Design and implementation of a camera based WSN device utilizing deep learning for resource optimization

M.Sc. Thesis Master of Science in Engineering

By

Ioannis Mavridis

Department of Electrical and Computer Engineering
University of Thessaly

A dissertation submitted to the University of Thessaly in accordance with the requirements of the Diploma degree in the department of Electrical and Computer Engineering.

September 2020

Περίληψη

Τα τελευταία χρόνια υπάρχει μεγάλη τάση ενασχόλησης και ανάπτυξης τεχνολογιών που συνδράμουν στην γεωργία ακριβείας. Τεχνολογίες όπως τα ασύρματα δίκτυα αισθητήρων και η μηχανική μάθηση αποτελούν μια καινοτόμο μέθοδο στην συλλογή δεδομένων που έχουν ως απώτερο σκοπό την αύξηση της παραγωγής και την μείωση του κόστους (άρδευση, λίπανση, κ.α.).

Μια από τις μεθόδους παρακολούθησης της καλλιέργειας είναι αυτή των φυτογραφιών. Μέσω των φωτογραφιών μπορούμε να αρχικά να εποπτεύομε την καλλιέργεια ενώ με την χρήση της μηχανικής μάθησης και της επεξεργασίας εικόνας μπορούμε να εξάγουμε χρήσιμη πληροφορία που σχετίζεται με τον ρυθμό ανάπτυξης, την υγεία της καλλιέργειας καθώς και τις ανάγκες του εκάστοτε καλλιέργειας. Παρόλα αυτά, η λήξη φωτογραφιών σε πεδίο αγρού έρχεται αντιμέτωπη με μια πληθώρα προβλημάτων όπως είναι η τροφοδοσία της συσκευής, ο απομακρυσμένος έλεγχος και η αναδιαμόρφωση της συσκευής ανάλογα με τις εκάστοτε ανάγκες τις εφαρμογής καθώς και η κατανάλωση ενεργείας και δεδομένων καθώς η ως τώρα επεξεργασία των δεδομένων σε τέτοιες εφαρμογές εκτελείται σε απομακρυσμένους διακομιστές Server έχοντας ως αποτέλεσμα την φόρτωση του δικτύου.

Η προσφορά της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η δημιουργία ενός ενεργειακά αυτονόμου κόμβου παρακολούθησης καρπού δένδρων ο οποίος θα έχει την δυνατότητα απομακρυσμένης επαναρύθμισης ενώ παράλληλα θα επεξεργάζεται τα δεδομένα του συλλεγεί με την χρήση μηχανικής μάθησης με σκοπό την μείωση της κατανάλωσης δεδομένων που αποστέλλονται πάνω από το δίκτυο.

Abstract

In recent years there is a great tendency to design and develop technologies that assist in precision agriculture. Technologies such as wireless sensor networks and machine learning are an innovative method of data collection that have the ultimate goal of increasing production and reducing costs (irrigation, lubrication, etc.).

One method for monitoring the crop is that of photographs. Through photographs we can initially monitor the crop while with the use of machine learning and image processing we can extract useful information related to the growth rate, the health of the crop as well as the needs of each crop. However, collecting field photos faces various problems such as powering the device, remote programming and reconfiguration of the device according to the needs of the application as well as energy and data consumption.

The purpose of this thesis is to design and implement of an energy-autonomous tree fruit monitoring node which will have the possibility of remote reconfiguration while at the same time will process the data collected using machine learning in order to reduce the consumption of data sent over the network.

Dedication and acknowledgements

In the first place I want to thank my lead supervisor, Dr. Athanasios Korakis for their assistance and guidance for the completion of this work. I would also like to thank my friends and my family that were by my side all these years.

To my family and friends.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .... DATE: ....

Table of Contents

Page

List of Tables **xi**

List of Figures **xiii**

1 Introduction **1**

1.1 Deep Learning on Wireless Sensor Networks 1

1.2 Propose of this Thesis 2

1.3 Thesis Organization 2

2 Background **5**

2.1 Wireless sensor network 5

2.1.1 WSN Architecture and Protocol Stack 5

2.1.2 Challenges and Constraints 8

2.1.3 WSN Applications 10

2.2 Neural Networks 10

2.2.1 Deep Learning 12

2.2.2 Convolutional Neural Networks 13

3 Implementation **15**

3.1 Hardware 15

3.1.1 Raspberry Pi 15

3.1.2 Waveshare Power Management Hat 16

3.1.3 Charging System 17

3.1.4 Sensor and Wireless communication 19

3.2 Communication Protocols 21

3.2.1 MQTT Protocol 21

3.2.2 UART Protocol 23

3.3 Remote Configuration Protocol 25

3.4 Object Detection 26

3.4.1 Create our own Dataset 27

3.5 The Edge Device 27

4 Experiments Results **29**

4.1 Experiments 29

4.1.1 Experiments Settings 29

4.2 Results of the experiments 30

Bibliography **35**

List of Tables

Table **Page**

3.1 Raspberry Pi 3 B+ Specifications . . . 16
3.2 Waveshare Power Management Hat Specifications . . . 17
3.3 Solar Panel Specifications . . . 18
3.4 Solar Lithium Battery Charger Specifications . . . 18
3.5 Lithium Batteries Specifications . . . 19
3.6 Publications and Subscriptions of each device . . . 23
3.7 Message Type . . . 24
3.8 Encapsulated Information . . . 24

4.1 Experiments Settings . . . 30

List of Figures

Figure **Page**

2.1 Wireless Sensor Network 5
2.2 Edge device components 6
2.3 Energy Consumption of WSN Edge Device 7
2.4 Centralized base station topology. 8
2.5 Multi-hop communication topology. 8
2.6 ISO-OSI protocol stack 8
2.7 Basic Unit of a Artificial Neural Network — Artificial Neuron 11
2.8 Feedforward Neural Network 12

3.1 Raspberry Pi3 B+ 16
3.2 Waveshare Power Management Hat 17
3.3 The components of the charging System 19
3.4 Raspberry Pi Camera v2 Module 20
3.5 Huawei E3372 USB dongle 21
3.6 MQTT communication 22
3.7 UART Communication 23
3.8 Remote Configuration Protocol 25
3.9 The Edge Device 28

4.1 Average current of the node. 31
4.2 The active remain time of the node. 31
4.3 Data consumption when the object existed 32
4.4 Data consumption when the object didn't exist 33

Chapter 1

Introduction

1.1 Deep Learning on Wireless Sensor Networks

Thanks to the availability of advanced and low-cost electronic components, and to the ubiquitous connectivity afforded by the internet, Internet-of-Things (IoT) applications are growing in number and the connected devices are expected to exceed 20 billion units by 2023 [1]. Within such applications, internet-enabled wireless sensor networks WSNs play an important role in many of today's most promising fields, such as smart home, smart city, smart industry and pression agriculture. In our houses, wireless sensor networks can help in optimizing heating and air conditioning control, or contribute to intrusion alarm systems. In the smart city context, WSNs are used, e.g., to monitor traffic and pollution, or to improve the management of public lighting and parking. In the industry, smart connected sensing devices contribute to enhance production efficiency, security, logistics and maintenance planning. In agriculture, the monitoring of environmental data provides an efficient management of agricultural facilities. WSNs potentially provide a huge quantity of data that become especially valuable when supported by mining techniques and artificial intelligence in the cloud.

Deep Learning is a subset of artificial intelligence (AI) Artificial Intelligence refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving. The rapid growth of this sector has come to the fore a drastic change in technological fields, where it can be implemented to automate the system for more efficiency and performance. The applications for artificial intelligence are endless. The technology can be applied to many different sectors and industries. AI is being tested and used in the healthcare industry for dosing drugs and different treatment in patients. In automotive industry, where new models of cars are able for self-driving, thus reducing car accidents. Moreover, in the agriculture sector, AI is able to provide advice to the farmers based on the health of his crop.

The combination of the WSN with the AI introduces a new study field know as Adaptive and Intelligent Wireless Sensor Network [2]. The edge devices of WSN are able to take decisions according to the data that are gathering from the sensors in order to be more robust, reducing the power and data consumption if there is any issue from external factors. This critical thought is too important especially when the WSN collect images that are going to be processed on cloud. Thus, the edge devices of WSN are not just silly platforms that only collect and transmit data, they acquire intelligence.

1.2 Propose of this Thesis

The specific propose of this thesis is to develop an energy-autonomous edge device of WSN that are able to take decisions depending on the data that are collected. The outcomes of this node will be utilized in precision agriculture. Specifically, the main responsibility of this edge device is to monitoring a fruit of a tree using a camera sensor, it will be sampling photos during the day, then it will be processing them utilizes deep learning techniques to extract the region of interest and finally, it will send only the useful information to the cloud for safely stored and for any further processing. However, there are several challenges that taken into account in order to design this node.

