Università degli Studi di Padova

Department of Mathematics "Tullio Levi Civita"

Master Thesis in Computer Science

Open LoRa mesh network for IoT-based air quality sensing

Supervisor
Claudio Enrico Palazzi

Candidate
Voinea Stefan Ciprian

Student ID
1237294

Academic Year 2020 - 2021

Open LoRa mesh network for IoT-based air quality sensing
Master Thesis, Università degli Studi di Padova, December 2021

Author

Voinea Stefan Ciprian — Università degli Studi di Padova
Study programme: — Master's Degree in Computer Science
E-mail: — stefanciprian.voinea@studenti.unipd.it
E-mail: — ciprian.voinea@outlook.com

Supervising professor

Claudio Enrico Palazzi — Assoc. Prof. in Computer Science, Università degli Studi di Padova
E-mail: — cpalazzi@math.unipd.it
Personal webpage: — https://www.math.unipd.it/~cpalazzi/

cosmopolitan adjective

cos·mo·pol·i·tan | \ ˌkäz-mə-ˈpä-lə-tən \
From the Greek **kosmopolitēs** ('*citizen of the world*')

Essential Meaning of *cosmopolitan*

1 : having worldwide scope or bearing

2 : showing an interest in different cultures, ideas, etc.

3 : composed of persons, constituents, or elements from all or many parts of the world

4 : someone who has experience of many different parts of the world

5 : informally a ***cosmo***, is a cocktail made with vodka, triple sec, cranberry juice, and freshly squeezed or sweetened lime juice

Abstract

In the last two decades, technology has evolved exponentially, confirming the validity of Moore's law and allowing devices to become smaller and smaller, to the point where they do not only fit in a pocket but can be placed seemingly everywhere. Such devices, which are mostly composed of sensors, have the purpose of interacting and gathering data from anything they are connected to, thus rendering such things "*smart*".

These devices compose the Internet of Things, or IoT: a new paradigm of digital devices that make up about 21.5 billion interconnected devices in the world, at the time of writing. Many of the publications regarding IoT have been studying the development of low-cost devices, focused on analyzing the air quality surroundings of the individual.

This thesis expands the architecture for connecting such devices and describes a mesh low-power wide area network, Open LoRa Mesh, created to accommodate low-powered IoT embedded devices that communicate via LoRa technology. Such network consists of FiPy boards, by Pycom, that talk with MegaSense, a personal air quality device developed by the University of Helsinki.

Sommario

Negli ultimi vent'anni, la tecnologia si è evoluta esponenzialmente, confermando la validità della Legge di Moore e facendo sì che i dispositivi diventassero sempre più piccoli, al punto da non solo poter essere messi in tasca, ma anche essere posizionati in ogni luogo. Tali dispositivi, principalmente sensori, hanno lo scopo di interagire e raccogliere dati da qualsiasi cosa a cui sono connessi, rendendo dunque tali oggetti "*intelligenti*".

Questi dispositivi compongono l'Internet delle cose, o in inglese Internet of Things (IoT): un nuovo paradigma dei dispositivi digitali che costituisce circa 21.5 miliardi di dispositivi connessi nel mondo, al momento della scrittura. Molte delle pubblicazioni scientifiche a proposito dell'IoT studiano lo sviluppo di dispositivi a basso costo, con lo scopo di analizzare la qualità dell'aria che circonda gli individui. Questa tesi tratta di un'architettura per l'interconnessione di tali dispositivi e descrive una rete, Open LoRa Mesh, con topologia mesh di dispositivi a basso costo che comunicano utilizzando la tecnologia LoRa. Tale rete è composta da dispositivi FiPy, prodotti da Pycom, che comunicano con MegaSense, un sensore personale della qualità dell'aria sviluppato all'Università di Helsinki.

Contents

List of figures

List of tables

Ideas are common.
Everybody has ideas.
Ideas are highly, highly overvalued.
Execution is all that matters.

Casey Neistat

1 Introduction

In an interview from 1988 about school education and its relationship with the Internet, Isaac Asimov, one of the 20^{th} century's most prolific science fiction writers, said "*now, with the computer, it's possible to have a one-to-one relationship for the many. Everyone can have a teacher in the form of access to the gathered knowledge of the human species*".

At the time of writing, many things have changed since that interview: people now live in an *hyperconnected world*, where accessing Internet is considered a right, so that everyone has the same learning opportunities.

According to Cisco's 2018 - 2023 Internet Report, "the number of devices connected to IP networks will be more than three times the global population by 2023" [1]. Authors also predict there will be 29.3 billion networked devices by 2023, a major increase from 18.4 billion in 2018. Of al these, 50% is represented by Internet of Things, or IoT, devices.

As stated in one of Forbes's insights, "*IoT is ranked as the most important technology initiative by senior executives; more important than artificial intelligence and robotics, among many others*", while, from an economical point of view, "*of all emerging technologies, the Internet of Things (IoT) is projected to have the greatest impact on the global economy*" [2]. The number of devices that are being connected to the Internet is growing day by day and the industry is at its highest peeks.

This new paradigm of computer science can be considered an enabler for the sciences that need large amount of data for creating algorithms and offering better services and more well tailored products. The ubiquity of mobile technology has opened the door to a new era of

mobile sensing.

Using the data gathered until the second half of the 20^{th} century, computer simulations showed how the rise in CO_2 levels that can lead to climate change and cause global temperatures to rise steadily in the years. Scientists started to take a more common approach to pollutants and climate change by developing instruments capable of more accurate readings and by better understanding how to analyze the gathered data. In the preface of 1981's "*The Design of Air Quality Monitoring Networks*", the author states that "*the number of publications in the environmental area is increasing exponentially*" [3]. This leaves one to think about the state of this research area up to date.

Many of the publications regarding IoT have been studying the development of low-cost devices, focused on analyzing the air quality surroundings of the individual.

This thesis takes a focus on a particular project, MegaSense, a personal air quality device developed by the University of Helsinki.

In this thesis, the architecture for connecting such devices is expanded with the use of a LoRa mesh network.

1.1 Contributions

The main contribution made in this thesis is the creation of a mesh network, *Open LoRa Mesh*, composed by microcontrollers which communicates via LoRa technology. This mesh has been designed to accommodate the message exchange between devices such as air quality sensors, particularly MegaSense, an air quality sensing device from the University of Helsinki.

Air quality sensing, as previously mentioned, is one of the applications of IoT, and this mesh has been designed keeping in mind the communication needed in between these set of devices.

1.2 Document outline

This document follows an hourglass structure by first explaining in general the necessary notions to understand the work done and then going more in detail. The content is organized as such in the following chapters:

1. *Introduction*: current introductory chapter;
2. *Background*: introduces concepts necessary to understand the project and why it has been developed;

3. *Technologies*: explains underlying technologies, starting from the general definition of a network, common network architectures, radio technologies and micro-controllers;
4. *Related work*: described related projects from which project of this thesis has drawn inspiration;
5. *Proposed solution*: describes in depth the mesh network developed;
6. *Experimental results*: details the experiments and the results made to test the mesh;
7. *Conclusions*: concluding chapter which revisits the concepts of this project and reflects on the possibilities to ameliorate it.

It is one thing to mortify curiosity,
another to conquer it.

Robert Louis Stevenson, Dr. Jekyll and Mr. Hyde

2 Background

This chapter introduces the background concepts necessary to understand the project presented in this thesis and why it has been developed.

It first explains the definition of IoT, afterwards, it describes the air pollution problem, to conclude with an overview on the background work and state or art of two devices capable of detecting pollutants in the air.

Mesh networks are better described in Chapter 4, where also previously made projects which use a similar architecture are described.

2.1 Internet of Things and other networking paradigms

Internet of Things, also abbreviated with IoT, has a longer history than many people think about: its name, now know all around the globe, has been attributed to Kevin Ashton, who used it in a presentation about Radio Frequency Identification (RFID) technology, at *Protector & Gamble*, in 1999 [4] to describe the network connecting objects in the physical world to the Internet.

This constantly expanding branch of Computer Science aims to turn physical objects, as small as they may be, into nodes of an interconnected system which opens the door to new interfaces between humans and machines and how these see the physical world. IoT's importance heavily relies on data gathered from these devices, since, in combination with other computer science paradigms such as Machine Learning and Artificial Intelligence, raw data

can be transformed into valuable information.

The creation of models from all these inputs has given a more efficient workflow in companies and has improved certain aspects of everyday life, from wearable technologies to Inter-Vehicular Communication (IVC). As explained in [5], the latter is a "*key technology in order to increase driving efficiency and safety, as well as support autonomous driving and provide many other connectivity-based services*". The biggest challenges though are *heterogeneity* of devices and *dynamicity* in vehicular scenarios.

Another important application scenario of IoT is emergency situations, where it supports first responders with adequate means to perform their operations in a safe and effective way. In this case, from a networking point of view, "*one of the main challenges is that of providing first responders with multimedia information about the emergency as soon as possible, even from a remote location*" [6]. Projects designed to provide connectivity off-grid not only to civilians, but firefighters, military operations, local law enforcement, search and rescue, are described in Section 4.3.

But first, below are two important key projects in the story of IoT the UPC, or Universal Product Code, and the Carnegie Mellon University (CMU) coke machine.

2.1.1 Universal Product Code and Barcode

One of the first technologies that can be considered part of the IoT family, is the "*Universal Product Code*", or *UPC*. Its first iteration is detailed in the patent issued to inventors Joseph Woodland and Bernard Silver on October 7, 1952, and can be described as a "*bull's eye*" symbol, made up of a series of concentric circles [7], as can be seen in Figure 2.1.

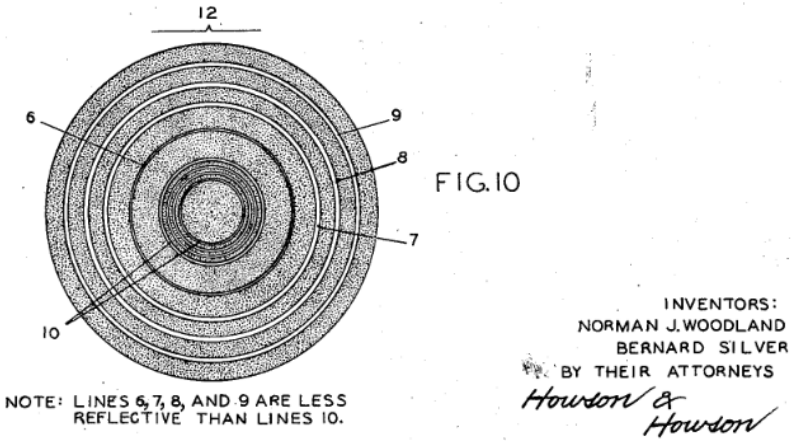


Figure 2.1: Diagrammatic view of the Universal Product Code [7]

Authors of the patent, state that "*one application of the invention is in the so called 'super-market' field*", indicating they already successfully identified a need to speed up and automatizing the process of paying at super-markets.

Due to the large size and low reliability of the equipment necessary to read the figure, this concept has not been immediately released for everyday use. Commercial adoption relied on the emergence of laser optics, which started to offer a more compact reading technology.

Although, printers were vulnerable to smudge the design coped with errors as ink bleeding would result in taller bars. Thus, the ameliorated UPC print was vertical and came in the form of a barcode printed with strict rules in order to avoid errors in scanning due to smear or marks on the label. This version, developed in 1971 by George Laurer at IBM [8], became the first wide appearance of the upc.

A centralizes super-system that provided the code translations was used in order to standardize codes among different super markets and shopping places. Such system has been the first way to track products and address them at large scale, thus rendering a "*thing*" at the super market capable of providing information to a larger scale, although not directly connected.

2.1.2 CMU's coke machine and modern vending machines

It may come as a surprise, but connecting everyday "*things*" for direct interaction, contrary to the previous example, started around the 1980s.

Figure 2.2: CMU's "*coke machine*"

One of the most famous and most quoted as the first IoT device, is the Carnegie Mellon University (CMU) coke machine at the Computer Science Department. Communication from and to the machine, which allowed remote access, took place via Arpanet at CMU as the system predated the Internet. Various sensors were used to detect whether shelves were empty and to track status of coke bottles (warm, cold, empty).

As explained in the official website[I] dedicated to this device by the University, there are

[I] www.cs.cmu.edu/~coke/history_long.txt

"*micro-switches in the Coke machine to sense how many bottles were present in each of its six columns of bottles*".

Modern day vending machines usually require continuous connectivity to the manufacturer's systems. This is not always achievable via a Wi-Fi connection where machines are placed, so other solutions, such as cellular connectivity, are used. Connection reliability in vending machines and other kiosks is important since these provide goods that can be payed by credit card, which need to establish a secure connection.

They contain multiple small, but complex, systems that interact with each other, thus it is implied that this kind of machines must have installed a secure software and that they need to be as hard as possible to be tampered with, either by brute force or by software bugs.

Otherwise it is not only possible that someone steals a snack, but some remote script may turn these machines into a botnet capable of bringing down the connectivity of an entire campus. Such attack has been described in Verizon's "*Data Breach Digest*" risk report from 2017, where the author states that "*the firewall analysis identified over 5,000 discrete systems making hundreds of DNS lookups every 15 minutes*" [9].

While credit card skimmers and chip card cloners remain viable risks to the end consumer, security measures to the environment where machines are placed must not remain an afterthought, especially when these are placed alongside other connected devices and not in their own separated network. Such kind of smart vending machines have helped bring a step closer old "*un-connected*" cities to become "*smart cities*", where these machines can be used as a mean to place devices such as routers or public Wi-Fi access points.

2.1.3 Trends, forecasts and research directions

IoT and related technologies have grown exponentially since the times of CMU's coke machine. According to data from Microsoft Academic[2], publications about "*Internet of Things*" have grown exponentially: from the 26 in the year 2000, to 534 in 2010, 4959 in 2015, to 22454 papers published in 2020. This shows how much interest IoT has gathered among the scientific community. Nonetheless, some IoT systems can still be considered not ready for mass deployment and many technical difficulties and problems need to be solved.

Research directions in this new area are vast, since every physical device now represents a possible "*thing*" connected in the network and that can be interacted with and provide data. Authors of [10] have highlighted ten particular topic areas that span across three layers of IoT

[2] www.academic.microsoft.com/topic/81860439

architecture: Application, Data and Physical, as represented in Figure 2.3.

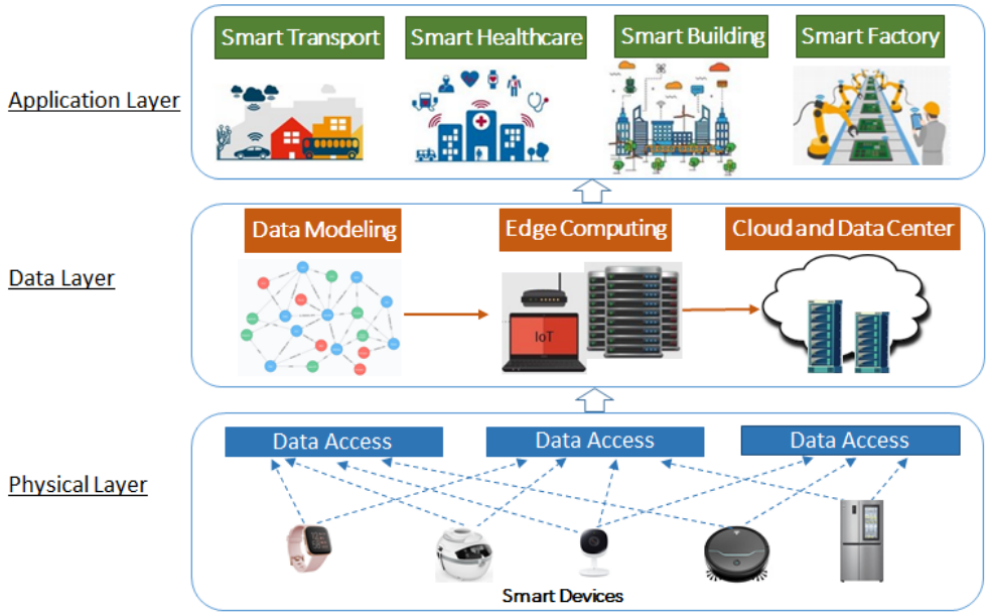


Figure 2.3: Most researched areas in IoT [10]

These topics include "*Data-driven IoT*", "*Security, Privacy, and Trust in IoT*", "*Social IoT*", and "*Edge Computing and IoT*", which have brought the need for new paradigms of computation.

Data can be created and collected at a very high speed when considering the number of devices connected. This has been stimulating the creation of faster and more reliable database management systems (DBMSs) and brokers that allow higher processing speeds and querying frequencies. Specialized versions of these are emerging, each fitted for different scenarios, that may range from a fully online (or as a service with products such as AWS IoT Core[3]) infrastructure to fully on premise one.

Another important aspect is the network's architecture, which needs to take in consideration aspects such as heterogeneity of connected devices, velocity of data that flows across and scalability. Thus, paradigms like Cloud Computing, Fog Computing and Edge Computing have emerged.

In Edge Computing, data points are used where they are produced, without traversing the network unnecessarily. This way, cloud infrastructure requirements are drastically reduced

[3] www.aws.amazon.com/iot-core/features

in three ways: firstly, less network traffic, secondly, less central storage and thirdly less computational power. Rather, Edge Computing makes use of all the capable hardware already deployed, e.g., in a smart home where all the data could stay within the house and be used on site.

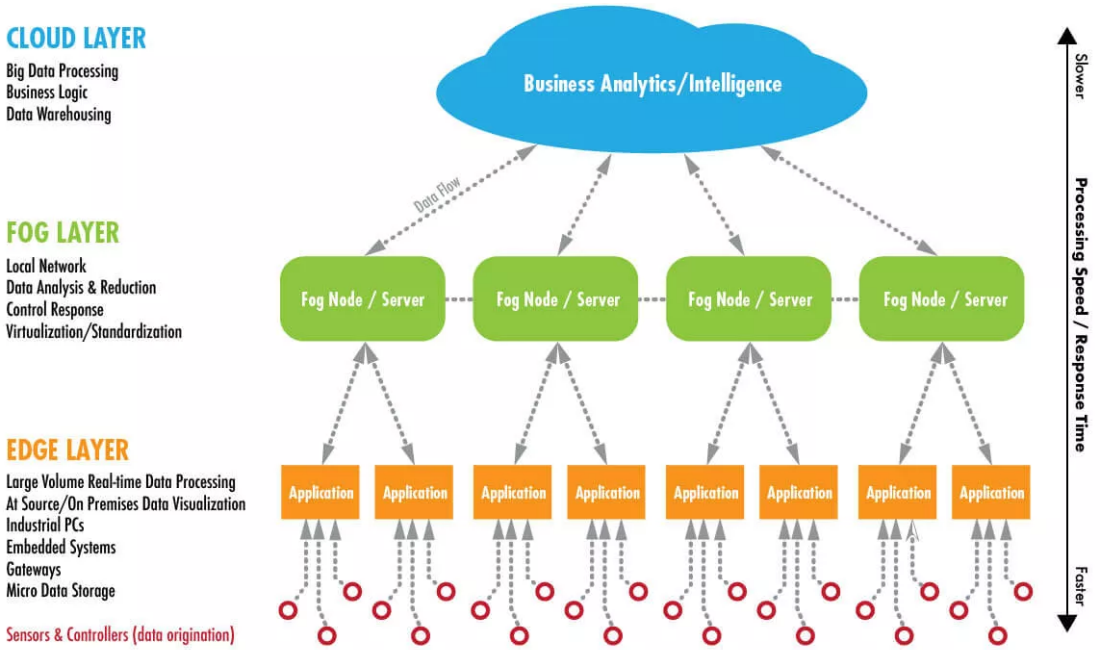


Figure 2.4: Edge, Fog and Cloud Computing

Each of these paradigms places computation on a different network layer, from Cloud Computing which lifts off all the need for end devices to compute data, to Edge Computing, where there might be specialized servers physically placed in strategic points so that they are closer to the end devices (lower latency), which may even have the ability to compute data by themselves. On the other hand, Fog Computing is less aggressive than Edge Computing, and does not require the same amount of services placed near the clients, but they can be sorted among the backbone of the network.

