FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

IoT na gestão remota de centrais de deteção de incêndio

Daniel Henrique Quintas Pereira

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Carlos João Ramos

September 15, 2022

Resumo

Os sistemas de deteção de fogo modernos incluem detetores, equipamentos de segurança e painéis de controlo simples e intuitivos que se encontram em constante evolução para melhorar o tempo de resposta em situações de emergência, beneficiando os utilizadores e os seus bens.

Geralmente, os utilizadores assumem que o sistema de deteção funciona de acordo com o esperado, no entanto, são necessários testes de manutenção regulares para evitar falhas de deteção em momentos de emergência. A gestão de falhas é um processo moroso e que pode desperdiçar recursos para efetuar este tipo de serviços.

A evolução de microcontroladores e telecomunicações permitiram a existência de novos conceitos tecnológicos tais como IoT (Internet of Things). O desenvolvimento de tecnologias IoT abrange diversas aplicações em diferentes ambientes, incluindo casas, escritórios, industrias entre outros estabelecimentos.

Neste sentido, é proposta uma solução baseada num sistema IoT que disponibiliza uma partilha de dados mais minuciosa relativamente aos estados de uma central de deteção de incêndio, como falhas de qualquer tipo e incêndios. O sistema é baseado no desenvolvimento de um modulo IoT incorporado numa central de deteção de incêndio que permanecerá conectado à internet.

O dispositivo IoT será capaz de comunicar com uma interface visual intuitiva desenvolvida e disponibilizada ao fornecedor. Desta forma será possível monitorizar à distancia por notificações recebidas pelo modulo IoT, os diferentes estados de uma central de deteção de incêndio. A comunicação com a interface é estabelecida preferencialmente por uma rede local de reduzida latência. Para além disso, no caso desta rede não estiver disponível o modulo poderá ainda comunicar através de uma rede de grande alcance.

O produto desenvolvido demonstrou como é possível obter uma comunicação rápida e flexivel num sistema IoT, introduzindo uma melhor gestão entre a central e o fornecedor para potencialmente, e de forma eficaz, reduzir os tempos de resposta numa situação de emergência ou falha.

Abstract

Modern fire detection systems composed of fire detectors, safety equipment and simple control panels are constantly evolving in order to improve the response time in case of an emergency to benefit people and their assets.

Property owners usually assume that these devices work properly but regular testing and maintenance is required to ensure protection against fire induced tragedies. This kind of management is usually best provided by the supplier, but it takes time and resources to make those kinds of services.

The microcontroller and telecommunication technology evolution has enabled new concepts like IoT (Internet of Things). The development of this technology is included in many applications and in different environments, including households, workspaces, industry buildings, and other establishments.

This paper suggests an IoT-based system, that enables a more spontaneous sharing of data regarding the fire detection system and thus a better surveillance of malfunctions or emergencies by the supplier. The proposal consists in the development of an online device, integrated with the fire detection system.

The IoT device will be able to communicate with a developed intuitive user interface available only to the supplier. This will enable remote monitoring of the fire detection system, through notifications on the state of a fire detection central. The IoT communication with the developed interface is made, preferentially, through a local network with low latency. Moreover, if the local network is not available then the IoT module may still communicate through a wide area network.

The developed product has demonstrated how it is possible to obtain an IoT system with fast and flexible communication, introducing improved management of the fire detection central by the supplier, to potentially and effectively reduce the response times for fault detection and fire emergencies.

Acknowledgments

First I'd like to manifest my gratitude towards my supervisors, Professor Carlos João Ramos as well as Engenheiro João Rodrigues for giving me the support needed.

I would like to thank my family, specially my mother for all the love and continuous support throughout the dissertation process.

Finally, to my friends that have been there for me I'd also like to sincerely thank you for supporting me.

Daniel Pereira

"You may not control all the events that happen to you,
but you can decide not to be reduced by them."

Maya Angelou

Contents

List of Figures

List of Tables

Abbreviations

3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
A	Ampere
AC	Alternating Current
ADC	Analog to Digital Converter
ACK	Acknowledgement
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ARP	Address Resolution Protocol
AT	Attention
BIOS	Basic Input/Output System
BLE	Bluetooth Low Energy
bps	bits per second
CoAP	Constrained Application Protocol
CON	Confirmable
CoRE	Constrained RESTful Environments
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
CTP	Collection Tree Protocol
DDOS	Distributed Denial of Service
DDS	Denial of Service
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSL	Digital Subscriber Line
DTLS	Datagram Transport Layer Security
EC-GSM-IoT	Extended Coverage GSM IoT
EDA	Electronic Design Automation
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMI	Electro Magnetic Interference
eMTC	enhanced Machine-type Communication
EPROM	Erasable Programmable Read-Only Memory
ETX	Expected Transmission Count
EXI	Efficient XML Interchange
FeRAM/FRAM	Ferroelectric RAM
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GPIO	General Pin Input-Output
GSM	Global System for Mobile communication
GSMA	GSM Association

