/LTA algorithm, and reports to a central computer.

2. Server-side application. The implemented server-side application monitors the specified address for new messages, creates a file, and saves the recorded accelerometer

data along with the timestamps and the GPS location data.

3. STA/LTA algorithm parameter setting. Through tests run on our physical device, the various parameters of the STA/LTA algorithm were set on values that aim to reduce the delay of the event detection and the detection of false events.

4. Experimental evaluation. An evaluation of the application's performance on recording and detecting possible events through everyday activities was made. Moreover, the application was evaluated about its resource consumption.

1.3 Thesis organization

The rest of the Thesis is organized as follows. Background information that supports the rest of the Thesis is in Chapter 2. In Chapter 3 there is an analysis about the implemented design. Chapter 4 includes the experimental evaluation and the STA/LTA algorithm parameter setting. Chapter 5 presents related projects and their differences with this implementation. Lastly, Chapter 6 summarizes the Thesis and proposes possible future expansions.

Chapter 2

Background

2.1 Seismic Waves

Seismic waves[9][10][11] are waves of energy that travel through Earth's layers, and are a result of earthquakes, volcanic eruptions, magma movement, large landslides, and large man-made explosions that give out low-frequency acoustic energy. For this Thesis we are interested mainly in the properties of these waves that support the EEW systems: the propagation velocity and the destructive power of each type of seismic wave.

Among the many types of seismic waves, one can make a broad distinction between body waves, which travel through the Earth, and surface waves, which travel at the Earth's surface. Figure 2.1 shows two seismograms of the same event. The main focus should be the time intervals between the arrival of the different type of waves, that indicates a difference in propagation speed.

2.1.1 Body Waves

Body waves arrive at the earthquake location before the surface waves are even emitted. They travel through the interior of the earth and have a higher frequency than the surface waves. There are two categories of body waves: Primary waves (P-waves) and Secondary waves (S-waves).

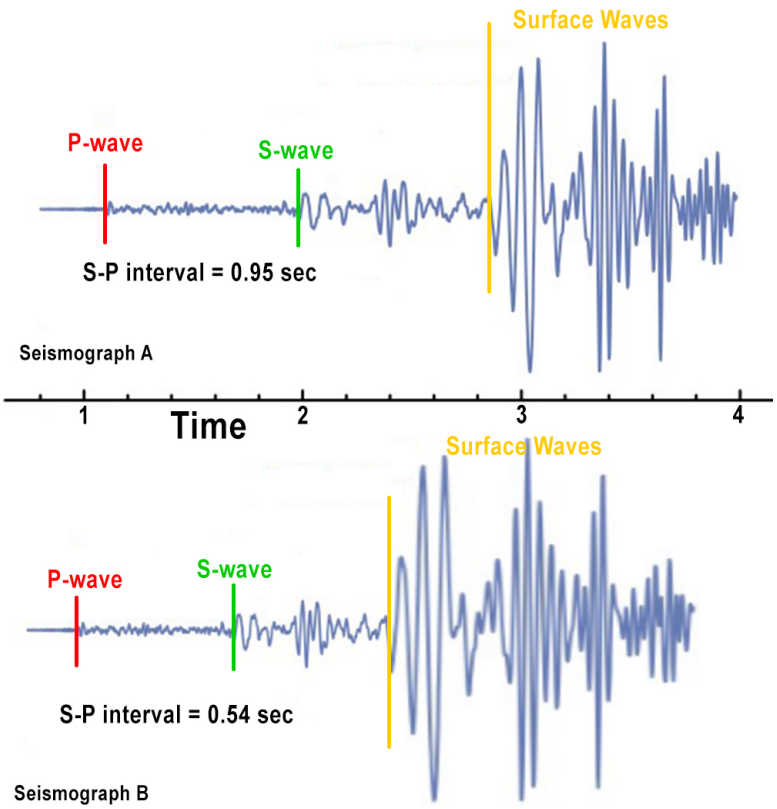


Figure 2.1: Comparison of the S-P intervals of a seismic wave in two seismographs. The P-waves are the first to arrive at each station, followed by the S-waves. Because the P-waves travel faster than S-waves, the great the distance between the two of them is, the further away the earthquake's epicenter is.[4]

Primary Waves

Primary waves (P-waves) are compressional waves that are longitudinal in nature. This means that the displacement of the medium they are traveling into is in the same direction as the wave propagation. They are the fastest type of seismic wave and thus the first ones to arrive at a specific location. They can travel through any type of material and their typical speeds are[10][11]:

1. 330 m/s in air

2. 1450 m/s in water

3. 5000 m/s in granite

Secondary Waves

Secondary waves (S-waves) are shear waves that are transverse in nature. This means that their oscillations are perpendicular to the direction of the wave propagation. These are the second waves to arrive, after the faster Primary waves. They are able to travel only through solids and their speed is typically around 60% of that of Primary waves in any material.

2.1.2 Surface Waves

Surface waves travel only through the crust. They have a lower frequency than the body waves and thus are easily distinguished. Even though they travel slower than the body waves, they are almost entirely responsible for the damage and the destruction an earthquake causes. All of the surface waves diminish as they get further from the surface. There are two categories of surface waves: Love waves and Rayleigh waves.

Love Waves

Love waves are transverse waves. They vibrate the ground in the horizontal direction perpendicular to the direction that the waves are traveling. Love waves are the fastest surface waves, however, they are slower than the body waves, traveling at approximately 90% of a S-wave's speed.

Rayleigh Waves

Rayleigh waves, also called ground rolls, are surface waves that travel as ripples with motions that are similar to those of waves on the surface of the water. A Rayleigh wave rolls along the ground just like a wave rolls across a lake or an ocean. It moves the ground up and down, and side-to-side in the same direction that the wave is moving. Most of the shaking felt from an earthquake is due to the Rayleigh wave, which can be much more intense than the other waves. They are the slowest wave, moving slightly slower than the Love waves.

2.2 Accelerometers

Acceleration sensors have multiple applications in today's industry and science. More specifically, in smartphones they can be used to change the screen orientation from portrait

to landscape and vice versa when the device is tilted, function as a pedometer by counting the user's steps. Additional uses include correcting the shaking in smartphone cameras while taking a picture or recording a video and even controlling characters in games.

An accelerometer is a tool that measures proper acceleration. Proper acceleration is the acceleration (the rate of change of velocity) of a body in its own instantaneous rest frame. This is different from coordinate acceleration, which is the acceleration in a fixed coordinate system.[12]

Theoretically, the acceleration of a device (α_D) is calculated by computing the sum of the forces that are exercised on the device (F_D) using the following formula:

$$\alpha_D = -(1/mass) \sum F_D$$

Nevertheless, the force of gravity is always affecting the measured acceleration according to the formula below:

$$\alpha_D = -g - (1/mass) \sum F_D$$

For this reason, when the device is at rest on the Earth's surface, the accelerometer reads an acceleration of $\alpha = g \approx 9.81 m/s^2$ straight upwards. By contrast, when the device is in free fall and thus accelerating towards the ground at $9.81 m/s^2$, its accelerometer reads a magnitude of $\alpha = 0 m/s^2$. Therefore, to obtain the real acceleration of the device, the input of the force of gravity must be eliminated from the accelerometer data. Hence, a separate embedded accelerometer is utilized, the gravity sensor. This particular sensor has the ability to measure the magnitude of gravitational force that is being exerted on the device.

The accelerometer data are presented to the smartphone applications in their three-component form. This means that the magnitude of the total acceleration of the device $\|a_d\|$ is calculated according to the following formula:

$$\|a_d\| = \sqrt{a_{dx}^2 + a_{dy}^2 + a_{dz}^2}$$

where a_{dx}, a_{dy}, a_{dz} is the acceleration of the device in the x, y and z axes respectively. The standard configuration of the axes is shown in Figure 2.2

The device used for the development and testing process of this application is a OnePlus 8. It includes the Bosch bmi261 inertial movement unit (IMU), which is a 6-axis IMU featuring a 16-bit tri-axial gyroscope and accelerometer specifically designed for Android operating

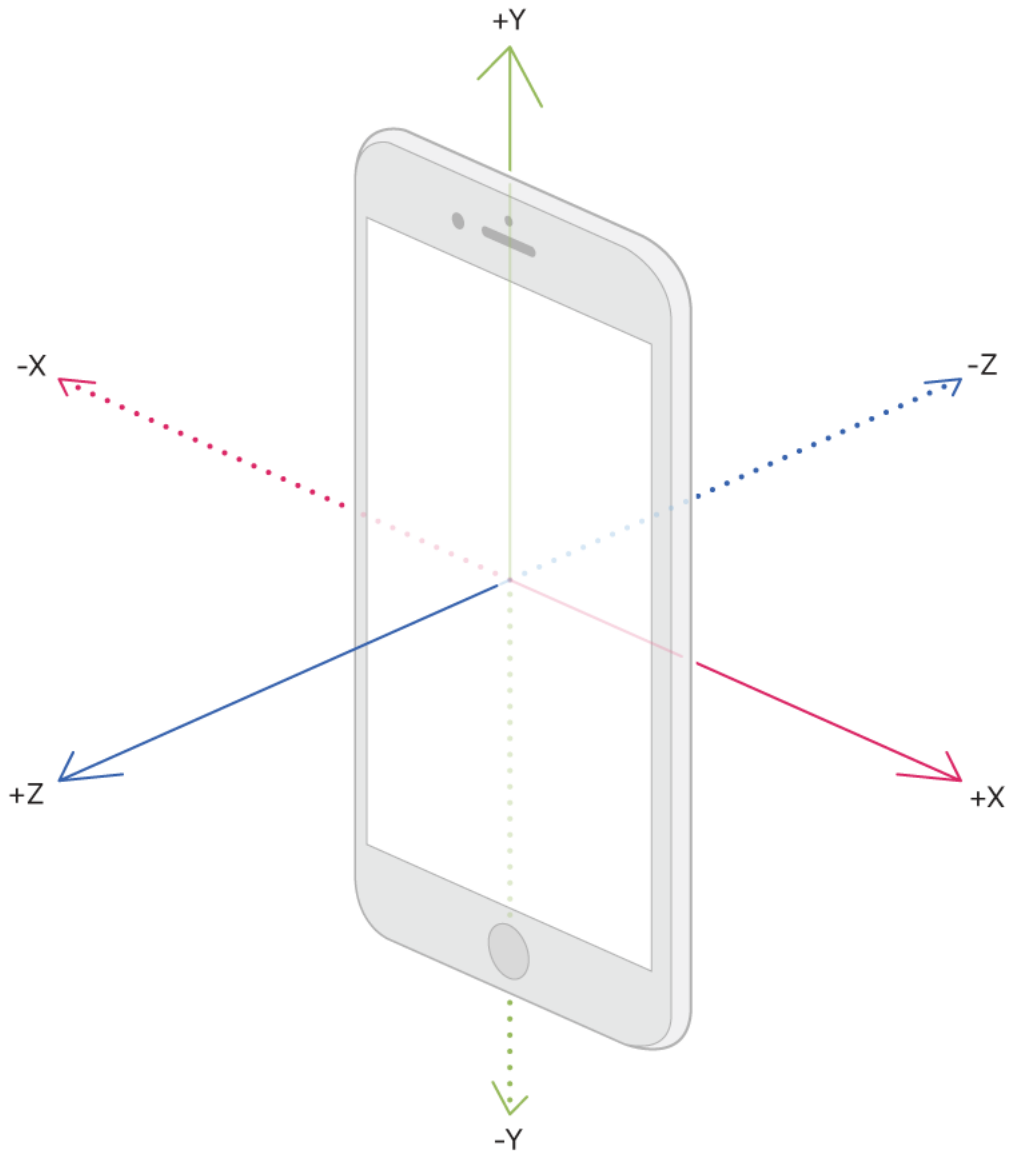


Figure 2.2: Smartphone axes [5]

systems[13]. According to the technical data, the accelerometer has a maximum capacity of 1.6 Khz data output rate. Moreover, the sensitivity of the sensor varies from 2048 LSB/g (least significant bits/g) up to 16384 LSB/g and a sensitivity error of $\pm$ 0.4%.