These communications do not take place only via Wi-Fi or Ethernet: given that "*things*" can be everywhere, the need for a network that can adapt to a fast paced environment is becoming a must. Here is where the 5^{th} generation of cellular connectivity comes into play.

As described in a 2019 whitepaper by the GSM Association on IoT and the use of 5G, a "*combination of 5G and wireless edge technologies will support demanding use cases, such as autonomous driving, time-critical industrial IoT manufacturing processes and augmented*

and virtual reality (AR/VR)" [11]. Compared to what is possible with other transmission technologies, 5G supports a massive number in connections, with very little latency.

All of this is not only interesting from a research point of view, but also from a market point of view, where new devices, for consumer and industrial purposes, are created to suit every possible need, that is why IoT can be considered as the "*next chapter of digital communication*".

The most notable example from a consumer's point of view, is the smartwatch, which started with the infamous Pebble watch[4], and is now considered almost a "*must-have*" extension of the smartphone. Not only it smartwatches can be used for recreational purposes, but some models are crossing the line to becoming medical devices, given the improving accuracy with which they record data. Data that, in conjunction with AI, can be used to predict heart attacks [12] or other diseases, like Hyperkalemia [13]. At the time of writing, IoT devices and frameworks can be used for contact tracing in order to prevent the spread of Covid-19 [14].

Consumers want remote control of simple household devices such as coffee pots so that they may wake up to freshly brewed coffee, or schedule it to be prepared at a given time.

On the other hand, from an industrial point of view, demands are more focused on fast connectivity among devices, "*Machine to Machine*" (M2M), to constantly improve production of goods and services in Industry 4.0 and with the use of Industrial IoT (IIoT). The more data available, the more there are opportunities for science, services, business etc. to understand and improve and offer better services and more well tailored products. The growing popularity of IoT use cases in domains that rely on connectivity spanning large areas and the ability to handle a massive number of connections is driving the demand for Low-Power Wide-Area-Network (LPWAN) access technologies, that is why the goal of fifth-generation (5G) wireless networks and beyond is to realize connecting "*anything, anyone, anytime, anywhere*" [15]. LPWANs are better explained in Section 3.2.

Given the importance of this economic sector, many companies, have analyzed the trends and have been producing forecasts about the growth of IoT. One analysis, made by The Economist's Intelligence Unit, and sponsored by Arm[5], states that "*more than two-thirds of respondents agree that understanding the value of data helps them articulate the business case for IoT investments*" [16]. In the same analysis, IoT has emerged as an enabler for AI, since many companies "*view IoT and AI as two components of an advanced analytics capability*", as previously mentioned.

[4] www.kickstarter.com/profile/getpebble/created

[5] www.arm.com

Underlying hardware challenges such as battery development and energy retention and consumption are among the main research areas that are being investigated, since they represent challenges to the realization of efficient IoT systems.

2.2 Air quality sensing

The ubiquity of mobile technology has opened the door to a new era of mobile sensing. Through this new paradigm that is IoT, physical phenomena can be observed in a distributed way, crowd-sourcing data measurement tasks to smartphones and/or other popular smart wearables. Mobile sensing and wireless communications can hence be employed to gather data and generate new information and services, benefitting society. Two general strategies can be used to deploy an environmental monitoring system: creating a network of *fixed sensors* or resorting to *mobile sensors*.

The presence of tiny particulate matter in air is a crucial factor in health, especially when considering urban scenarios. In this context, smart mobility coupled with low-cost sensors can create a distributed and sustainable platform for social sensing able to provide pervasive data to citizens and public administrations. Sustainable and eco-aware decisions can then be supported by empirical evidence, resulting in an improved life and better city administration.

Some of the most harmful airborne agents are fine particles (PM2.5), sulphur dioxide (SO_2), nitrogen dioxide (NO_2), PM10, carbon monoxide (CO), benzene and ozone. The maximum amount for these particulates is regulated by various local and international authorities, for example the EU has set standards[6] that are reinforced by constant monitoring.

One of the most important instruments in air quality measurement is the *Keeling Curve*: a meticulous record of the amount of carbon dioxide in the atmosphere.

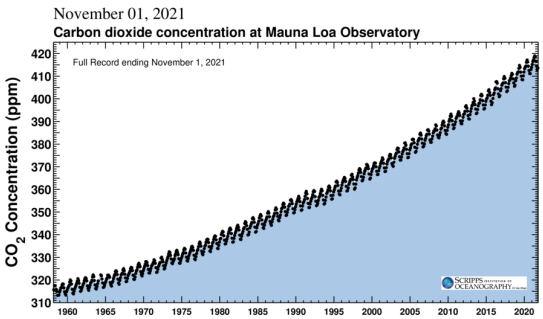


Figure 2.5: Full record Keeling curve until Nov. 1 2021

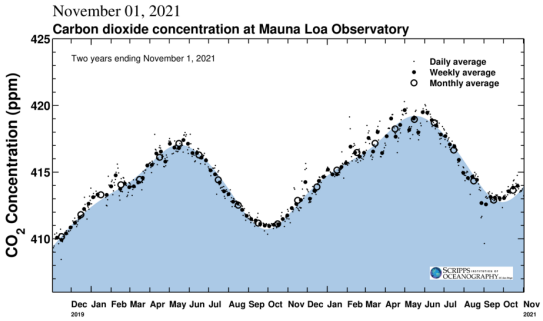


Figure 2.6: Two years Keeling curve until Nov. 2021

[6] www.ec.europa.eu/environment/air/quality/standards.htm

The Keeling Curve tracks changes in the concentration of CO_2 in the Earth's atmosphere using data from a research station on Mauna Loa, Hawaii. Figure 2.5 shows the result of daily readings that have continued almost uninterrupted for more than 60 years, while Figure 2.6 shows the data between October 2019 and November 2021. The importance of this instrument lies in the fact that, over those six decades, the zig-zag has trended steadily upward. Real time readings can be found on the UC San Diego, Institution of Oceanography's dedicated website[7].

Authors of [17] have surveyed the use of machine-learning-based calibration for low-cost air quality sensors. This represents the "*main technique for improving the usefulness of measurements provided by low-cost air quality sensors*". Authors explain how the use of low-cost air quality monitoring technology is complementary to professional-grade air quality stations, which are high cost of professional-grade stations that limit "*limits the granularity at which they can capture pollutant concentrations, whereas low-cost sensors can be deployed densely to increase the spatial granularity of collected information*" [17].

Many projects have been made, both for research and commercial purposes, to detect pollutants in the air and raise awareness of the conditions people are living in. Below are two research solutions that have been made for air quality sensing with low-cost sensors.

2.2.1 ArduECO

ArduECO is a wireless device based on an Arduino-like board, esp8266, capable of gathering data about air quality (and more with simple extensions) and sending them to the cloud, to be processed and displayed.

This solution has been proposed in the paper "*Air Quality Control through Bike Sharing Fleets*"[18], presented at the 2020 IEEE Symposium on Computers and Communications.

One of the main design goals of this device is to be easily fit on shared means of transportation, such as bikes and electric scooters, which been advantaged by the COVID-19 pandemic [19], contrast other shared transportation approaches, such as car sharing, that have severely suffered from the pandemic.

ArduECO is built around four core components: the NodeMCU ESP8266-based development board, a microSD card reader for local data caching, a GPS-based global navigation satellite system receiver for location data and an MQ-7 carbon monoxide (CO) sensor, which can easily be replaced with other sensors using the same pinout. Figure 2.7 shows the full cir-

[7] keelingcurve.ucsd.edu

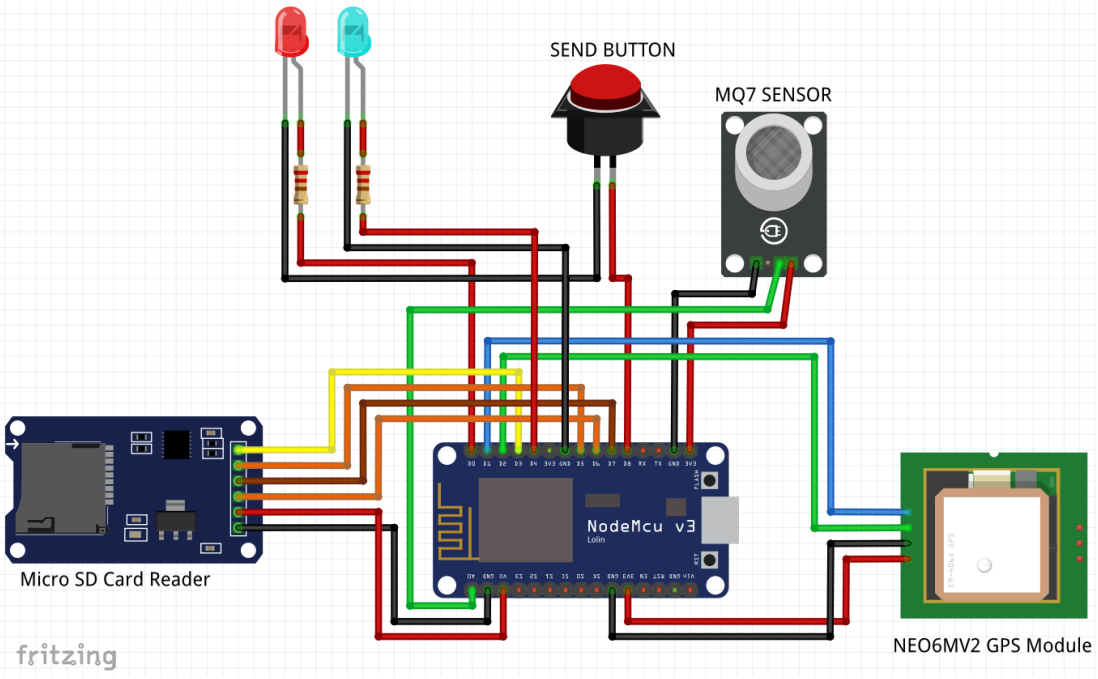
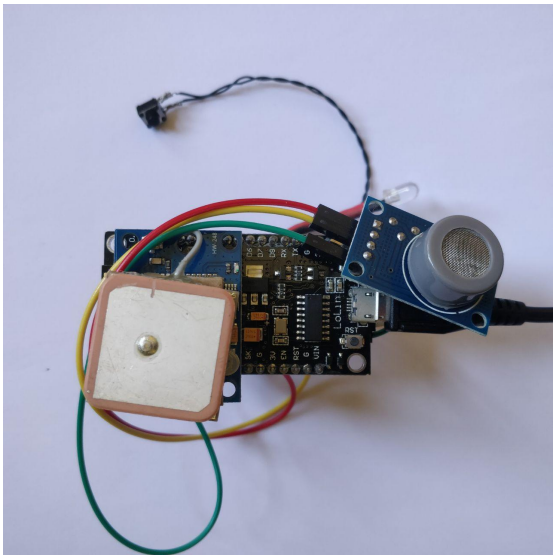


Figure 2.7: Circuit of the ArduECO prototype

cuit of the prototype, while Figure 2.8 is a picture of the built device.

As well as capturing the air quality data locally, data is cached and later sent in the cloud to an MQTT server running on Amazon Web Services IoT Core.



Server costs aside, the sensor platform can be built in a bill of materials of around 15€, or less if the materials are bought in bulk. As described in the paper, this project is more of a proof of concept that shows the possibility of creating a small and cheap device which can be placed on shared means of transportation. Therefore there are various aspects that can be improved, such as adding multiple sensors to detect different pollutants and tweak the software so that the board can sleep and save battery.

Figure 2.8: Built ArduECO prototype

2.2.2 MegaSense

MegaSense is a project that has been developed by the Departments of Computer Science at the University of Helsinki[8] and brings forward accurate portable low-cost sensing devices and an online data platform integrating multiple sources of urban data and leveraging AI network calibration producing hyper-local air quality information in real time.

Compared to the previously described ArduECO, MegaSense, represented in Figure 2.9, is a more complete project which has been tested by citizens and companies in City of Helsinki and EU projects. It is also encased in a 3D printed case and has an rechargeable battery, allowing it to be immediately given to the users.

The back-end system receives data from sensing platforms, the individual MegaSense devices, measuring local pollution exposure and other variables affecting it. These data are processed into air quality information such as maps and advice on how to reduce personal exposure, take healthier routes, and direct participants to improve measurements in areas that have limited sensor coverage.

Figure 2.9: MegaSense prototype

As explained in the main paper [20] regarding this device, "*the core of MegaSense consists of two layers: the Edge and Cloud*". The Edge layer receives data from available node and delivers pollution maps to the mobile app, which provides users with personal air pollution exposure information as well as local exposure maps. This layer is responsible for data pre-processing, filtering and data cleaning. The cloud layer, which runs on AWS, is responsible for storing cleaned data and aggregating the crowd-sourced data while preserving the privacy of participants.

Registered citizens that have been given the device, download the HOPE Exposure App from Google play store and tether their Android smart phone to the sensor. The phone gathers data from the sensor, alongside the GPS coordinates, and it sends them in the cloud.

Calibration of the mid-cost sensor that are contained in the MegaSense is done against a reference station in the Kumpula Campus at the University of Helsinki. Authors "*use a deep learning model consisting of convolutional layers, fully connected neural network layers, and*

[8] www.helsinki.fi/en/computer-science

long short-term memory (LSTM) layers that model temporal dependencies" [21] against the data gathered from the main station.

The exact sensors [20] contained in the MegaSense device are: BME-280 (temperature, humidity, air pressure), battery voltage, Sensirion SPS30 (PM3), SI1133-AA00-GM (UV), MiCS-4514 (CO, NO2), MQ-131 (O3). These sensors have been chosen after an careful analysis contained in [17].

This project develops the clean air journey planner application that creates optimal walking and cycling routes based on air quality of Helsinki.

A better understanding of the architecture of MegaSense is explained in Section 5.1, while the papers in which this project is detailed are [20, 21].

The Internet is becoming the town square
for the global village of tomorrow.

Bill Gates

3 Technologies

This chapter explains more in detail the underlying technologies of this project. Starting from the general definition of a network and the most common architectures, radio technologies and micro-controllers.

3.1 Fundamentals of network communication

> "*A computer network is a structure that makes available to a data processing user at one place some data processing function or service performed at another place.*" [22]

Starting from the definition of a computer network by Paul E. Green, it is easy to understand its importance in today's society. Smartphones, personal computers and other interconnected devices have become omnipresent in modern society, where people have the urge to be connected to each other via these devices. Not only they are used for fun, leisure and other social activities, but they allow connection to services such as online banking, government services and healthcare, which require a stable and secure connection among the systems they use in order to provide a safe experience for their users. All this to say, networks are everywhere underneath today's technology. There are no services or devices that can stand on their own without sharing data to other devices, to synchronize and provide a better user experience, to get updates from the manufacturer or simply to send a keep-alive message.

While this raw data is important for computers, people, the final users, process it to gain information, and this exchange of information from all around the world has brought radical

changes many levels, from a cultural and economical point of view. The possibility of having a network to share information is the next step of globalization, which started with the trade of goods among countries and now brings everyone together, allowing for a cultural exchange that lets people unite across the globe.

This big network that is used to exchange information all around the world has a special name: Internet. Many countries, such as Finland, Spain and Greece, have recognized the importance of this network and have given people the "*right to Internet access*", also known as the *right to broadband* or *freedom to connect*. In these countries, service providers must be able to supply a mandatory minimum connection capability to all desiring home users in the regions of the country they serve.

It is important to note that *Internet*, with a capital *I*, is a particular set of worldwide interconnected networks [23], but a common network of networks is called *internetwork*, shortened by internet, with a lowercase *i*.

Such distinction began in the 1980s and has been described in RFCs[1,2] by computer scientists that understood how ARPANET was expanding and its dimensions were not enough anymore to accommodate the amount of data traveling from one computer to another. At that time, computers such as the *IBM 5150* and the infamous *Commodore 64* were starting to become more and more available, even if highly priced, not only to companies and universities, but also to consumers who brought them in their households, especially with the advent of *MS-DOS*, the dominant operating system throughout the 1980s, now open-source[3].

As described by IBM in one of their technical books from the time, "*it is possible to divide the Internet such as the following groups of networks*" [23]:

- *backbones*: large and strategical data routes among core networks and routers that compose and connect the Internet;
- *regional networks*: that connect large facilities such as universities and colleges;
- *commercial networks*: that provide to their subscribers access to the Internet;
- *local networks*: which run, for example, across a campus university or private property.

Given this increase of computers connecting to the Internet, there came the need for a revised structure that could better organize these components in a more robust, but also flexible, large network. With more accessible operating systems, such as *Windows 95* and

[1] RFC 871 (1982): A PERSPECTIVE ON THE ARPANET REFERENCE MODEL
[2] RFC 872 (1982): TCP-ON-A-LAN
[3] www.github.com/microsoft/MS-DOS

Windows 98, and the advent of Tim Berners-Lee's *World Wide Web* (or *WWW*), computers became a common commodity. The invention of the web and its ease to navigate, using hyperlinks and search engines, culminated in the *dot com* bubble, a stock market bubble in the late 90s that caused rapid rise of technology companies in stock market.

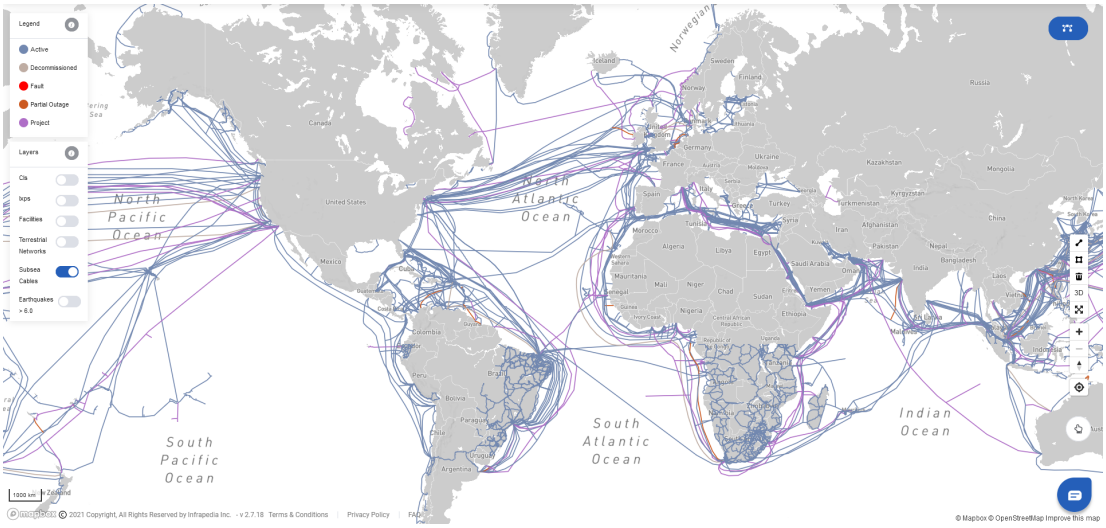


Figure 3.1: Subsea Internet backbone cables between US and Europe[4]

Networks can be categorized based on the area they cover and serve:

- *Wide Area Network*, or *WAN*: also called long haul networks, provide communication over long distances;
- *Metropolitan Area Network*, or *MAN*: provide communication inside a metropolitan area, which could be a single large city, multiple cities, or any given large area with multiple buildings;
- *Local Area Network*, or *LAN*: provide the highest speed connections among computers in a small and circumscribed area;
- *Personal Ara Network*, or *PAN*: connects devices within a user's immediate area.