GUI	Graphical User Interface
HCI	Human-Computer Interaction
HD	High Definition
HDMI	High Definition Multimedia Interface
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
IBM	International Business Machines
ICMP	Internet Control Message Protocol
ID	IDentification
IDE	Integrated Development Environment
IDF	IoT Development Framework
ID-6lowpan-hc	6LoWPAN Header Compression
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IoT	Internet of Things
IP	Internet Protocol
IPV4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISM	Industrial, Scientific, and Medical
JID	Jaber ID
JSON	JavaScript Object Notation
LED	Light Emitting Diode
LoRa	Long Range Wide Area
LPWAN	Low-power WAN
LSB	Least Significant Bit/Byte
LTE	Long Term Evolution
LTE Cat-M1	Long Term Evolution (4G) category M1
LTE-M	LTE Cat-M1
lwIP	Lightweight Internet Protocol
M2M	Machine-to-Machine
MAC	Media Access Control
MBS	Most Significant Bit/Byte
mIP	Micro IP
MQTT	Message Queuing Telemetry Transport
MQTT-SN	Message Queuing Telemetry Transport for Sensor Networks
NB-IoT	Narrow Band IoT
NFC	Near Field Communication
NFPA	National Fire Protection Association
NON	Non-confirmable
OASIS	Organization for the Advancement of Structured Information Standards
OTP	One Time Programmable
OWL	Web Ontology Language
PCB	Printed Circuit Board
PHP	Hypertext Preprocessor
PMI	Purchasing Managers Index
PPPoE	Point-to-Point Protocol over Ethernet
PPPoS	Point-to-Point Protocol over Serial
PWM	Pulse Width Modulation

QoS	Quality of Service
RAM	Random Access Memory
RC	Resistor-Capacitor
RDF	Resource Description Framework
REST	Representational State Transfer
RFID	Radio Frequency Identification
RJ35	Registered Jack-45
ROLL	Routing Over Low Power and Lossy Networks
ROM	Read-Only Memory
RST	Reset
RX	Receiver
SASL	Simple Authentication and Security Layer
SCL	Serial Clock Line
SD	Solid State Disk
SDA	Serial Data Line)
SHF	Super High Frequency
SIG	Special Interest Group
SIM	Subscriber Identity Module
SMTP	Simple Mail Transfer Protocol
SSID	Service Set Identifier
SSL	Secure Sockets Layer
TCP	Transmission Control Protoco
TLS	Transport Layer Security
TV	Television
TX	Transmitter
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
UNB	Ultra-Narrow-Band
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UV	Ultraviolet
V	Volt
VAC	AC Voltage
VDC	DC Voltage
VE	Virtual Environment
VHDL	Very High-Density Logic
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPA	Area Network
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WWAN	Wireless Wide Area Network
XCTP	eXtend Collection Tree Protocol
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Context

Detectors in fire detection systems are usually triggered by many occurrences like smoke, flames and heat. The control panel receives signals from these detectors and then enables audio-visual or sound fire alarm gadgets. These detectors, like any device, can run into many problems during their operation. One of the most common problems in these detectors are battery leaks, dust build up, environment issues and ground faults. All of these problems are not always apparent to the consumer.

NIBBLE has proposed, for this thesis, an IoT (Internet of Things) based management system that adds another layer of safety to traditional fire detection systems.

1.2 Motivation

As much prevention and safety measures homes or retail have, fires are impossible to avoid completely. However, repercussions can be highly diminished if an effective and fast response is implemented in the emergency system. This is why it makes sense to adopt an IoT (Internet of Things) system that can monitor and intervene remotely even when people are not in the property. Additionally, the level of service each company can deliver to the client becomes an important factor to standout in the marketplace.

1.3 Objective

The project consists in the development of a small IoT module that can be placed within any fire control system to keep track of any signal available. The most common signals are the state of the power supply grid, output of the fire alarm siren and the output of malfunction state. Additionally the battery of the IoT module will also be monitored. The IoT module is directly connected to the fire detection system pins. To monitor the signals previously mentioned our module will have input readings on existing relays or voltage signals from the output pins of a fire control system.

The module is responsible for communicating with a dashboard, that will allow the reading of a fire detection system monitored signals, and display their state through an intuitive graphical user interface.

Wi-Fi was suggested, by the company, as an approach to communicate with the dashboard but it was also encouraged to study other alternatives for local networks in Chapter 2 and verify what technology makes more sense in this project, taking into account power shortages, communication reliability and data rate.

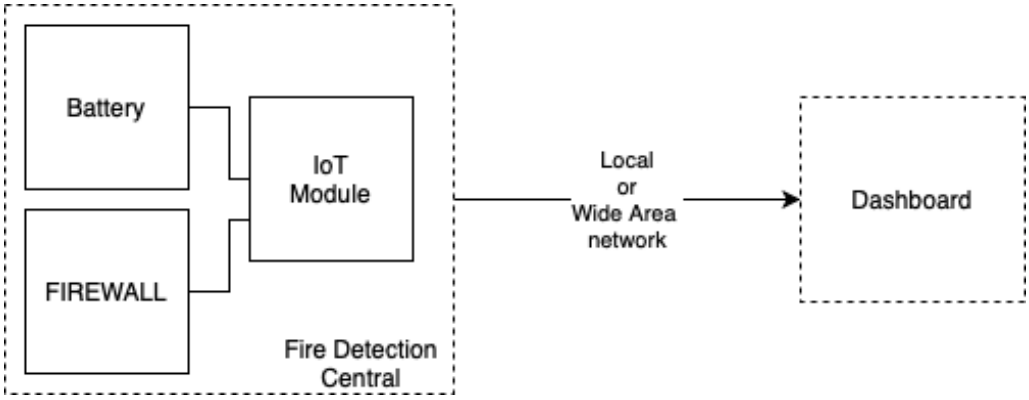


Figure 1.1: Proposed system and possible communication technologies.