- **Power consumption:** This node is going to be deployed at a rural field. Thus, it is extremely difficult to connect to any power source. Therefore, a power management system with solar panel and rechargeable batteries has been developed. Hence, the node is independent of any power infrastructure and makes it portable.
- **Management at a distance:** Edge devices of WSN will be deployed at a rural field. It is difficult to reconfigure the device operation when the network escalates due to the distance of each node. Every time you have to change the sampling time you have to get there and upload a new configuration code. Hence, a remote reconfiguration protocol has been developed in order to chance from the distance the operation of the device.
- **Reduction of bandwidth:** In contrast with the WSNs nodes that gather data from sensors such as temperature and humidity, when you sampling images the reduction of bandwidth is vital. In our case the node collects photos in high resolution and no compression is allowed as the photo is to be processed. Thus, the node must decide when the photo contains the specific object and when not. To address this challenge, an object detection algorithm utilized for two reasons, 1) when the image contains the desired object, this is cropped and only the useful data sent. 2) when the image does not contain the desired object, a warning message sent instead of the image. In both cases achieved a reduction of bandwidth.

1.3 Thesis Organization

This research thesis is organized as follows. In the chapter **Background** we present the background information on the topics of Wireless Sensor Networks and Artificial Neural Networks. We present their architecture and various use case applications that deployed using them. Moreover, in chapter **Implementation** we described the edge device that we designed. Specifically, we present the hardware we used to implement this node, the communication protocols of the edge device, as well as a Remote Configuration Protocol that we implemented to manage the edge device remotely.In addition, we present the object detection algorithm that we used and the operation of the whole device. Finally, in chapter**Experiments Results** we evaluated the node in terms of power consumption and bandwidth, implementing experiments in field conditions.

Chapter 2

Background

2.1 Wireless sensor network

2.1.1 WSN Architecture and Protocol Stack

WSNs, as shown in Figure 2.1, are composed of a number of sensor nodes, which are densely deployed either inside a physical phenomenon or very close to it. The sensor nodes are transceivers usually scattered in a sensor field where each of them has the capability to collect data and route data back to the sink/gateway and the end-users by a multi-hop infrastructureless architecture through the sink. They use their processing capabilities to locally carry out simple computations and transmit only the required and partially processed data. The sink may communicate with the task manager/end-user via the Internet or satellite or any type of wireless network (like WiFi, mesh networks, cellular systems, WiMAX, etc.), making Internet of Things possible. However, in many cases the sink can be directly connected to the end-users. Note that there may be multiple sinks/gateways and multiple end-users in the architecture.

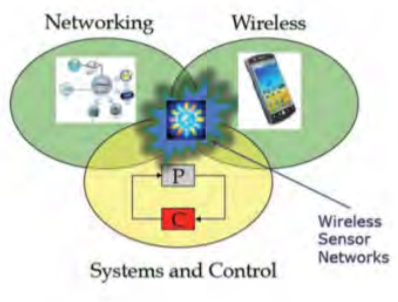


Figure 2.1: Wireless Sensor Network

As illustrated in Figure 2.2, each sensor node is consisting of five main components; a microcontroller unit, a transceiver unit, a memory unit, a power unit and a sensor unit. Each one of these components is determinant in designing a WSN for deployment. The microcontroller unit is in charge of the different tasks, data processing and the control of the other components in the node. It is the main controller of the wireless sensor node, through which every other component is managed. The controller unit may consist of an on-board memory or may be associated with a small storage unit integrated into the embedded board. It manages the procedures that enable the sensor node to perform sensing operations, run associated algorithms, and collaborate with the other nodes through wireless communication. Through the transceiver unit a sensor node performs its communication with other nodes and other parts of the WSN. It is the most power consumption unit. The memory unit is for temporal storage of the sensed data and can be RAM, ROM and their other memory types (SDRAM, SRAM, EPROM, etc.), flash or even external storage devices such as USB.

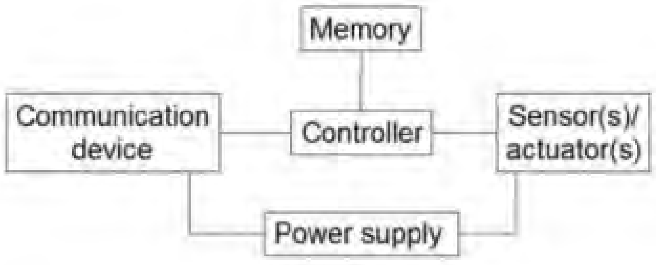


Figure 2.2: Edge device components

The power unit, which is one of the critical components, is for node energy supply. Power can be stored in batteries (most common) rechargeable or not or in capacitors. For extra power supply and recharge, there can be used natural sources such as solar power in forms of photovoltaic panels and cells, wind power with turbines, kinetic energy from water, etc. Last but not least is the sensor unit, which is the main component of a wireless sensor node that distinguishes it from any other embedded system with communication capabilities. It may generally include several sensor units, which provide information gathering capabilities from the physical world. Each sensor unit is responsible for gathering information of a certain type, such as temperature, humidity, or light, and is usually composed of two subunits: a sensor and an analog-to-digital converter (ADC). The analog signals produced by the sensor based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit.

In WSNs, the sensor nodes have the dual functionality of being both data originators and data routers. Hence, communication is performed for two reasons:

- **Source function:** Each sensor node's primary role is to gather data from the environment through the various sensors. The data generated from sensing the environment need to be processed and transmitted to nearby sensor nodes for multi-hop delivery to the sink.

- **Router function:** In addition to originating data, each sensor node is responsible for relaying the information transmitted by its neighbors. The low-power communication techniques in WSNs limit the communication range of a node. In a large network, multi-hop communication is required so that nodes relay the information sent by their neighbors to the data collector, i.e., the sink. Accordingly, the sensor node is responsible for receiving the data sent by its neighbors and forwarding these data to one of its neighbors according to the routing decisions.

Except for their transmit/receive operation state, transceivers can be put into an idle state (ready to receive, but not doing so) where some functions in hardware can be switched off, reducing energy consumption. The breakdown of the transceiver power consumption in Figure 2.3 shows that a transceiver expends a similar amount of energy for transmitting and receiving, as well as when it is idle. Moreover, a significant amount of energy can be saved by turning off the transceiver to a sleep state whenever the sensor node does not need to transmit or receive any data. In this state, significant parts of the transceiver are switched off and the nodes are not able to immediately receive something. Thus, recovery time and startup energy to leave sleep state can be significant design parameters.

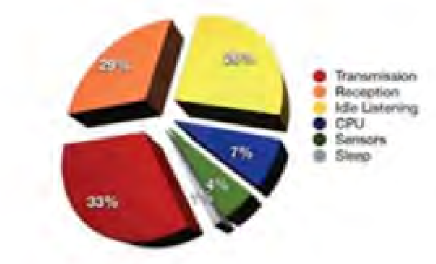


Figure 2.3: Energy Consumption of WSN Edge Device

When the transmission ranges of the radios of all sensor nodes are large enough and the sensors can transmit their data directly to the centralized base station, they can form a star topology as shown in Figure 2.4. In this topology, each sensor node communicates directly with the base station using a single hop.

However, sensor networks often cover large geographic areas and radio transmission power should be kept at a minimum in order to conserve energy; consequently, multi-hop communication is the more common case for sensor networks (shown in Figure 2.5). In this mesh topology, sensor nodes must not only capture and disseminate their own data, but also serve as relays for other sensor nodes, that is, they must collaborate to propagate sensor data towards the base station. This routing problem, that is, the task of finding a multi-hop path from a sensor node to the base station, is one of the most important challenges and has received large attention from the research community. When a node serves as a relay for multiple routes, it often has the opportunity to analyze and pre-process sensor data in the network, which can lead to the elimination of redundant information or aggregation of data that may be smaller than the original data.

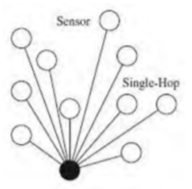


Figure 2.4: Centralized base station topology.

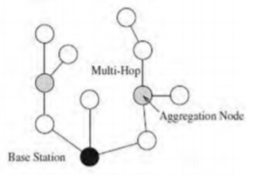


Figure 2.5: Multi-hop communication topology.