2.3 Global Positioning System

Global Positioning System (GPS) is one of the global navigation satellite systems (GNSS) that provides geolocation and time information to a GPS receiver anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites[14]. Initially, GPS signals were used by specific dedicated devices. Nowadays, there are GPS receivers

embedded in many electronic devices, including smartphones.

However, modern Android software offers developers the chance to use an Assisted GPS (A-GPS) interface. Using this approach, the location information doesn't have to rely on GPS satellites solely or not at all. Instead, the software can utilize WiFi or cell tower positioning in order to calculate the device's coordinates. This option offers less accuracy on the real position of the device but helps to reduce the battery drain of the application on the device. The options can be low power, meaning accuracy of approximately 10 kilometers, balanced power, meaning accuracy of approximately 100 meters, and high accuracy, which is usually the data from GPS sensor, as it is mentioned on Android Developers' page[15].

The OnePlus 8 we used for testing the application supports the most known GNSSs which are:

1. GPS (L1+L5 Dual Band)
2. GLONASS
3. Galileo (E1+E5a Dual Band)
4. Beidou
5. A-GPS

2.4 STA/LTA

The short-time-average through long-time-average (STA/LTA)[16] trigger is an algorithm designed to identify potential earthquake events, that should be recorded and further analyzed. STA/LTA continuously calculates the average absolute values of the amplitude of the acceleration over a short-time and a long-time window. Consequently, a ratio of both those values (STA/LTA ratio) is calculated, and when the ratio of those windows exceeds a specific value, an event is declared and the data are recorded. The recording process continues until this ratio falls below a distinct value, a detrigger threshold.

The trigger works by identifying sections of an incoming data stream when the signal amplitude increases. The STA window is relatively small; however, it is not instantaneous, and there is less of a possibility of triggering at spikes or short-duration events. On the other hand, the purpose of LTA is to provide a measure of the variation in the background seismic

noise. To fully capture a seismic event, recording a few seconds of buffered data from before the event is needed. It is distinctly beneficial for emergent type signals, where the system is triggered only after the first arrival. Similarly to the trigger, the detrigger works by identifying when the signal amplitude decreases to its normal levels.

Figure 2.3 shows the raw accelerometer data recorded from a minor event. The smartphone is resting on the table recording when the event, someone placing a coffee mug on the table, happens. The usual noise levels of the table are $\approx \pm 0.04$, thus in the x- and y-axis we can see that the plot before the event is noisy, due to the minimal acceleration levels on those axes. On the other hand, the acceleration on the z-axis is quite significant, reaching a peak value of ≈ 5, and thus the plot before the event seems like a straight line. Figure 2.4 shows the calculated STA and LTA values for the same event. Additionally, Figure 2.5 shows the calculated STA/LTA ratio. Finally, we should mention that for these examples the STA window is 400 samples, meaning $\approx$ 1 second, the LTA window is 4000 samples, meaning $\approx$ 10 seconds and the refresh interval of STA and LTA is 80 samples or $\approx$ 0.2 seconds since the sampling rate is $\approx$ 400 Hz.

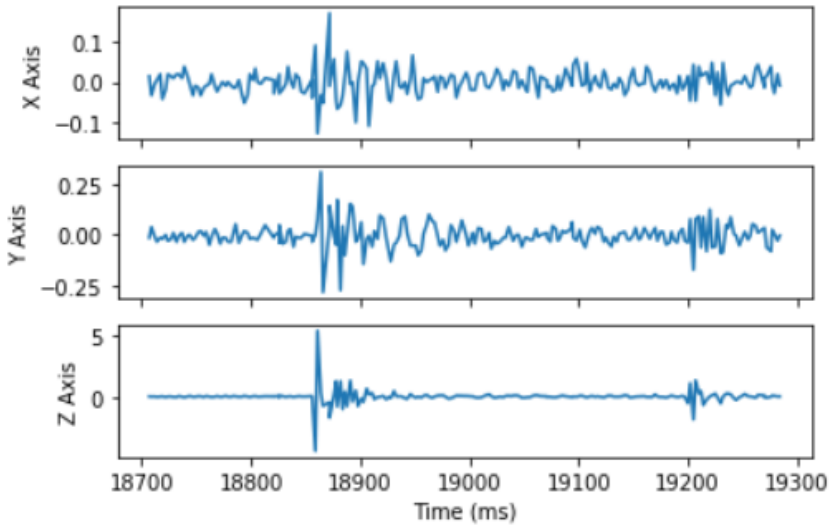


Figure 2.3: Accelerometer raw data from placing a coffee mug on a table.

STA/LTA algorithm is associated with the following parameters:

1. STA window duration
2. LTA window duration
3. STA/LTA event trigger parameter

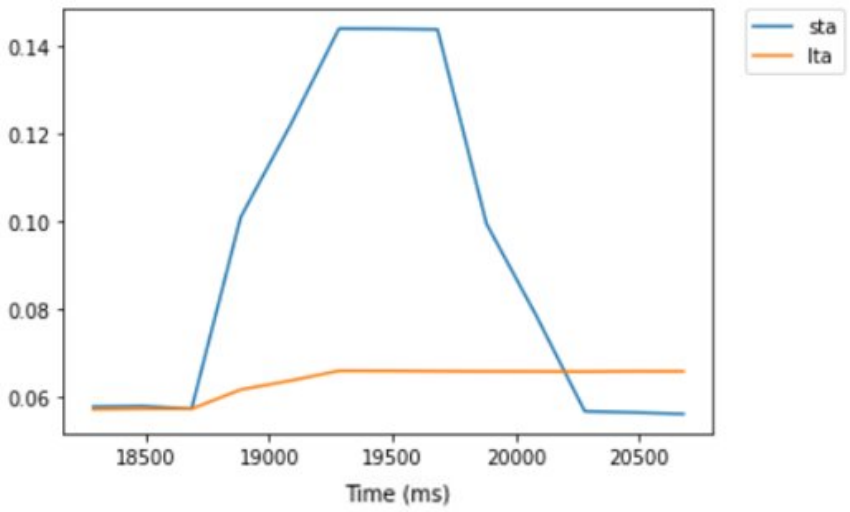


Figure 2.4: Calculated STA and LTA of placing a coffee mug on a table.

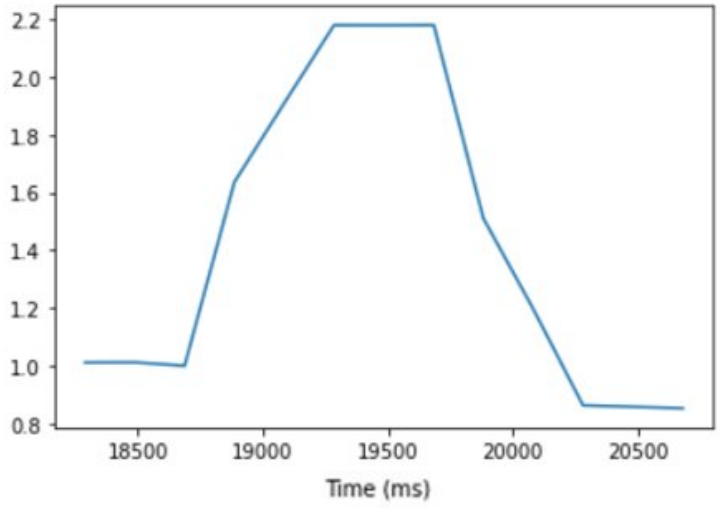


Figure 2.5: Calculated STA/LTA ratio of placing a coffee mug on a table.

4. STA/LTA detrigger parameter

2.4.1 STA window duration

STA measures the (almost) instant absolute values of the acceleration. It monitors these values and checks for possible events. The duration of the STA window affects the sensitivity of the algorithm. When the duration of the STA window is short, the STA value becomes more sensitive, especially on short-lasting spiky movements. For instance, even a minor spiky incident such as placing a coffee mug on the table can potentially trigger an event, if the STA window is too small. On the contrary, the same activity with a larger STA window might not

even come near to triggering an event.

On the other hand, if the STA window gets too large, the STA value could potentially become non-sensitive, thus requiring multiple windows for STA to ramp up. This could cause an unwanted delay in triggering the algorithm and consequently the earthquake warning.

2.4.2 LTA window duration

LTA measures the average absolute values of the acceleration over a long period of time. A shorter window for LTA means that LTA can be updated fast enough and get used to man-made noise and thus possible false triggers might be prevented. On the other hand, an LTA that updates too quickly, might delay, or even prevent, the triggering of an event, by keeping the STA/LTA ratio lower than it should be. Moreover, with short windowed LTAs, events may be terminated sooner than they should.

2.4.3 STA/LTA event trigger/detrigger parameter

The trigger parameter mainly decides the significance of the recorded events. Obviously, there is some element of trade-off in setting a value for the trigger ratio. As a result, the lower the trigger parameter gets, the less powerful earthquakes can be recorded. However, this means that the algorithm becomes more susceptible to man-made noise as well. Although some false events are expected to be recorded, the trigger parameter should not be set too high as this will cause many actual events to be missed. Determining an appropriate value, one that maximizes the number of detected events, while also minimizing the number of false alarms, is a matter of trial and error.

The detrigger parameter determines the duration of the recorded events. Consequently, the bigger this parameter is, the smaller the duration of the recordings becomes. Nevertheless, the value of this parameter should not be too low, because a never-ending recording might occur. This can happen when, due to the increasing environmental noise, the STA/LTA ratio can never fall lower than the detrigger parameter.

2.5 MQTT

Message Queue Telemetry Transport (MQTT), which has nowadays become the standard protocol for internet of things (IoT) communications, is a lightweight protocol that was in-

vented by IBM to facilitate machine-to-machine communication[17]. MQTT is built on top of the TCP/IP stack and can also run on SSL/TLS, which is a secure protocol built on TCP/IP. It has officially become an OASIS open standard since 29 October 2014[18], and it is currently supported by many popular programming languages by using multiple open-source implementations.

MQTT is a Client-Server publish/ subscribe messaging transport protocol, with a small transport overhead and protocol exchanges minimized to reduce network traffic. In this project, the MQTT protocol is used to facilitate communication between the mobile device and the computer. For this reason, an MQTT client library written in Java for developing applications on Android, the Paho Android Service was used[19].

There are a couple of reasons that this protocol is chosen for the application. To begin with, the more widely known and used Hypertext Transfer Protocol (HTTP)[20] web service, is heavyweight with a lot of header files, making it unsuitable for restricted networks like mobile roaming. HTTP also has a bigger average response time than MQTT[21], making HTTP not the best option. Another very popular messaging protocol used in enterprise middleware systems is the Advanced Message Queuing Protocol (AMQP)[22]. AMQP is very similar to MQTT, however, it has a bigger header size in its messages, so we opted for the second one.

MQTT doesn't create a direct connection between the server and the client in order to transfer messages. Instead, it requires an intermediate broker, like Mosquitto[23] which is a lightweight open-source that implements the MQTT protocols. The broker is necessary because the clients have no unique addresses or identifiers and messages cannot be sent directly to them. For this reason, the mobile device publishes the message on a specific topic. After the message is published, the broker identifies the subscribers of this topic and forwards the message only to them. As shown in Figure 2.6, the client-publisher sends the message to the broker and then the broker forwards the message to the subscribed devices.

MQTT invokes the TCP/IP protocol for the connection between the clients and the broker. This guarantees that the packages will arrive in the correct order. Moreover, MQTT is a command-response protocol, which means an acknowledge message is sent back from the broker. As we can see in Figure 2.7, the client initiates the communication by sending a connection request, the broker answers with a connection acknowledge. Then the client publishes the message and finally, the broker confirms that the message was received.