1.4 Document Structure

For this thesis report the first chapter presents some context and motivation for the IoT solution as well as the main objective of the proposed thesis.

The second chapter will introduce some theoretical aspects about technologies, protocols and alternatives for IoT applications, review and compare some bibliographic papers that present adequate solutions for management with IoT.

The third chapter will encapsulate the IoT system overall functionality and will explain the project's implementation.

The fourth chapter will present testing and validation of our module.

Finally, the last chapter in this thesis report will highlight the main conclusions and results of this project.

Chapter 2

State of Art

An IoT management solution requires a good knowledge about the benefits and pitfalls of network technologies and protocols. Therefore in this chapter we will explore in more detail these topics and review research that was previously done related to Internet of Things (IoT) more specifically in the scope of our project. The following sections will cover data regarding fire hazards, emergency faults and response times as well a background in IoT technology, software and development environments for IoT, components for IoT monitoring systems, an introduction to microcontrollers and a discussion on communication protocols. The state of art depends on the gathering of conference papers and journal articles distributed in electronic databases like IEEE Xplore and Scopus. Grey Literature comprising of non-commercial articles, technical reports, among others, likewise upheld the research process.

2.1 Fire emergencies

Fire has always been the landmark for human evolution and also one of it's greatest threat. Forest and urban fires happen daily, hence the need to minimize the loss of property assets and lives. The latter being the most critical aspect of a fire crisis. The main causes for fire emergencies are rooted in faulty electrical grids in households as well as unsafe usage of small fires [18].

Minimizing the destructive power of a fire requires detection, an alarm for awareness, notifying the fire department and activation of security equipment as soon as possible. This way the intervention and evacuation in a fire emergency can be better handled by everyone, avoiding it's spread to neighbor facilities or, given a short response time, stop fire growth up to life threatening proportions.

Monitored fire alarm systems may be connected directly to the fire department either by phone, cellular dialer, a network connection or as an alternative, they can be connected to other reception and management alarm centrals. These automatic systems are specially relevant to alert fire departments and help firefighters achieve a desired response time to arrive at the fire location.

When discussing about fire crisis it is clear that response time is a pivotal safety measure. The USA's National Fire Protection Association (NFPA) has provided an extensive data and research

with regards to fire hazards and safety, stating that within "eight minutes, a fire is beyond the room of origin, and people are eight times as likely to die in a building fire after those eight minutes" [19].

Occupancy characteristics in buildings may lead to a manual fire detection approach rather then an automatic fire detection. The latter is better suited for low levels of occupancy in establishments where the exit may be blocked by the fire before the occupants are aware of the fire.

Since the beginning of the COVID-19 pandemic and lockdown in 2020, the number of workers within commercial buildings has decreased dramatically while working remotely began being the new norm. This shift in occupancy levels has enforced the need for automatic fire detection to supplement already existing manual detection systems and often times required by legislation in countries such as England for example [20].

In case of power failure, the emergency battery power supply is enabled for a period time that allows the normal function of the fire detection system and possible corrective interventions. The battery capacity may need to be superior to that of the required by up to 25% as the original battery capacity decreases every year.

Automatic fire detection systems may have to be monitored for any occurring fault within the detection system, fire alarms and control over safety equipment.

Management Routines

Upon implementation of the fire detection system management routines and assistance must be conducted to establish a continuous and reliable system. These routines can be handled by the owner and/or the operator every day, month, trimester or annually. They are usually meant to monitor the normal function of the system. However, some occurrences might need special assistance[21], these include:

- Any type of fire emergency;
- Continuous abnormal false alarms;
- Rearrangement of the establishment;
- Change of occupancy characteristics in monitored areas;
- Change of level of noise that might alter the alarm siren;
- Damage of any system component regardless of any malfunction;
- Any change of the assistance equipment;
- The usage of the fire detection system before a complete building setup.

The efficiency of the automatic fire detection system relies on it's periodic management routines. Failing these routines may compromise response time or even cause a full system failure which in turn, in case of a fire emergency, leads to a greater loss of assets and lives.

Beyond the scheduled routines, a remotely monitored fire detection system introduces an improved real-time and practical solution. Supplementing the system with an Internet of Things based module can aid the general public with a more meticulous management and a safer fire detection system.

2.2 Internet of Things

Physical objects are increasingly becoming part of the linked world. Machines, products, infrastructure, and devices are being outfitted with networked sensors that allow them to monitor their surroundings. These devices incorporate computing and networking capabilities that enables them to receive instructions, query their state and, if possible, change it by using embedded actuators, processors, and transceivers. Sensor-enabled gadgets can be configured to track people's health status, or establishment's emergencies like fires and trespassing, to name a few examples. IoT gadgets make use of wireless technologies as they are by and large introduced at geologically scattered areas.

Academic and relevant industrial areas have numerous definitions for the Internet of Things. When the terms 'Internet' and 'Things' are combined semantically, it refers to a global network of devices that can be uniquely addressed within a network using standard communication protocols. In this context, an IoT solution makes sense as it represents an inclusive and easy way to assist a greater scope of individuals like the old and differently abled people when it comes to the management of system appliances in their properties.