The reduced ISO-OSI protocol stack used by the sink and all sensor nodes is given in Figure2.6. This protocol stack combines power and routing awareness, integrates data with networking protocols, communicates power efficiently through the wireless medium, and promotes cooperative efforts of sensor nodes. The protocol stack consists of the physical layer, medium access control layer, routing layer and application layer. The physical layer addresses the needs of simple but robust modulation, transmission, and receiving techniques. Since the environment is noisy and sensor nodes can be mobile, the medium access control layer is responsible for ensuring reliable communication through error control techniques and manage channel access to minimize collision with neighbors' broadcasts. The routing layer takes care of routing the data and depending on the sensing tasks, different types of application software can be built and used on the application layer.

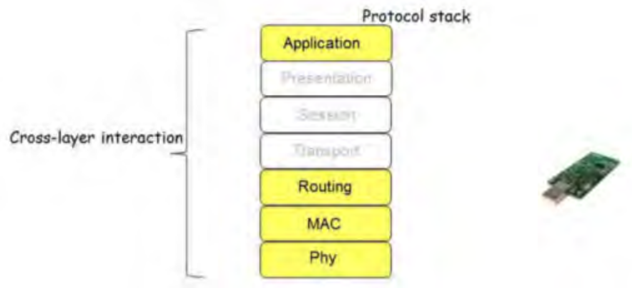


Figure 2.6: ISO-OSI protocol stack

2.1.2 Challenges and Constraints

While WSNs share many similarities with other distributed systems, they are subject to a variety of unique challenges and constraints. These constraints impact the design of a WSN, leading to protocols and algorithms that differ from their counterparts in other distributed systems.

Energy

The intrinsic properties of individual sensor nodes pose additional challenges to the communication protocols primarily in terms of energy consumption. As will be explained in the following chapters, WSN applications and communication protocols are mainly tailored to provide high energy efficiency. Sensor nodes carry limited power sources. Typically, they are powered through batteries, which must be either replaced or recharged (e.g., using solar power) when depleted. For some nodes, neither option is appropriate, that is, they will simply be discarded once their energy source is depleted. Whether the battery can be recharged or not significantly affects the strategy applied to energy consumption. Therefore, while traditional networks are designed to improve performance metrics such as throughput and delay, WSN protocols focus primarily on power conservation.

Node Deployment

The deployment of WSNs is another factor that is considered in developing WSN protocols. The position of the sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this random deployment requires the development of self-organizing protocols for the communication protocol stack. In particular, sensor nodes must be selfmanaging in that they configure themselves, operate and collaborate with other nodes, and adapt to failures, changes in the environment, and changes in the environmental stimuli without human intervention. Moreover, many sensor networks, once deployed, must operate unattended, that is, adaptation, maintenance, and repair must be performed in an autonomous fashion. In energy-constrained sensor networks, all these self-management features must be designed and implemented such that they do not incur excessive energy overheads.

Wireless Medium

The reliance on wireless networks and communications poses a number of challenges to a sensor network designer. Large and small-scale fading limit the range of radio signals, that is, a radio frequency (RF) signal attenuates while it propagates through a wireless medium. The received power is proportional to the inverse of the square of the distance from the source of the signal. As a consequence, an increasing distance between a sensor node and a base station rapidly increases the required transmission power. Therefore, it is more energy-efficient to split a large distance into several shorter distances, leading to the challenge of supporting multi-hop communications and routing. Multi-hop communication requires that nodes in a network cooperate with each other to identify efficient routes and to serve as relays.

Hardware Constraints

While the capabilities of traditional computing systems continue to increase rapidly, the primary goal of wireless sensor design is to create smaller, cheaper, and more efficient devices. The five node components described before should also fit into a matchbox-sized embedded system. A sensor's hardware constraints also affect the design of many protocols and algorithms executed in a WSN. For example, routing tables that contain entries for each potential destination in a network may be too large to fit into a sensor's memory. Instead, only a small amount of data (such as a list of neighbors) can be stored in a sensor node's memory. Further, while in-network processing can be employed to eliminate redundant information, some sensor fusion and aggregation algorithms may require more computational power and storage capacities than can be provided by low-cost sensor nodes. Therefore, many software architectures and solutions (operating system, middleware, network protocols) must be designed to operate efficiently on very resourceconstrained hardware.

2.1.3 WSN Applications

The emergence of the WSN paradigm has triggered extensive research on many aspects of it. The applicability of sensor networks has long been discussed with emphasis on potential applications that can be realized using WSNs. In this section, an overview of certain applications developed for WSNs is provided.

Military or Border Surveillance Applications

WSNs are becoming an integral part of military command, control, communication and intelligence systems. The need of rapid deployment and selforganization characteristics of sensor networks make them a very promising sensing technique for military applications. Since sensor networks are based on the dense deployment of disposable and low-cost sensor nodes, which makes the sensor network concept a better approach for battlefields. Sensors can be deployed in a battle field to monitor the presence of forces and vehicles, and track their movements, enabling close surveillance of opposing forces.

Environmental Applications

The autonomous coordination capabilities of WSNs are utilized in the realization of a wide variety of environmental applications. Some environmental applications of WSNs include tracking the movements of birds, small animals, and insects; monitoring environmental conditions that affect crops and livestock; temperature, humidity and lighting in office buildings; irrigation; large-scale earth monitoring and planetary exploration. These monitoring modules could even be combined with actuator modules which can control, for example, the amount of fertilizer in the soil, or the amount of cooling or heating in a building, based on distributed sensor measurements.

Agriculture

Using wireless sensor networks within the agricultural industry is increasingly common; using a wireless network frees the farmer from the maintenance of wiring in a difficult environment. Gravity feed water systems can be monitored using pressure transmitters to monitor water tank levels, pumps can be controlled using wireless I/O devices and water use can be measured and wirelessly transmitted back to a central control center for billing. Irrigation automation enables more efficient water use and reduces waste.

2.2 Neural Networks

An Artificial Neural Network (ANN) is a computational model that is inspired by the way biological neural networks in the human brain process information. Artificial Neural Networks have generated a lot of excitement in Machine Learning research and industry, thanks to many breakthrough results in speech recognition, computer vision and text processing.

A Single Neuron

The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output. Each input has an associated weight, which is assigned on the basis of its relative importance to other inputs. The node applies a function f to the weighted sum of its inputs.

Feedforward Neural Network

The feedforward neural network was the first and simplest type of artificial neural network. It contains multiple neurons (nodes) arranged in layers. Nodes from adjacent layers have connections of edges between them. All these connections have weights associated with them.

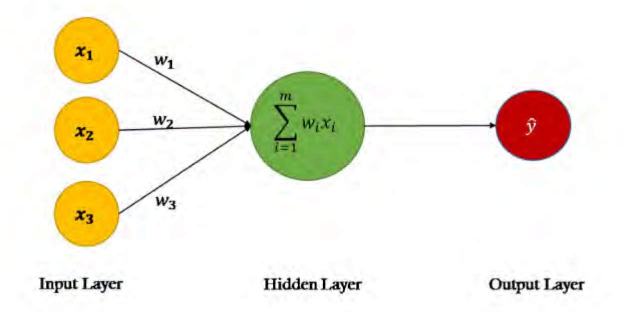


Figure 2.7: Basic Unit of a Artificial Neural Network — Artificial Neuron

A feedforward neural network can consist of thee types of nodes:

- **Input Nodes:** The inputs nodes provide information from outside world to the network and are together referred to as the “Input Layer”. No computation is performed in any of the Input Nodes and they pass on information to the hidden nodes.
- **Hidden Nodes:** The Hidden Nodes have no direct connection with the outside world. They perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a “Hidden Layer”. While a feedforward network will only have a single input layer and a single output layer, it can have a zero or multiple Hidden Layers.
- **Output Nodes:** The Output nodes are collectively referred to as the “Output Layer” and are responsible for computations and transferring information from the network to the outside world.

In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes and to the output nodes. There no cycles or loops in the network. Two examples of feedforward networks are existed a) Single Layer Perceptron(2.8): This is simplest feedforward neural network and does not contain any hidden layer. b) Multi Layer Perceptron : A Multi Layer Perceptron has one or more hidden layers.