Another level of distinction among networks, which is made based on the power consumed by the transmission medium, is explained in Section 3.2.

Since now everyone can connect to the Internet and access its services, there is no need for the average user to understand what happens between his machine and the rest of the network, which means he only sees the information displayed, without knowing where it arrives from or what path it took to arrive on his monitor.

[4] https://www.infrapedia.com/app

For computer scientists though, it is important to understand the difference between *network architecture* and *network topology*. A *network architecture*, as described by by Paul E. Green, "*is a complete definition of all the layers necessary to build the network*" [22]. This is focused on the network software, which needs to be highly structured in order to allow for heterogeneous systems to communicate with each other. One example of network architecture is the *ISO/OSI reference model*[5], which is implemented by the *TCP/IP protocol stack*.

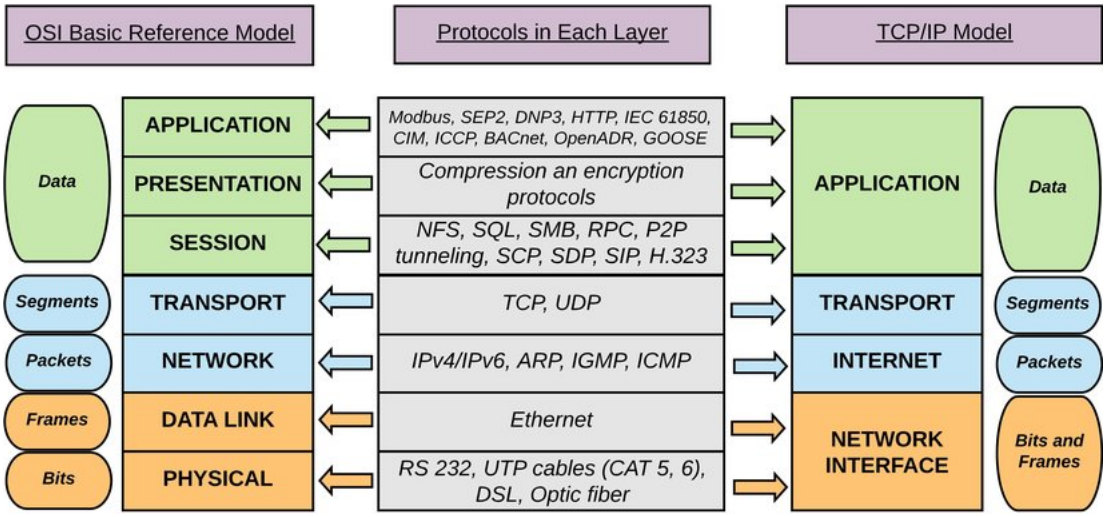


Figure 3.2: The ISO/OSI reference model against the TCP/IP protocol stack

Thus comes the definition of a protocol as "*a set of agreements for interaction of two or more parties and is expressed by three components, syntax (e.g., a set of headers, a set of commands/responses), semantics (the actions and reactions that take place, including the exchange of messages), and timing, the sequencing and concurrency aspects of the protocol*" [22]. Different types of network use distinct architectures, based on the transmission medium and how well this performs (errors, speed, etc.).

On the other hand, the *network topology* refers to the manner in which the links and nodes of a network are arranged to relate to each other.

As shown in Figure 3.3, some of the most common network topologies are:

- *point-to-point*: in which devices are connected directly;
- *bus*: devices are connected to each other via a backbone cable;
- *ring*: two dedicated point-to-point links connect a device to the two devices located on either side of it, creating a ring of devices through which data is forwarded via

[5] www.iso.org/standard/20269.html

repeaters until it reaches the target device;

- *star*: connects each device in the network to a central hub. Devices can only communicate with each other indirectly through the central hub;
- *tree*: parent-child hierarchy in which star networks are interconnected via bus networks;
- *mesh*: a dedicated point-to-point link connects each device on the network to another device on the network, only carrying data between two device;
- *hybrid*: any combination of two or more topologies.

An important factor that decides the industrial acceptance of any network topology is its "*scalability*", concept reiterated in Section 4.2 for mesh networks.

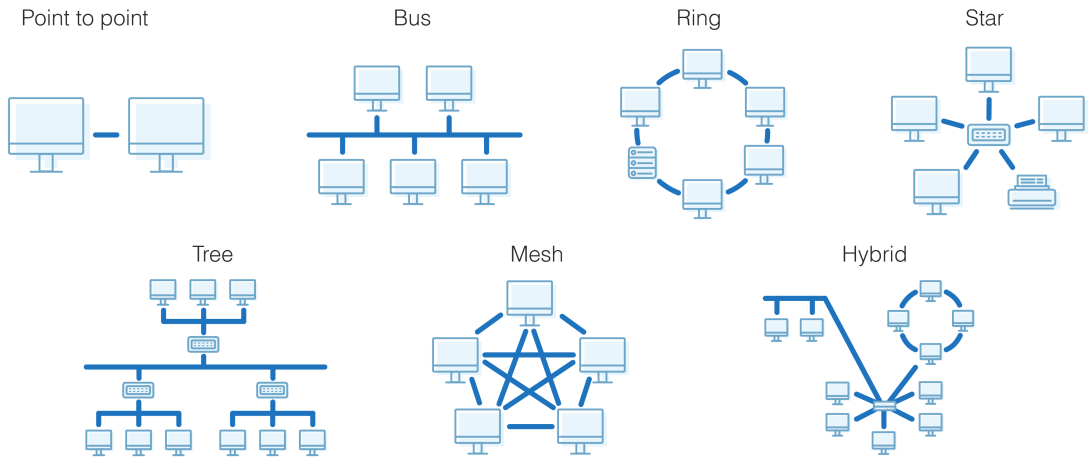


Figure 3.3: Common network topologies

The project presented in this thesis regards a network with a mesh topology, which can be considered a *multi-hop network*, since data passes from node to node until it reaches its destination. This is better described in Chapter 5, alongside the technical details of mesh proposed in this thesis, which has a span of LAN / MAN, since it connects devices that are in a circumscribed area but can be also placed further from each other, in order to cover longer distances.

Another key distinction is the one between *control plane* and *data plane*. The control plane is the part of a network that controls how data packets are forwarded, which means how data is sent from one place to another. The process of creating a routing table, for example, is considered part of the control plane. Data plane, on the other hand, is the part of the software that processes the data requests and handles the data which needs to be forwarded.

The organization responsible for Internet standards is the Internet Engineering Steering Group (IESG)[6]. It is necessary to have an organization looking over the Internet itself since it gives the regulations that allow all devices to interconnect with each other.

3.2 Radio technologies

Although Guglielmo Marconi is usually credited as the inventor of radio due to the creation of the first commercially successful wireless communication system [24], many scientists before him have studied the subject of radio waves. The discovery of electromagnetic waves, including radio waves, by Heinrich Rudolf Hertz in the 1880s, came after theoretical development on the connection between electricity and magnetism that started in the early 1800s. Scientists tried to achieve the idea of a wireless telegraph via electric conduction and electromagnetic induction for a while before the establishment of radio-based communication.

Other important experiments were made by Nikola Tesla, who invented the *Tesla coil*, a device essential to sending and receiving radio waves, during efforts to develop a "wireless" lighting system. This Tesla coil has been used by Marconi in his experiments and is present in the patent he presented for radio transmission of data.

More than a century later, radio technology has massively evolved and is used on a daily basis. Devices has shrunk in dimension and the amount of transmission meanings have increased far from what both Tesla and Marconi could have thought of. Thus, in order to give a complete picture of radio transmitting technologies, it is important to make a distinction among the ones that are made for internal or nearby use compared to the ones which are used for longer distances.

Many users opt for wireless transmission media because it is more convenient than installing cables, even if they might sacrifice some performance. Also, using wireless technology allows transmission in locations where it is impossible to install cables. Types of wireless transmission media used in communications include:

- *infrared*: wireless transmission medium that sends signals using infrared light waves;
- *broadcast radio*: wireless transmission medium that distributes radio signals through the air over long distances such as between cities, regions, and countries and short distances such as within an office or home. Bluetooth, UWB, Wi-Fi, and WiMAX communications technologies use broadcast radio signals;

[6] www.ietf.org/about/groups/iesg

- *cellular radio*: form of broadcast radio that is used widely for mobile communications, specifically wireless modems and cell phones, which use high-frequency radio waves to transmit voice and digital data messages;
- *communications satellite*: space station that receives microwave signals from an earth-based station, amplifies (strengthens) the signals, and broadcasts the signals back over a wide area to any number of earth-based stations. Applications such as air navigation, television and radio broadcasts, weather forecasting, video conferencing, paging, global positioning systems, and Internet connections use communications satellites.

With new transmission technologies, new network architectures and topologies that are better suited for the transmission method have emerged. Topologies that bring computation closer to the edge are also rising in popularity, since they allow for faster computation and they bring data closer to the user. These are Cloud, Edge and Fog Computing, as mentioned in Chapter 2.

LAN, MAN and WAN are not enough anymore to describe the new topologies. An important distinction is now made by other factors such as power consumption, cost of the devices and range of transmitter and receiver. The most important new category of wireless communication, that interests IoT and represents a large portion of the market at the time of writing, as described in Chapter 2, is *LPWAN*, which stands for *Low Power Wide Area Networks* and is composed by nodes that are capable of *long range communication*, *low power consumption* and *low cost*. Lightweight protocols also allow reducing complexity in hardware design and lower device costs. Its long range combined with a star topology reduces expensive infrastructure requirements, and the use of license-free or licensed bands reduces network costs. Growing popularity of IoT use cases in domains that rely on connectivity spanning large areas and the ability to handle a massive number of connections is driving the demand for LPWAN access technologies [25], and the project presented in this thesis falls under this category of communication. LPWAN allows connectivity in many applications, from crowded areas in smart cities to smart farming and smart environment, from security and emergencies to e-health.

Various wireless transmission methods can be used to create a LPWAN, some of the most used in IoT are Sigfox, LoRa and Narrow-Band IoT (NB-IoT). In Section 3.2.1, follows a more complete explanation on LoRa and LoRaWAN. One crucial factor for these transmission methods is to "*support a massive number of simultaneously connected devices with low data rates*" [25].

Other important communication technologies in IoT are RFID, NFC, for contact pur-

poses, Bluetooth, Zigbee for a personal network and IEEE 802.11, or Wi-Fi, for a local area. The latter was used in the ArduEco project, described in Sec. 2.2.1. Fig. 3.4 contains a representation of the geographic coverage in meters of the various aforementioned wireless transmission methods.

3.2.1 LoRa and LoRaWAN

LoRa (short for *long range*) is a spread spectrum modulation technique derived from the *chirp spread spectrum* (*CSS*) technology. LoRa devices provide a long range, low power wireless platform that is being widely adopted for the IoT networks worldwide. In conjunction with the LoRaWAN protocol, LoRa devices enable smart IoT applications in various contexts, like: energy management, natural resource reduction, pollution control, infrastructure efficiency, disaster prevention, and more.

LoRaWAN, on the other hand, is a comprehensive network architecture that details a LPWAN technology. It was designed to interconnect battery operated "things" and provides a bi-directional communication service with end-to-end security features.

The architecture is based on a star-of-stars topology in which specific devices called gateways relay messages between end-devices and a central network server. The wireless communication takes place through a single-hop link between the end-device and one or many gateways. The specification defines the device-to-infrastructure physical layer parameters (LoRa) and the specific LoRaWAN protocol to allow for full interoperability among manufacturers. It defines the technical implementation but it does not define any commercial model or type of deployment, e.g., public, shared, private, enterprise.

The LoRaWAN specification is developed and maintained by the LoRa Alliance, an open association of collaborating members. With LoRa and LoRaWAN the intention was to provide the possibility to deploy Internet of Things applications fast in areas where large distances are involved, yet low bandwidth is needed.

Low bandwidth makes it ideal for practical IoT deployments with less data or where data transmissions are not constant, for example with waste management where the information that is required is whether a waste bin is full or not. Moreover, the *end-to-end* delay (or the *RTT*, *Round-Trip-Time*) can be relatively high, with values below one second being acceptable.

Conversely, packet loss is a more crucial factor. In LoRaWAN, once a message has been delivered, there is no acknowledgement of receipt. However, nodes can request acknowledgements. In this case, if many gateways receive the same packet, the cloud has to choose

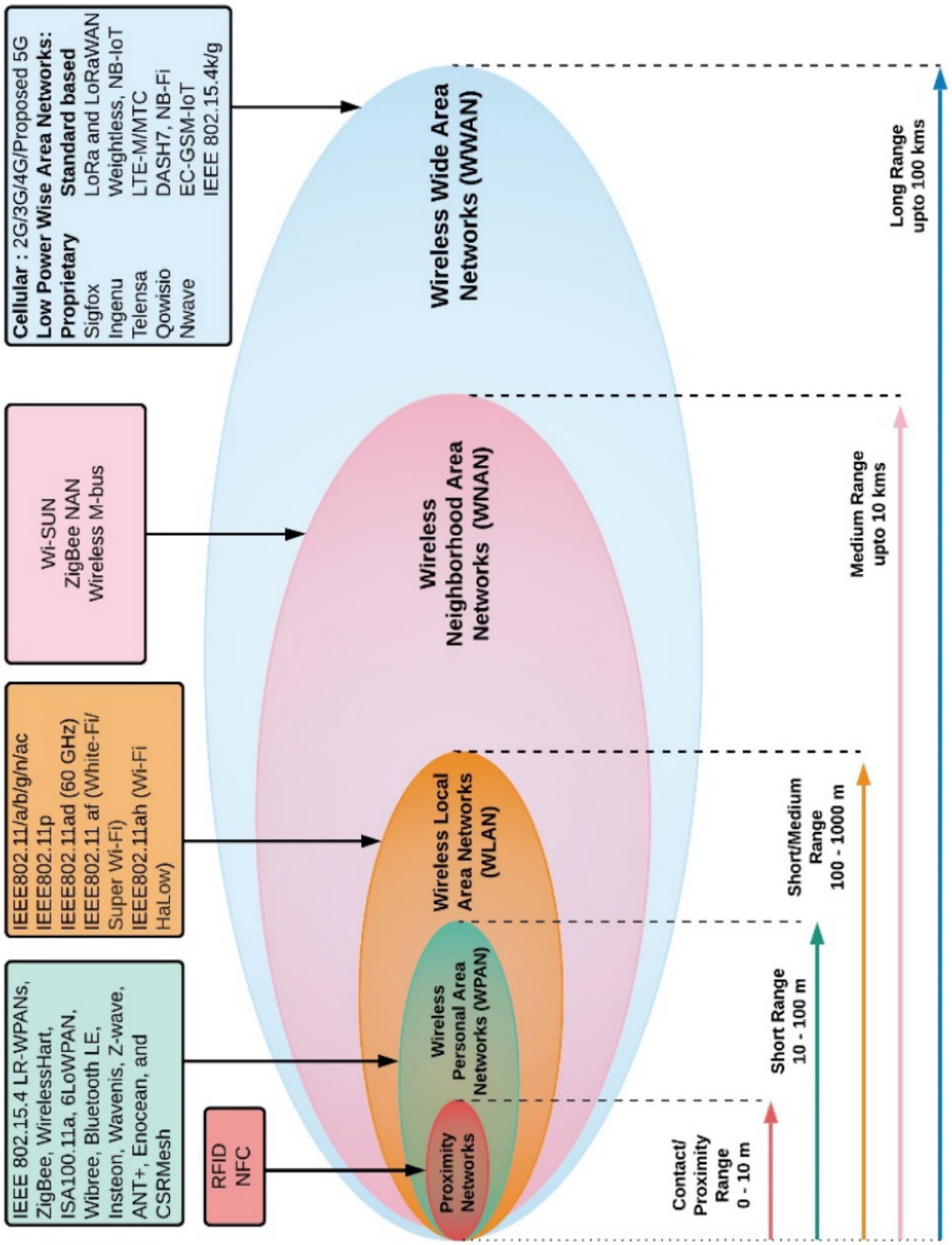


Figure 3.4: Wireless access geographic coverage [25]

one gateway to respond at a fixed time, usually a couple of seconds later. The problem is that when a gateway is transmitting back to the node, it stops listening to everything else. So, if an application needs a lot of acknowledgements, the gateway will very likely spend more time transmitting acknowledgements than listening, which will eventually lead to a network saturation.

Therefore, classical applications do not require acknowledgments and that's why having a reliable network with very small packet loss rate is crucial to not lose relevant information.

Chirp spread spectrum has been used in military and space communication for decades due to the long communication distances that can be achieved and robustness to interference, but LoRa is the first low cost implementation for commercial usage.

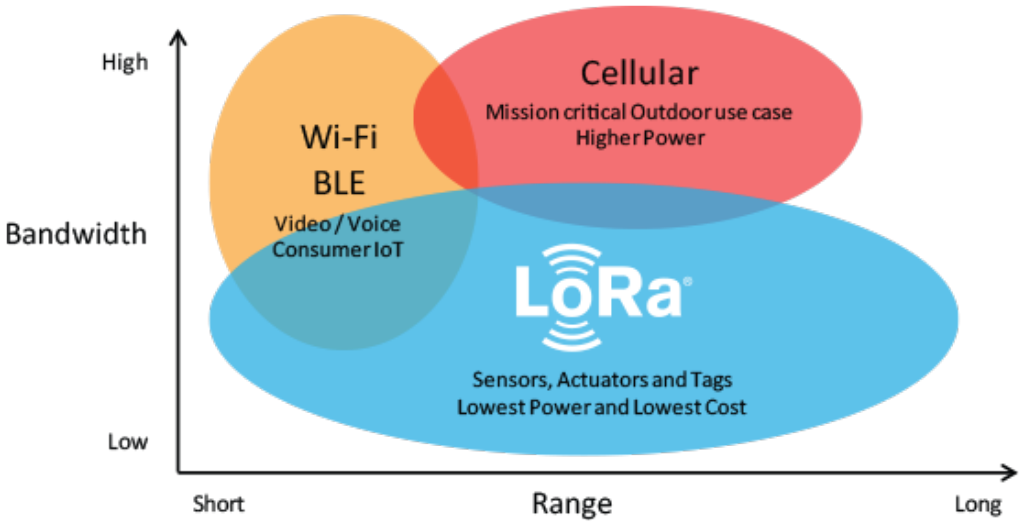


Figure 3.5: Range of LoRa transmission compared to Wi-Fi, BLE and Cellular

The advantage of LoRa is in the technology's long range capability, as can be seen in Figure 3.5 and in Figure 3.4. Range highly depends on the environment or obstructions in a given location, but LoRa and LoRaWAN have a link budget greater than any other standardized communication technology.

Although LoRa shows good performance for long-range transmission in open areas, its signals "*can be attenuated over distance, and buildings, trees, and other radio signal sources may interfere with the signals*" [26].