Evolving dynamics of IoT

The Internet of Things (IoT) is the result of advances in fields such as embedded systems, microelectronics, telecommunications and sensors. IoT is of great interest in academia and industry due to its potential applications in various fields of human activities.

In short, IoT is an extension of the current Internet and allows (any) everyday object to be connected to the Internet, provided it has computing and communication capabilities. By connecting to the network, objects can be remotely controlled and the objects themselves can be used as service providers. These new possibilities for everyday objects open up a wide range of opportunities for science and industry. However, these opportunities are also associated with significant technical and social risks and challenges. The Internet of Things is slowly changing the concept of data networks. In this context, the evolution of this concept over time can be traced as follows. According to Tanenbaum (2002)[22], a computer network is a collection of independent computers connected by a single technology. This connecting technology is assumed to be of different types (copper wires, optical cables, electromagnetic waves or others).

In 2011, Peterson and Davie stated that the most important characteristic of computer networks is that they are universal, i.e. based on general-purpose devices and not optimised for specific applications, such as telephone or television networks [23]. Given the many non-traditional tools and

technologies used on the Internet, the term 'computer network' is becoming somewhat obsolete.

Smart objects, as defined below, play a key role in this evolution. This is because objects have sensor-related communication and processing capabilities that change the way they are used [24].

The features on these objects will allow them to define and manage their context, exchange information between them, access online services and interact with people. At the same time, a number of new applications (e.g. smart cities, healthcare, smart homes) and challenges (regulation, security, standardisation) are emerging.

It should be noted that one of the keys to the success of IoT is the standardisation of the technology. This will increase the heterogeneity of devices connected to the Internet and make IoT a reality. It should also be noted that the most important period for the IoT sector in terms of defining the IoT infrastructure is the last months and the next few years. If we use the word "Internet" in the above concept of IoT, an analogy can be made with the Internet of today, where "things" will soon be able to communicate with each other, provide and use services, provide information and react to events [25].

A more technical analogy is that the Internet of Things can be seen as a set of protocols used in smart objects. The basic hardware modules of the IoT will end up having at least one of the following characteristics [26] [27]:

(i) A processing unit;

(ii) A memory device;

(iii)A communication module;

(iv) A sensor or actuator module.

Devices with these characteristics are called smart objects. When the objects interact with other devices, they indicate that they are connected to the network.

Motivation for maintaining the evolution of IoT

The term "Internet of Things" was first used in 1999 in the article "I Did It at Procter Gamble" [28]. At that time, the Internet of Things was associated with the use of RFID (Radio Frequency Identification) technology. Nevertheless, this notion has not been widely explored.

By 2005, wireless sensor networks (WSNs) were the most popular term for IoT (in academia and industry). These networks led to the development of methods to investigate various device constraints (e.g. memory and power), network scalability and robustness [27], as well as home and industrial automation [29] [30].

In the following years (2008-2010), the concept of the "Internet of Things" spread rapidly. This was driven by the development of WSNs and the growing prospects of IoT. In 2012, experts recognised IoT as an emerging technology [31].

In 2012, IoT was expected to be established in 5-10 years, and today academia and industry have the highest expectations of this technology. It can also be noted that when the first IoT platforms appeared, there were high expectations for their use.

New applications can be developed by connecting objects to different network resources. By connecting these objects to the Internet, the Internet of Things is created. In IoT, objects can provide a connection between users and devices. This leads to new applications such as collecting data from patients, monitoring the elderly, monitoring inaccessible and hostile environments [32]. In this scenario, the potential for new applications increases, but at the same time new challenges arise in connecting objects to the Internet with limited processing, storage, communication and energy requirements [27].

In this case, objects are clearly heterogeneous, i.e. they differ in design, features and quality. Theoretical and practical issues include considerations of ease, finding good routes and preserving limited resources at the local level. This emphasises the need to adapt existing protocols. It is also recognised that the communication and routing paradigm in smart object networks is not the same as in networks such as the Internet [33].

New challenges arise when developing new IoT applications. The data provided by objects can now be incomplete (sensor misconfigurations), inconsistent (wrong layout, outliers) and contain different types of data (from humans, physical sensors, data fusion). Therefore, applications and algorithms must be able to cope with these data issues. Another example concerns the trust in the data collected by IoT devices and how and where this data should be used in certain scenarios. Therefore, the challenges related to these new applications need to be analysed and solutions proposed to ensure that IoT meets expectations in the near future.

Some authors point out that IoT will be a new revolution in computing [28] [34] [35]. Therefore, IoT should perhaps not be seen as an end in itself, but as a means to a greater end, such as ubiquitous computing.

Basic Building Blocks of IoT

The Internet of Things can be seen as a combination of different technologies that complement each other and enable the integration of objects in the physical environment into the virtual world. IoT consists of some basic building blocks of IoT, namely:

- Identification: This is one of the most important parts, as objects need to be identified to connect to the internet. Technologies such as RFID, NFC (Near Field Communication) and IP addresses can be used to identify objects;

- Sensor and operator: Sensors collect information about the environment the object is in and then store or transmit this data to a data warehouse, cloud or storage centre. The operator can change the environment or react to the observed data;

- Communication: Refers to the various methods used to connect smart objects to each other. It also has a significant impact on the object's power consumption and is therefore a very important factor. The technologies used are Wi-Fi, Bluetooth, IEEE 802.15.4 and RFID;

- Computing: Includes computing devices such as microcontrollers and FPGAs used to implement localised algorithms for smart objects;

- Services: IoT can provide several categories of services, including identification services responsible for matching physical objects (of interest to users) and virtual objects (VEs), such as temperature values for a physical location, geographic coordinates for a sensor, and collection time; a data collection service that collects and aggregates homogeneous and/or heterogeneous data from smart objects; collaboration and information services that make decisions based on data collection services and react to scenarios; ubiquitous services to provide collaboration and information services anywhere and anytime;

- Semantics: Refers to the ability to extract knowledge from IoT objects. It involves knowledge discovery and efficient use of IoT resources, leveraging existing data to provide specific services. Various methods such as RDF (Resource Description Framework), OWL (Web Ontology Language) and EXI (Efficient XML Interchange) can be used for this purpose.