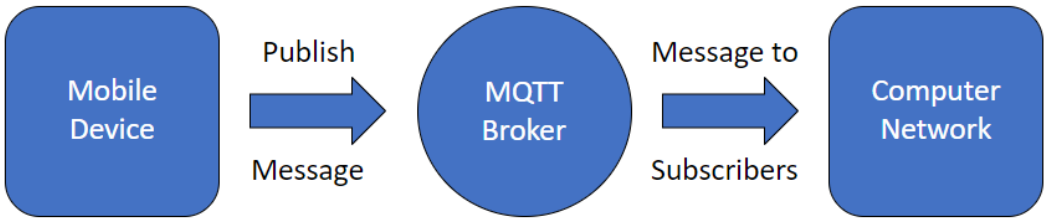


Figure 2.6: MQTT Publish-Subscribe protocol

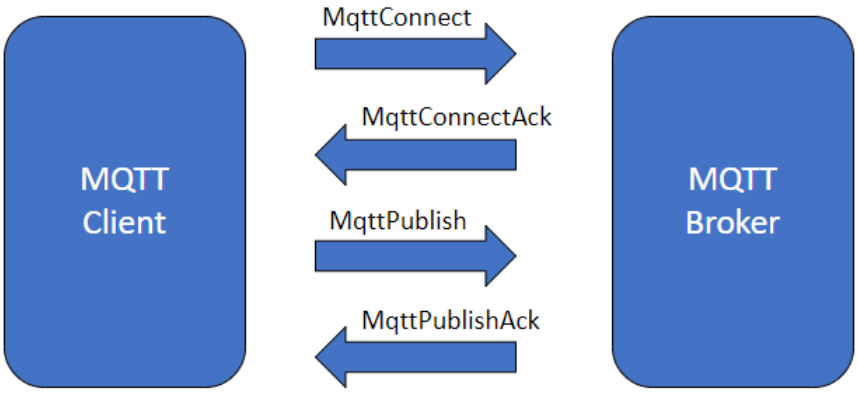


Figure 2.7: MQTT command-response protocol

2.6 Android Interface

Android[24] is one of the most popular smartphone operating systems (OS) and many smartphone companies use it as a base for their devices. It is free and open-source software. Its source code is known as Android Open Source Project (AOSP). It is a mobile operating system based on a modified version of the Linux kernel. More specifically, Android uses versions 4.4, 4.9 or 4.14 of the Linux kernel since 2020[6]. On top of the Linux kernel, there are the libraries and the application programming interfaces (APIs), which are written in C. The standard C library for Android is Bionic, which is developed by Google to be used in environments with less memory and processing power. Along with the libraries, there is the Android Runtime (ART) environment which uses ahead-of-time (AOT) compilation to entirely compile the application bytecode into machine code upon the installation of an application [6]. On the highest level, there is the application software, commonly known as

apps, running on an application framework that includes Java-compatible libraries. Figure 2.8 shows the Android layers.

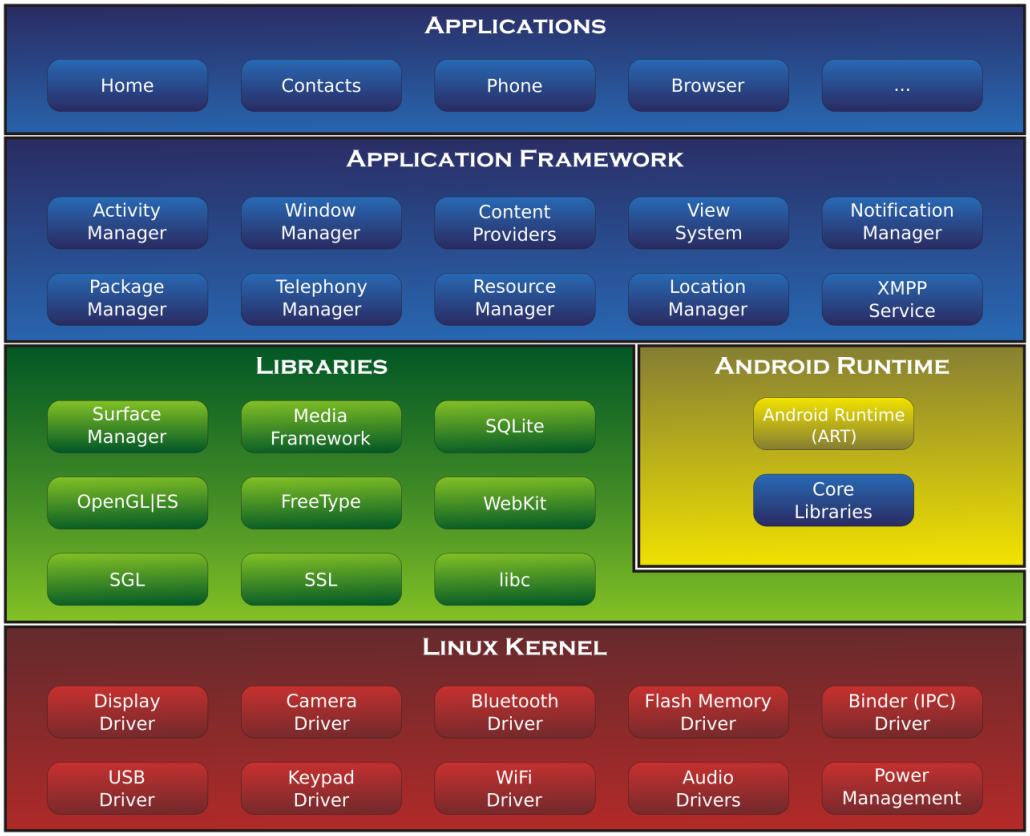


Figure 2.8: Android System Architecture[6]

Android uses events to collect data from various sources such as user's interaction with the phone (button presses or screen touches) or read data from sensors. The Android framework maintains an event queue structured like a first-in, first-out (FIFO) queue. In order to be able to handle the event, the application must have the proper listener registered.

The Android Event Management includes the following:

1. Event Listeners: an event listener is an interface that contains a single callback method. These methods will be called by the Android framework when the component, for instance, the sensor, to which the listener has been registered is triggered.

2. Event Listeners Registration: this is the process by which an Event Handler gets registered with an Event Listener so that the handler is invoked when the Event Listener receives an event.

3. Event Handlers: this is the method that handles the event. It is called by the Event Listener when the event happens.

The tool used to develop this application was Android Studio, which is the official integrated development environment (IDE) for Google's Android operating system. The most notable features that it offers to the developers are:

1. tools to help you in debugging (inline debugging) and performance analysis tools,
2. an integrated Gradle-based tool as the foundation of the build system,
3. Lint tools to catch performance, usability, version compatibility, and other problems,
4. a build-in emulator which allows for further testing and debugging of the application,
5. a layout editor that allows users to drag-and-drop or code UI components, while also being able to preview the layouts on multiple screen configurations.

Chapter 3

Design and Implementation

3.1 System Architecture

The EEW system's overall proposed design can be described in Figure 3.1. First and foremost, there are smartphone devices. These devices are constantly monitoring their respective accelerometer for potential events. The devices are then divided into clusters based on their current location. Each cluster publishes the messages to a local MQTT broker, which then forwards them to a subscriber-server application. This server along with the MQTT broker creates a local fog server in each area. The final decision takes place in another server, the Decision Server, which is responsible for declaring an event as true or false. This is accomplished with the implementation of a voting system. This means that the event will be declared only if the majority of the devices in the area sent simultaneously a trigger message to the server. If the server decides that an event is detected, it notifies the devices that might be affected by this event, by publishing a warning message.

3.1.1 Implemented Design

In this Thesis, the implemented design can be described in Figure 3.2. It includes the smartphone application and the fog server, which contains the MQTT broker along with a simple server implementation that subscribes to the channel and reads the data sent from the mobile publisher.

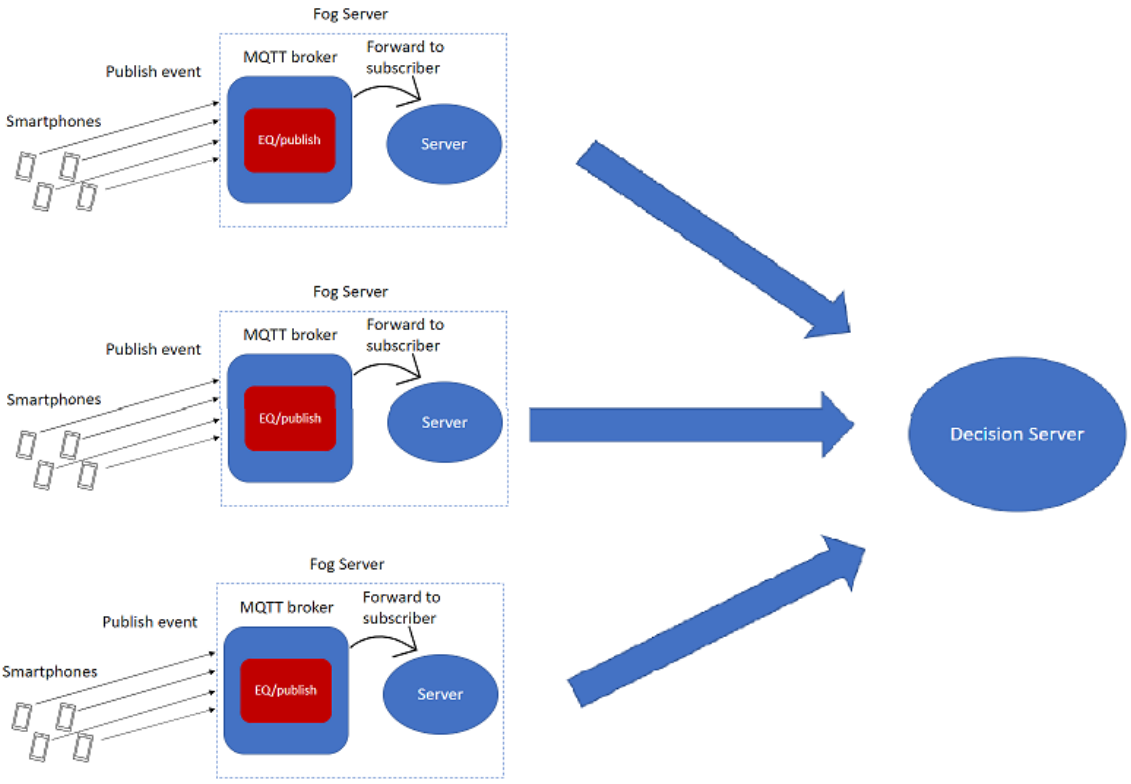


Figure 3.1: Overall design of the full EEW system.

3.2 Smartphone Application implementation

The general functionality of the implemented application can be shown in Figure 3.3. This block diagram shows the core modules of the application, along with the inputs/output they use to communicate with each other. Figure 3.4 shows a more detailed flow diagram of the application. The smartphone application can be separated into three distinct parts:

1. the data collecting function
2. the function that calculates STA/LTA ratio
3. the function that handles MQTT communications

3.2.1 Data collecting function

This function, as its name suggests, is in charge of collecting the data. As we mentioned in 2.6, the data from a sensor arrive at the application in the form of an event. This means that in order to collect and store the data one should register a listener. Since our data (events) are related to sensors, the listener utilized is a sensorEventListener. These listeners are registered

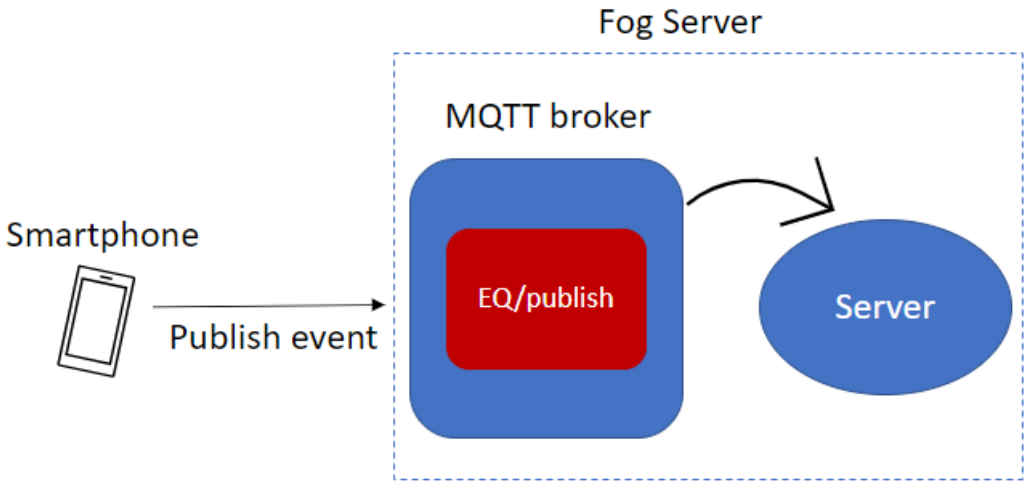


Figure 3.2: Implemented design of this Thesis.

during the onCreate function of the application, which is the default function that is executed when the application is being initialized. During the registertration process of the listener, the programmer can also specify the sampling rate of each sensor.