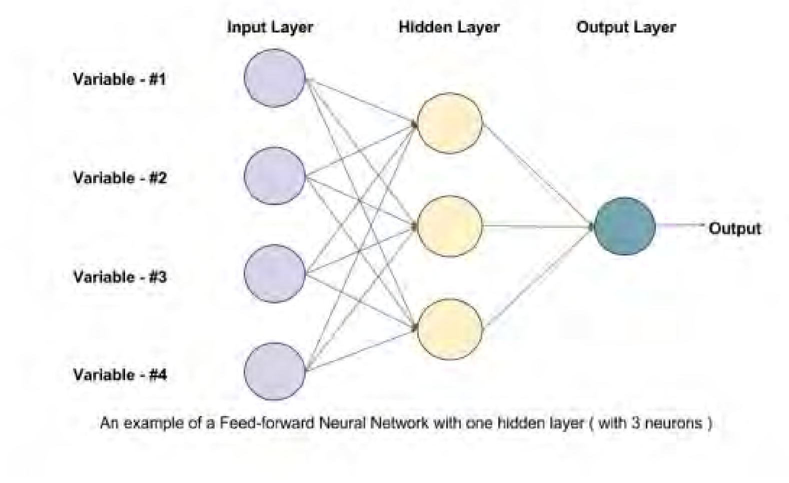


Figure 2.8: Feedforward Neural Network

Training a MLP: The Back - Propagation Algorithm

The process by which a Multi Layer Perceptron learns is called the Back Propagation algorithm. Backward Propagation of Errors, often abbreviated as BackProp is one of the several ways in which an artificial neural network can be trained. It is a supervised training sheme, which means, it learns from labeled training data. There is a supervisor to guide its learning. BackProp is like "learning from mistakes". The supervisor corrects the ANN whenever it makes mistakes. An ANN consists of nodes in different layers, input layer, intermediate hidden layers and the output layer. The connections between nodes of adjacent layers have "weights" associated with them. The goal of learning is to assign correct weights for these edges. Given an input vector these weights determine what the output vector is. In supervised learning, the training set is labeled. This means, for some given inputs we know the desired/expected output (label).

2.2.1 Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans, learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state of the art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

What's the Difference Between Machine Learning and Deep Learning?

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs end-to-end learning, where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically. Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network. A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

2.2.2 Convolutional Neural Networks

Convolution Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. ConvNets are an important tool for most machine learning practitioners today. CNNs is used at vision, audio, and even natural language processing applications which researchers and engineers have built amazing applications using CNNs.

The LeNet Architecture

Lenet was one of the very first convolutional neural networks which help on Deep Learning. This pioneering work by Yann LeCun was named LeNet5 after many previous successful recognition tasks such as reading zip codes, digits, etc. There have been several new architectures proposed in the recent years which are improvements over the LeNet, but they all use the main concepts from the LeNet.

There are four main operations in the ConvNet. These operations are the basic building blocks of every Convolutional Neural Network.

- Convolution
- Non Linearity (ReLU)
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

Convolution:

ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. To produce a convolved feature, we slide the 3x3 Matrix over the image array (5x5 matrix) and for every position, compute element wise multiplication (between the two matrices) and add the multiplication outputs to get the final integer which forms a single element of the output matrix. The 3x3 martix is called a filter or kernel or feature detector and the matrix formed by sliding the filter over the image and computing the dot product is called Convolved Feature or Action Map or Feature Map. That filters acts as feature detectors from original input image.

A CNN learns the values of these filters on its own during the training process, although we still need to specify parameters such number of filters, filter size, architecture of the network etc before the training process. The more number of filter we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images. The size of the Feature Map (Convolved Feature) is controlled by three parameters:

- **Depth:** Depth corresponds to the number of filters we use for the convolution operation.
- **Stride:** Stride is the number of pixels by which we slide our filter matrix over input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.
- **Zero-padding:** It is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is called wide convolution, and not using it wound be a narrow convolution.

Non Linearity (ReLU)

An addition operation called ReLU has been used after every convolution operation. ReLU stands for Rectified Linear Unit and is non-linear operation. ReLU is an element operation where applied per pixel and replaces all negative pixel values in the feature map by zero. Other non linear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

Pooling Layer

The Pooling layer can be seen between Convolution layers in a CNN architecture. This layer basically reduces the number of parameters and computation in the network, controlling overfitting by progressively reducing the spatial size of the network. Spatial Pooling, also called subsampling or downsampling, reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types, Max, Average, Sum etc. In the network, pooling operation is applied separately to each feature map. The function of Pooling is to progressively reduce the spatial size of the input representation.

Fully Connected Layer The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer, other classifiers like SVM can also be used. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

Chapter 3

Implementation

This section presents the design of the energy autonomous WSN edge device that can make a decision based on its collection data. This development takes into account the key concerns of WSN edge devices such as energy consumption, limited bandwidth recourses and remote configuration of device. To achieve this, we combined various embedded hardware and developed a firmware to act on them as a single unit. More specifically, in this section are described the hardware that is used, the communication protocol, the remote configuration protocol, the object detection algorithm and the edge device operation

3.1 Hardware

3.1.1 Raspberry Pi

One of the main tasks of this device is to make decisions using a deep learning algorithm as we want to extract from an image if the object of desire existed. Unfortunately, these kinds of algorithms cannot be executed on traditional WSN platforms (Arduino, Waspmote, ESP, Teensy) as they are microcontroller-based platforms. To address this, we used a single board computer namely Raspberry Pi 3 B+(Figure 3.1). The Raspberry Pi is a low cost, credit-card sized computer made by the Raspberry Pi Foundation. Early on, the Raspberry Pi project leaned towards the promotion of teaching basic computer science in schools and in developing countries. Later, the original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It is now widely used in many areas, such as for weather monitoring, because of its low cost and high portability.

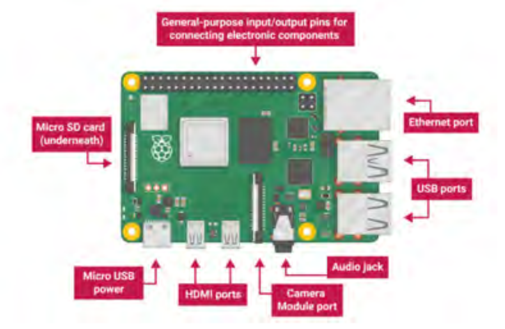

General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Audio jack
Micro USB power
HDMI ports
Camera Module port

Figure 3.1: Raspberry Pi3 B+

Specifications

SoC	Broadcom BCM2837B0
CPU	4× Cortex-A53 1.4 GHz
GPU	Broadcom VideoCore IV @ 250 MHz
Memory (SDRAM)	1 GiB
USB 2.0 ports	4
On-board storage	MicroSDHC slot, USB Boot Mode
WiFi IEEE 802.11 wireless	b/g/n/ac dual band 2.4/5 GHz
Low-level peripherals	17× GPIO plus the same specific functions, and HAT ID bus
Power ratings	459 mA average when idle, 1.13A maximum under stress

Table 3.1: Raspberry Pi 3 B+ Specifications

In our case, the Raspberry Pi is a convenient platform due to its small size that makes it portable, its processing power that allows the execution of deep learning algorithms and its peripheral ports (GPIO, USB) that make it combined with other devices, integrated platforms or sensors. Since the Raspberry Pi does not have a built-in RTC and micro-controller, it is impossible to operate as a WSN platform with duty cycles. For these reasons it was necessary to combine it with an external micro-controller platform.

3.1.2 Waveshare Power Management Hat

Power Management HAT (Figure 3.2) is a smart power bank designed for Raspberry Pi. With the embedded Arduino MCU and RTC, the HAT features auto power management function which allows the Pi to work more power-efficient and more safely. By auto starting up on specified period, and auto shutdown at other time, the HAT can greatly save the battery life. It monitors the operating voltage/current status of the Raspberry Pi in real time, and can be configured to shut the Pi down according to the operating status.

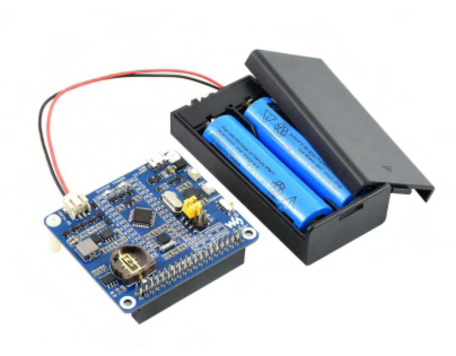


Figure 3.2: Waveshare Power Management Hat

Specifications

Controller	ATmega328P-AU
Communication interface	UART + GPIO
Baudrate	115200bps by default (programmable)
Power supply interface	USB port or PH2.0 connector
USB power input voltages	5V
PH2.0 power input voltage	DC 7 28V (regulated power supply or lithium battery)
Embedded circuits	power supply anti-reverse, counter current proof voltage monitoring, current monitoring
Mounting hole size	3.0mm

Table 3.2: Waveshare Power Management Hat Specifications

In practice, this platform is powered by two rechargeable batteries and controls the power of the Raspberry Pi. It can be programmed to turn the Raspberry Pi on / off using a combination of tiny transistors. Thus, we achieve the duty cycles of the node. The Waveshare Power Management Hat and the Raspberry Pi have to operate as a single unit so a communication protocol has developed that is descripted in following sections.