IoT Architecture

Connecting billions of smart objects to the Internet requires a flexible Architecture. Several advanced architectural designs that meet the needs of science and industry have been reported in the literature [36][37]. By the time being there's no standard IoT architecture.

IoT describes a group of devices or systems that are able to: collect data using sensors, process and exchange data between each other and use actuators. Given that we'll have a single server communication with different IoT modules in each client's fire detection system, the main idea of the Internet of Things implementation can be defined as a basic three-layer architecture model. The first layer is the smart object layer or perception layer. This layer represents physical objects that collect and process information, for example, by using sensors. The network layer should include abstractions of communication technologies, management services, routing and identification. Just above this is the application layer, which is responsible for providing services to clients. For example, applications require temperature and humidity measurements for clients that need this information.

IPv4 is the standard for addressing network devices and "binds" to the Internet, i.e. IP is required to connect to the Internet. However, the Internet was not designed to grow to tens of thousands of endpoints on a single subnet, as is the case today with the Internet of Things.

The growth of the World Wide Web has led to the exhaustion of available IPv4 addresses. IPv4 has proven not to be scalable enough to meet the needs of IoT.

IPv6 is a more effective way to deal with the scarcity of IPv4 addresses. The 32 bits originally allocated in IPv4 have been expanded to 128 bits, significantly increasing the number of addressable devices on the Internet. In the Internet of Things, network elements are uniquely addressed using IPv6 and are typically designed to transmit small amounts of data collected by devices.

However, the IPv6 packet size is larger than the frame size of the protocols used in IoT devices (IPv6 packets are sent in the data field of the media access protocol frame). For example, the IEEE 802.15.4 standard used for media access limits packets to 128 bytes. To solve this problem, the IETF has created 6LoWPANs [38].

6LoWPAN is an adaptation layer originally developed for IEEE 802.15.4. The basic idea is IPv6 packet compression (ID-6lowpan-hc), which allows devices with low computing power to use IPv6. 6LoWPAN compression is enabled using information from other layer protocols. For example, 6LoWPAN can use part of a device's MAC (Media Access Control) address to assign an IPv6 address to a smart object. Therefore, 6LoWPAN requires fewer bits for addressing than IPv6.

Researcher Adam Dunkels has made many contributions to the IoT field. Three of his most important contributions are related to the implementation of the TCP/IP stack for low power devices. These implementations are known as Low Weight IP (lwIP), Micro IP (mIP) and Contiki operating system for IoT [39].

The lwIP is a reduced implementation of the TCP/IP stack for embedded systems. The lwIP stack has more features than the mIP stack, as described below. However, higher performance can be achieved with lwIP. This protocol stack is currently supported by developers worldwide [39].

Some embedded system manufacturers such as Xilinx and Altera use lwIP. lwIP has the following protocols: IP, ICMP, UDP, TCP, IGMP, ARP, PPPoS, PPPoE. Applications include: HTTP server, DHCP client, DNS client, Ping client, SMTP client, and others [39].

Micro IP (mIP) is one of the smallest TCP/IP protocols available. It is designed for small microcontrollers (8- or 16-bit processors) where code size and available RAM are limited - at most 5 kilobytes of code and several hundred bytes of RAM [39]. Currently, mIP has been ported to several systems and includes Contiki.

Connectivity models in smart object networks

The forms of connectivity are categorised in a spectrum ranging from standalone smart object networks without Internet connectivity to the "real" Internet of Things, where objects have access to the Internet. Smart object networks will become a part of our lives in the coming years. Therefore, it is very important to understand the basics of IoT in terms of communication paradigms and connectivity models, as these two aspects will be crucial for the development of new IoT applications. There are three connectivity models for smart grids:

- Autonomous smart object networks: In this model, the object network is not connected to the "public" Internet. This model has many possible applications, for example in industrial automation networks (e.g. in nuclear power plants). There may be a need for a network of

objects that are interconnected but completely inaccessible from the outside, i.e. a special internal use of the network. In this connectivity model, smart objects should be handled using IP, given the previous experience with IP in traditional networks. IP has several features that encourage a positive response. For example, the TCP/IP architecture is inter-operable since it works with different connection characteristics and physical layers. Another argument is that IP is flexible and can evolve because its architecture is based on the end-to-end principle, where the network intelligence (application) is at the end point and the network is simple (it only transmits packets). This allows the protocol to evolve continuously over time. However, using IP in this and subsequent models, it can be said that the network remains compatible with the architecture of the Internet and is in line with the trends set by the RoLL and 6LoWPAN groups [39];