In this part, we came across some limitations of the Android application development. It is mentioned in a previous subsection that the accelerometer has a capacity of 1.6 Khz data output rate, yet the programmer has access to a maximum data rate of $\approx$ 400 Hz. Moreover, the programmer is not able to define the exact sampling rate of the accelerometer interface. According to the Android developer page[25], programmers can only give a ”hint” to the system about their preferred sampling rate. The results of our testing showed that usually, the samples arrive on time, but sometimes they may be received faster or slower.

After registering a listener, there are some core functions that need to be declared. These are:

1. onPause: this function is called by the system every time the activity stops being in the foreground (without meaning the activity is getting destroyed).

2. onResume: when the activity gets back to the foreground, this function is executed. The application then enters and stays in the resumed state until something happens that will take the focus away from the application (like a phone call)

3. onAccuracyChanged: this function is invoked when the accuracy of a registered sensor changes

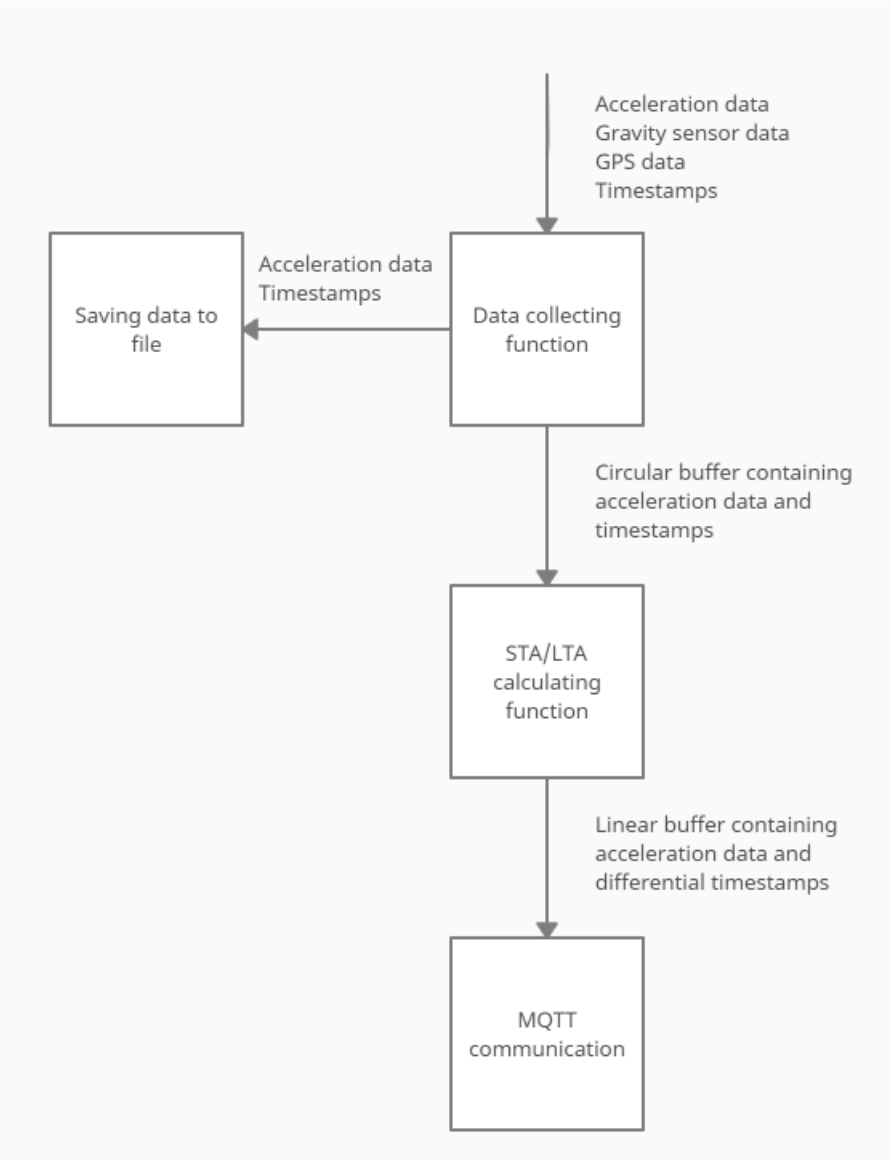


Figure 3.3: Block diagram of the implemented application. It shows the main functions and their inputs/outputs.

4. onSensorChanged: this function gets called when new data arrive from a registered sensor. Despite what the name implies, the new data could be exactly the same as the previous ones

In this application, the data collecting and storing part is incorporated in the onSensorChanged function. Due to the fact that the same function is being invoked for all the registered sensors, which in this case are the accelerometer and the gravity sensor, this function is split into two parts, that are executed according to the identity of the sensor that generated the event.

The first half belongs to the gravity events. We mentioned in 2.2, that the acceleration

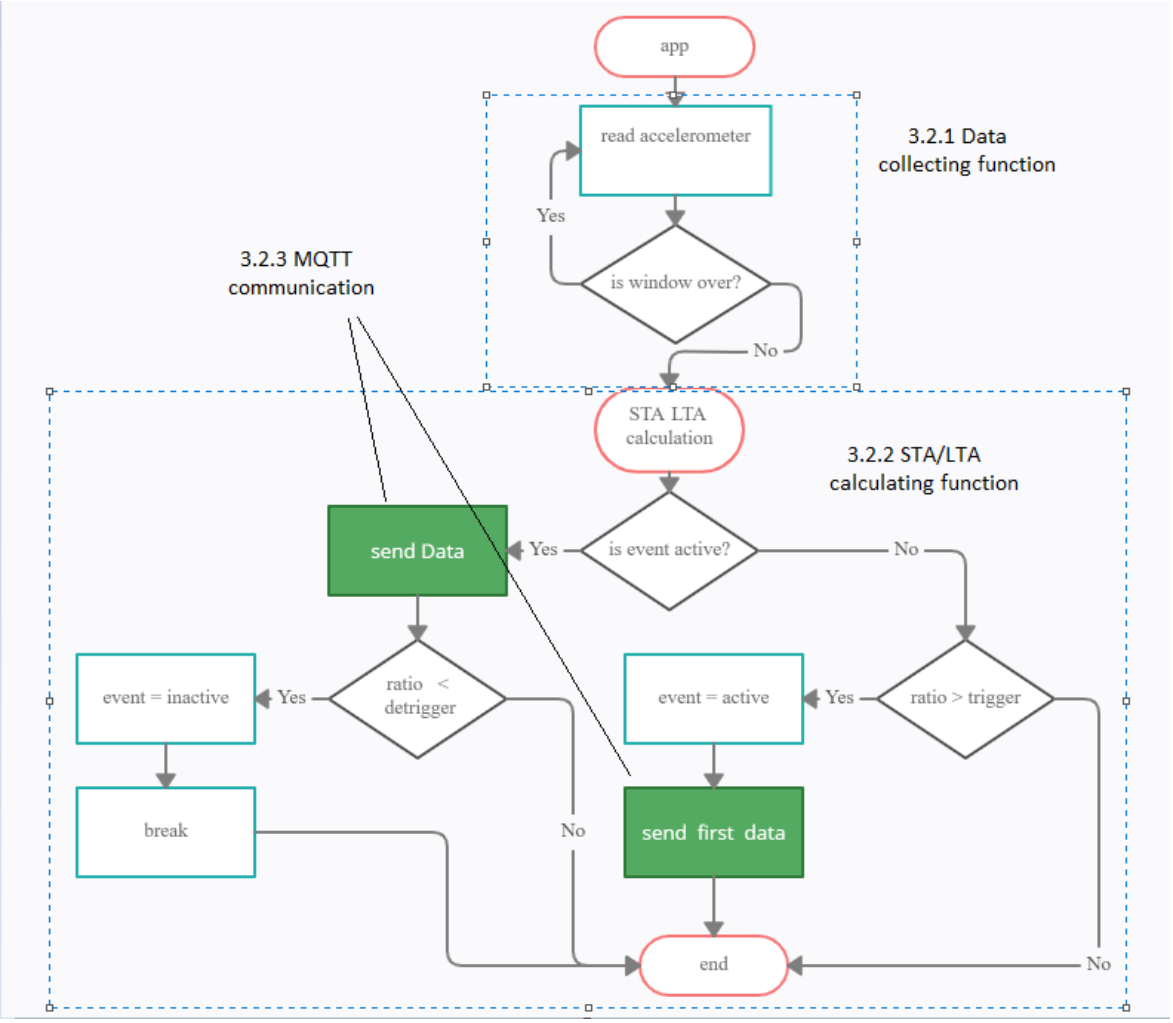


Figure 3.4: Flow diagram of the implemented application.

values given by the accelerometer are affected by the force of gravity. However, in order to detect earthquake motions from the acceleration data, the contribution of the force of gravity must be first eliminated. Thus, the gravity sensor is being sampled, in a relatively low frequency (50-55 samples/s), and these values are stored in order to be deducted from the accelerometer values.

The other half of this function is executed when the accelerometer events arrive. First and foremost, after the acceleration values arrive, the current gravity value is deducted from them. Consequently, the data are being stored in a circular buffer. This buffer consists of four fields, one for every component of the acceleration and a fourth one containing the timestamp of the sample's arrival. This part of the onSensorChanged is also responsible for calling the function that calculates STA/LTA at the end of each window. For this reason the index of the circular buffer's position is used to determine the end of a window. For this application, the

window was set in eighty (80) samples, which is approximately every 0.2 seconds since the sampling rate of the accelerometer is $\approx$ 400 Hz.

Location data collecting function

Another type of data being collected in this application is location data. Location data are acquired by similar means to the motion sensors data. In order to use this feature, the application should request permission from the user. Afterward, a fused location provider client along with a fused location request is created. Moreover, the normal and the fastest update intervals, as well as the required accuracy are set. The last step in the registering process is the creation of the callback function, the function that is getting executed when the new data arrive.

In this application, the callback function begins by checking the new location value fields for null values, which can happen because the user turned off the location settings. Next, it updates the stored location values with the new ones. The values used are:

1. latitude
2. longitude
3. accuracy
4. altitude
5. speed

The last two are not available on every device, thus before updating a check about their availability is performed. If they are not available a ”NaN” string is sent instead.

3.2.2 STA/LTA calculating function

This function, as mentioned earlier, is executed at the end of every window. It is responsible for processing the data, calculating the STA, LTA and STA/LTA ratio values, identify the events as well as initializing the communications.

It begins by calculating the magnitude of the acceleration of every sample in the last window by using the following formula:

$$a_{total} = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

where a_x, a_y and a_z are the three components of the device's acceleration. These sums are then stored in a circular buffer, similar to the one the accelerometer data are being stored. Consequently, there is the calculation of the new STA and LTA values. The function utilizes the circular sum buffer and updates the STA and LTA values by subtracting the oldest sum and adding the new one. However, STA and LTA are average values, which means that they contain sums from multiple windows, as shown in Figure 3.5. Thus before adding (or subtracting) the sums they are divided by the number of samples STA or LTA include. After updating the STA and LTA values the new ratio is calculated.

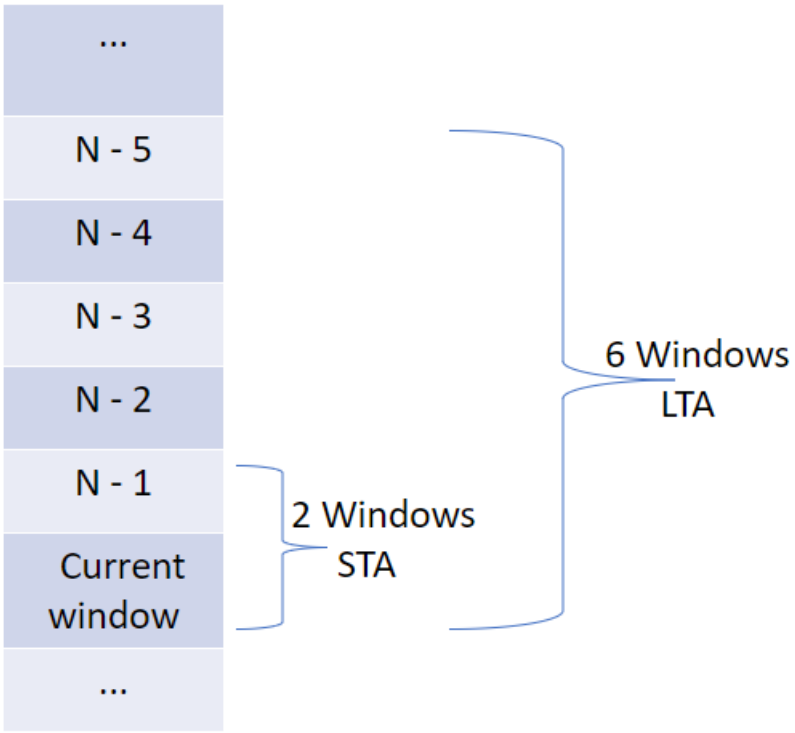


Figure 3.5: Example of two-windowed STA and six-windowed LTA.