3.1.3 Charging System

As we want to build an energy-autonomous edge device a charging system need. In our case, we use a solar charging system that consists of two solar panel 1W, two Solar Lithium Battery Charger and two Lithium Batteries.

3.1.3.1 Solar Panel

A solar panel (Figure 3.3(a)) is actually a collection of solar (or photovoltaic) cells, which can be used to generate electricity through photovoltaic effect. These cells are arranged in a grid-like pattern on the surface of solar panels. Thus, it may also be described as a set of photovoltaic modules, mounted on a structure supporting it. A photovoltaic (PV) module is a packaged and connected assembly of 6×10 solar cells.

Specifications

Chemical type	Monocrystalline silicon solar cells
Structure type	Homojunction solar cells
Use the state	Flat-panel solar cells
Output power	1W
Current @ Pmpp	166mA
Voltage @ Pmpp	6V
Operating current	0-200mA
Current - Short Circuit (Isc)	180mA
Operating Temperature	0...70°C

Table 3.3: Solar Panel Specifications

3.1.3.2 Solar Lithium Battery Charger

In our case we used the Solar Lithium Battery Charger – CN3065 (Figure 3.3(b)), is a super mini Solar Lipo charger based on the CN3065 - a single lithium battery charge management chip. This Solar charger provide you with the ability to get the most possible power out of your solar panel or other photovoltaic device and into a rechargeable LiPo battery. The output of the Solar Charger is intended to charge a single polymer lithium ion cell. The load should be connected in parallel with the battery. By default, the Solar charge comes set to a maximum charge current of 500mA with a maximum recommended input of 6V (minimum 4.4V). It's recommended that batteries not be charged at greater than their capacity rating.

Specifications

Solar panel input	4.4 - 6V
Max charge current	500mA
Interface	2-pin JST connectors (or PH2.0)
Short circuit protection	Yes
Continuous Charge Current	Up to 500mA
Battery status indication	Red: Charging Green: Charged
Support USB Charge	Micro-USB Connector
Dimensions	20x40mm

Table 3.4: Solar Lithium Battery Charger Specifications

3.1.3.3 Lithium Batteries

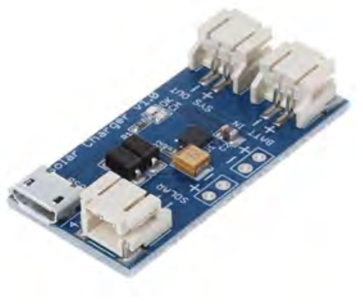
A lithium-ion battery or Li-ion battery (Figure 3.3(c)) is a type of rechargeable battery. Lithium-ion batteries are commonly used for portable electronics and electric vehicles and are growing in popularity for military and aerospace applications.

Specifications

Nominal Voltage	3.6V
Nominal Capacity	3350 mAh
Minimum Discharge Voltage	3V
Maximum Discharge current	1C
Charging Voltage	4.2V (maximum)
Charging current	0.5C
Charging Method	CC and CV
Cell Weight	48g (approx)
Cell Dimension	18.4mm (dia) and 65mm (height)

Table 3.5: Lithium Batteries Specifications

(a) Solar Panel


(b) Solar Lithium Battery Charger

(c) Lithium Battery

Figure 3.3: The components of the charging System

3.1.4 Sensor and Wireless communication

The purpose of this node is to collect and send images of tree fruits for further processing to a remote server. Through this processing information on the health status of the fruit could be extracted. Thus, the images that are going to be sent over the network should be in the best resolution. In our node we used the Raspberry Pi Camera v2 Module (Figure 3.4).

The Camera v2 is the new official camera board released by the Raspberry Pi foundation. The Raspberry Pi Camera Module v2 is a high quality 8-megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSi interface, designed especially for interfacing to cameras.

- 8-megapixel native resolution sensor-capable of 3280 x 2464 pixel static images.
- Supports 1080p30, 720p60 and 640x480p90 video.
- Camera is supported in the latest version of Raspbian, Raspberry Pi's preferred operating system.

The board itself is tiny, at around 25mm x 23mm x 9mm. It also weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. It connects to Raspberry Pi by way of a short ribbon cable. The high-quality Sony IMX219 image sensor itself has a native resolution of 8 megapixel, and has a fixed focus lens on-board. In terms of still images, the camera is capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p90 video.

Figure 3.4: Raspberry Pi Camera v2 Module

The communication with the cloud performed using the Huawei E3372 USB dongle (Figure 3.5). This device is a portable and your connection moves with you, this is ideal for students and commuters who need a reliable but cost-effective option for Internet access. The device has a secure download speed of 150 Mbps and upload speed of 50 Mbps, allowing you to stream or download videos and films in no time.

Figure 3.5: Huawei E3372 USB dongle

3.2 Communication Protocols

In this thesis we worked on two communication protocols for the exchange of information over the network as well as between devices. More specifically, we use the MQTT protocol for the communication of edge device with the Server. Through the MQTT the edge device publishes the data that gather during its operation and server is able to publish a new configuration for a specific edge device (descripted at 3.3). The second communication protocol is the UART for the communication between the Raspberry Pi and the microcontroller unit of the Waveshare Power Management Hat. Through, the UART the Raspberry Pi and the Waveshare Power Management Hat operates as a single unit.

3.2.1 MQTT Protocol

The MQTT (Message Queuing Telemetry Transport) is an open OASIS and ISO standard (ISO/IEC 20922) lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT. It is designed for connections with remote locations where a ”small code footprint” is required or the network bandwidth is limited.

The MQTT protocol defines two types of network entities: a message broker and a number of clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients. An MQTT client is any device (from a micro controller up to a full-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connected broker. The broker then distributes the information to any clients that have subscribed to that topic. The publisher does not need to have any data on the number or locations of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers.

If a broker receives a message on a topic for which there are no current subscribers, the broker discards the message unless the publisher of the message designated the message as a retained message. A retained message is a normal MQTT message with the retained flag set to true. The broker stores the

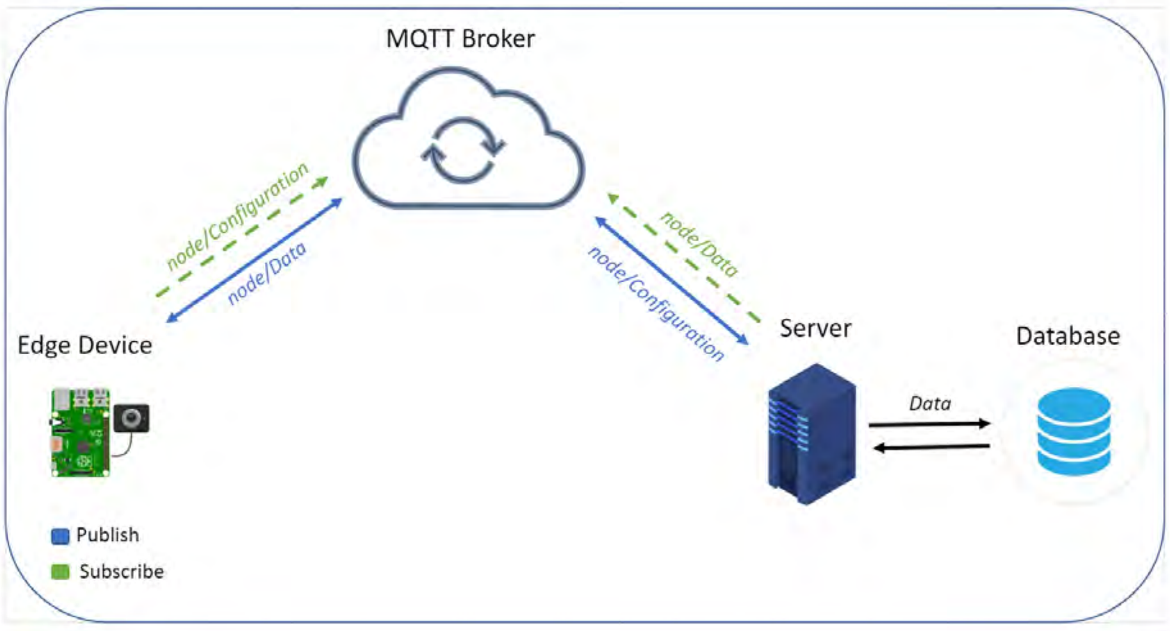


Figure 3.6: MQTT communication

last retained message and the corresponding QoS for the selected topic. Each client that subscribes to a topic pattern that matches the topic of the retained message receives the retained message immediately after they subscribe. The broker stores only one retained message per topic. This allows new subscribers to a topic to receive the most current value rather than waiting for the next update from a publisher. When a publishing client first connects to the broker, it can set up a default message to be sent to subscribers if the broker detects that the publishing client has unexpectedly disconnected from the broker. Clients only interact with a broker, but a system may contain several broker servers that exchange data based on their current subscribers' topics.