Microcontrollers (or MCUs, short for Microcontroller Unit) are compact integrated circuits designed to fit in specific environments or to perform specialized functions, which usually do not require any particular computation, memory capacity or power. This has been made possible thanks to the continuous shrinking of transistors, which makes almost all the components more compact, and the improved power sources. In junction with the previously described wireless technologies, microcontrollers are at the hearth of IoT devices, described in Chapter 2, and they require long-lasting, low-cost, and sustainable batteries.

The number of IoT connected devices is expected to grow up to 75 billion worldwide by 2025 [27], and connection density is expected to be one million devices per square Km [28]. Hence, these devices will generate massive data and consume significant energy.

Given the amount of specific functions an MCU can perform, there are many boards on the market. Some are very alike, while others are very different, since they are expected to be used in other types of environments. All boards consist on a similar architecture, which contains the processing unit (CPU), along with memory and programmable input/output (or I/O) peripherals.

Figure 3.6: Attiny 85 on the left, two boards based on the ESP32 in the middle, Asus Tinker Board 2 on the right

In Figure 3.6, there are four different boards, the one farthest on the left is the *ATtiny 85*[7], a low-power, 8-bit microcontroller that is made for general purpose and can be programmed for easy tasks, from simple LEDs flashing, to more "*elaborate*" small sensor projects. The two boards in the middle are based on the ESP32 chip, a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. They both are more powerful compared to the ATtiny 85, and the right one offers an integrated LoRa antenna

[7] www.microchip.com/en-us/product/ATTINY85

on board. Far on the right, there is the Asus Tinker Board 2[8], a board powered by an Arm 6-core system on a chip (SoC), with a 64-bit Armv8 architecture. This board provides much more computing power compared to the previous ones and is able to run operating systems such as Linux and Windows.

One of the strong points of these boards is the price: the ATtiny is priced around 1€ when bought in bulk, the boards on the middle cost around 7€ and 15€ respectively, while the board by Asus is the most expensive of the four and starts from 70€.

It is important to note though that using a generic board in a production environment might not be ideal, since it might lack of customer support and documentation from the company. The boards described subsequently are from three of the major MCU producers, Arduino, Raspberry Pi and Pycom, which have built hardware that is well documented and suited for many different environments, from hobbyists to industrial use.

The simplified architectures and the constraints of embedded devices are reflected in the narrow choice for programming languages. It is hard to find an MCU programmed in Java since this would require the JVM running in the background. Popular languages for MCUs are, for example, C/C++, Assembly, Rust, Ada, Erlang, *etc.* All these have in common the fact that they are compiled and the bytecode has a small footprint. A particular microcontroller company, Arduino, has developed a version of C++ specific for their boards, but given the simplicity of this new dialect, many boards on the market can be programmed with it. Arduino is explained further in Section 3.3.1.

Another programming language that is quickly taking hold at the time of writing, is Micropython. As explained in their website it is an "*efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments*"[9]. Python's fast learning curve and talkative code advantages are reflected in this smaller version, available for most microcontrollers.

3.3.1 Arduino

Arduino is a company founded by Massimo Banzi *et al.* in Ivrea, Italy, in 2005, and has released the first commercially available microcontroller. They wanted a device that was simple, easy to connect to various "*things*" (such as relays, motors, and sensors), and easy to program, besides being inexpensive.

[8] https://tinker-board.asus.com/product/tinker-board-2.html

[9] www.micropython.org

They selected the AVR family of 8-bit microcontroller devices from Atmel and designed a self-contained circuit board with easy-to-use connections, wrote bootloader firmware for the microcontroller, and packaged it all into a simple *Integrated Development Environment* (*IDE*) that uses programs called "*sketches*" and Arduino was the result.

The most famous version of their board is the *UNO* (*one* in English). Arduino UNO, the one on the left in Figure 3.7, is the most used and most documented board of the whole Arduino family, even if this board does not have any integrated sensors or particular ports for peripherals. At the time of writing, the current revision of the board is the Arduino UNO Rev 3[10], which consists of 14 digital pins, 6 analog inputs, a power jack and USB connection.

The Arduino family of products can be programmed in a particular programming language based on C/C++, using a special open-source IDE. Arduino was so disruptive in the market that many boards, included the ones in Figure 3.6, support the special C++ dialect.

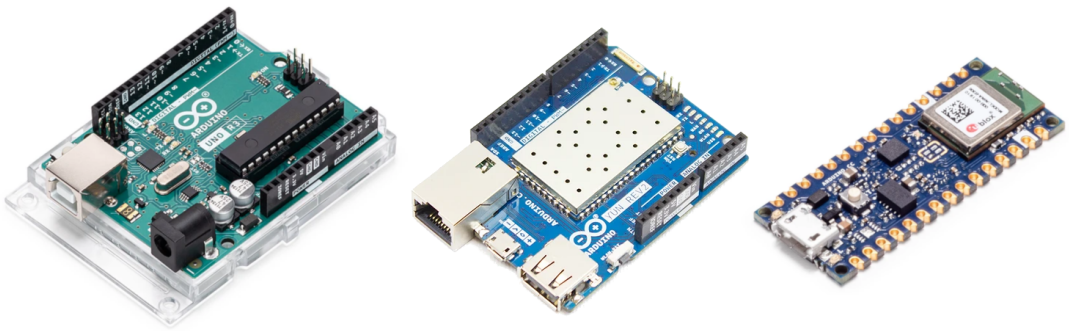


Figure 3.7: Arduino UNO Rev 3 on the left, Arduino Yún in the middle, Arduino Nano 33 BLE on the right

Shields are modular circuit boards that can be added to extend capabilities to different application needs. These can be attached directly on top of the board and provide sensors, interfaces, peripherals on a single board, rather than attaching to the Arduino singularly. Some of the functionalities that can be added by a shield are Ethernet, Wi-Fi, GPS, displays and cameras, motor drivers. Most of the additional shields on the market have been developed for the Arduino UNO, since it is the most common board.

The choice of making the Arduino schematics open-source and accessible to anyone has largely favored the development of newer boards, similar in capacity to the Arduino but more specialized, since producers and board makers are able to keep only the components needed or add different ones. An example can be the two middle boards in Figure 3.6, which rode

[10] https://store.arduino.cc/products/arduino-uno-rev3

the wave of Arduino's popularity. Not only the datasheets are available for all boards, but also the Arduino IDE software is open-source.

The versatility of Arduino and its easy-to-understand interface makes it a leading choice for a wide range of users around the world from hobbyists, designers, and artists to product prototypes.

Newer Arduino boards offer many integrated functionalities, for example:

- *Arduino MKR NB 1500*: offers an all-in-one solution for Narrow-Band IoT large-coverage solutions;
- *Arduino MKR Wi-Fi 1010*: offers integrated Wi-Fi and Bluetooth;
- *Arduino Nano 33 BLE Sense*: contains BLE connectivity and multiple sensors, such as 9 axis inertial, humidity, and temperature, barometric, microphone, gesture, proximity, light color and light intensity.

Particularly, this last model, the board on the right in Figure 3.7, has been considered as one of the possible choices as development board for this project. As better explained in Section 5.2, it has been discarded since it does not offer LoRa connectivity and an additional module would have been necessary to connect the board in a mesh.

3.3.2 Raspberry Pi

Another important microcontroller on the market is the Raspberry Pi, developed by Eben Upton at the University of Cambridge in the United Kingdom with the aim of teaching and improving programming skills of students in developing countries.

Compared to the Arduino specifications, it offers more functionalities, since the flagship board of the company includes an ARM processor, a GPU with HDMI output connectivity, an Ethernet port, USB ports to connect mouse and keyboard, a camera interface, more RAM memory and more I/O pins. ARM processors, or Advanced RISC Machine processors, are better suited to mobile computing, since they use a simplified, less power-hungry method of processing. This allows the Raspberry Pi to run full operating systems such as some Linux distributions, included the official operating system Raspbian OS.

Since the entire Computer (the Processor, RAM, Storage, Graphics, Connectors, etc.) is sitting on a single Printed Circuit Board (PCB), the Raspberry Pi (and other similar boards) are called as Single Board Computers (SBC).

Letting their differences aside, both are very popular boards among hobbyists and even professionals. Some projects involve the use of both boards in a master-slave architecture,

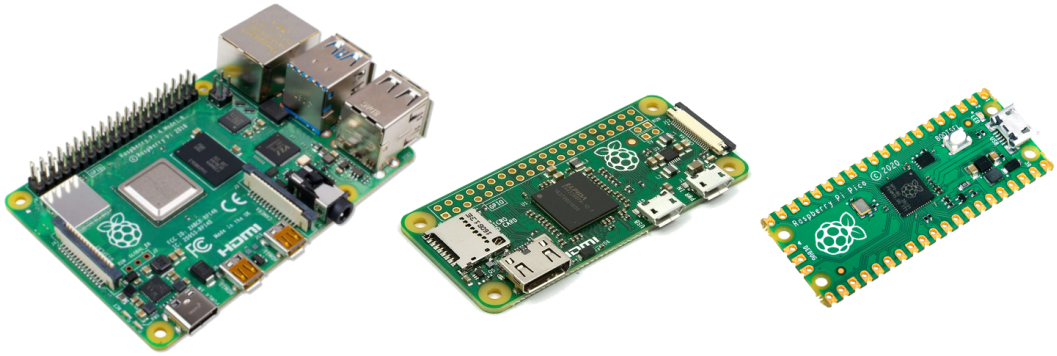


Figure 3.8: Raspberry Pi Model 3B+ on the left, Raspberry Pi Zero in the middle, Raspberry Pi Pico on the right

where the Raspberry Pi acts as a master and gathers the data from the Arduinos, which are equipped with the sensors.

Some of the main competitors of the Raspberry Pi are the Banana Pi and the Asus Tinker Board (in Figure 3.6). Since, like the Arduino, the Raspberry Pi boards schematics are open-source and available online[11], board makers have been able to adapt them in their own way.

There are now different Raspberry Pi boards, or models, each a bit more specialized than the other. Some of the most important models are:

- *3B+/4*: the model 3B+ and 4 are their key products, marketed as a "*tiny, dual-display, desktop computer*"[12];
- *Zero*: the smallest form factor Raspberry Pi;
- *Pico*: a low-cost microcontroller board with flexible digital interfaces.

They are represented in Figure 3.8 and can be considered as the latest evolution of what is needed to learn programming in a Unix like environment at a low cost, in fact these boards cost 35$, 5$ and 3$ respectively. A complete list of the available boards can be found on their online store[13].

The Raspberry Pi Pico was considered for this project, but was rejected since, as the Arduino, it would have needed additional modules for LoRa and Bluetooth Low Energy (*BLE*) connectivity, while the other models have a computational power much greater than the one needed.

[11] www.raspberrypi.org/documentation/computers/raspberry-pi.html
[12] www.raspberrypi.org/products/raspberry-pi-4-model-b
[13] www.raspberrypi.org/products

Figure 3.9: FiPy on the left, PyTrack 2X in the middle, PySense on the right

3.3.3 Pycom

While Raspberry Pi and Arduino share a longer history, Pycom is a younger company. It was founded in 2015 via a crowdfunding campaign on Kickstarter with the goal to create a new board for immediate development in the world of IoT, with all the possible connectivity. As the other two previously described companies, Pycom offers multiple board choices, such as the FiPy, represented in Figure 3.9, the WiPy and the LoPy. These boards are very similar to each other, since all of them offer Wi-Fi and BLE, have the same chipset, interfaces and memory.

In particular, the FiPy board, has been chosen for this project since it packs five networks in one small board. This choice is furthered in Chapter 5.

As described in the product page[14], it is capable to communicate via Wi-Fi, Bluetooth, LoRa, Sigfox and dual LTE-M (CAT-M1 and NB-IoT), and gives access to global LPWAN networks. All the boards offered by Pycom at the time of writing are equipped with an Espressif ESP32 chipset, 4MB of RAM and an flash memory of 8MB.

Contrary to Arduino and Raspberry Pi, Pycom has decided to maintain the datasheets and the firmware of their boards proprietary, which means there is far less support from the community when comes to finding bugs in the software or improving the component placement on the board. Nonetheless, Pycom boards have been chosen for the affability of the company producing them, also because of their high density of hardware on a board with a small footprint.

Additional sensors and functions can be added to Pycom boards via shields, just like the Arduino. Particularly for this project, the FiPy has been integrated with the Pytrack 2.0, which add accelerometer and GPS, and the Pysense, which add ambient light, pressure and

[14] www.pycom.io/product/fipy

humidity. Both expansion boards are represented in Figure 3.9.

For the programming language, Pycom boards can be programmed using Micropython via their Pymakr suite of IDE plugins, the Pymate mobile app, and Pybytes an online middleware platform and desktop application to remotely manage the boards.

About the cost of the boards, the price of the single major components used for this project are, at the time of writing:

- FiPy: 59.40€
- Pytrack 2.0: 40.65€
- Pysense: 29.65€

Although the costs are higher compared to those of Arduino boards, it is important to consider that these offer and all-in-one solution. The additional cost in buying an external generic shield for an Arduino board would be reflected not only on the money spent on the shield itself, but also in the time and effort of configuring and troubleshooting it in case of errors.

An overall advantage of Pycom's products is the tight ecosystem, which allows for faster and easier troubleshooting, configuration and programming. All these factors are well described in Pycom's documentation on their website[15].

A more in depth description of the chosen technologies is described in Chapter 5, while a personal consideration on working with such boards is expressed in Section 7.2.1.

[15] https://docs.pycom.io

To succeed, planning alone is insufficient.
One must improvise as well.

Isaac Asimov, Foundation series

4
Related work

IoT is one of the most discussed topics in both academical and industrial research, as mentioned in Section 2.1.3.

By contrast, mesh networks that use wireless technologies have been researched for decades, but have yet to be put into use in large scale. The correct definition of a *wireless mesh network*, or WMN, is "*a communications network made up of radio nodes organized in a mesh topology instead of star topology*" [29]. This particular *multi-hop network*, Figure 4.1, where data "*hops*" from a node to another until it reaches its destination, can make a difference when it comes to the interconnections in the IoT world. It is important that "*future IoT deployments need to focus on sustainability since huge centralized data centers (Cloud Computing) have become a critical part of the infrastructure*" [30].

This chapter anticipates the technical one, and shows some related projects from which the project has drawn inspiration. At first, challenges and solutions are presented for these WMS, alongside their advantages and disadvantages. Below are also presented some of the projects which have been made at various levels from homemade projects, to research, to the ones already available on the market.

4.1 Overview of wireless mesh networks

With new technologies, such as the ones presented in Section 3.2, wireless mesh networking has been evolving and has reached an ideal point of maturity, which allows it to be useful

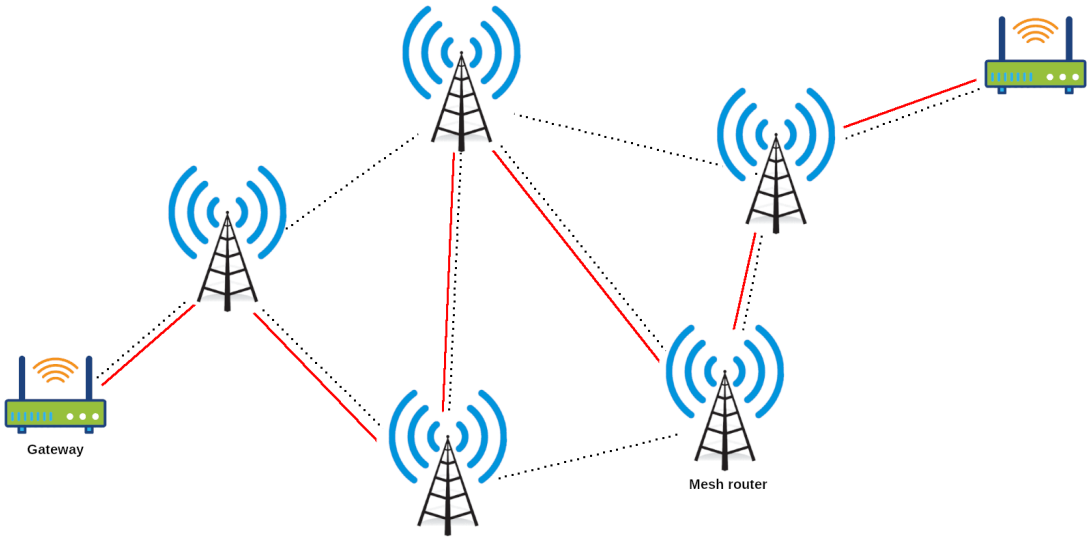


Figure 4.1: Multi-hop network

in the development of IoT devices for mass production. Also, the rising number of connected homes and industry support on open-source resources has made mesh networking truly accessible and low-cost. Such networks are starting to be regarded as more viable and real choices for commercial as well as industrial IoT applications. Besides the possibilities such networks can bring in optimizing communication among nodes, a big advantage is the possibility of bringing connectivity in a system where nodes might have limited connection.

Some of the applications for this network topology are:

- *smart cities*: extending radio signals through campus grounds, business parks, parking garages, and other outdoor facilities;
- *healthcare equipment*: monitoring and locating medical equipment, also serve as a backup for medical devices that always require to stay online;
- *smart home*: track and manage data from sensors all around the house;
- *farming*: in areas where power and connectivity are highly limited, mesh networking is ideal to track sun exposure and water levels across the crops and fields, for example.

Before choosing a mesh network topology it is important to evaluate aspects such as *installation*, *device management* and *support*.

Energy management represents an important factor as well, since the connected devices could be small battery operated. Thus, efficient transmission techniques and protocols that consider the amount of energy used, must be developed in order to optimally use the batteries

on such devices That is why there is a particular network topology, *Low Power Wireless Area Network*, focused on the interconnection of these devices, described in Section 3.2.

Solutions to optimize the networks can be implemented both via hardware, via repeaters, and software. The latter is mainly implemented with with AI techniques, which allow protocols to "*learn*" how to spatially coordinate and adapt contention patterns. Authors of [31], for example, have developed new distributed MAC scheduling algorithms by combining synchronous two-level priority RTS/CTS[1] handshaking with randomized time slot selection. From these "*adaptively biasing time-slot selection probabilities based on past history, one can develop variations that are also provably throughput-optimal and exhibit better convergence rates*".

Classic network topologies, such as the ones in Figure 3.3, are not designed for IoT communication and might not be able to meet the requirements for such, and more, dynamic devices, which are also low-powered sensors. Another important setback of such networks is their single point of failure nature, which "*makes the entire system extremely vulnerable when it comes to disasters or even difficult environment as the sensors may need to be deployed into some hardly reachable locations*" [30].

An example of a multi-hop wireless mesh network is *VANET* (*Vehicular Ad-Hoc Network*), a particular case of wireless multi-hop network, which has the constraint of fast topology changes due to the high node mobility, Figure 4.3.