- Advanced Internet: The term "advanced Internet" refers to the "core" position of this connectivity model in relation to autonomous object networks and the "real" Internet of Things. In other words, this model presents a network of smart objects that are fully or partially connected to the Internet but with similar protection and security requirements [40]. Smart City applications are an example of this connectivity model. Applications will provide residents with useful information, allowing them to improve their quality of life and make important daily decisions. In the context of smart cities, a model can be applied where access to a facility's network is provided through firewalls and proxy servers that act as secure access controllers to the smart facility's private network. In this way, the connectivity model "extends" the Internet by providing access to previously isolated smart objects [39];

- Internet of Things: This model is the opposite of the autonomous object network model in the spectrum. In IoT, smart objects are connected to the Internet. Objects can therefore provide services like other entities on the Internet, for example, a smart object can be a web server. All users of the Internet, both humans and machines have access to the resources of smart objects. Access is possible through a direct connection to the object or through proxy servers. These servers can be located inside or outside the subnet of the smart object and their purpose is to collect information from the device. If a proxy server is used, the user connects to the Internet through a proxy server installed in the device. In this way, network processing techniques can be applied to the network between the proxy server and the device to save resources and maintain the end-to-end principle [39].

Communication paradigms

The Internet of Things is an evolution and a combination of many different technologies. In this respect, WSN and IoT overlap in terms of communication paradigm. This section gives an overview of these paradigms and categorizes them into four categories. These paradigms are very different, so the way they communicate can have more or less impact on resource utilization, especially on the sustainability of storage, energy and applications in the network. In the following section, each example is described and illustrated along with the protocol used to implement it:

- Many-to-one: smart objects transmit information from base stations. This paradigm is called data aggregation and is the most common paradigm as it meets the communication needs of many applications. For data aggregation, a routing tree with a root is created at the base station. This paradigm is usually not associated with high memory and power consumption. However, it is impossible to implement applications that require data confirmation because there is no return path between the root and the objects in the network. An example that illustrates the current state of this paradigm is the CTP or Collection Tree Protocol[41]. CTP uses Expected Transmission Count (ETX) [42] and Trickle [43] to generate high quality routes or to obtain such routes at low cost;

- One-to-Many : it is called data dissemination and has the opposite characteristics of the data aggregation paradigm. In data dissemination, the base station typically sends commands to one or more network locations. It usually involves reconfiguration or reconfiguration of network devices. For example, in data distribution, the network may be flooded to reach the target device. However, as with CTP data collection, there is no way to confirm that the data sent has been delivered. The cost per number of messages can also be high if there is a lot of flooding in the network. An example of a dissemination protocol is Deluge [44]. This protocol is optimised for the dissemination of large amounts of data, using the ETX metric to find high quality paths;

- One-to-Many and Many-to-One: this is an example that combines the two already mentioned. In this way, smart objects can communicate with base stations and vice versa. This makes previously impossible applications possible, such as reliable transport protocols. However, this paradigm requires additional memory to support a bidirectional path. The eXtend Collection Tree Protocol (XCTP) is an example of this category. In [45] the authors state that XCTP is an extension of CTP and therefore retains all the characteristics of CTP, but extends them with the ability to use reverse paths between the root and network elements;

- Any-to-Any: This paradigm is the most common, since it allows communication between two smart objects in the network. Although this approach is the most common, it is also the most complex and often requires more resources, especially storage, since paths must be maintained for all available devices in the network. On the other hand, it is not a major problem for applications unless the computing resources of the devices are very limited. The main routing protocols for the Internet of Things fall into this category.

Market Opportunity

In addition to traditional computers, various devices such as televisions, laptops, cars, smartphones, game consoles and webcams are networked and the list is growing day by day. In this new scenario, diversity has increased and it was predicted that more than 40 billion devices would be connected by 2020 [34]. Today more than nine billion devices, including computers and cellphones, are currently connected to the Internet throughout the world. Furthermore, during this very decade, that number has been increasing, with predictions ranging from a quintupling to 50 billion devices reaching one trillion [46].

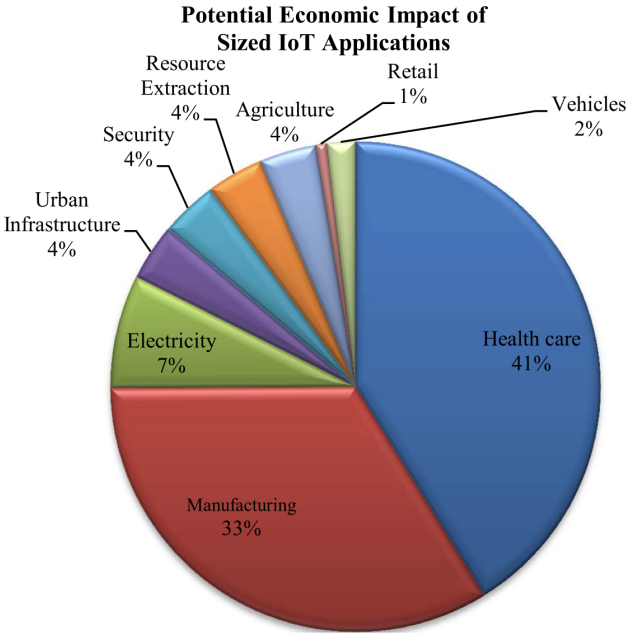


Figure 2.1: Potential market predictions for IoT technologies [1].