A minimal MQTT control message can be as little as two bytes of data. A control message can carry nearly 256 megabytes of data if needed. There are fourteen defined message types used to connect and disconnect a client from a broker, to publish data, to acknowledge receipt of data, and to supervise the connection between client and server. MQTT relies on the TCP protocol for data transmission. A variant, MQTT-SN, is used over other transports such as UDP or Bluetooth. MQTT sends connection credentials in plain text format and does not include any measures for security or authentication. This can be provided by using TLS to encrypt and protect the transferred information against interception, modification or forgery.

The Figure 3.6 shows how the data is stored in the server database via the MQTT protocol and how the server sends new configuration packets to a specific edge device. Each edge device publishes its own data on a topic that describes its location and the type of data. Moreover, the edge device subscribes to the topic for the new configuration packets.

Device	Action	Topic
Edge Device	Subscribe	deployment/device-type/device-id/configuration
	Publish	deployment/device-type/device-id/value-type
Server	Subscribe	deployment/device-type/device-id/value-type
	Publish	deployment/device-type/device-id/configuration

Table 3.6: Publications and Subscriptions of each device

3.2.2 UART Protocol

In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART

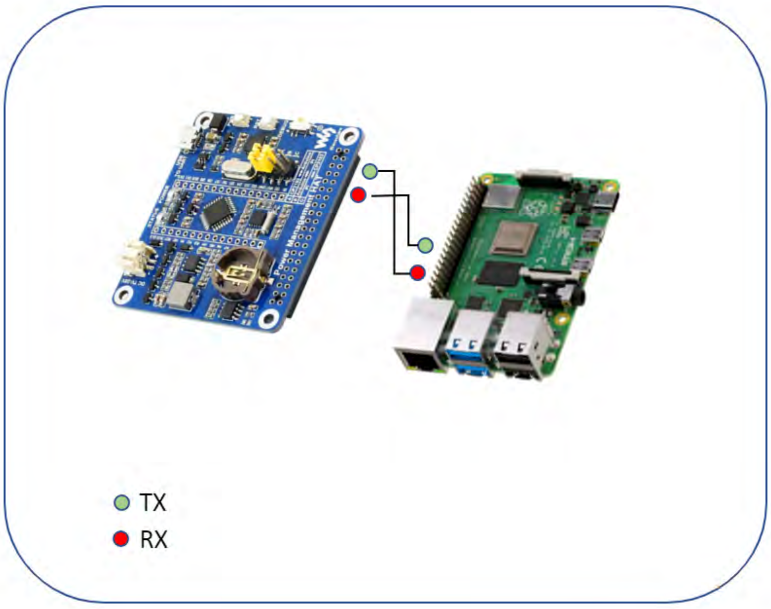


Figure 3.7: UART Communication

UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits. When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off. Both UARTs must also must be configured to transmit and receive the same data packet structure.

The Figure 3.7 shows the connection of the Raspberry Pi and Waveshare Power Management Hat. Over the UART we have built our communication protocol for these two devices. With this protocol the devices are able to exchange information, while is used in the Remote Configuration mechanism in order to change the configuration of the node from a distance.

The messages of this protocol have the following structure as show in Table 3.7

Msg Type	Further Info	Description	Sender	Receiver
UART Open	No	UART port connection has been established	Raspberry Pi Waveshare	Waveshare Raspberry Pi
End Process	No	The particular task of the Raspberry Pi has completed successfully	Raspberry Pi	Waveshare
Update	No	Raspberry Pi task for this cycle is to check the broker for any reconfiguration message	Waveshare	Raspberry Pi
Sampling	No	Raspberry Pi task for this cycle is to take a picture	Waveshare	Raspberry Pi
Data	Yes	This message contains statistic data	Waveshare	Raspberry Pi
Config	Yes	This message contains information about restructuring the device	Raspberry Pi	Waveshare

Table 3.7: Message Type

In the above communication scheme, there are six types of messages where each of them incorporates specific information. However, it has a modular design that makes it easy to integrate a new type of message. There is full-duplex communication between two devices. Thus, each message can have both of the devices as sender or recipient. The following Table 3.8, describes the encapsulated information in the Data and Config message types, respectively.

Msg Type	Encapsulated Info	Description
Data	Vbat	The voltage level of the battery
	AVG_Curr	The average current consumption of the entire node
Config	Sampling timestamps	The list of timestamps that Raspberry Pi should take a picture
	Update timestamps	The list of timestamps that Raspberry Pi should check the Broker
	Sampling iteration	The number of images it collects

Table 3.8: Encapsulated Information

The parsing procedure exists on both devices (Raspberry Pi, Waveshare) and used to extract the information from the messages. First, on the sender side, the messages converted according to the Table to hex to pass it over the UART. After that, the receiver read the messages (Hex buffer) by the following method:

- Read the first byte that identify the message type.
- Read the following byte that identify if the message contain encapsulated information.
- If there is encapsulation information, read each value.
- The procedure stop when read the end message mark (Delimiter 0x7E 0xFF).

3.3 Remote Configuration Protocol

The Remote Configuration Protocol has implemented to achieve the reconfiguration of the edge device of a WSN from a distance. In a real-world deployment, the needs during the time conceivably change. For instance, a node that monitors a specific object during the day could be inconvenient when captures images at the wrong time. To solve this issue with the traditional way you have to get to the node and upload a new code with a property configuration file, changing the times of sampling. This approach may work when you have two or three nodes nearby, but as the scale increases, it becomes ineffective. To address this, we design an asynchronous method to upload new configurations on the edge devices independently the numbers of nodes and the location of them.

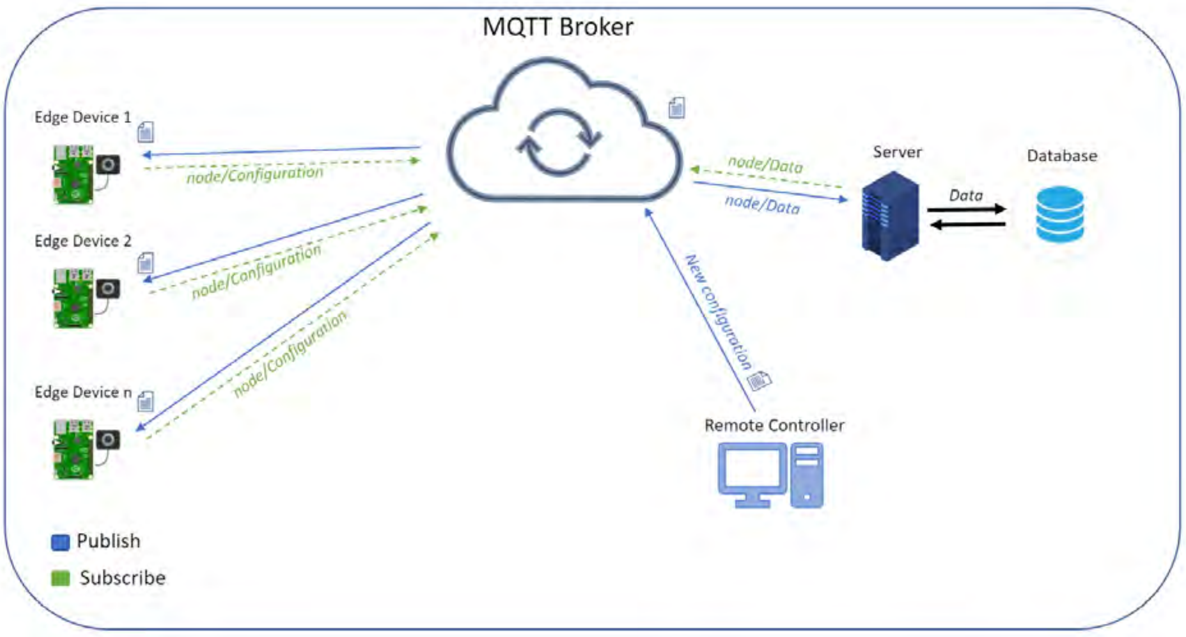


Figure 3.8: Remote Configuration Protocol

To implementing the Remote Configuration Protocol, we utilize a combination of MQTT and UART communication protocols that already described in Chapter 3.2. Specifically, the reprogramming of a node achieved by sending the new configuration over the MQTT to the Broker, there the message will remain until the edge device wakes up and receives the message. As the node consists of two devices (Raspberry Pi and Waveshare), the information that received has to pass equally to both using the parsing procedure described in Chapter 3.2.2. The Figure 3.8 shows how a new configuration sent to the edge Devices.