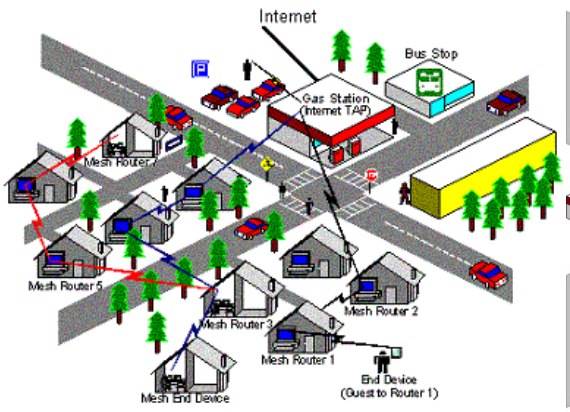


Figure 4.2: Self organizing wireless mesh networks [32]

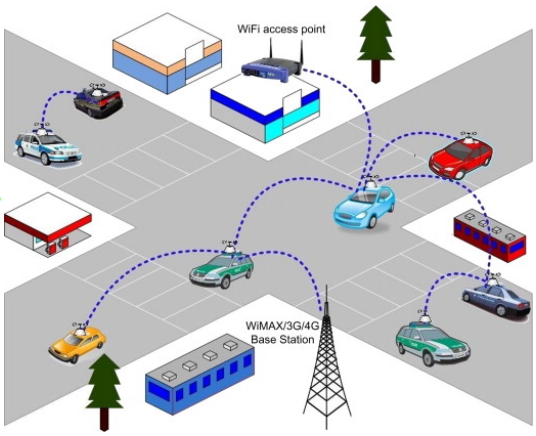


Figure 4.3: An example of a VANET [33]

This is a very important type of network, giving the increasing number of vehicles equipped with computing technologies and wireless communication devices. Authors of [33] explain

[1] *Request To Send / Clear To Send*

how such increase "*intervehicle communication is becoming a promising field of research, standardization, and development*". A tangible example can be seen in the rising of electric vehicles and their constant rise in sells on the market, which brought more attention on the aforementioned factors.

Microsoft has also tried to apply mesh networking to normal Internet household use [32], however, given the improvements in telecommunication infrastructures and the advancement of fiber optics and Internet via cable, the project has been abandoned. Their proposed topology is represented in Figure 4.2. This shows how each network topology is better suited for particular scenarios.

Compared to the example of Microsoft's project, a VANET is more suited for a wms since the data transfered among cars is less than the one transfered among houses.

A more personal approach consideration about this networking topology is made in Section 7.2.2.

The components of a mesh network usually are:

- *nodes*: devices that communicate data with each other;
- *gateway*: allows devices to transfer data in the network, also provides a backhaul to the Internet for the local mesh network;
- *repeater*: repeaters that maintain signal and forward messages between endpoints
- *endpoint*: mesh-only devices that don't route messages for other devices, but send them to other nodes.

4.1.0.1 Advantages of WMS

Mesh Network for IoT devices offers enormous benefits that make it sought-after in enterprises and significant in an IoT app development company.

Mesh networks can offer substantial benefits in the development of IoT applications:

- *self-healing*: they automatically adjust allowing data to take the best path to transfer even if a few nodes lose connection;
- *self-configuration*: new nodes calibrate automatically and connect to the network without any previous setup;
- *reliability*: more nodes create multiple routes in which data packages can travel, which makes the network reliable and error-resistant in case of a node failing;
- *cost*: when considering the nodes that typically compose a mesh network, they are inexpensive, thus adding more nodes and sensors to expand the network should not be a substantial cost.

These are just some of the advantages that using a mesh network brings, but it is also important to consider the scenario where the network is deployed. Not always a mesh network is necessary, but when it is well suited, it reduces expenses in many other ways like better management, optimization of resources usage, and more.

4.1.0.2 Disasvantages of WMS

Despite the numerous advantages previously described, there are also some drawbacks when using mesh networks in IoT. That is how come is so important to know when such network topologies can fit in the project.

These disadvantages are:

- *low capacity*: such networks are well suited for sending small data packages and they do not perform well while transferring larger file sized data;
- *latency*: sending data from a node to another requires longer time than direct transfer, it might not be an issue when the system requires few packages, but, it might take too long for some systems;
- *maintenance*: finding bugs in a mesh network might be time-consuming since there could be a lot of factors that could create errors. Also, even if there is not a single-point-of-failure that could take down the whole network, it might not be easy to untangle logs from the nodes.

The amount of data that a mesh network is capable to handle also depends on the transmission technology which is used, as explained in Section 3.2. For large amount of data Wi-Fi represents a better option then LoRa or BLE, for example.

4.2 Scalability of WMS

Scalability of mesh networks is treated in a separate section, since it cannot be considered as an advantage, nor as a disadvantage.

Alongside topological stability, these are two of the most challenging issues in current wireless mesh networks deployments, as explained in [34].

In the case of wireless mesh networks, scalability has to be considered "*both in terms of increased geographical area and coverage and number of users*" [35]. Each solution for scalability brings advantages and disadvantages, since it may impact one or the other terms from above.

There are numerous various scaling relationships in this network topology: as number of nodes increases, the load increases and load increases as the communication distance increases and the total one-hop capacity increases as the coverage of the network increase.

Some of the factors that affect the scalability of a wireless mesh network in particular are:

- *co-channel interference*
- *routing protocol overhead*
- *half-duplex nature of radio antennas*
- *difficulties in handling multiple frequency radio systems*
- *deployment architecture*
- *medium access control*
- *density of nodes and degree of nodes*
- *communication pattern (locality and number of hops).*

A formula that determines the capacity of an arbitrary mesh network is $\theta(1/\sqrt{n})$, where n is the number of nodes in the network. In other words, with the increase of nodes in the network, the information carrying capacity becomes negligibly small. Such formula is described in depth in a seminal paper by Gupta and Kumar [36].

Although this is a theoretical description of the effects of scaling on the amount of data that can be exchange in a network, it is very different to what can happen in practice. Such comparison is described in the paper "*Scalability of Mobile Ad Hoc Networks: Theory vs practice*" [37], where the author inspects how different relationships between the previously mentioned factors affect scalability in real life mesh networks.

4.3 Projects

This sections describes three category of projects: open-source, research and commercial. These categories should represent a sample of what is the current state of the art for wireless mesh networking, particularly the ones communicating via LoRa.

4.3.1 Open source projects

4.3.1.1 LoRa Mesh Chat

This project consist in an add-on for mobile phones that enable text messaging in a group when outside cellular and Internet coverage.

Figure 4.4: LoRa Mesh Chat ESP32 device

As explained on the project's webpage [2], an ESP32 microcontroller with a LoRa antenna and an OLED display is programmed to connect to the phone either via and OTG cable or BLE. By using an application on the phone, the user is capable of sending text messages in a group.

4.3.1.2 Meshtastic

Meshtastic is an open-source[3] LoRa signal extending mesh communicator that uses LoRa. This project allows the use of inexpensive (around 30$) GPS radios as an extensible, long battery life mesh GPS communicators for who practices sports like hiking, skiing, paragliding. Essentially for who needs connectivity in places without Internet access.

This project consists of private meshes created by its users, and allow the share of GPS location and text messages. The radios automatically create a mesh to forward packets as needed, so everyone in the group can receive messages from even the furthest member. Optionally devices can connect with phones, but no mobile is required.

4.3.2 Research projects

4.3.3 LoRa mesh networks

The idea of creating a wireless mesh network in which nodes communicate via LoRa technology is certainly not new. Authors of [26] designed and tested a LoRa wireless mesh network system for collecting data from IoT sensors distributed across a mixed scenario, with nodes both indoors and outdoors. Their measurements show that in urban areas "*LoRa requires dense deployment of LoRa gateways (GWs) to ensure that indoor LoRa devices can successfully transfer data back to remote GWs*".

Other papers, such as [38], [39] and [40] analyze study cases of LoRa mesh networks in different scenarios.

[2] www.hackster.io/scottpowell69/lora-mesh-chat-5267d9

[3] www.github.com/meshtastic

The first paper, "*Exploring Multi-Hop LoRa for Green Smart Cities*" [38] describes an experimental case study on multi-hop LoRa systems with the goal of understanding the amount of energy used by each node and the range in both transmission and reception of such nodes, as represented in Figure 4.5. Authors of such paper conclude by stating that "*the smart city concept would require hundreds of thousands of end devices to be deployed across a city, multi-hop assisted star-of-stars topologies may be considered a viable option for clustering many end devices together and making the network more energy efficient*".

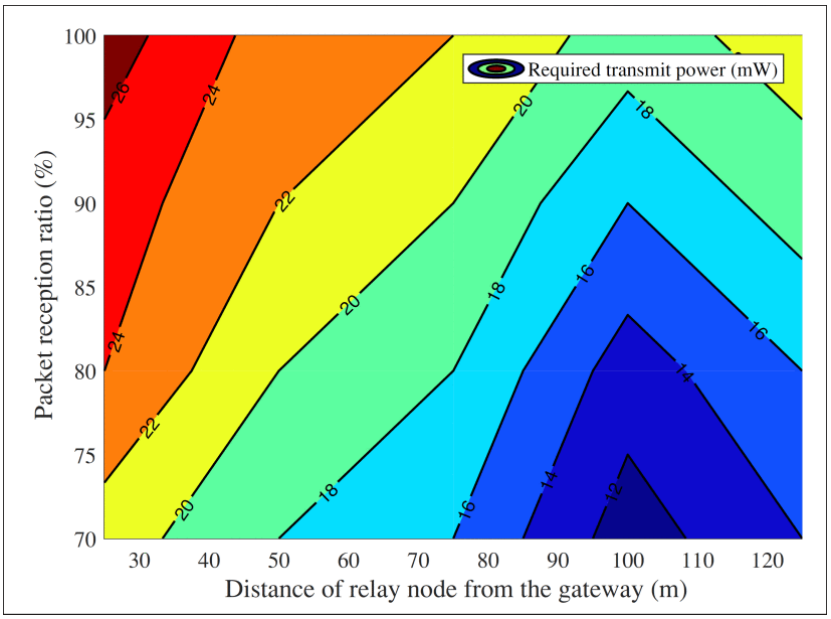


Figure 4.5: Required transmit power as a function of packet reception ratio and location of relay node [38]

In the second paper, "*Black Powder Flow Monitoring in Pipelines by Means of Multi-Hop LoRa Networks*" [39], authors propose "*a multi-hop linear topology that allows to deploy a monitoring infrastructure that can be used to cover extremely long pipeline structures thanks to the wide transmission range of the LoRa technology*".

Authors of the third and last paper, "*LoRa-based Mesh Network for Off-grid Emergency Communications*" [40], propose the use of a LoRa mesh network to monitor natural calamities in the Philippines. Such network would integrate with the existing emergency system, based on GSM base stations, and would be composed of embedded devices capable of sending information via LoRa and communicate with a user's phone via BLE. This architecture resembles the one of the projects later described in Section 4.3.4.1, where a phone is needed to generate the data sent in the network.

4.3.3.1 LoRaCTP

LoRaCTP, which stands for *LoRa Content Transfer Protocol*, is a "*flexible protocol based on LoRa technology that allows for the transfer of "content" to large distances with very low energy*", as explained by the authors in [41].

The goal of this protocol is to make LoRa transmissions reliable, by "*introducing a lightweight connection set-up and ideally allowing the sending of an as-long-as necessary data message*". By "*content*", authors refer to a "*self-contained piece of data*", which in essence is a `JSON` encoded message with a virtually unlimited length. The structure of a LoRaCTP packet is depicted in Figure 4.6.

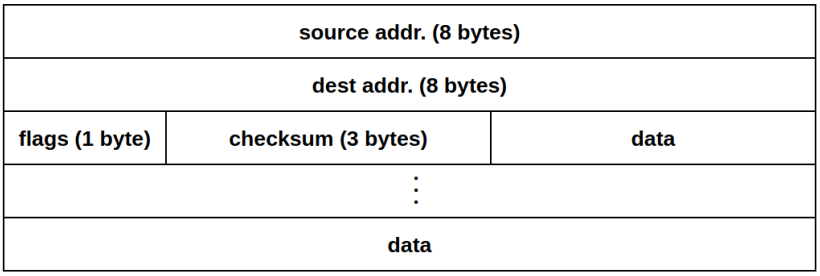


Figure 4.6: Structure of the packet used by the stop-and-wait ARQ [41]

LoRaCTP's structure is based on a unicast protocol that uses a textbook stop-and-wait Automatic Repeat Request (ARQ) approach with a "*dynamic and adaptive value for the retransmission delay*". The protocol ensures data is not lost due to dropped packets and that these are received in the correct order. Each packet is sent by using a "three attempts scheme": if after three attempts no `ACK` is received, the protocol supposes the channel is currently busy or too noisy and the content sending is dropped.

Authors also introduced a "*lightweight transport protocol that is used to establish a connection between a master node, and a client node*", Figure 4.7, which allow for greater flexibility. Such connections are established in a unicast manner, by providing the address of the master device, or using an anycast approach.

Authors have made the code publicly available online on a GitHub repository[4]. Section 5.3 explains how `loracpt.py` is used in Open LoRa Mesh in forwarding messages from one node to another.

[4] www.github.com/pmanzoni/loractp

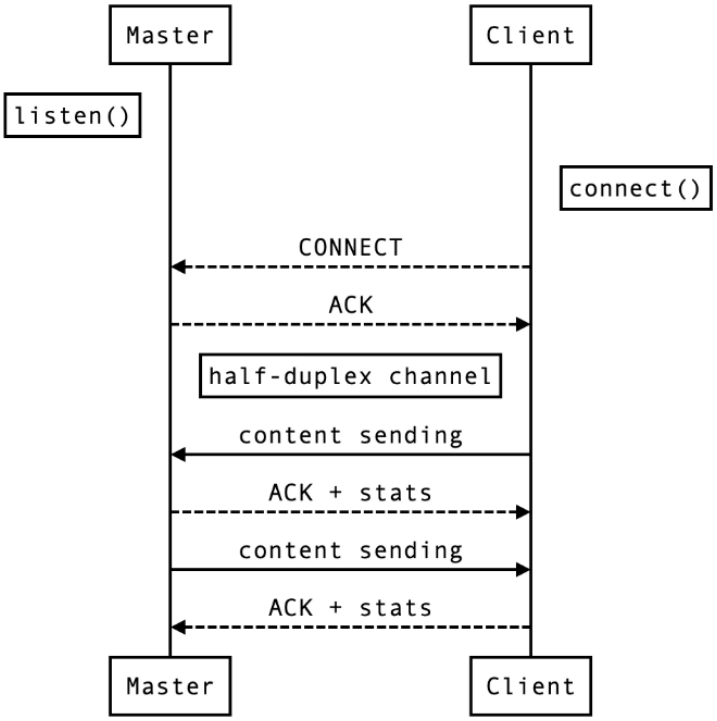


Figure 4.7: Flow of the establishment and interchange of data in LoRaCTP [41]

4.3.4 Commercial applications

4.3.4.1 Off grid mesh devices: Sonnet and goTenna

Both of the following devices use LoRa to create a mesh network and have been designed for emergency off-grid communication, where cellular towers or land-line phones are not physically reachable.

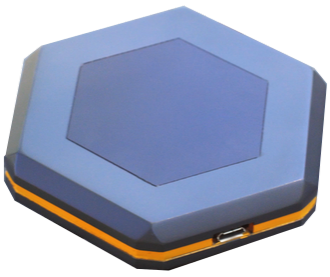


Figure 4.8: Sonnet

Sonnet[5], in Figure 4.8, connects to the smartphone via Wi-Fi allowing the user to send texts, voice messages, images, data, files and share GPS locations to any other Sonnet users up to several kilometers away, thanks to LoRa. This completely removes smartphones' dependency on cellular grid and other network infrastructure, and allows Sonnet to be used even when there is no cellular connectivity or Internet access.

[5] www.sonnettech.com

According to the product's Kickstarter page[6] "*with Sonnet, data can be relayed up to 16 times to achieve a maximum range of 80 km (50 miles)*".

Figure 4.9: goTenna Mesh

Figure 4.10: goTenna Pro

The goTenna[7], in Figure 4.9, offers similar functions as the Sonnet, allowing to send text and GPS locations without the use of a cellphone with Internet connectivity. Mesh-networking allows to relay messages from a node to another until they reach destination.

A compact and ruggedized kit, the goTenna Pro X, in Figure 4.10, allows control for larger teams that operate in complex environments where it is not possible to be offline.

As Sonnet, goTenna has raised the funds necessary to enter the market as a crowdfunded project on Kickstarter[8].

Compared to the projects in Section 4.3.1.1 and Section 4.3.1.2, Sonnet and goTenna offer a more stable network, thanks to the fact that they can be considered finished products which have been thoroughly tested and produced on a bigger scale.

4.3.4.2 Wi-Fi mesh

Wi-Fi mesh, or *whole home Wi-Fi* systems, consists of a main router that connects directly to the main modem, and a series of satellite modules, or nodes, placed around the house for full Wi-Fi coverage. Each node serves as a hop point for other nodes in the system and are all part of a single wireless network and share the same SSID and password, unlike traditional

[6] www.kickstarter.com/profile/sonnet/created
[7] www.gotenna.com
[8] www.kickstarter.com/profile/gotenna/created

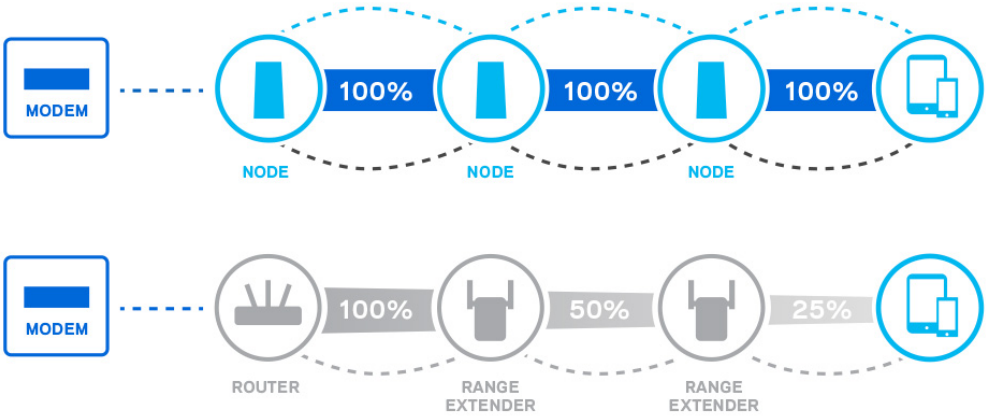


Figure 4.11: Wi-Fi 6 mesh nodes compared to Wi-Fi range extenders performance

Wi-Fi routers. Weakened signal or Wi-Fi dead spots of the latter are the result of physical obstructions (floor, doors, and walls).

A modular mesh whole home Wi-Fi system is flexible and scalable, giving a customizable method of expanding Wi-Fi coverage without the need to add range extenders, which are certainly effective when it comes to increasing the router range, but they do so at the expense of Wi-Fi performance, which gets cut in half.

Wi-Fi 6 is an evolution of 802.11ac technology, which promises increased throughput speeds (up to 9.6Gbps), less network congestion, greater client capacity, and better range performance thanks to the improved wireless technologies, including *Orthogonal Frequency-Division Multiple Access* (*OFDMA*). The latter improves overall throughput by breaking Wi-Fi channels into sub-channels, allowing up to 30 users to share a channel at the same time.

Many important manufacturers for network hardware, like Netgear and Cisco, have already implemented such technology and functions in their products available not only for industrial purposes, but also for average users.

***K**eep*
***I**t*
***S**imple*
***S**tupid*

Kelly Johnson

5 Proposed solution

This chapter contains the technical description of "*Open LoRa Mesh*", a mesh network composed by FiPy devices that communicate with each other via LoRa. Open LoRa Mesh is able to accommodate small messages, such as the ones sent by MegaSense, described in Section 2.2.2. All network related functions are contained in a library that describes main components of this project.

Even though it would have been possible to use a simulator, such as "*The one*" [1], for demonstrating the usefulness of such network, the final project has been realized with Pycom hardware, discussed in Section 5.2.

The code created for this project is open sourced and available on GitHub under the repository *Open LoRa Mesh* [2], and better explained in the following Section 5.3.