2.3 Wireless technologies

Wireless network are usually employed in difficult wiring areas, they provide network management and extension as well as flexibility to move the network. However, wireless channels are notorious for having significant levels of distortion and being unstable, thus reliably communicating data without too many retransmissions is a significant problem that motivates research in IoT communication technologies. In the context of the server communication, given a power outage or an access point malfunction, we're able to communicate through a modem to establish an isolated network connected to the internet. Nevertheless, for a faster response time our system will always give priority to a no modem communication where the module shares the same local network as other devices in the establishment where the fire detection system will be implemented. It should be noted that the latter implementation may compromise the functionality of our module if the local network is congested.

2.3.1 WLANs and WPANs

This subsection will explore technologies in regards to wireless local area networks (WLAN) as well as wireless personal area networks (WPAN) for IoT communication which will then be narrowed down to more suitable solutions by taking into account the requirements of our module and NIBBLE specifications.

Both networks follow the IEEE 802 standard, developed by the Institute of Electrical and Electronics Engineers (IEEE), which is composed by more than 423,000 members over 160 countries [47]. This society of professionals aims for technological innovation rooted on humanity needs and convenience, therefor providing new standards and common practices with regards to numerous technologies.

WLANs encapsulate two modes of operation, one being the connection to a network infrastructure and the other a connectivity between devices known as an adhoc network [2]. WPAN encapsulate wireless technologies of the IEEE 802.15 standard for short range communications like Zigbee and Bluetooth that fit within this standard as a subgroup, IEEE 802.15.4 and IEEE 802.15.1 respectively.

WLAN is usually implemented in small areas such as houses, stores, office spaces and other establishments, following the IEEE 802.11 standard dedicated to higher data rates when compared to the IEEE 802.15 standard. These wireless technologies complement wired solutions, the same applies to Wireless Wide Area Networks (WWANs) and Wireless Personal Area Networks(WPANs).

Only wireless LAN devices allow true mobility (although slow) and connectivity for indoor or outdoor environments, without compromising it's throughput [2].

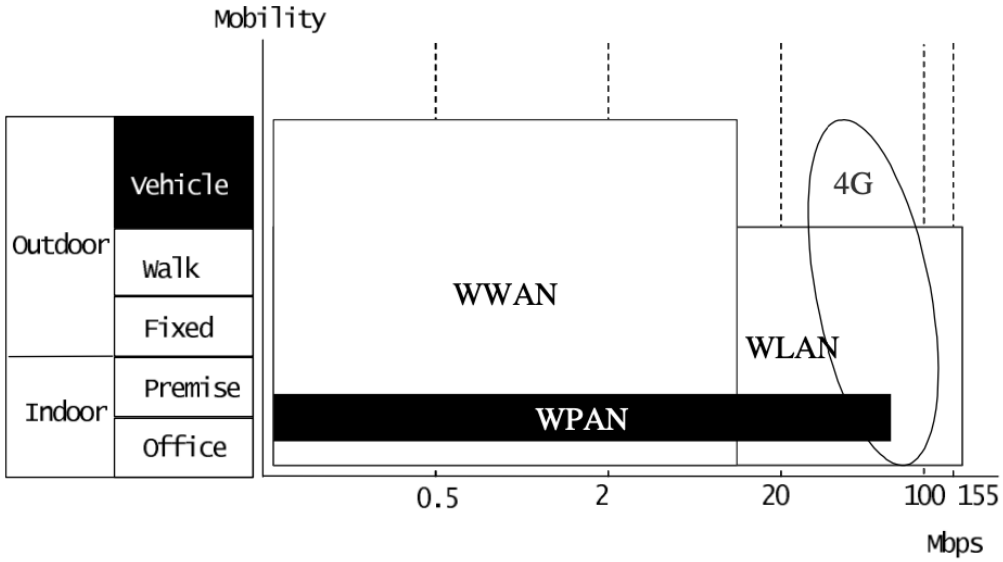


Figure 2.2: Wireless technologies performance placement [2]

Bluetooth/BLE

In order to facilitate the communication between devices Bluetooth was created by Ericsson in 1994 [48] as a low power radio that can currently stream data over 79 channels in the 2.4GHz unlicensed industrial, scientific, and medical (ISM) frequency band at a coverage range of up to 10 meters. This technology implements a point-to-point device communication, with an architecture that is commonly called master-slave. Where the master will control device access to the network and the slaves synchronizes with it's master.

Bluetooth Classic is usually employed in wireless audio streaming and has become the standard radio protocol behind wireless devices and in-car entertainment systems, also enabling data transfer applications such as mobile printing.

This wireless technology popularity has increased quite a lot since it's first release in the market in 2001 [48], with projected annual shipments of bluetooth enabled devices reaching 7 billion users by 2026 [49]. The Special Interest Group (SIG), comprised of leaders like Ericsson, Nokia and many others in the telecommunications, automotive, consumer electronics, computing and network industries, is leading the development of Bluetooth wireless technology and bringing it to market.

Bluetooth Low Energy (BLE) developed by Nokia following the standard by Bluetooth SIG, can transmit data over 40 channels in the 2.4GHz unlicensed ISM frequency band. It is meant to resolve disadvantages for wireless sensor systems of traditional Bluetooth technology through the expansion the traditional point-to-point wireless communication to broadcast or mesh which allows for a larger and reliable device network.