Subsequently, the function checks in what state the application is in. The two possible states are:

1. an event is currently active

2. no event is active at the time

In case there is not an active event, the function checks if the STA/LTA ratio is higher than the trigger limit. If this is not true, the application continues its normal function. However, if the ratio is greater than the trigger threshold, an event is declared active and the communication with the broker is initialized. Moreover, the timestamp of the event declaration is stored (it will be used in 3.2.3).

On the other hand, if there is an already active event, the function checks if the STA/LTA ratio is higher than the detrigger limit. In case this is true, the function continues to send data via the MQTT connection. When the ratio gets lower than the threshold, the application publishes the final data along with a terminating message. Moreover, the event is declared finished and a timer, which puts the application on a break until the countdown hits zero, is initialized. The timer is used to deactivate and then reactivate the functionality of the data collecting function by changing the value of a boolean flag variable. The break functionality is implemented so that a device cannot detect false events too often. The following Algorithm 1 describes this function in pseudo-code.

3.2.3 MQTT communication

The functionality of the communication can be split in four parts:

1. commands that initialize the connection with the broker
2. commands that are executed when an event starts
3. commands that are executed to publish data
4. commands that are executed when an event is finished

The commands that initialize the connection with the broker are executed during the onCreate function, which is the function that the system invokes when an application is started. These commands are used to create the connection of the application with the broker, set up information regarding the username and the password of the publisher device as well as create the callback functions that are executed upon the arrival of the acknowledge messages, the loss of connection or the reception of a message (in case the application subscribes to another topic). After these, the application connects to the broker. It should be mentioned, that at the beginning of the other functions a check of the connection state occurs and in case of a disconnection, the application tries to reconnect to the broker.

Algorithm 1 STA/LTA calculating algorithm

```
1:  Procedure StaLtaCalc
2:  for each sample s in this window do do
3:      Calculate total acceleration
4:  end for
5:  Update STA and LTA
6:  Calculate new ratio
7:  if event == active then
8:      publishData(data)
9:      if ratio <= detriggerThreshold then
10:         event = inactive
11:         Publish ending message
12:         initiateTimer()
13:     end if
14: else if ratio >= triggerThreshold then
15:     event = active
16:     Publish starting message
17:     publishData(data)
18: end if
19: End Procedure
```

When a potential event is detected a starting message containing specific information is being published. This message begins with a standard string, by which the subscriber/server is informed that data from a new event will arrive shortly after. This string is followed by the geographical longitude and latitude along with the timestamp of the event, which are separated by a predefined separator character. After this message has been sent, the function that is executed when an event is detected also sends data from windows prior to the event's detection in order to provide a better understanding of the event. As we mentioned in 3.2.1, the buffer which contains the saved accelerometer data and the timestamps is circular. This means that it needs to be untangled before being copied to a smaller buffer used by the function that publishes the data since not all the windows are to be sent via the MQTT connection. The untangling process is visualized in Figure 3.6. Additionally, timestamps are being converted to differential values, by subtracting the timestamp of the event detection, while also being typecasted from longs to integers and thus reducing memory and network usage.

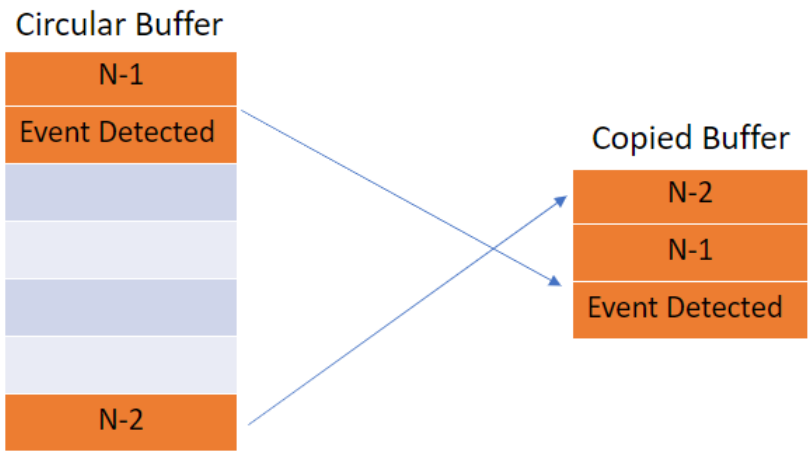


Figure 3.6: Example of circular array untangling.

The next part of the communication process is the function that publishes the data. This function is executed at the end of every window from the time an event is detected until the event is declared finished. This function takes as an argument the buffer with the copied accelerometer data and the differential timestamps. Since each sample has three floats, the three components of the acceleration, and one integer, the differential form of the timestamps, the total payload is $3 * 4 + 4 = 16$ bytes. Since the maximum payload of an MQTT message normally is approximately 260 MB, more than 16.000.000 samples can be sent at once, which in our case is more than enough. It should be mentioned in this part that, MQTT requires the data payloads to be in byte form in order to be able to publish them. For this reason, a byte buffer equal to the size of the copied buffer is allocated. The data are then transformed into their byte forms before they are published as an MQTT message.

We mentioned in 3.2.2 that after a window of data is published, the STA/LTA calculating function checks whether the ratio is lower than the detrigger value in order to declare the event finished. When this happens, in order to inform the subscriber/server that there are no more data from this event to be sent, the function that publishes an ending message is executed. This ending message consists of a standard string payload, that informs the receiver that there are no more data from this event.

3.2.4 Data recording function

The user interface (UI) depicted in Figure 3.7 is used to control data collection for the experimental evaluation. Each of these buttons has a registered onClick function that is ex-

ecuted every time the user clicks them. Through the save button the user has the ability to create a file that has the name the user typed in the text box above. This button's function creates an Intent[26], which asks the user to confirm the creation of the file in a pop-up window. After a file has been created, the user can decide when to start and when to stop recording accelerometer data. This is controlled using a boolean flag variable. When the start button is pressed, the timestamp is noted as the beginning of the recording and the flag's value becomes true. While it is true, another block of code inside the data collecting function is executed. This block is executed every time the accelerometer sends data. When the recording is on, the data are written into a buffered output stream, after they have been saved in the circular buffer. The timestamps are written in their differential value (like 3.2.3). Finally, when the stop button is pressed, the boolean flag's value becomes false and the output stream is closed.

An issue that appeared during the development of this functionality is the storage permissions. Since Android 11, Application Programming Interface (API) level 30, asking for external storage permissions prompts the user to allow the application to access media files only. This can still be solved by requesting legacy external storage if the application is targeting Android 10, API level 29.

Figure 3.7: User interface for data recording.

3.3 MQTT Broker (Mosquitto)

The broker that was used for this project was Eclipse Mosquitto[23]. It is an open-source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1

and 3.1. The Mosquitto project also provides a C library for implementing MQTT clients, as well as the mosquitto_pub and mosquitto_sub command line MQTT clients. The latter two were used for testing the communication mid-development since they function both as a broker and as a publisher/subscriber.

3.4 Fog Server Implementation

As we discussed in Chapter 2, the MQTT protocol requires one or more publishers and one or more subscribers. In this project, a simple subscriber/server has been implemented. The server is a Java[27] application that consists of the following parts. Figure 3.8 describes the functionality of the server-side implemented application.

To begin with, the server application makes a connection with the MQTT broker, whose IP (Internet Protocol) address is already known. Afterward, the server subscribes to the respective topic. When this is completed, the subscriber continuously monitors the connection, waiting to receive a message. Then the server application checks if the message has the correct format, which is described in 3.2.3. If the message begins with the correct string the server application extracts the information about the location and the time of the incoming event and creates a file. After the creation of the file, the server enters a loop, writing the payload of every message it receives into the file. Finally, the server exits the loop when the predefined standard string message, which signals the end of the event, arrives.

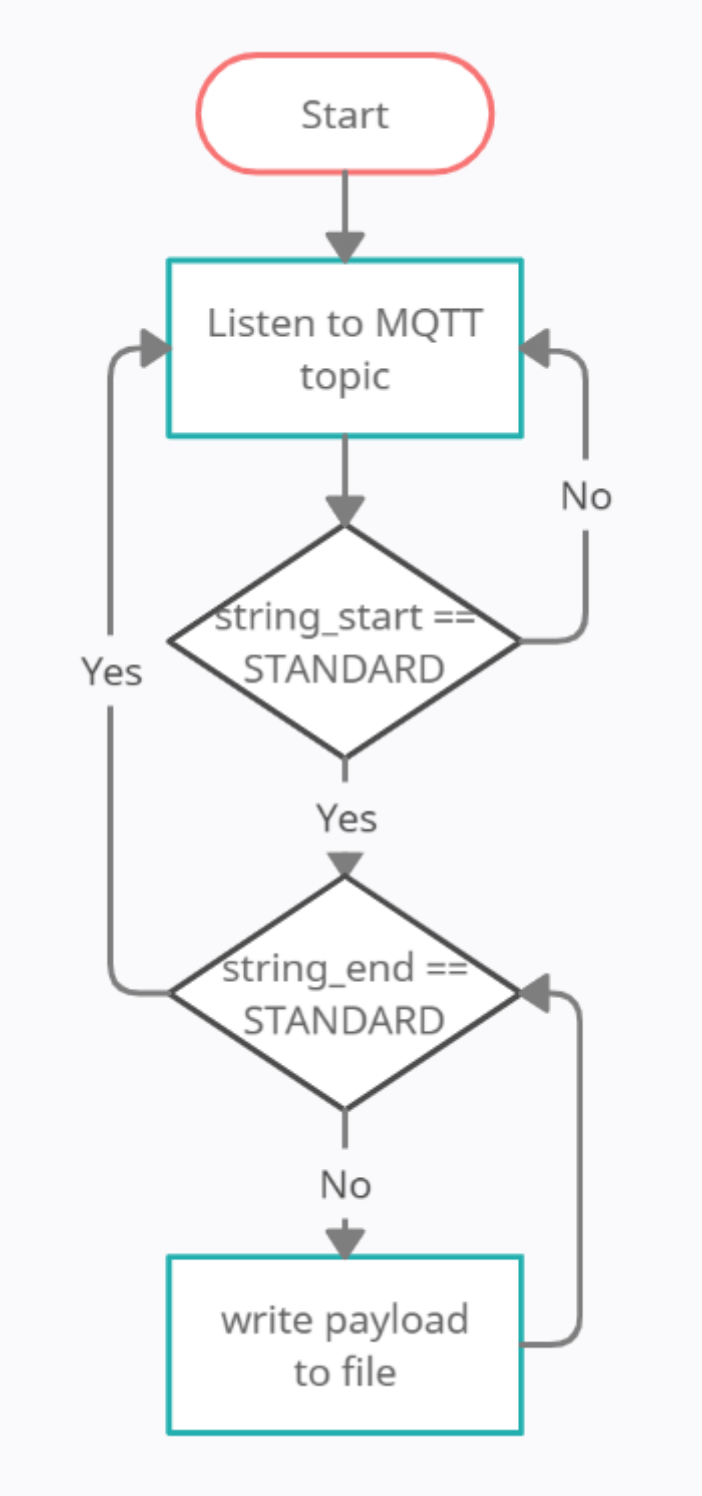


Figure 3.8: Flow diagram of the implemented server-side application.