5.1 Network architecture

Before talking about the architecture of Open LoRa Mesh, it is important to understand the one of MegaSense. As can be seen in Figure 5.1, since each MegaSense device is made to be carried by a person, devices connect each one to the assigned user's phone. The latter then communicates with the MegaSense sensor and, via the application developed by the Computer Science Department at the University of Helsinki, sends the air-quality readings

[1] akeranen.github.io/the-one

[2] www.github.com/cipz/OpenLoRaMesh

and the GPS location, which has been read by the application from the phone itself, to the servers in the cloud.

Communication between the MegaSense sensor and phone is done via BLE, and exchanges data such as sensor readings (like NO_2, O_3, CO, battery percentage, *etc.*) and calibration measurements, in `JSON` format. Thus each device has to be linked to a person's phone, which might enhance its portability, but not its possibility to read data independently if placed in a specific location. Such communication resembles the one from devices described in Section 4.3.4.

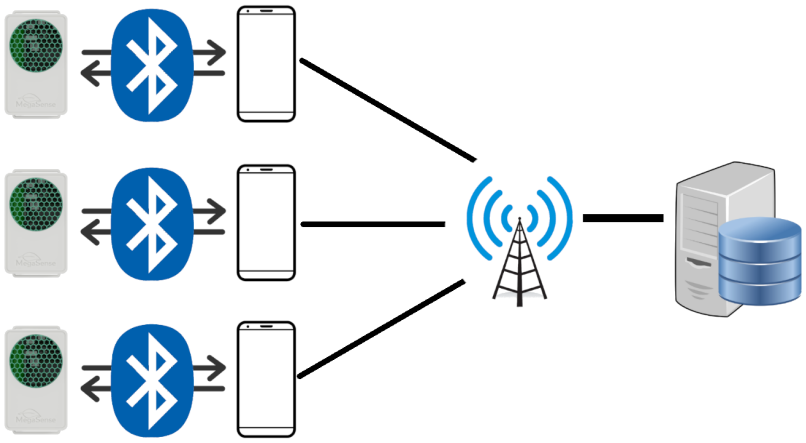


Figure 5.1: MegaSense architecture

One of the goals Open LoRa Mesh tries to achieve is to give independence to these devices so that they can communicate with one another and allow information to be passed along between nodes without the aid of a user's phone.

This is why the architecture would evolve from the one in Figure 5.1 to Figure 5.2: the phone is replaced by a FiPy, which also connects to the MegaSense via BLE, and sends the data acquired from the sensors in the network. Nodes communicate with each other via LoRa, detailed in Section 3.2.1, and exchange data using mainly *LoRaCTP*, described in Section 4.3.3.1.

Each FiPy board represents a node of the network, which in this case is a "*Flat Wireless Mesh*", where each node acts both as data provider and as data forwarder. Although simple, such network architecture is well known from the previously mentioned, Section 4.2, Gupta and Kumar [36]. Besides not scaling well and having the potential to put very high resource constraints, "*addressing schemes and service discovery would prove to be a major bottleneck against scalability*" [35].

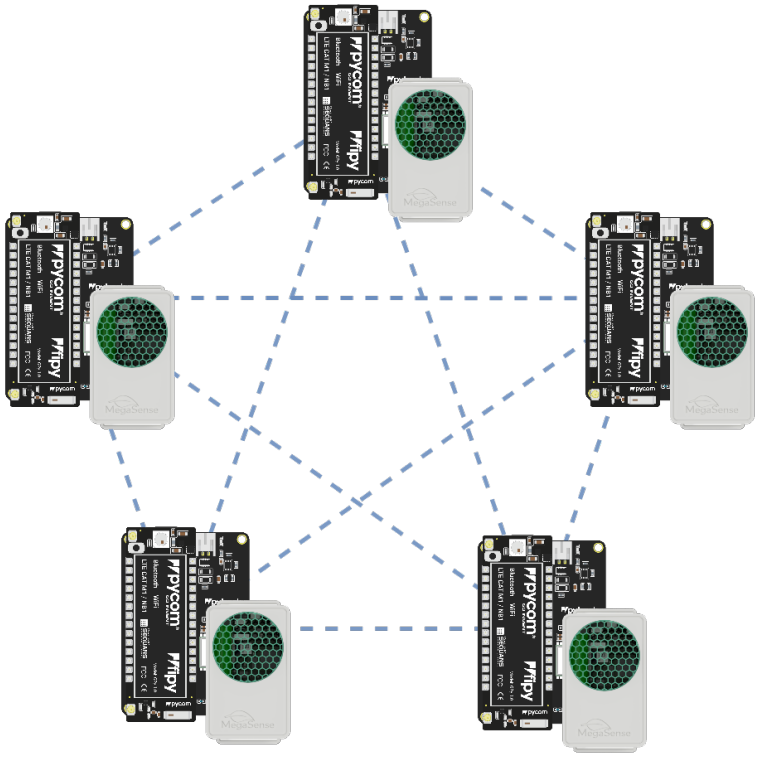


Figure 5.2: Open LoRa Mesh architecture

Despite having these disadvantages, it is important to keep in mind the small environment and the limited number of devices that will be connected in such network. Since communication is not frequent and data packets are not large, this network layout can satisfy the MegaSense's requirements for interconnection.

5.2 Open LoRa Mesh Hardware

For the development of Open LoRa Mesh, the chosen hardware to replace the phone in the original MegaSense architecture is composed by Pycom's FiPy microcontrollers, along with its shields, like the PyTrack and the PySense. A prototype of the MegaSense device was also given from the developers of the project in order to test the connectivity with the Pycom boards and test its capacity to send and receive data from other nodes in the network. All devices can be seen together in Figure 5.3.

Although the expansion board is important, the FiPy is the one that contains all the chips that allow connectivity, as said in Section 3.3.3. After an initial evaluation on the require-

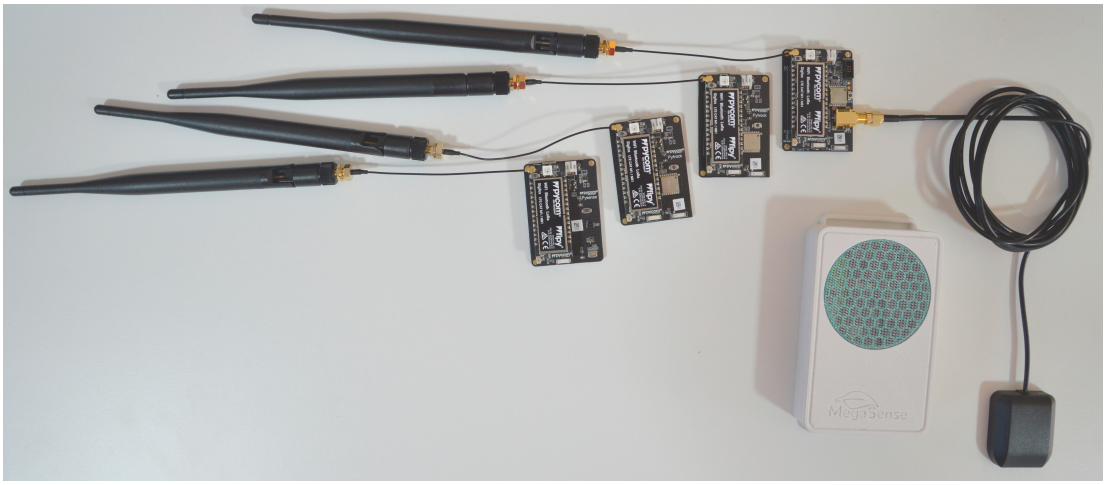


Figure 5.3: FiPy with PyTrack, LoRa antenna, GPS antenna and MegaSense prototype

ments necessary in creating a network for exchanging messages among air quality sensing devices, Pycom's boards have come first considering the numerous possibilities it allows for prototyping.

The exact hardware with which this project has been completed is: four FiPy boards, one PyTrack 2X, one PyTrack, two PySense shields and one MegaSense prototype, Figure 2.9. Pycom boards do not have a USB connection directly on them, so they can be programmed either using a shield or via an UART (Universal Asynchronous Receiver-Transmitter) to USB.

Alternative microcontrollers, such as the Raspberry Pi Pico, Raspberry Pi 4, Arduino Nano 33 BLE Sense, and ESP32 boards have been discard given either their excessive power or their need for multiple expansion shields.

5.3 Open LoRa Mesh Software

While Arduino's C++ dialect separates the code in two main functions, the `setup()` and `loop()`, as mentioned in Chapter 3, the Pycom boards use two files to separate an initial bootstrap of the board and a main section of the code. These two special files are called `boot.py` and `main.py` respectively.

The following section explains the software made for this project and how this is composed and implemented. Afterwards, the addressing scheme in LoRa used in Open LoRa Mesh is described.

All network related functions are contained in a library, `mesh.py`, that contains the main components of this project, while connectivity with the MegaSense sensor is managed through

a second library, `megasense.py`, that contains the necessary functions to connect and exchange data with the boards.

The FiPy boards have been programmed using Micropython, and the firmware flashed[3] on the boards, at the time of writing, is `v1.16.5`.

5.3.1 Important functions

This subsection shows in depth how the software works, for the most significant building block of the code there are finite-state automata (or FSA) that show the state nodes find themselves in. For a graphical reason, each state of the following automata is abbreviated, and, when needed, the full state is described in a table underneath it.

At first an overview of the entire software is presented, then for each important component of the software there is a subsection that describes how it functions and connects to the rest. Such vision should allow to understand the system from a broader point of view to a more detailed one, where every function is analyzed.

5.3.1.1 Overview of the whole system

As previously said, Pycom boards require the presence of two files `boot.py` and `main.py`. When the board is powered up, the first file executed is `boot.py`, while the second one is `main.py`, unless otherwise specified. If neither of these are present, the board will boot in an interactive mode, where commands are sent in input via the terminal.

In the case of Open LoRa Mesh, the board can either boot normally and create the network, or join one if already exists, or boot in a state of receiver. The receiver state allows for the device to be used as a station that only sniffs the messages traveling in the air. Continuing in the normal sequence, thus executing the code in `main.py`, will allow the node to connect to an existing mesh or to create one, for sending and receiving data afterwards.

Open LoRa Mesh heavily relies on the use of LoRaCTP, described in Section 4.3.3.1, for message transmission. This is because LoRaCTP builds an important layer of stability on top of raw LoRa transmission.

[3] *Flashing* involves the overwriting of existing firmware or data, contained in EEPROM or flash memory module present in an electronic device, with new data.

5.3.1.2 Boot sequence

As mentioned earlier, the `boot.py` file contains the code that is initially executed by the FiPy. The first operation done is to disable Wi-Fi communication and the default blinking ("*heartbeat*") of the FiPy's onboard LED, Figure 5.4. This is because when powered up FiPy's firmware enables Wi-Fi, which can draw unnecessary power from the device even though it is not used in the system.

```
wlan = WLAN(mode=WLAN.STA)
wlan.deinit()
pycom.heartbeat(False)
```

Figure 5.4: Code for Wi-Fi de-initialization and heartbeat stop

During the boot procedure, the board checks the availability of a `config.json` file on an external SD card and, if present, uses the parameters on such files. If not, it checks for such file locally on the flash memory, which contains one with default configurations and is uploaded with the `*.py` files.

The configuration file contains various custom parameters that can be used by the board, like `BOOT_MODE`, which can either be set to `DEFAULT_BOOT` or to `PLAIN_RECEIVER`. If set to `DEFAULT_BOOT`, the board will normally proceed to executing the software in `main.py`. When set to `PLAIN_RECEIVER`, the board will execute code which makes it listen to the packets traveling in the air. Such process can be seen in Figure 5.5, described by Table 5.1

Other modes could be added to the software as an improvement, but at the time of writing only these have been implemented. Improvements on this topic, and others, are later talked about in Section 7.1.2.

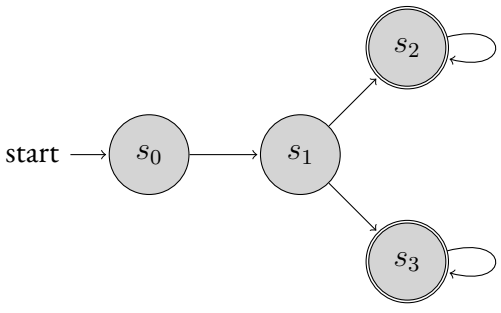


Figure 5.5: FSA for the overview of boot sequence

State abbreviation	State description
$start$	device is powered up
s_0	Wi-Fi is de-initialized and heartbeat is set to `False`
s_1	configuration file is read
s_2	`PLAIN_RECEIVER` mode is initialized
s_3	`DEFAULT_BOOT` mode is initialized

Table 5.1: Boot sequence FSA description

5.3.1.3 Plain receiver

The mode in which the device is set only to listen is called "*PLAIN_RECEIVER*", and the code executed is contained in the `plain_receiver.py` file. Such state lets the FiPy listen to messages that are in its range, without the possibility of sending messages or participate into the mesh. This state can be used for debugging and catching the different messages that are sent in an area.

```
def plain_receiver():
  ctpc = loractp.CTPendpoint()
  print("Executing plain_receiver.py")
  while True:
    print('plain_receiver.py: waiting for data')
    try:
      rcvd_data, addr = ctpc.recvit()
      print("plainreceiver.py: got {} from {}".format(rcvd_data, addr))
    except Exception as e:
      print ("plainreceiver.py: EXCEPTION!! ", e)
```

Figure 5.6: `PLAIN_RECEIVER` mode code

The principal function in the file `plain_receiver.py` is `plain_receiver()`, in Figure 5.6. Such function is an adaptation of the example one contained in the LoRaCTP repository. It first creates an endpoint, which is then used in an infinite loop that reads data received at the endpoint and logs it.

5.3.1.4 Main sequence and key variables

This subsection describes the course of the `main.py` file and the `main()` function it contains. The code in this file is automatically executed by the board, unless specified otherwise, after the code in the previously described `boot.py` file.

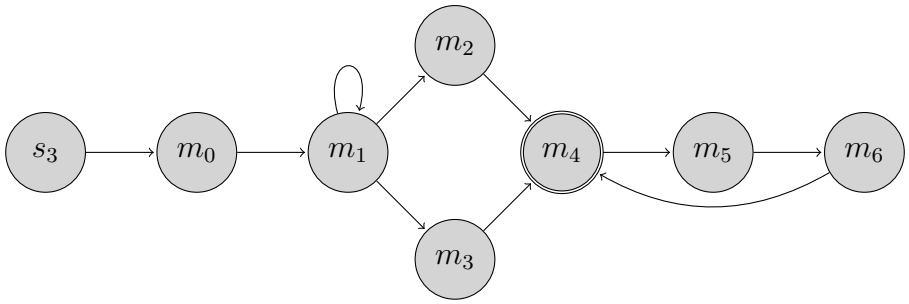


Figure 5.7: FSA for the overview of the main code

State abbreviation	State description
s_3	state s_3 from Figure 5.5 where the code `main.py` starts
m_0	variables,objects and communication with MegaSense device, if available, are initialized
m_1	listen loop for messages for existing LoRa mesh network
m_2	if the mesh exists, the node exchanges information about it with the advertising node
m_3	if the mesh does not exist, the node creates a new mesh and advertises the mesh information
m_4	the node listens for any messages in the air and processes them
m_5	if the MegaSense device is available, the node connects to it, reads data, and sends it into the network
m_6	the node broadcasts a message advertising the mesh network

Table 5.2: Main sequence FSA description

The most important objects used in the main part of this software, and that are initialized at the beginning, are:

- `LORA_ANTENNA_LOCK`: a `LOCK` is placed on the use of the LoRa antenna so that it can be used only in one part of the program at the time;
- `node`: represents the current node and its properties;
- `mesh`: contains the data about the mesh network;
- `routing_table`: contains data about the other nodes connected to the mesh.

`node`, `mesh` and `routing_table` are all objects initialized from the `mesh.py` library, later described in Section 5.3.2.

Like in the `boot()` function, `main()` also searches for a configuration file, at first on an SD card, then on the flash memory. Important entries in this `JSON` file used by `main()` are:

- `GPS`: a boolean that tells the FiPy to either find or not the GPS coordinates of the board. This only happens if the board is connected to a PyTrack shield;
- `MEGASENSE_CONNECT`: another boolean that tells the board either to connect or not to a MegaSense device, of which the MAC address is contained in `MEGASENSE_ADDR`;
- `MEGASENSE_ADDR`: address of the MegaSense device the board will connect to;
- `DEFAULT_MESH_ID`: name of the mesh network the node will create if none is found;
- `DEFAULT_NODE_ID`: identifier of the node.

The functions related to the communication with the MegaSense device are contained in the `megasense.py` file, described in Section 5.5.

5.3.1.5 Mesh initialization

As can be seen in the Figure 5.7, after initializing the necessary variables and objects in m_0, the node listens for any messages that advertise an existing mesh, represented by state m_1. Particularly, this is done via the code in Figure 5.8. An important thing to keep in mind when working with devices that communicate wirelessly is the channel availability. The node listens for *at least* 180 seconds: the function `listen_for_mesh` of the `Node` object contains a line of code that adds a random amount of time so that nodes which start at the same time, do not overlap and cause the creation two separate mesh networks.

```
mesh_id, advertising_node, request_connection = node.listen_for_mesh(180)
```

Figure 5.8: Code that listens for mesh network advertisements

```
if request_connection:
  mesh = Mesh(mesh_id)
  mesh_info = node.request_mesh_info(mesh_id, advertising_node)
  mesh_info = json.loads(mesh_info.decode())

  received_routing_table_content = mesh_info["CURR_ROUTING_TABLE_CONTENT"]
  received_routing_table_version = mesh_info["CURR_ROUTING_TABLE_VERSION"]

  routing_table = RoutingTable(
    received_routing_table_content,
    received_routing_table_version
  )
else:
  mesh = Mesh(force_new_mesh_id = DEFAULT_MESH_ID)
  new_routing_table_dict = { node.LORA_MAC : "" }
  routing_table = RoutingTable(new_routing_table_dict)
```

Figure 5.9: Code use to initialize the mesh network by requesting data from the advertising node or creating a new mesh

The `node.listen_for_mesh()` function is invoked on the `Node` object from the library in `mesh.py`, better explained later in Section 5.3.2. `request_connection`, in Figure 5.8, is a boolean variable that tells if a message has been received.

If positive, the receiving node sends a request for such mesh information to the sending node, this will send the data, which includes the routing table, the routing table version and the mesh id. Otherwise, the current node will create a new mesh network, thus initializing the routing table with itself as parent and using the mesh id given in the configuration file. Such code is present in Figure 5.9 and represents respectively states m_2 and m_4 from Figure 5.7.

5.3.1.6 Main loop

After the code in Figure 5.9 is executed, the code contains an infinite `while True` loop. Unlike the Arduino, which has the `loop()` function that runs infinitely unless ordered differently, Pycom devices only execute the contents of `main.py` once. That is why this infinite loop is needed.

In the FSA in Figure 5.7, state m_4 coincides with the beginning of this loop. It is marked as an accepting state since the program might finish due to possible exceptions. Such exceptions

are part of the ones described later in Section 7.1.2.

At each start of the loop, `LORA_ANTENNA_LOCK` is acquired, thus allowing the node to send and receive data. This infinite loop can be divided in three parts, which are respectively states m_4, m_5 and m_6 in Figure 5.7:

- m_4, listen for incoming messages and send responses;
- m_5, connection with MegaSense sensor and data gathering;
- m_6, broadcasting mesh advertisement.