The available configuration that you can send over this protocol is the following:

- Sampling Timestamps
- Update Timestamps
- Sampling Iteration

As each of these affects the power consumption of the node, experiments have been performed comparing the power consumption in relation to the number of Sampling timestamps, Update timestamps and sampling iterations.

3.4 Object Detection

The code was written in python and was needed to install Tensorflow API to run deep learning objects models. Then, we implemented an algorithm by the OpenCV library to extract feature such as the size of the fruit. The main duty of these algorithms is to detect the desired object in an image and crop it in bounding boxes to reduce the size of bytes so that as few as possible are sent over the network.

We used Tensorflow's Object Detection API to train an object detection classifier for multiple objects. We chose to train our own convolution neural network which it's consist of three kinds of fruits but it could be larger with more classes or smaller with only one class, and it depends on the use of the application. In our system, we need only one class that is peaches but we configured it with three classes to see how robust is the model. At the end of this step, we will create a program that can identify and draw boxes around specific objects in pictures and crop it removing the undeserved information.

We used Tensorflow-GPU version which allows our computer to use the video card that provides extra processing power. Tensorflow-GPU, instead of regular TensorflowCPU, reduces training time in much better performance. Tensorflow provides several object detection models, which is pre-trained classifiers with specific neural network architectures, in its model zoo. The model zoo is a collection of detection models pre-trained on existing datasets, such as COCO dataset, the Kitti dataset, the Open Images dataset, the iNaturalist Species Detection dataset etc. These models can be useful for out of the box inference if someone interests for categories in those datasets. Also, datasets can be used for initializing a model when training a novel dataset. The categories in these models are not satisfied with our goals and we created a new customized dataset with the new class of peaches.

There are models, such as the SSD-MobileNet model, which have an architecture that allows faster detection but with less accuracy, while on the other hand, there are some models, such as the Faster R-CNN model, which gives slower detection but more accuracy. In our system, we chose to re-train a detector with Faster R-CNN Inception V2 model because it worked considerably better and we did not need speed in outputs results or real-time efficiency. The goal is to take results and update the platform in the same periods of times and this assumption satisfies the needs of our system. To sum up, the selection of model deals with criteria such as speed, accuracy and the computational power of the device

3.4.1 Create our own Dataset

As it was referred previously, we decided to create our own dataset and train a new classifier. The first step was to gather and label pictures. Tensorflow needs hundreds of images to train a good detection classifier. The training of a robust classifier was to gather images in variation backgrounds scenes and lighting conditions. For better results were incorporated in classifier some images which the desired object is partially obscured and overlapped with something else, or was only halfway in the picture.

For our classifier, we used different search engines to download a variety of images for fruits, such as Google and Flickr. One other option was to use a camera and capture a lot of images with different backgrounds and conditions of lightning. Finally, we chose the search engines because of the large variety. Then, we categorized images depending on the size because we wanted those that size was approximately 200KB each and their resolution was not more 1280x720. Larger images are more complicated and it takes a longer time to train the classifier. After all, we separated images in two directories, the test and train directory that the test had 20% of images and the other had 80%.

With all the pictures in our collection, we needed to label the desired objects in every picture. We used LabelImg tool to do that but there are a lot of labelling tools to do this job with different front-end environment and different output format files. The LabelImg is an easy to use graphical image annotation tool for labelling images with analytical instructions. It is written in python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format used by ImageNet. With the images labelled, the next step was to generate data that served as input data to the Tensorflow training model. At this point was generated TFRecords a special file format of Tensorflow.

When training begins each step of it reports the loss. It will start high and get lower and lower as training progresses. For the training in Faster R-CNN Inception V2 model, it started at about 3.0 and quickly dropped below to 0.8. Good results are achieved when model train until loss consistently drops below 0.05, which it will take about 40000 steps or two or three hours and depending on how powerful is GPU. The loss numbers will be different if use another model.

3.5 The Edge Device

In this section described the overall functionality of the device, how the individual components combined to create an energy-autonomous node that monitors a specific fruit on the tree. This edge device as shown in the Figure 3.9 is equipped with LTE interface to send the collected data, power management system that powers the node through rechargeable batteries and collects energy by the solar panels, single-board computer that captures images and executes ANN to extract the appropriate information and microcontroller board that manages the duty cycles of the node in order to be more robust in terms of the power consumption.

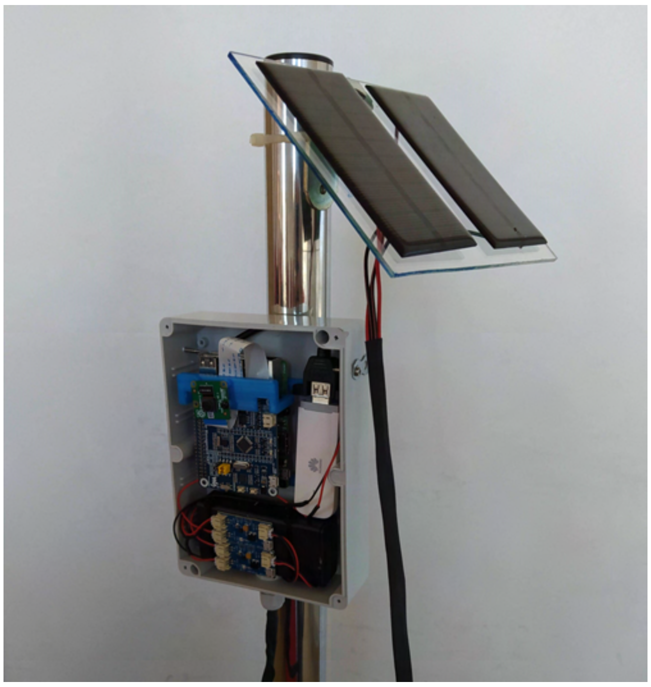


Figure 3.9: The Edge Device

The operation of the edge device shown in Figure. As mentioned above this node consists of two devices the single-board computer Raspberry Pi and the microcontroller board Waveshare Power Management Hat. The Raspberry Pi in our approach performs predefined actions assigned to it by the microcontroller. For this reason, we consider it as an external platform of the microcontroller with specific operations. Particularly, the Waveshare microcontroller activates the Raspberry Pi at predetermined times, when the Raspberry Pi wakes up properly send a message to Waveshare that it is ready for a task, then the Waveshare response with a message (Sampling or Update). If Raspberry takes the Sampling task, it will take a new photo and send it to the server, otherwise, if it receives the Update it will check the Broker for any restructuring action. When the Raspberry Pi finished the task assigned to it, it will send the message to the Waveshare to cut off the power supply, then the Waveshare it will be in the sleep mode and it will start the same process in the next predefined cycle.

Chapter 4

Experiments Results

In this chapter, we evaluate the performance of the proposed edge device in terms of the power consumption and transmitted bytes over the LTE.We compared two scenarios, the performance of the edge device with the Object Detection algorithm and without it. In addition, a study on the rate of power charging is presented, using the results of the experiment.

4.1 Experiments

4.1.1 Experiments Settings

The experiments performed on the edge device were executed in the laboratory and aimed at simulating its operation when it will deploy in the field. For this reason, we performed four experiments. For this reason, we performed four experiments. In the first, we performed 10 image sampling where the Object Detection algorithm was activated and there was the desired object in the field of view of the camera node. The second, executed under the same conditions on the node, however, does not have the desired object. In the third, we performed again 10 image samplings of the desired object but this time on the node does not execute the Object Detection algorithm. Finally, we performed 10 samplings of images without the Object Detection Algorithm on the node and without the desired object in the field of view of the camera. From each experiment, we get metrics such as the average current consumption, the transmitted bytes, the execution time of the node.

Experiments	Sampling	Object Detection Algorithm	Desired object
Experiment 1	10	Yes	Yes
Experiment 2	10	Yes	No
Experiment 3	10	No	Yes
Experiment 4	10	No	No

Table 4.1: Experiments Settings

4.2 Results of the experiments

Power consumption: The power consumption of the node consists of two parameters, the average current where the node consumes and the amount of time it remains active. In the Figure 4.1 shown the needs of the current of the node with the object detection algorithm and without it, while in Figure 4.2 shows the amount of time that the node remains active for sampling and sending the image. Both of them has the desired object in the field of view of the camera.