Chapter 4

Experimental Evaluation

We discussed in Section 2.4 the different effects the STA/LTA algorithm's parameters have in the detection of a potential event. In this section, we will review the proper setting of those parameters in order to avoid as many false triggers as possible in everyday activities. The initial plan was to test the application on a shake table with recordings of real earthquake events. However, access to the equipment was not possible due to the SARS-CoV2 pandemic, therefore the experimental evaluation was limited to sampling and analyzing everyday activities. The parameters under discussion are:

1. the trigger parameter, which should be set accordingly in order to avoid as many false triggers as possible,
2. the de-trigger parameter, which should should be set at a value that allows the algorithm to accurately identify the end of events (neither prematurely, nor too late or - even worse - never),
3. STA and LTA windows, which should be set in such a way that the algorithm's sensitivity to man-made noise will be minimized.

We should also remind, that the size of the window is 80 samples, which is ≈ 0.2 seconds since the sampling rate is at ≈ 400 Hz.

4.1 General Graph Form

To begin with, one should analyze the general form of an STA/LTA ratio graph. Figure 4.1 shows the calculated STA/LTA ratio recorded from a smartphone during a walk. In the begin-

ning, the STA/LTA ratio has values of approximately 1, due to it being in a pocket of a person who stands still. When the person starts moving (when the event starts) STA/LTA ratio rises until it hits its max value and then it begins falling. Here one can observe many interesting things.

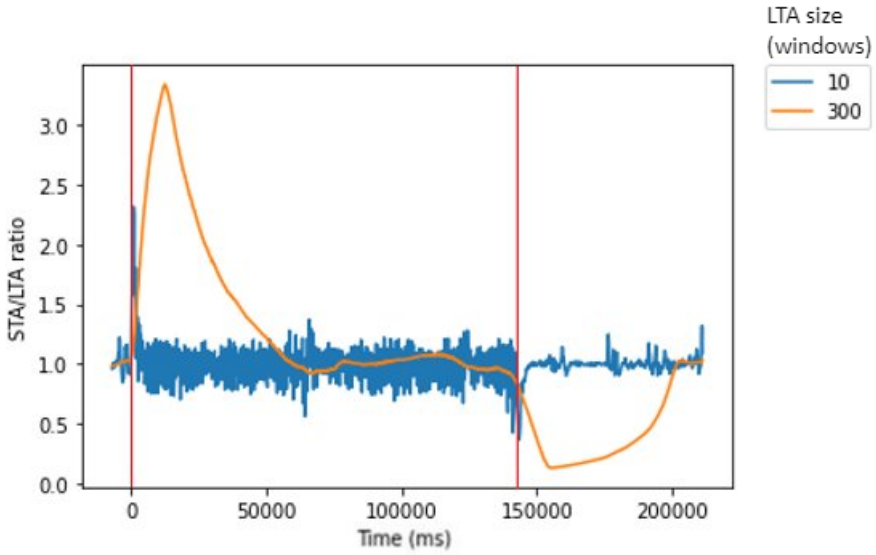


Figure 4.1: STA/LTA ratio of walking with the smartphone in a pocket. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows. The red lines indicate when the event starts and when it finishes.

First of all, the maximum value of the STA/LTA ratio depends on the size of the STA and LTA windows as well as on the ratio of the STA window size/LTA window size, which can be seen in Figure 4.2. This means that with the same size ratio of 1/5 the graph with an LTA window size of 800 samples achieves the peak ratio value of 2.31360 while the graph with the LTA window size of 2400 samples achieves the peak ratio value of 4.02012. On the other hand, with a stable LTA window size of 800 samples, we can observe that the graph with the STA window size of 160 samples achieves a peak ratio of 2.31360 while the graph with the STA window of 400 samples achieves a peak ratio of 1.55705.

Moreover, the rising and falling speed of the STA/LTA ratio is also dependant on the sizes and the ratio of the sizes. This means that with the same size ratio of 1/5 the graph with an LTA window size of 800 samples achieves the peak ratio value in 0.749 seconds after the events start while the graph with the LTA window size of 2400 samples achieves the peak ratio value 12.501 seconds after the events start. On the other hand, with a fixed LTA window size of 800 samples, we can observe that the graph with the STA window size of 160 samples

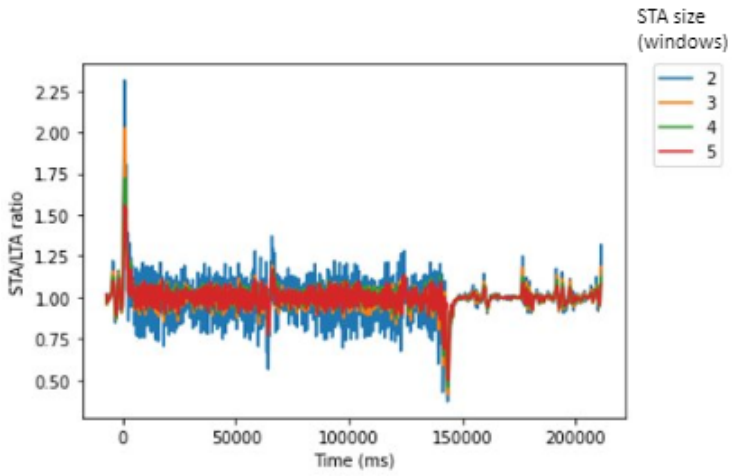


Figure 4.2: STA/LTA ratio of walking with the smartphone in a pocket. The legend describes the number of STA windows each graph has. LTA is set on 10 windows.

achieves the peak ratio 0.749 seconds after the events start, while the graph with the STA window of 400 samples achieves the peak 0.949 seconds after the beginning of the event.

Following, the ratio falls until it reaches values around 1, where it remains for the rest of the duration of the event. In events that have a small duration this part is not observed. Then, when the event is ending, in this case when the person stops moving, the ratio falls even lower, reaching the lowest value which is dependant on the STA and LTA window sizes, similarly to the peak values. This means that with the same size ratio of 1/5 the graph with an LTA window size of 800 samples achieves the lowest ratio value of 0.37057 while the graph with the LTA window size of 2400 samples achieves the lowest ratio value of 0.13520. On the other hand, with an LTA window size fixed at 800, we can observe that the graph with the STA window size of 160 samples achieves the lowest ratio of 0.37057 while the graph with the STA window of 400 samples achieves the lowest ratio of 0.49981.

Finally, the ratio stabilizes again, due to the phone standing still.

4.2 Different Everyday Activities

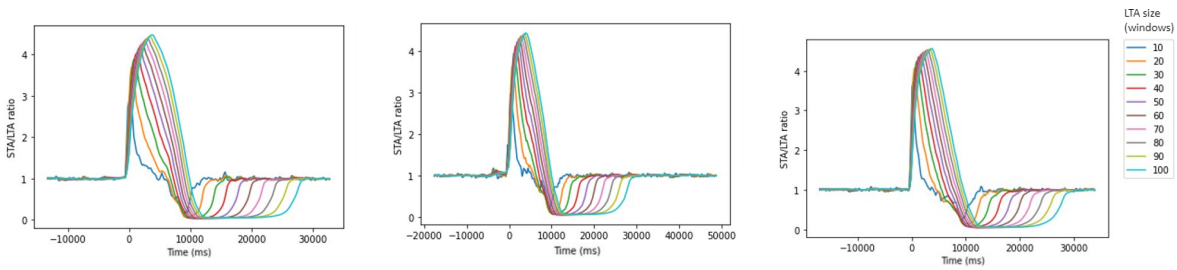
Due to the global pandemic going on during the composing of this Thesis, this is essentially the only experimental evaluation we could do. Had we the experiments on the shake table, the ones in 4.2 would be indications on what not to detect as an event.

After discussing the general form of a graph, one should further analyze the data from some everyday activities that could trigger a false event. In the following subsections, we

experiment on the most ordinary activities. However, first, we should set a reference point. For this reason, we used the event that most closely resembles an earthquake. That event is the shaking of a table.

4.2.1 Shaking the table (earthquake imitation)

In this experiment, a smartphone is placed on a table. After a while, the table is shaken for 5-10 seconds in an attempt to imitate an earthquake-like motion. We can observe from Figure 4.3 that the graph contains most of the basic properties that were under discussion in the previous section. The graph begins with the ratio being stable with values of ≈ 1. After that, the STA/LTA ratio raises when the event starts. However, here we can observe that the ratio might start declining for a little while before it raises to the peak value, instead of going straight for the maximum value. This happens due to the fact that during the shaking of the table there were differences in the intensity of the shaking. Especially, the graphs with the lower LTA sizes include many spikes due to the increasing and decreasing of the intensity, while the graphs with the larger LTA sizes are smoother. Consequently, when the event ends, the ratio drops to its lowest value, where it stays for a while. Then, it returns to its normal levels of ≈ 1. The peak and lowest ratio values of this experiment, along with the timestamps these values are detected, are shown in Table 4.1.



(i) (ii) (iii)

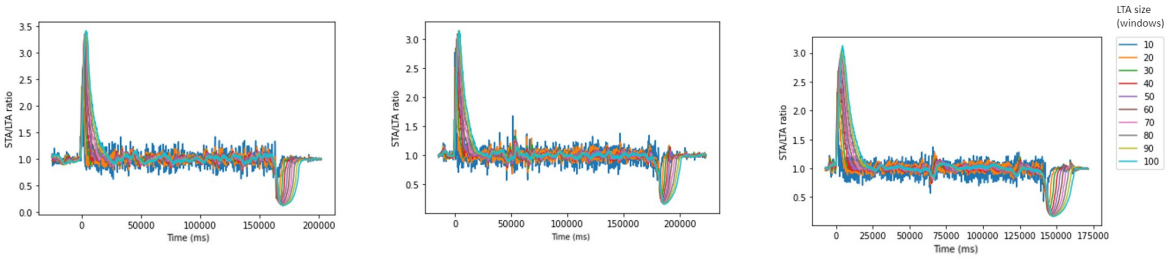
Figure 4.3: STA/LTA ratio of shaking the table where the smartphone rests. This is trying to imitate an earthquake event. The legend describes the number of LTA windows each graph corresponds to. STA is set on 1/5 of LTA windows.

Table 4.1: Highest and lowest values from shaking the table where the smartphone rests. These values depict Figure 4.3i.

LTA size (windows)	Peak ratio	Peak ratio time (ms)	Low ratio	Low ratio time
10	3.095806	160	0.152698	8917
20	3.685992	359	0.064540	9514
30	3.881555	956	0.044640	10310
40	4.039886	1354	0.036528	10708
50	4.164926	1753	0.034937	11704
60	4.249399	2150	0.036327	12699
70	4.321893	2548	0.040071	13097
80	4.381726	2946	0.044900	15087
90	4.432071	3344	0.050297	14689
100	4.470444	3743	0.055435	15087

4.2.2 Walking

In this experiment, a smartphone is placed in the pocket of a standing person. Then the person begins to walk for a while and later they come to a full stop and stands still for some moments. We can observe from the figures in Figure 4.4 that the graph starts off with a stable ratio, with values of ≈ 1. When the event begins, the ratio rises for some milliseconds until it reaches its peak value and then starts to fall off. As we mentioned in the general example, due to the fact that walking is an activity with a long duration, one can observe a phase where the STA/LTA ratio has values near 1 with a deviation of $\pm$ 0.5. We can also notice here that in the graphs with the smaller LTA sizes, after the ratio falls off to values around 1, it has some periodical spikes that do not diverge more than 0.5 from the stabilizing values. The bigger the LTA sizes are the smoother these spikes become and they diverge less. These spikes are caused by the person holding the phone taking steps, which also explains their periodical nature. When the person with the device comes to a stop the ratio drops to its lowest value. Then, after a while, the ratio stabilizes again to normal values of ≈ 1. The peak and lowest ratio values of this experiment, along with the timestamps these values are detected, are shown in Table 4.2.



(i) (ii) (iii)

Figure 4.4: STA/LTA ratio of walking with a smartphone in a pocket. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows.