Broadcasting mesh advertisement messages is done at the very end of the loop: Figure 5.10 is the line of code in the cycle which calls the code in Figure 5.11, implemented in the mesh library.

```
node.broadcast_mesh_advertisement(mesh.MESH_ID)
```

Figure 5.10: Code for mesh advertisement in `main.py` loop

```
def broadcast_mesh_advertisement(self, mesh_id):
  msg = self.MESH_ADVERTISE_PREAMBLE.decode() +
      "=" + mesh_id.decode() + "." + self.LORA_MAC.decode()
  self.broadcast(msg)
```

Figure 5.11: Code for broadcasting advertisement messages in mesh library

5.3.1.7 Message forwarding

As previously mentioned, LoRaCTP's code is used in Open LoRa Mesh to rely the messages from one node to another. This brings all the advantages that are described in Section 4.3.3.1, especially the fact it "ensures that information is not lost due to dropped packets and that packets are received in the correct order" [41].

Since LoRaCTP is based on a unicast protocol, messages contain a field stating which is the final destination of a message, expressed in the first 8 bytes of the MAC address of the destination node. At every hop the message makes to arrive at its destination, it is acknowledged, thus it is not necessary for the first sending node for a acknowledge of its own.

5.3.1.8 MegaSense communication

If the boolean variable `MEGASENSE_CONNECT` in the configuration file is set to `True`, then at the beginning of `main()` a `MegaSense` object is created. Such object is implemented in the `megasense.py` file, a library that contains the functions used to interact with the device.

The architecture of Open LoRa Mesh implies that each node of the network communicates with a MegaSense device via BLE. Thus, the configuration file of each node contains `MEGASENSE_ADDR` variable, a string with the MAC address of a specific MegaSense. If such sensor is in reach, then the node will connect to it and interact, gathering data about air quality in its surroundings. Such interaction is done via the functions in the MegaSense class, described in Section 5.5.

5.3.2 Mesh library

The mesh library is contained in the `mesh.py` file and implements the code for the objects `Node`, `Mesh` and `RoutingTable`.

5.3.2.1 Node

The `Node` object represents the device itself, which, when instantiated, sets the correct LoRa network in which the FiPy operates, the socket, gets the MAC address and sets a node identifier to either the default value or to a given one. The complete list of the methods made available by this object and the implementation of its constructor is represented in Figure 5.12.

5.3.2.2 Mesh

The `Mesh` object is used track the mesh network to which the device is connected to. It implements the constructor of the object, the get and set methods for the mesh identifier, and a method that generates a mesh identifier based on the MAC address of the current device, if no mesh identifier is given.

The signature of the methods contained in the `Mesh` object and the implementation of its constructor are in Figure 5.13.

5.3.2.3 RoutingTable

The `RoutingTable` object is the object that keeps track of all the adjacent nodes. In order to keep track of the various connections among nodes, this object uses a dictionary in which en-

```
class Node:
  LORA_NETWORK = None
  LORA_SOCKET = None
  LORA_MAC = None
  NODE_ID = None

  def get_mac_addr(self): ...
  def listen(self): ...
  def broadcast(self, msg): ...
  def send_mesh_info(self, rcv_addr, routing_table): ...
  def request_mesh_info(self, mesh_id, sndr_addr, routing_table): ...
  def broadcast_mesh_advertisement(self, mesh_id): ...
  def listen_for_mesh(self, LISTEN_TIMEOUT = 180): ...
  def __init__(self, default_node_id = 0x00) -> None:
    self.LORA_NETWORK = LoRa(mode=LoRa.LORA, region=LoRa.EU868)
    self.LORA_SOCKET = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
    self.LORA_MAC = binascii.hexlify(LoRa().mac())[8:]
    self.NODE_ID = default_node_id
```

Figure 5.12: List of methods of the Node object and implementation of its constructor

tries are organized so that each node has a parent node assigned. This is true for all nodes but the root, which has None as a parent, since it is considered to be the first node that initiated the mesh.

Nodes are identified via the first 8 bytes of their MAC address, while the routing table has an identifier that keeps track of its version. Such variable changes when there are changes in the routing table, for example a node is added or another one is removed.

The signature of this object's function and the implementation of the constructor method can be seen in Figure 5.14.

5.4 LoRa addressing scheme

As any other transmission method, LoRa needs has an addressing scheme that allows devices to have an unique hardware address (MAC address) that uniquely identifies each node of a network. Such address is assigned by the vendor or manufacturer and saved to the device memory.

LoRa's MAC address is built up with 16 hexadecimal characters, but Open LoRa Mesh

```
class Mesh:
  MESH_ID = None
  def generate_mesh_id(self) -> bytes: ...
  def set_mesh_id(self, new_mesh_id) -> None: ...
  def get_mesh_id(self): ...
  def __init__(self, existing_mesh_id = None,
               force_new_mesh_id = None) -> None:
    if existing_mesh_id:
      self.set_mesh_id(existing_mesh_id)
    elif force_new_mesh_id:
      self.set_mesh_id(force_new_mesh_id)
    else:
      self.set_mesh_id(self.generate_mesh_id())
```

Figure 5.13: List of methods of the `Mesh` object and implementation of its constructor

adapts to LoRaCTP's address scheme, where only the first 8 are used. This can be seen in Figure 4.6 where the *source addr.* and the *dest. addr.* are both 8 bytes.

This has been done to save 8 bytes, 4 for the sender address and 4 for the destination address, that would be added to each packet. A lighter packet means a faster transmission time and less chances of overlapping communications.

The default frequency plan of LoRa devices for Europe is called `EU_863_870868` and is 868.1 to 868.5 MHz.

5.5 MegaSense library

The `megasense.py` file contains the implementation of the MegaSense object. Code in this file is based on the script used by the authors of [20] to connect to the sensor via Linux, which has been adapted to work on Pycom's firmware and implementation of the Micropython BLE library.

Figure 5.15 shows the methods that are implemented in this object. When invoked, the constructor method initiates a BLE scan searching for the device with the given Bluetooth MAC address. This address is previously read from the configuration file and passed to the constructor when invoked in the `main.py` file.

```
class RoutingTable:
  VERSION = None
  ROOT_ADDR = None
  ROUTING_TABLE = dict()
  def check_version(self, other_version) -> bool: ...
  def calculate_version(self): ...
  def set_version(self, new_version) -> None: ...
  def get_version(self): ...
  def get_routing_table(self): ...
  def set_routing_table(self, new_routing_table) -> None: ...
  def add_child(self, parent_addr, child_addr) -> None: ...
  def remove_child(self, remove_addr) -> None: ...
  def get_root(self): ...
  def set_root(self, new_root_addr) -> None: ...
  def __init__(self, existing_routing_table_content = {},
               existing_routing_table_version = None) -> None:
    self.set_root(list(existing_routing_table_content.keys())[0])
    self.set_routing_table(existing_routing_table_content)
    if existing_routing_table_version:
      self.set_version(existing_routing_table_version)
    else:
      self.set_version(self.calculate_version())
```

Figure 5.14: List of methods of the `RoutingTable` object

5.6 Other files

Figure 5.16 shows a list of all files that contribute to this project. Some of these files are, for example, `LED.py`, `L76GNSS.py`, `pymakr.conf`. `LED.py` is an LED library that allows the definition of threads in which LED functions are controlled. These are used in the software and have not been explained in the previous sections since their use is secondary compared to the other functions. `L76GNSS.py` and `pyproc_2.py` are libraries used for the GPS antenna.

5.7 Microcontroller sleep cycle

Most IoT devices that compose LPWANs have three main modes, or profiles: *sense*, *connect* and *sleep*. Sense and connect refer respectively to gathering data from the sensors and sending it into the network, while sleep refers to a sleep-mode energy consumption where most pro-

```
class MegaSense:
  MEGASENSE_ADDR = None
  CONNECTED = False
  def is_connected(): ...
  def read_data(): ...
  def __init__(self, default_megasense_addr = None) -> None: ...
```

Figure 5.15: List of methods of the `MegaSense` object

```
cip ~/Desktop/UNI/TESI/CODE/OpenLoRaMesh  main tree
.
├── fipy_firmware
│   ├── boot.py
│   ├── config.json
│   ├── lib
│   │   ├── L76GNSS.py
│   │   ├── LED.py
│   │   ├── loractp.py
│   │   ├── megasense.py
│   │   ├── mesh.py
│   │   └── pycoproc_2.py
│   ├── main.py
│   ├── plain_receiver.py
│   └── pymakr.conf
└── README.md
```

Figure 5.16: List of files that compose the Open LoRa Mesh project

cesses in the device are halted, and some circuits may be powered down. It can be considered a design trade-off between functionality, size, and battery lifetime.

This energy consumption profile is important because it helps maximizing the battery life, which currently is one of the most considerable bottlenecks in IoT low-power devices, as explained in Section 2.1.3.

However, in the implementation of the Open LoRa Mesh, nodes do not have such functionality implemented, since might it be the cause for network disruptions.

For example, if a new node might want to join a network, but the nearest physical node is in sleep mode, there would be no possibility for the new node to request data about the network, thus wasting time and resources while waiting to the other node to come back online.

Such mode could be implemented but would require an careful analysis on the evolution of the network in time, also would require a network made entirely, or mostly of static nodes. This thought is reviewed again in Section 7.1.2.

5.8 Use cases

Each network topology is better suited for particular scenarios, and it is not possible to have a "*one size fits all*" network. Which is why the final section of this chapter explains where Open LoRa Mesh is better suited and where it will not perform as well.

5.8.1 Mobile network

Mobile networks, such as bike sharing and other shared transportation environment, require high speed for adjusting. These are dynamic environments where speed is important, especially for messages exchanged in the control plane, as explained for VANETs in Section 4.1.

Open LoRa Mesh would not be well suited for such type of networks, this is because the "*raw maximum data rate of 27 kbps*" [42] and, when moving at high speed, devices can become very sensitive to Doppler effect [43].

5.8.2 Fixed network

This is the type of network Open LoRa Mesh is better suited in. Although MegaSense works with mobile sensors, Open LoRa Mesh would allow it to work with monitoring sensors spread in a defined area. This could either be inside or outside a a building.

Such type of network works best since it is mostly static and connections are rarely broken, thus data that needs to be sent in the control plane does not exceed the one sent in the data plane, as it can happen in the previous scenario.

5.8.3 Hybrid network

Such scenario includes, for example, the integration of mobile sensors given to users with a network of fixed sensors in an area. In this case, the usability of Open LoRa Mesh depends on the number of mobile devices that need connectivity and how many times they connect and disconnect.

There's no reason to have a plan B
because it distracts from plan A.

Will Smith

6 Experimental results

This chapter expands the previous one by verifying the usability of Open LoRa Mesh. Particularly, it describes four experiments in which Open LoRa Mesh has been set up and tested, in order to record its rough performances.The following sections describe such experiments done with the available hardware in two particular locations: inside a building and in an open field.

The first location is *Archimedes Tower* [1] (*Torre Archimede* in Italian), the building which houses the department of mathematics and computer science at the University of Padova, while the second place is the *Mazzini Square* [2] (*Piazza Mazzini* in Italian) in Monselice, a city in the south of Padova.

These places have been chosen especially for their opposite characteristics: Archimedes Tower, in Figure 6.1, is a 31m tall building which develops on seven levels above the ground and three underground levels[3], while Mazzini Square, in Figure 6.2, is an open space that allows an unobstructed view between nodes.

An important remark that has to be done before describing these experiments is that they are not exhaustive. When thinking about the amount of possibilities in which nodes can be arranged, it is certain that more tests could have been executed. Such kind of system test-

[1] For simplicity, pictures representing this location are taken from a publicly available online 3D model (https://3dwarehouse.sketchup.com/model/a4d327dd45162ee0557d96e459f19076/)

[2] Images of this location in this thesis are taken from Google Maps and OpenStreetMap

[3] https://phaidra.cab.unipd.it/view/o:10783

ing has to take in consideration also the surroundings of devices, since they communicate wirelessly.

Figure 6.1: 3D model of Archimedes Tower

Figure 6.2: View from above of Mazzini Square

6.1 Board placement

For these experiments, all four boards have been used to test the functionalities of the network. Placement of the boards is represented for both locations, respectively in Figure 6.3 for Archimedes Tower and in Figure 6.4 for Mazzini Square.

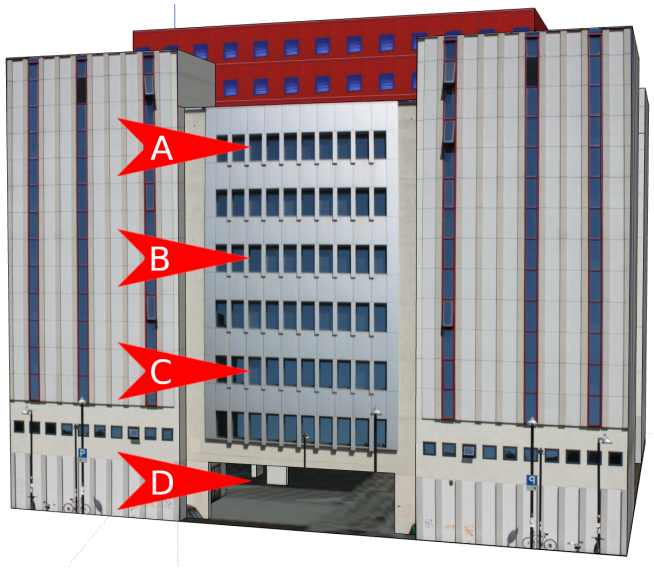


Figure 6.3: Placement of boards in Archimedes Tower

Regarding Archimedes Tower, boards have been placed so that board ***A*** is on the 6^{th} floor, ***B*** is on the 4^{th} floor, ***C*** is on the 2^{nd} floor, ***D*** is on the first underground level. All boards have been placed with the antenna facing the internal courtyard of the building.

For Mazzini Square, on the other hand, boards have been placed so that they are directly visible to each other, almost one for each corner of the square.

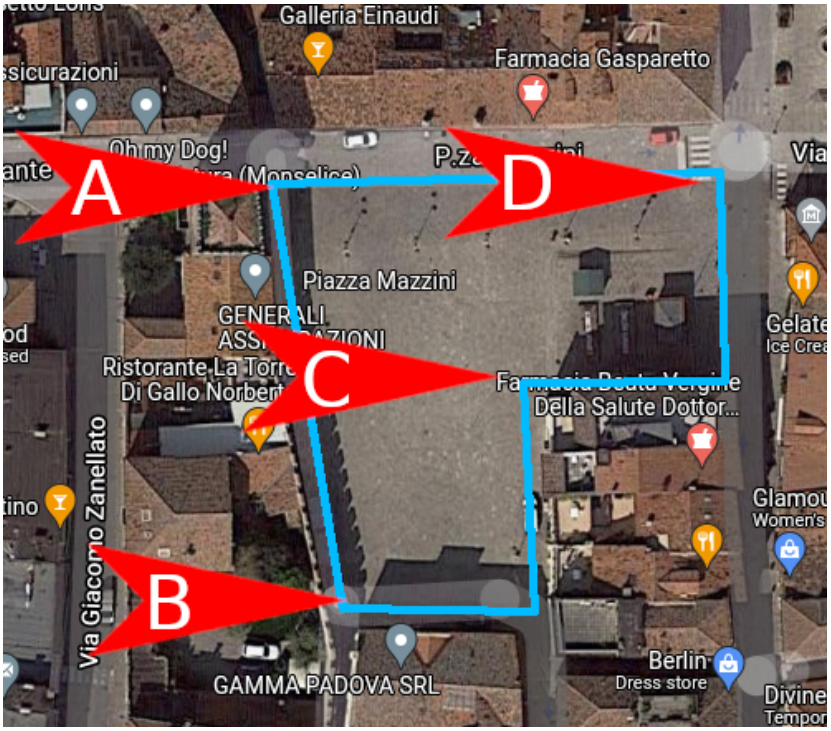


Figure 6.4: Placement of boards in Mazzini Square

These placements have been made so that boards do not have direct obstructions that can block the signal in between them. Such placement can be considered ideal because nodes are not far from one another, and their signal does not lose much power.

In both scenarios boards have been tested as if they were part of a network of fixed devices, which, as explained in Section 5.8.2, is the best case scenario for Open LoRa Mesh.

6.2 Experiments

Following there are the four experiments done to test the usability of the network: *message range*, *mesh creation*, *node addition* and *node removal*. These sections contain a description on how these experiments have been executed and the results achieved.

6.2.1 Message range

This first experiment is intended to understand nodes' range and how they can be placed in order to achieve a strongly connected network.

In the case of Archimedes Tower, this test has shown the weakness of LoRa when devices are placed inside a building. Considering the placement of nodes as represented in Figure 6.3, various cases have been examined.

To test the transmission range of the nodes in the worst-case scenario, at first device ***D*** has been booted on the library's floor. After booting up, the device listens for around 180 seconds for any mesh advertisement messages, as explained in Section 5.3.1.5, but since there are no other nodes on it will create a new mesh with the specified identifier. Subsequently, device ***A*** was the second one started, but, due to the number of concrete layers in between these devices, not all packets were correctly exchanged in between these two nodes. This has caused device ***A*** not being able to exchange data with node ***D*** and to be shut down so that a better placed node could be booted up. Node ***B***, placed on the 4^{th} floor, showed better results when exchanging mesh data, although some messages were still lost when communicating with node ***D***.

The best placement of nodes in Archimedes Tower is thus represented by the placement in Figure 6.1. Nodes have shown a strong connection when placed roughly two levels apart and booted up in order, allowing the mesh to be created without any packet loss. FiPys have been booted in alphabetical order and two minutes apart from each other, so the first one would have time to generate the mesh, and the others would be able to join it. The creation of a mesh and the addition of a node are further described in Section 6.2.2.

Contrasting with the thick walls of Archimedes Tower, Piazza Mazzini is an open space where no obstructions are present. In this scenario, whatever is the order in which devices are booted, all of them have a clear line of sight with one another. This resulted in a network where nodes have a strong connection with one another and do not require any particular though on antenna placement, as long as these are facing the inside of the square.

Such differences in results are the reason why places so divergent have been chosen for testing the mesh. They show how space around nodes influences the network and that it is important to study an ideal placement that can allow the mesh to function correctly and with as little packet loss as possible. A scenario similar to Mazzini Square can be an orchard, where each node is strategically placed to monitor part of the field and relates data on air measurements, since for every FiPy there should be a MegaSense paired.

6.2.2 Mesh creation and node addition

As long as nodes are in reach of one another, the creation of the mesh and the addition of a node are quite straightforward. When the first node boots, it needs around 180 seconds to go beyond the initial listening phase, since there are node preexisting networks.

In Archimedes Tower's case, by booting devices in alphabetical order, or reverse, the mesh would create without any issues since, as stated in the previous section, FiPy's range would not be affected as much when devices are separated by two floors.

Creating a mesh in Mazzini Square produced similar results compared to Archimedes Tower. Since here devices' range is not obstructed, only the time in mesh creation is considered.

In both scenarios, the minimum time to correctly initiate the mesh network has proven to be around 5 minutes, considering the first node needs 3 of them to create the mesh and then booting the others 30 seconds apart after.

Node addition can be tested while testing the correct creation of the mesh network. Booting up node by node, it can be seen that the routing table correctly creates for each device. Environment, as previously mentioned, has to be taken in consideration. This is because a strong reception is required in between nodes.

6.2.3 Node removal

This experiment tests the robustness of the network when a node is abruptly removed. Results of such test diverge, according to the environment.

In Archimedes Tower, the removal of either of the external nodes results in a mesh that keeps functioning and allows messages to be exchange in between the remaining devices. The mesh heals by recreating the routing table and removing the device that was suddenly shut down.