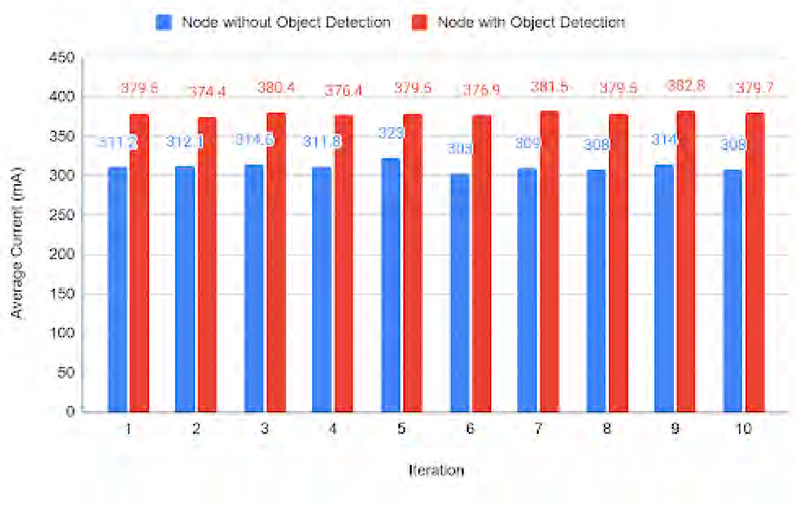


Figure 4.1: Average current of the node.

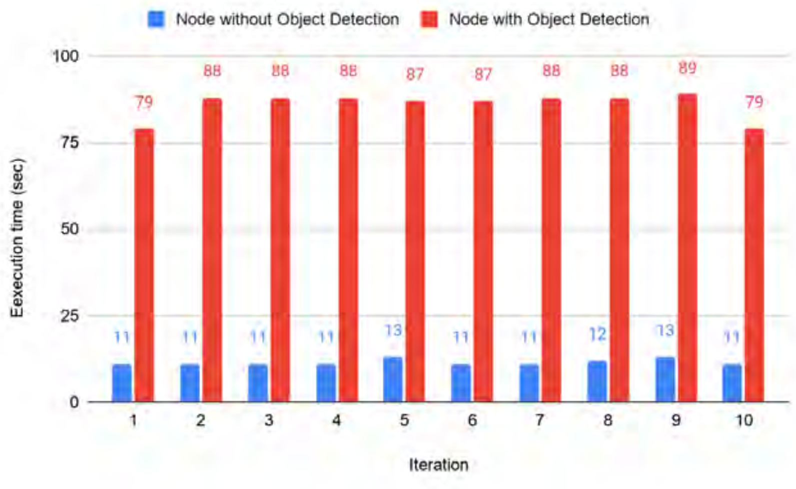


Figure 4.2: The active remain time of the node.

We observed that there is a small increase in the average current consumption when the Object Detection Algorithm is active, from 307 mA the average current consumption is rising at 380 mA, that makes sense because the ANN consumes more CPU resources and MEM that have an impact in power consumption. A larger increase is observed in the execution time between two nodes. When the node simply sends the image it records via the LTE interface it remains active for 10 seconds otherwise when it executes the Object Detection Algorithm, it remains active for 89 seconds.

Based on the above results, we can compare these two approaches in terms of energy consumption, calculating the mAh of each operating cycle. For the node with the Object Detection algorithm the energy consumption calculating by the following expression

$$1mAh = 1mA * 3600sec$$

$$\frac{380mA * 81sec}{3600} = 8.5mAh$$

While for the node without the Object Detection algorithm the energy consumption is

$$1mAh = 1mA * 3600sec$$
$$\frac{309mA * 10sec}{3600} = 0.85mAh$$

According to the above calculations, the integration of the Object Detection algorithm makes the node 10 times more expensive in terms of energy consumption. This is an extremely high consumption that does not suit for applications where they have high sampling repetitions.

Data consumption: These experiments were intended to visualize the data used by the edge device when the object detection algorithm is built-in and when it is not. However, one parameter to consider is that if the desired object is in the camera field of view. For this reason, we performed four experiments as shown in the Table 4.1.

In Figure 4.3 illustrate the data consumption of the edge device when the object existed. The object was from 15 to 50 cm away from the node when experiments performed. It is obvious that we achieve significant reduction in the data we send when the Object Detection Algorithm built-in the node. The reason for this reduction is that the node perceives the position of the desired object in the image and cuts only the region of interest, unlike the node which does not have the algorithm where it just sends the whole image.

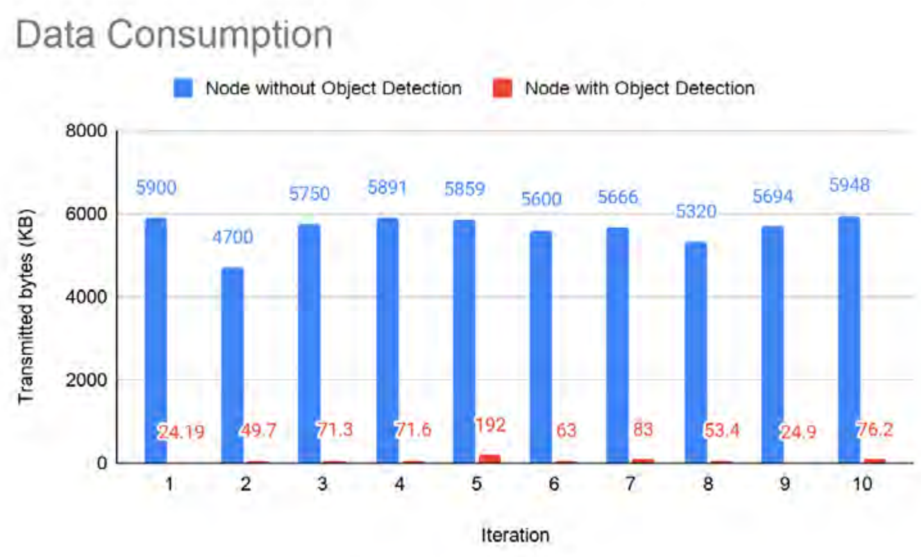


Figure 4.3: Data consumption when the object existed

A stronger reduction was observed when the desired object didn't exist in the camera field of view. These results presented in the Figure 4.4. This time, the node with the built-in Object Detection Algorithm realizes that there is no area of interest for a particular image and only sends a specific message as opposed to the node that does not have the algorithm that continues to send the entire image, even if the desired object does not exist.

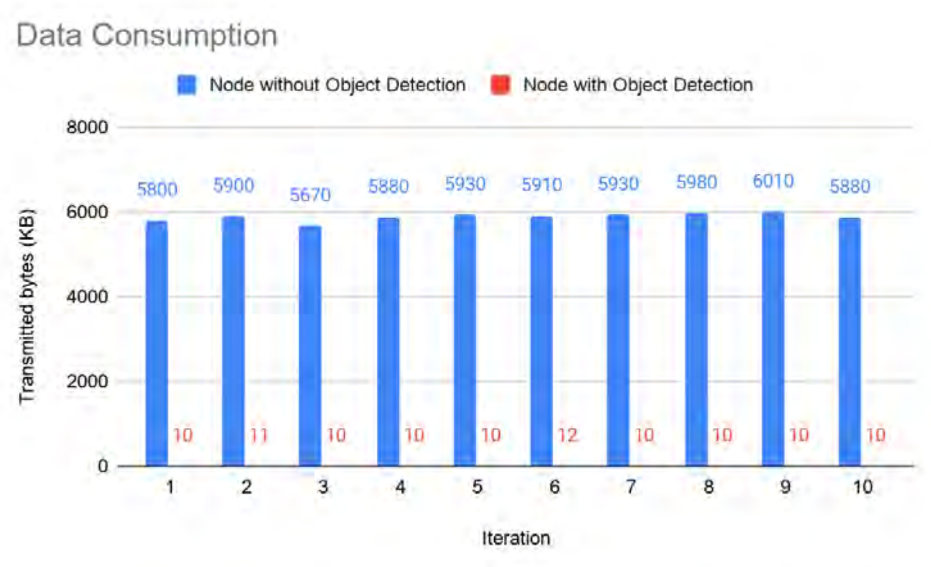


Figure 4.4: Data consumption when the object didn't exist

According to the above results, we achieve a reduction of 98.7% of the data when the object exists and 99.7% when the object does not exist in the field of view of the camera.

Bibliography

[1] *Ericsson mobility report*, Nov 2017.

[2] J. L. Gursel Serpen* and L. Liu, *Ai-wsn: Adaptive and intelligent wireless sensor network*, in Conference Organized by Missouri University of Science and Technology, 2013.