4.2.3 Desk bump

In this experiment, a smartphone is resting on the table. Then the table is getting bumped by hand, or by throwing a heavy object, like a book, on top of it. We can observe from Figure 4.5 that the ratios start by being stabilized on ≈ 1 due to the phone resting on the table. When the bump (the event) happens, the ratio suddenly spikes to its peak value. In this case, we can observe that even though the LTAs with the larger sizes delay a couple of seconds to reach their peak value, in all the graphs, the ratio appears to spike to a value near the peak almost immediately. After the spike, the ratios stabilize to values $\approx$ their peak values. The duration of the flat part of the curve is proportional to the STA window size since the only acceleration the phone receives is from the bump. When a full STA window passes, the STA/LTA ratio spikes again, falling off to its lowest values. There it stays for a while, before climbing up and stabilizing again to normal values of ≈ 1. The peak and lowest ratio values of this experiment, along with the timestamps these values are detected, are shown in Table 4.3. It should also be noted here that in contrast with all the other experiments, the values of the ratio are inversely proportional to the size of the LTA window, meaning that the bigger the LTA and STA windows become the smaller the peak ratio gets.

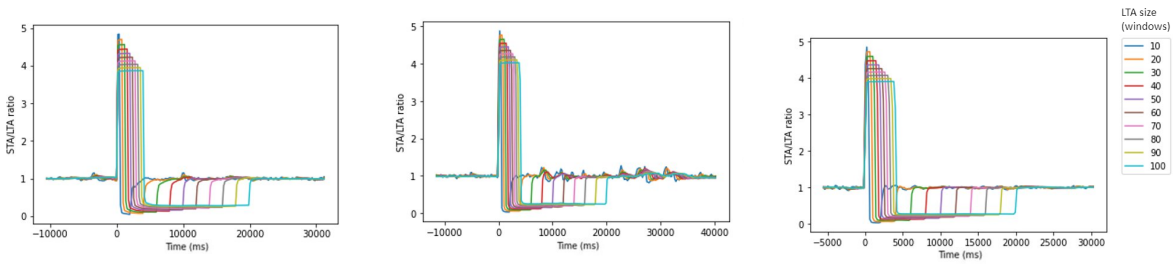
4.2.4 Smartphone pick up

In this experiment, a smartphone was set on the table. After a while, the phone is getting picked up and put down a couple of seconds later. We can observe from Figure 4.6 that like the previous cases the ratio starts being stable at a value of ≈ 1. In this case, before the event, a small noise is observable. This noise is caused by small movements on the device while trying to grab it. Consequently, when the phone gets picked up, the ratio rises to its peak

Table 4.2: Highest and lowest values from walking with the smartphone in a pocket. These values depict Figure 4.4i.

LTA size (windows)	Peak ratio	Peak ratio time (ms)	Low ratio	Low ratio time
10	2.225809	43	0.259855	164427
20	2.461799	2033	0.265191	164626
30	2.878690	1834	0.226176	166417
40	3.083738	2033	0.181241	167213
50	3.242368	2431	0.153797	168010
60	3.300375	2431	0.139432	168407
70	3.343488	2431	0.134290	168805
80	3.363833	2829	0.128662	169203
90	3.396341	3426	0.123721	169601
100	3.416700	3625	0.121409	169999

value. Then the STA/LTA ratio falls off. One can observe here that in the graphs with the small LTA window sizes the ratio spikes again after it falls below 1. This is caused by the second movement, setting the phone to the table again. It is also noticeable that the longer the LTA window size is the less distinguishable the second spike is. After the two movements are over, the ratio drops to its lowest values. Then, after a while, it stabilizes to a value of ≈ 1. The peak and lowest ratio values of this experiment, along with the timestamps these values are detected, are shown in Table 4.4.



(i) (ii) (iii)

Figure 4.5: STA/LTA ratio of bumping the table where the smartphone is resting. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows.

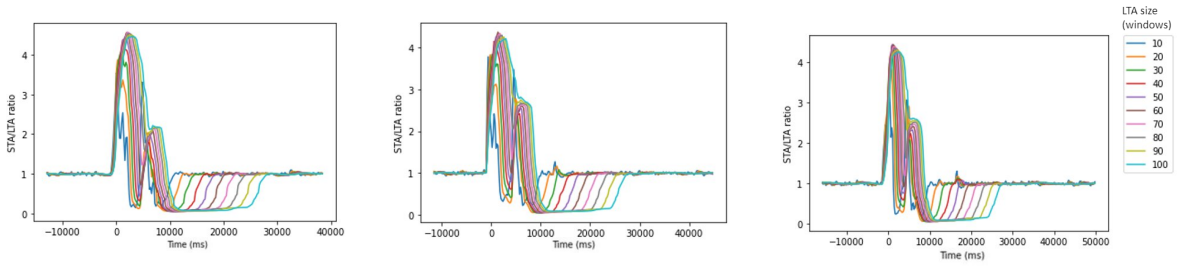
Table 4.3: Highest and lowest values from bumping the table where the smartphone is resting. These values depict Figure 4.5i.

LTA size (windows)	Peak ratio	Peak ratio time (ms)	Low ratio	Low ratio time
10	4.845900	351	0.044575	1943
20	4.704198	749	0.073163	3741
30	4.567620	1154	0.105081	5924
40	4.449641	1545	0.130321	7715
50	4.334980	1943	0.158244	8113
60	4.231240	2341	0.184443	8511
70	4.133628	2739	0.208466	8909
80	4.041489	3138	0.230975	9307
90	3.953614	3535	0.253082	9307
100	3.871175	3933	0.274425	9506

4.2.5 General Conclusions

In the previous subsections, a discussion about the properties of some common everyday activities was made. With these observations in mind, we came to a conclusion about setting the parameters that are under discussion, meaning:

1. the trigger parameter
2. the de-trigger parameter
3. STA and LTA window sizes



(i) (ii) (iii)

Figure 4.6: STA/LTA ratio while picking up a smartphone resting on a table. The legend describes the number of LTA windows each graph has. STA is set on 1/5 of LTA windows.

Table 4.4: Highest and lowest values from picking up the smartphone. These values depict Figure 4.6i.

LTA size (windows)	Peak ratio	Peak ratio time (ms)	Low ratio	Low ratio time
10	3.544719	5	0.147162	3587
20	3.885472	403	0.064933	9360
30	4.120685	1000	0.052278	10355
40	4.379988	1199	0.063787	10355
50	4.491607	1796	0.044343	10555
60	4.569806	1995	0.047351	11549
70	4.568850	2194	0.054892	12743
80	4.536406	2394	0.061458	15330
90	4.496223	2791	0.068059	17520
100	4.460234	2990	0.074790	18117

However, let us analyze the observations that we considered when making our choice.

LTA Window Size

LTA window size affects two performance metrics. The first and most important aspect is that it affects the responsiveness of the algorithm. We can observe in the tables presented earlier that a ratio with an LTA window size of 10 achieves its peak value 3-4 seconds earlier than a ratio with an LTA window size of 100. Due to the fact that for this application any latency matters as it reduces the time between receiving warning and the arrival of the seismic event, we opted for the smaller LTA window sizes. The other aspect that LTA window size affects, is the peak and lowest values. This means that the choice of the trigger and de-trigger parameter values should be made after choosing the LTA window size. Lastly, STA size should be small in comparison to LTA size which means that LTA window size was the first parameter we decided on.

STA Window Size

STA window size has similar effects on the algorithm. With these observations in mind, we opted for smaller STA window sizes too. However, due to the fact that STA window sizes

have smaller deviations from each other, since the values are smaller than LTA window sizes, the effect on the outcome of the algorithm is also smaller. Lastly, this was the second parameter we decided on, since it also has an effect on the values of the two remaining parameters, the trigger and de-trigger thresholds.

Trigger Parameter

We observed in the previous sections, that the relation in general (except the bump experiment), between the peak values and the size of the LTA window is as follows: the larger the LTA window becomes the larger the peak value is. The inverse is also true between the peak values and the size of the STA window, with the peak ratio values getting bigger the smaller the STA window size becomes. However, due to the fact that the imitation of the earthquake event, the experiment with the table shake, achieves the lowest peak ratio values from all the activities under discussion, choosing a value that only the ”real” events will manage to exceed is impossible. Thus, the choice was made so that no ”real” event will be missed.

De-trigger Parameter

Choosing a proper value for this parameter was easier. The value of this parameter should be set with two concerns in mind. First of all, the value should not be set too low, in order for endless events to be avoided. Also, the value should not be set too high, so that the application will not lose any part of the event.

Final Parameters

After considering the above, the choice we made for the LTA window size was 10 windows (800 samples). LTA window size should not be too small, however, we chose this LTA window size due to the fact that for this application the delay is very important and should be minimized as much as possible. With the same thought process, the STA window size was chosen to be 2 windows (160 samples).

After deciding on the STA and LTA window sizes, choosing a trigger parameter value means that the chosen value should be lower than the lowest ”real” event value recorded, such as no real event will be missed. The lowest recorded peak value was 2.236124 which means that setting the trigger parameter value to 2.0 seems a safe choice. Similarly, the chosen de-trigger parameter value should be higher than the highest of the recorded lower pick

ratio values. Since this was 0.259855 setting the de-trigger parameter to 0.4 seemed a proper choice.

4.3 Fog Server Notification Latency

Another fact we analyzed and took under consideration was the delay of the application, in other words, the delay between the start of the event and the time the warning message reaches the MQTT server. In order to study this delay more in-depth, we separated the overall delay into two:

1. the algorithm's buffering delay
2. the communication delay

4.3.1 Algorithm's Buffering Delay

With the term algorithm's buffering delay, we refer to the time between the start of the event and the time the event was detected. This delay is caused due to the fact that the acceleration data come in windows and thus is influenced by the size of the LTA window along with the size of the STA window. We mentioned in the previous section that we chose STA and LTA size mainly considering the minimization of the algorithm delay, however, the elimination of this delay is not possible. Moreover, the computational delay was measured and was found negligible, as it was < 1 ms.

In the previous section, we summarized in Tables 4.1 4.2 4.3 4.4 the elapsed time between the start of the events and the time these events were detected. The average algorithm's buffering delay across all experiments, for STA and LTA windows size equal to 2 and 10 windows respectively, was measured to 139.75 ms.

4.3.2 Communication Delay

With the term communication delay, we refer to the time between the detection of the event and the time the MQTT message arrives at the MQTT server. This delay is heavily influenced by the wireless internet connection of the mobile device, meaning roaming data or WiFi.

In this Thesis, the MQTT message route is from the mobile device to the router via WiFi and then to the desktop, where the MQTT broker and the MQTT server are running, via Ethernet cable. Due to the fact that we do not know if the mobile device's clock and the desktop's clock are in perfect sync, we calculated the round-trip delay, by adjusting the server to reply with an MQTT message back to the phone when the warning message arrives, and then dividing the total elapsed time by 2. The average one-way communication delay was calculated at 10.76 ms by executing the above experiment multiple times.

This means that the total average delay is the sum of the algorithm's buffering and the communication delay, summing up to 150.51 ms.

4.4 Resource Consumption

The resources an application uses while it is running are:

1. processing power (CPU)
2. memory (RAM)
3. network bandwidth
4. power

For inspecting the CPU and the memory usage we used the Android Profiler[28] utility, bundled with Android Studio. For the battery consumption, we used the Android Profiler along with testing it on our physical device.

In Figure 4.7 these values are shown in further detail during a small duration event. Starting from the application's CPU usage, one can see that while the phone is staying still the usage of the processor is between 0 and 3%. Around the 1:25 minute mark, the minor event takes place. Then we can see the application's CPU usage momentarily reach up to 12% before it falls again to 0-3%.

Next, the memory consumption is stable, ranging from 54 to 55 MB. Bellow memory consumption, we can see the network bandwidth usage of the application. While idling, the application uses no network bandwidth. When the event takes place the event uses the network to send the MQTT messages and receive the acknowledgement messages back, reaching a peak network usage of 1,1 KB/s sent and 0,7 KB/s received.

Lastly, concerning power (battery) consumption, according to the Android Profiler, it is light at all times. It is visible that in general, the application consumes negligible power, with a spike in consumption during the event. However, in order to get a more conclusive result, we tested the application in our physical device by letting it run in the background for several hours and then consulting Android's detailed battery usage. At this point, we should mention that many Android-based operating systems have an aggressive approach to optimizing battery consumption by pausing any background activity of an application that spends more than a few minutes in the background. This happened in our case as well. Thus, to calculate the power consumption we ran the application on the foreground. After 8 hours of usage, the application had consumed 2% of the phone's battery (this figure does not include the power consumption for the screen, which was much larger).