If the device shut down is one in the middle of the building, the one on floor 4 for example, the mesh will have a harder time to recreate, given the previously talked about connectivity problem of nodes in such environment.

In Mazzini Square the removal of a node resembles the first case described for Archimedes Tower. Since devices have a clear line of sight, the mesh does not have any problems to heal.

6.3 About the results

In order to achieve an optimal connectivity in Archimedes Tower, nodes would need to be placed near the window on each of the aforementioned floors (-1, 2, 4, 6). Placing them near the window, either facing externally or on the internal courtyard, would allow them to have a clear visual: a stronger connection could then be achieved in between nodes ***A*** and ***D***.

The same approach applies to Mazzini Square, where devices have been placed on almost each corner of the square, thus being outside. If they where to be placed inside of the buildings facing the square, antennas would need to be near windows or outside.

Placing the devices inside of Archimedes Tower means collecting air quality information in the building, possibly in the staircase, or in classrooms or offices. This allows the MegaSense device, or other sensors that could attach to the network, to record this data and create o model of air quality inside the building.

On the other hand, placing the FiPys in Mazzini Square would allow MegaSense devices to collect data on air quality in regarding a large open area. If thoroughly recorded, data from this area could be an interesting start for a research on pollutants of cement plants, given that Monselice has two of them.

By booting multiple nodes at the once, the random time added to the initial 180 seconds in which each node is listening, prevents devices to create separate multiple meshes.

6.4 Other possible experiments

Nothing is ever perfect, thus it is important to keep improving and test the work done. The aforementioned tests are not conclusive to declare Open LoRa Mesh ready for a "*production environment*", where nodes can rely on the structure of the network to send their data. These tests only show that Open LoRa Mesh is a valid starting point to build upon and provide an interesting possibility in using LoRa as a transmission method. Also, as said earlier, the placement of nodes in the test can be considered ideal when thinking about signal strength and reach of LoRa antennas.

A consideration on how other transmission methods would work compared to LoRa is talked about in Section 7.1.3.

Be fearful when others are greedy,
and greedy when others are fearful.

Warren Buffet

7 Conclusions

This thesis contains the description of a mesh network, Open LoRa Mesh, of low powered devices in which nodes communicate using LoRa technology. Such solution has been thought to work with IoT devices, particularly with an air quality sensing device: MegaSense, described in Section 2.2.2.

It is important to understanding characteristics of pollutants in urban environments for counteracting problems linked to poor air quality and for assessing the effectiveness of initiatives designed for tackling it. Open LoRa Mesh helps in the placement of devices across a larger space where connectivity might be an issue.

As said by authors of [34], mesh networks are "*decentralized, easy to deploy, and characterized by dynamic self-organization, self-configuration, and self-healing properties*". Such properties enable fast deployment, low installation cost, and reliable communication.

Implementation of Open LoRa Mesh has been done in Micropython for Pycom hardware, particularly it has been programmed for the FiPy board. Software is available on GitHub under the repository *Open LoRa Mesh* [1].

This conclusive chapter summarizes the previously discussed arguments in this thesis, presents possible improvements, both hardware and software, and contains some personal considerations by the author.

[1] www.github.com/cipz/OpenLoRaMesh

7.1 Future work

Open LoRa Mesh can be considered as a proof of concept that shows LoRa's validity for creating a network which allows devices like MegaSense to communicate and exchange useful data, like on updating calibration. However, there are various aspects of this project that can be better improved, both from a hardware and a software point of view.

7.1.1 Hardware improvements

In this project, the hardware that composes each node is quite straightforward: a FiPy device with an expansion board, an antennas for LoRa communication and, where needed, an antenna for GPS and, last but not least, the MegaSense.

A possible improvement could be adding a hard 3D printed shell to each node. This would contain the FiPy and the Megasense, with openings for antennas and MegaSense's air sensors.

If functionalities such as the SD card and the GPS antenna are not needed, also if nodes have already been flashed with the software and do not need the micro USB port, shields can be removed. Without the shield, FiPys can be powered by either a battery or a direct power source that connects to a 5V wall charger.

About powering the device, an external battery, either LiPo (Lithium Polymer) or alkaline cell, can be added when devices are placed in locations harder to reach. Upon these, battery sensors, such as the ones in the MegaSense, can be added to evaluate the battery charge and communicate when they are almost out of energy.

7.1.2 Software improvements

The most important software improvement is on routing table, which could be populated keeping in mind the *Received Signal Strength Indicator* (RSSI) of incoming messages. With the use of this indicator a more stable network can be created, since nodes messages would be better routed through more stable connections among nodes.

Besides the `DEFAULT_BOOT` and `PLAIN_LISTENER` modes, other modes could be added to the nodes. These can include, for example, a relay station which only forwards packets and does not participate in data creation or communication with air quality sensing devices. Such nodes can strategically be placed so that nodes hard to reach, because inside a building or really far away from other nodes, can communicate with the rest of the network.

As explained in Section 5.3.1.6, there can be Exceptions that terminate the main loop, thus better Exception handling can be added to the software. This would allow for a more stable

communication between nodes when, for example, transmissions overlap and packets get corrupted. Retransmission backoff methods, for a more controlled system, can be added as a possible solution.

Another factor that would improve the performance of each node on the network is the implementation of a sleep mode, Section 5.7. This would allow each node to have a longer battery life and only activate when necessary. Although adding such mode requires more data and in depth testing, it is not an impossible upgrade.

7.1.3 On LoRa and other transmission technologies

LoRa is a promising transmission technologies, but it also comes with some disadvantages, such as the low data rate and high transmission time. There are other technologies that could be used for a project such as this one, particularly NB-IoT and 5G would allow for transferring higher amounts of data.

They would not allow thought to have a mesh network since they rely on the cellular technology, thus need a cellular tower and a SIM card. Wi-Fi, on the other hand, does not supply a substantial range for these devices to work, as proved by the project described in Section 2.2.1. In this case, LoRa is the most logical choice since it allows devices to rely on themselves.

7.2 Personal considerations

This final section of the thesis contains some personal considerations from the author regarding the work with Pycom boards and mesh networks.

7.2.1 About the Pycom boards

Compared to working with other boards, such as the Arduino, Pycom boards are more complete, since they offer better specifications, for both computing power and memory, connectivity and support. For prototyping, Arduino boards offer very good opportunities and have been on the market for a longer time, allowing consumers to get acquainted and know the brand. Sometimes, working with Arduino feels like "*everything that can be invented has been invented*", as said by Charles H. Duell, since there is so many people who have made all kind of projects and are sharing them online.

The FiPy, on the other hand, offers more connectivity and more power, at the expense of the open source advantages of most of its competitors. As explained in Section 3.3.3, besides

keeping the blueprints of their boards private, Pycom also has a proprietary firmware on their boards, which sometimes gets in the way of developers that require to fully use the hardware capabilities. This also reflects on the software, where some functionalities that are functioning on other boards are not implemented in Pycom's firmware for their devices. Nonetheless, the official forum[2] available for Pycom's boards is full of useful comments and the users are fairly active.

An example where the FiPy would offer a great improvement over an Arduino board would be this smart home application[3]. Such project contains an Arduino Mega board with various sensors and relays, all connected via a specifically build website, with the use of a third-party API handler for the Arduino. The Arduino connects to the Internet via an Ethernet shield, thus sends data by cable. A FiPy would allow a smaller footprint and no need for a shield, besides avoiding the use of a third-party API handler.

In my personal opinion, these boards are quite powerful and resilient for research purposes. They can be used as multiple testbenches for different projects, but might result excessive if their powerful specifications are not necessary.

7.2.2 About mesh networking

As said in Chapter 3, each network topology is better suited for particular scenarios. It is not possible to have a "*one size fits all*" network for all possible applications that need to exchange data. As for all questions in computer science, the answer to the question where a mesh network is useful is "*it depends*".

Mesh networking is a very interesting network topology which, compared to all the others in Figure 3.3, has still a lot of research possibilities. In conjunction with AI and machine learning, it is possible to achieve an optimal level of connectivity in such networks. The bottlenecks would then be represented by the computing power in the nodes, and in the algorithm used in case one of the nodes fail, or new nodes are added into the network.

Scalability of such networks, as spoken about in Section 4.2, is a very interesting research scenario since it is one of the major deciding elements for any new networking technology to be accepted. Factors such as multi-hopping, protocol overhead, and wireless interference limit the scalability of such networks. Mesh networks will be more and more common, but I believe they will be used in conjunctions with other topologies: for example a local mesh

[2] https://forum.pycom.io/

[3] www.instructables.com/Smart-Home-With-Arduino-Ethernet-Shield-and-Teledu

network that allows communication between sensors in a house and a gateway node interacts with other networks in a different topology.

Security is another important factor when considering in mesh networking. Since each node receives and analyzes packages, there is no totally effective solution for a secure network. Adding security features such as encryption/decryption algorithms and other checks, would result in a big overhead on small IoT devices. Also, like other radio technologies, anyone can read messages, encrypted or not, "*flying*" through the air.

Acknowledgments

At the beginning of this thesis, instead of having a dedication, I choose to have the meaning of a word which I believe describes me the best: *cosmopolitan*. This word means a lot to me because I love traveling and meeting new people, getting involved in international projects and exchanges. That is how I got to meet some of the most interesting people I know, and why thanks to them, I want to expand my horizon even more and seek out new experiences.

Networking for me does not only mean connecting computers, but getting to know new people, either in a hot Finnish sauna in Helsinki, at a pub in Dublin, or at a convention center in Milan.

I would like to thank my supervisors for their continued support and encouragement: Prof. Claudio Palazzi, from Università degli Studi di Padova, and Prof. Pietro Manzoni, from Universitat Politècnica de València. Also thanks to Samu Varjonen, Ashwin Rao, Andrew Rebeiro Hargrave and Naser Hossein Motlagh, who are Professors and Researchers at the University of Helsinki. They have provided me with the MegaSense device and part of the hardware to develop this project.

I offer my sincere appreciation for the learning opportunities provided by all of them, both as teachers, in Padova and Helsinki, and as supervisors.

"*Perfect is the enemy of good*", and this thesis is surely not perfect, but I would like to consider it as a stepping stone for what comes next. It has made me dive into a broader view on LoRa communication and has made me connect with people that otherwise I would have not had a chance to interact with.

References

[1] Cisco, "*Cisco Annual Internet Report (2018–2023) White Paper*," *Cisco Annual Internet Report*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html No citations.

[2] Forbes, "*Intelligent World: the state of IoT*," *Forbes insight*. [Online]. Available: https://www.hitachivantara.com/en-us/pdfd/ebook/intelligent-world-state-of-iot-forbes-insights-ebook.pdf No citations.

[3] R. E. Munn, "*The Design of Air Quality Monitoring Networks*," *Palgrave Macmillan, London*, 1981. No citations.

[4] K. Ashton, "*That 'Internet of Things' Thing*." [Online]. Available: https://www.rfidjournal.com/that-internet-of-things-thing No citations.

[5] A. Bujari and O. Gaggi, "*A mobile sensing and visualization platform for environmental data*," *Pervasive and Mobile Computing*, vol. 66, p. 101204, 06 2020. No citations.

[6] M. Roccetti, M. Gerla, C. E. Palazzi, S. Ferretti, and G. Pau, "*First Responders' Crystal Ball: How to Scry the Emergency from a Remote Vehicle*," pp. 556–561, 2007. No citations.

[7] N. J. Woodland and S. Bernard, "*Classifying apparatus and method*," no. US2612994A. [Online]. Available: https://patents.google.com/patent/US2612994A/en No citations.

[8] D. Savir and G. J. Laurer, "*The characteristics and decodability of the Universal Product Code symbol*," *IBM Systems Journal*, vol. 14, no. 1, pp. 16–34, 1975. No citations.

[9] Verizon, "*Verizon Data Breach Digest*," *Verizon Data Breach Digest*. [Online]. Available: https://enterprise.verizon.com/resources/reports/data-breach-digest-2017-perspective-is-reality.pdf No citations.

[10] W. E. Zhang, Q. Z. Sheng, A. Mahmood, D. H. Tran, M. Zaib, S. A. Hamad, A. Aljubairy, A. A. F. Alhazmi, S. Sagar, and C. Ma, "*The 10 Research Topics in the Internet of Things*," pp. 34–43, 2020. No citations.

[11] GSMA, "*Internet of things in the 5G era - Opportunities and Benefits for Enterprises and Consumers*," *GSMA IoT Report*. [Online]. Available: https://www.gsma.com/iot/resources/iot-5g-ltem-nbiot-opportunities-benefits/ No citations.

[12] S. Ali and M. Ghazal, "*Real-time Heart Attack Mobile Detection Service (RHAMDS): An IoT use case for Software Defined Networks*," pp. 1–6, 2017. No citations.

[13] C. Galloway, A. Valys, F. Petterson, V. Gundotra, D. Treiman, D. Albert, J. Dillon, Z. Attia, and P. Friedman, "*NON-INVASIVE DETECTION OF HYPERKALEMIA WITH A SMARTPHONE ELECTROCARDIOGRAM AND ARTIFICIAL INTELLIGENCE*," *Journal of the American College of Cardiology*, vol. 71, p. A272, 03 2018. No citations.

[14] L. Garg, E. Chukwu, N. Nasser, C. Chakraborty, and G. Garg, "*Anonymity Preserving IoT-Based COVID-19 and Other Infectious Disease Contact Tracing Model*," *IEEE Access*, vol. 8, pp. 159 402–159 414, 2020. No citations.

[15] M. Agiwal, A. Roy, and N. Saxena, "*Next Generation 5G Wireless Networks: A Comprehensive Survey*," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016. No citations.

[16] T. Economist, "*The IoT Business Index 2020: a step change in adoption*," *The Economist, Intelligence Unit*. [Online]. Available: https://armkeil.blob.core.windows.net/developer/Files/pdf/report/economist-iot-business-index-2020-arm.pdf No citations.

[17] F. Concas, J. Mineraud, E. Lagerspetz, S. Varjonen, X. Liu, K. Puolamäki, P. Nurmi, and S. Tarkoma, "*Low-Cost Outdoor Air Quality Monitoring and Sensor Calibration: A Survey and Critical Analysis*," vol. 17, no. 2, May 2021. No citations.

[18] S. C. Voinea, A. Bujari, and C. E. Palazzi, "*Air Quality Control through Bike Sharing Fleets*," pp. 1–4, 2020. No citations.

[19] S. Hu, C. Xiong, Z. Liu, and L. Zhang, "*Examining spatiotemporal changing patterns of bike-sharing usage during COVID-19 pandemic*," *Journal of Transport Geography*, vol. 91, p. 102997, 2021. No citations.

[20] A. Rebeiro-Hargrave, N. H. Motlagh, S. Varjonen, E. Lagerspetz, P. Nurmi, and S. Tarkoma, "*MegaSense: Cyber-Physical System for Real-time Urban Air Quality Monitoring*," pp. 1–6, 2020. No citations.

[21] N. H. Motlagh, E. Lagerspetz, P. Nurmi, X. Li, S. Varjonen, J. Mineraud, M. Siekkinen, A. Rebeiro-Hargrave, T. Hussein, T. Petaja, M. Kulmala, and S. Tarkoma, "*Toward Massive Scale Air Quality Monitoring*," *IEEE Communications Magazine*, vol. 58, no. 2, pp. 54–59, 2020. No citations.

[22] P. E. Green, "*Computer network architectures and protocols / edited by Paul E. Green, Jr.*," pp. xvii, 718 p. :, 1982. No citations.

[23] L. t. Parziale, "*TCP/IP Tutorial and Technical Overview*," *IBM Redbooks*. No citations.

[24] P. K. Bondyopadhyay, "*Guglielmo Marconi - The father of long distance radio communication - An engineer's tribute*," vol. 2, pp. 879–885, 1995. No citations.

[25] B. Chaudhari, M. Zennaro, and S. Borkar, "*LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations*," *Future Internet*, vol. 12, p. 46, 03 2020. No citations.

[26] H.-C. Lee and K.-H. Ke, "*Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation*," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 9, pp. 2177–2187, 2018. No citations.

[27] "*Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions).*" [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ No citations.

[28] P. Xu, Z. Ding, X. Dai, and H. V. Poor, "*NOMA: An Information Theoretic Perspective*," 04 2015. No citations.

[29] "*Wireless Mesh Networks*," 2009. No citations.

[30] Y. Liu, K.-F. Tong, X. Qiu, Y. Liu, and X. Ding, "*Wireless Mesh Networks in IoT networks*," pp. 183–185, 2017. No citations.

[31] Y. Yi, G. de Veciana, and S. Shakkottai, "*Learning contention patterns and adapting to load/topology changes in a MAC scheduling algorithm*," pp. 23–32, 2006. No citations.

[32] V. Bahl, R. Chandra, P. Lee, V. Mishra, J. Padhye, D. Rubenstein, and Y. Yu, "*Opportunistic Use of Client Repeaters to Improve Performance of WLANs*," *IEEE/ACM Transaction on Networking*, vol. 17, no. 4, pp. 1160–1171, August 2009. [Online]. Available: https://www.microsoft.com/en-us/research/publication/opportunistic-use-of-client-repeaters-to-improve-performance-of-wlans-3/ No citations.

[33] T. O. Olwal, K. Djouani, B. J. V. Wyk, Y. Hamam, and P. Siarry, "*Optimal Control of Transmission Power Management in Wireless Backbone Mesh Networks*," 2011. [Online]. Available: https://doi.org/10.5772/12993 No citations.

[34] S. Sampaio, P. Souto, and F. Vasques, "*A review of scalability and topological stability issues in IEEE 802.11s wireless mesh networks deployments*," *International Journal of Communication Systems*, vol. 29, 01 2015. No citations.

[35] S. Misra, S. Misra, and I. Woungang, "*Guide to Wireless Mesh Networks*," *Guide to Wireless Mesh Networks*, 01 2009. No citations.

[36] P. Gupta and P. Kumar, "*The capacity of wireless networks*," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000. No citations.

[37] R. Ramanathan, R. Allan, P. Basu, J. Feinberg, G. Jakllari, V. Kawadia, S. Loos, J. Redi, C. Santivanez, and J. Freebersyser, "*Scalability of Mobile Ad Hoc Networks: Theory vs practice*," pp. 493–498, 2010. No citations.

[38] M. S. Aslam, A. Khan, A. Atif, S. A. Hassan, A. Mahmood, H. K. Qureshi, and M. Gidlund, "*Exploring Multi-Hop LoRa for Green Smart Cities*," *IEEE Network*, vol. 34, no. 2, pp. 225–231, 2020. No citations.

[39] A. Abrardo, A. Fort, E. Landi, M. Mugnaini, E. Panzardi, and A. Pozzebon, "*Black Powder Flow Monitoring in Pipelines by Means of Multi-Hop LoRa Networks*," pp. 312–316, 2019. No citations.

[40] K. C. V. G. Macaraeg, C. A. G. Hilario, and C. D. C. Ambatali, "*LoRa-based Mesh Network for Off-grid Emergency Communications*," pp. 1–4, 2020. No citations.

[41] K. Nakamura, P. Manzoni, M. Zennaro, J.-C. Cano, and C. T. Calafate, "*LoRaCTP: a LoRa based Content Transfer Protocol for sustainable edge computing*," pp. 539–545, 2020. No citations.

[42] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "*Understanding the Limits of LoRaWAN*," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017. No citations.

[43] T. Ameloot, M. Moeneclaey, P. Van Torre, and H. Rogier, "*Characterizing the Impact of Doppler Effects on Body-Centric LoRa Links with SDR*," *Sensors*, vol. 21, no. 12, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/12/4049 No citations.