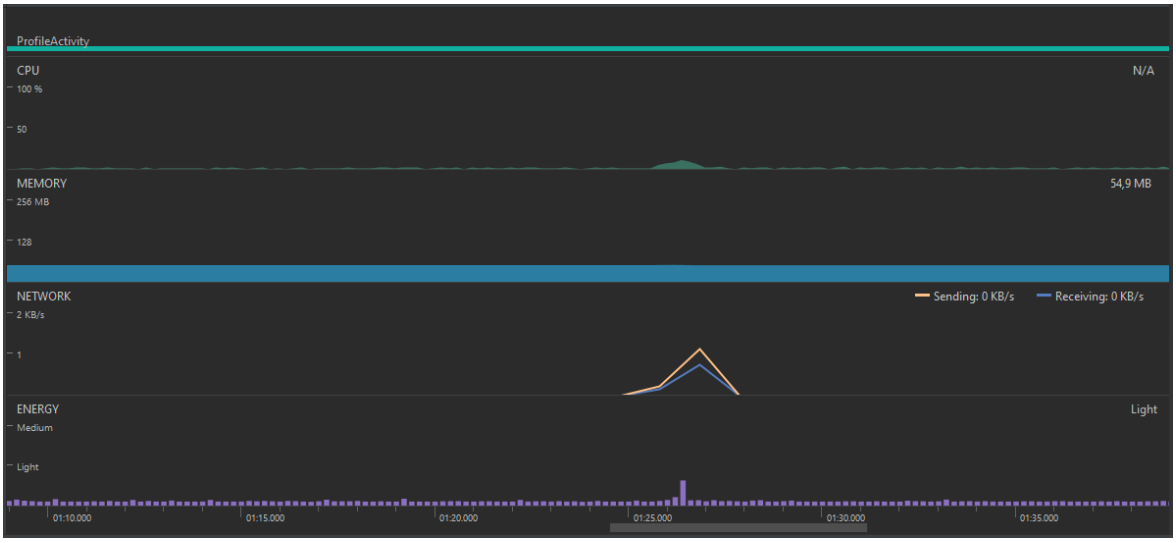


Figure 4.7: Screenshot from the Android Profiler, while running the application on the One-Plus 8.

Chapter 5

Related Work

In this chapter, there will be a short reference in some major EEW systems that, similarly to this project, utilize the smartphone's accelerometer as an earthquake sensor.

5.1 Earthquake Network Project

Earthquake Network Project's[29] application is Earthquake Network - Realtime alerts. The application, similarly to the one developed in this Thesis, detects and reports anomalies in a smartphone's acceleration. However, Earthquake Network's application monitors the accelerometer only when the smartphone is charging and it is not being used. Then, the application sends a signal to a server which, with the gathered signals from all the smartphones and by utilizing a statistical algorithm, will decide in real-time if there is an earthquake going on. In order for an earthquake to be declared, there should be at least six smartphones that are actively monitoring the accelerometer and at least six smartphones should send a signal around the same time. Regarding accuracy, the Earthquake Network's EEW design has some blind spots due to the fact that a smartphone is working as a sensor only when it is charging and not being used. This means that the system has better accuracy during night time, when people typically charge their smartphones, while it is possible to miss the earthquakes that happen during the day. However, it should be noted that the project researchers claim that when the minimum number of active smartphones during the day exceeds 500 in an area, the time of the day does not matter anymore. For the alert distribution, the Firebase Cloud Messaging (FCM)[30] platform is utilized and the messages are being sent according to the distance of the calculated earthquake's epicenter.

5.2 MyShake

MyShake[31] is another widespread EEW system that utilizes smartphones. This system's application uses an artificial neural network (ANN) in order to distinguish the possible earthquake motions from everyday activities, achieving a $\approx$ 93% accuracy. Then, the application sends two messages to a central computer:

1. the trigger information (time of the trigger, location and the maximum amplitude of the three components of acceleration)

2. the waveform data of the three-component acceleration for a short time prior, during and after the event

In order to save resources, the waveform data are sent only when the smartphone has access to WiFi and power. Then, the central computer declares an event if at least 60% of the operating devices in a 10km radius trigger. Moreover, it calculates the earthquake's epicenter, the time the event started as well as its magnitude. Lastly, the central computer calculates the shaking intensity and time until shaking at a user's target location, before sending an alert.

5.3 Android Earthquake Alerts System

Android Earthquake Alerts System[32] is Google's EEW system. It started in partnership with the United States Geological Survey (USGS) and was powered by ShakeAlert[33] as a warning system for earthquakes on the West Coast of the United States. However, it recently expanded to countries that do not have an EEW system, Greece and New Zealand. In these countries, the Android Earthquake Alerts System utilizes the smartphone's accelerometers to detect seismic waves. After the phone detects a possible earthquake it sends a signal to Google's earthquake detection server along with the location and the server then decides, by cross-referencing the data from multiple sources if there is an ongoing event, its location, and its magnitude.

5.4 ARIS Project

ARIS project[34] is another EEW system that is currently being developed in Greece. By using special seismographs, project ARIS's researchers are able to detect the P waves

an earthquake emits. After the detection of the wave, special algorithms are utilized for the computation of the size and the epicenter of the earthquake. Moreover, computations about the remaining time until the destructive earthquake waves arrive are taking place. Finally, the system sends the data to the near vicinity, in order for precautionary actions to take place.

Chapter 6

Conclusion

6.1 Summary

In this diploma Thesis, we implemented a part of an EEW system. To begin with, an application that detects potential earthquake activity and then sends out a warning message to a central computer with information about the time, the location, and some accelerometer data was implemented. For the STA/LTA algorithm that was used to detect the potential earthquake motions, an experimental evaluation was made. This evaluation was also used to properly configure the parameters of the algorithm in order to reduce the elapsed time between the start of the event and the event's detection and increase the accuracy of the algorithm. In addition, the smartphone application offers the functionality of recording accelerometer data along with their respective timestamps in differential form in a file. Moreover, a server-side application was implemented. This server-side application monitors the network for new warning messages and then stores the collected information in files.

6.2 Future Work

A first future step would be to fine tune the detection algorithm. As we discussed earlier, the experiments for the fine-tuning would have included events with the smartphone placed on top of a shake table, in order to get samples about a real event. This would potentially make the detection algorithm able to distinguish more accurately earthquakes from human motions. However, due to the pandemic, this plan was canceled. Another potential future expansion could be an update to the algorithm the smartphone application is using, which

could lead to a better classification between real earthquakes and human motions.

Furthermore, the focus of this Thesis was on the edge application (and partially the fog interface) and an natural next step is the implementation of the full fog server, as well as the cloud server functionality. The completed fog server would decide if an event is ongoing or if the warnings are false alarms, calculate the earthquake's epicenter, and sends out warning messages to the users inside the afflicted areas.

Last but not least, one should test the possible latency that could be caused by a high number of users. More specifically, there are two main reasons for such a latency to appear:

- Network latency (wifi and, mainly, cellular data network) under load.
- Fog server processing under load.

So in order to avoid these types of problems, there needs to be a research based on the systems infrastructure.

Bibliography

[1] Global injuries from natural disasters, earthquake, 1912 to 2019. https://ourworldindata.org/grapher/number-injured-from-disasters?country=~Earthquake.

[2] Economic damage from natural disasters, earthquake, 1906 to 2019. https://ourworldindata.org/grapher/economic-damage-from-natural-disasters?tab=chart&country=~Earthquake.

[3] Greek national seismograph network. http://geophysics.geo.auth.gr/ss/ethniko-diktyo.htm#3.

[4] Benjamin J. Burger. S-P interval seismogram https://commons.wikimedia.org/wiki/File:S-P-interval-seismogram-distance-to-earthquake.jpg.

[5] Smartphone's accelerometer axes. https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events#2904020.

[6] Kharisma, Awal. "What Is Android?" http://developer.android.com/guide … (2011).

[7] Nejat Anbarci, Monica Escaleras, and Charles A. Register. Earthquake fatalities: the interaction of nature and political economy. *Journal of Public Economics*, 89(9):1907 – 1933, 2005.

[8] Number of smartphone users worldwide. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[9] Seismic waves michigan tech. http://www.geo.mtu.edu/UPSeis/waves.html.

[10] S.J Baxter. Earthquake basics. http://udspace.udel.edu/handle/19716/3592, 2000.

[11] Science Learning Hub Pokapū Akoranga Pūtaiao.Seismic waves https://www.sciencelearn.org.nz/resources/340-seismic-waves.

[12] Wikipedia, accelerometer. https://en.wikipedia.org/wiki/Accelerometer.

[13] Bosch bmi261 data sheet. https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi261.html.

[14] Global positioning system (gps). https://www.gps.gov/systems/gps/.

[15] Android developers gps. https://developer.android.com/training/location/change-location-settings.html.

[16] Trnkoczy, A. (2009 online): Understanding and parameter setting of STA/LTA trigger algorithm. - In: Bormann, P. (Ed.), New Manual of Seismological Observatory Practice (NMSOP), Potsdam : Deutsches GeoForschungsZentrum GFZ, 1-20. https://doi.org/10.2312/GFZ.NMSOP_r1_IS_8.1.

[17] [mqtt-v3.1.1-errata01]. MQTT Version 3.1.1 Errata 01. Edited by Andrew Banks and Rahul Gupta. 10 December 2015. OASIS Approved Errata. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/mqtt-v3.1.1-errata01.html.

[18] Oasis open standards. https://www.oasis-open.org/standards/.

[19] Paho android service. https://www.eclipse.org/paho/index.php?page=documentation.php.

[20] Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee, Tim (June 1999). Hypertext Transfer Protocol – HTTP/1.1. IETF. doi:10.17487/RFC2616. RFC 2616.

[21] Google cloud http vs. mqtt. https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols.

[22] John O'Hara. Toward a commodity enterprise middleware: Can amqp enable a new era in messaging middleware? a look inside standards-based messaging with amqp. *Queue*, 5(4):48–55, May 2007.

[23] Roger A. Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265, 2017.

[24] Android. `https://www.android.com`.

[25] Android developers. `https://developer.android.com`.

[26] Android developers intent. `https://developer.android.com/reference/android/content/Intent`.

[27] Java. `https://www.java.com/en/`.

[28] Android profiler. `https://developer.android.com/studio/profile/android-profiler`.

[29] Francesco Finazzi. The Earthquake Network Project: Toward a Crowdsourced Smartphone-Based Earthquake Early Warning System. *Bulletin of the Seismological Society of America*, 106(3):1088–1099, 05 2016.

[30] Firebase cloud messaging (fcm). `https://firebase.google.com/docs/cloud-messaging`.

[31] Kong, Q,. R. M. Allen, L. Schreier, Y.-W. Kwon (2016), MyShake: A smartphone seismic network for earthquake early warning and beyond, Sci. Adv., 2, doi: 10.1126/sciadv.1501055.

[32] Android earthquake alerts system. `https://blog.google/products/android/introducing-android-earthquake-alerts-outside-us/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+blogspot%2FMKuf+%28The+Keyword+%7C+Official+Google+Blog%29`.

[33] Given, D.D., Allen, R.M., Baltay, A.S., Bodin, P., Cochran, E.S., Creager, K., de Groot, R.M., Gee, L.S., Hauksson, E., Heaton, T.H., Hellweg, M., Murray, J.R.,

Thomas, V.I., Toomey, D., and Yelin, T.S., 2018, Revised technical implementation plan for the ShakeAlert system-An earthquake early warning system for the West Coast of the United States: U.S. Geological Survey Open-File Report 2018–1155, 42 p., https://doi.org/10.3133/ofr20181155. [Supersedes USGS Open-File Report 2014–1097.].

[34] «Ολοκληρωμένο Σύστημα Έγκαιρης Προειδοποίησης & Διαχείρισης Σεισμικού Κινδύνου με εφαρμογή σε Βιομηχανικές Υποδομές (ARIS)». ΕΣΠΑ 2014-2020, ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΤΗΤΑ ΚΑΙ ΚΑΙΝΟΤΟΜΙΑ 2014-2020, ΕΡΕΥΝΩ-ΔΗΜΙΟΥΡΓΩ-ΚΑΙΝΟΤΟΜΩ.