Despite being known for enabling device communications, given it's features that determine distance, presence and direction of other devices, BLE is now commonly used for device positioning in response to the high demand for accurate indoor location services [49].

BLE based wireless sensor systems present low power consumption, good data throughput, small and simple software stack as well as a simple implementation.

Zigbee

Zigbee is based on the IEEE 802.15.4 communication protocol standard and is used usually for personal area networks (PANs). Transmission requires very little power, which is only one percent of that used in Wi-Fi or cellular networks. It operates at frequency band of 868 MHz, 915 MHz and 2,4 GHz with an approximate 250Kbps of maximum data rate [50]. However, the range of Zigbee device communication is very small (10–100 meters). This technology presents various routing schemes which depend on star, tree, and mesh topologies, integrated by three different Zigbee devices: the Coordinator, Router and Endpoint. The latter is the device that will be controlled, while the Coordinator leads the communication between devices by communicating with the Router which in turn will forward information regarding the network devices.

The star topology, is named after the star shape architecture where at the center the coordinator will receive every information and communicate with endpoint devices.

The tree topology similarly to the previous star architecture holds the coordinator at the centre of the communication, and with the added routers or endpoints connected a network hierarchy is established.

In the mesh topology, the coordinator roll is to register added or removed devices from the network. Router devices are responsible for routing information and the topology management process is automatic, always prioritizing the traffic of data.

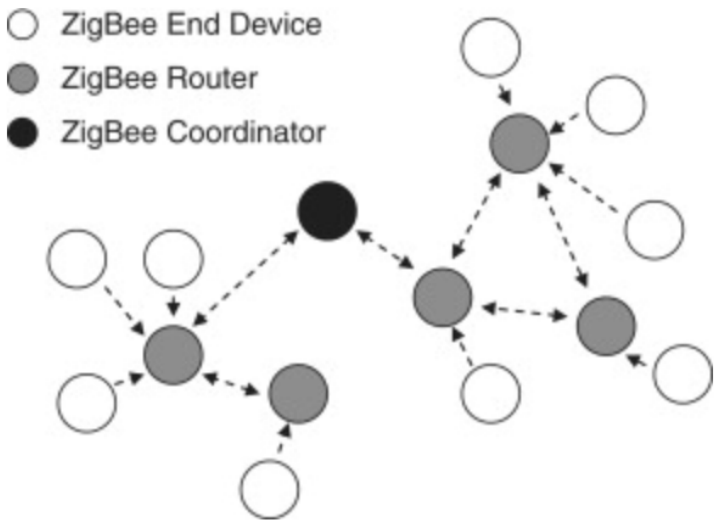


Figure 2.3: Zigbee routing scheme and devices[3].

Wi-Fi

The IEEE 802.11 standard commonly called Wi-Fi, which stands for *Wireless Fidelity*, was developed and standardized by Wi-Fi Alliance and consolidated it's dominance in the market given their associates such as Apple, Facebook, Sony Corporation, Cisco Systems, Intel and many others [51].

The first certified products started in April 2000, with IEEE 802.11b devices. After more than 20 years later, it is estimated that more than 15 billion Wi-Fi enabled products are in use according to the Wi-Fi Alliance[52]. Wi-Fi's popularity stems from a great trade-off between distance and data rate which stands out from previously mentioned wireless technologies Bluetooth and Zigbee 2.5, but for also providing a wide range of implementation environments.

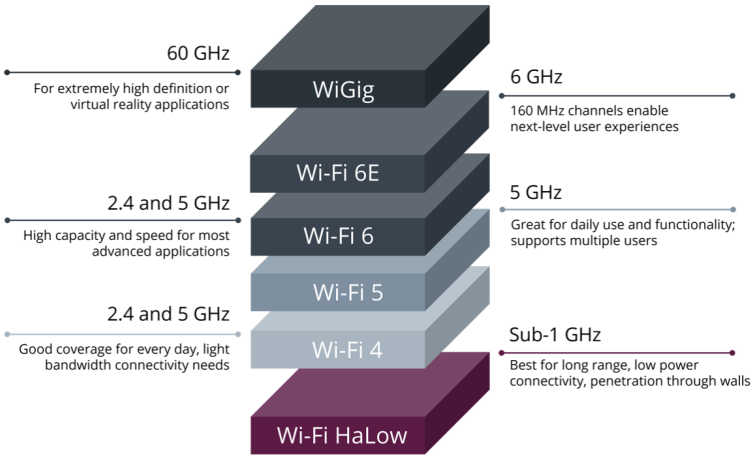


Figure 2.4: Wi-Fi technologies for various enviorments [4]

For most establishments the Wi-Fi network works with both 2.4GHz and 5GHz within the Super High Frequency band (SHF) in the ISM spectrum bands. Internet connection is made through Access Points but the Wi-Fi architecture may be employed as an Ad-Hoc or Infrastructure network.

The Ad-Hoc architecture allows direct communication between the devices and the Access Point maybe be implemented as an extension of the network, while in the Infrastructure architecture, every device is connected to the Access Point. The network is identified with an SSID (Service Set Identifier), which holds the name of the Wi-Fi network that the user wants to connect to as well as password for authentication. The Access Point is connected to the fixed network either through a copper wire DSL (Digital Subscriber Line), a faster hybrid fiber-coaxial cable, or even Fiber, which is the fastest form of wireline connectivity.

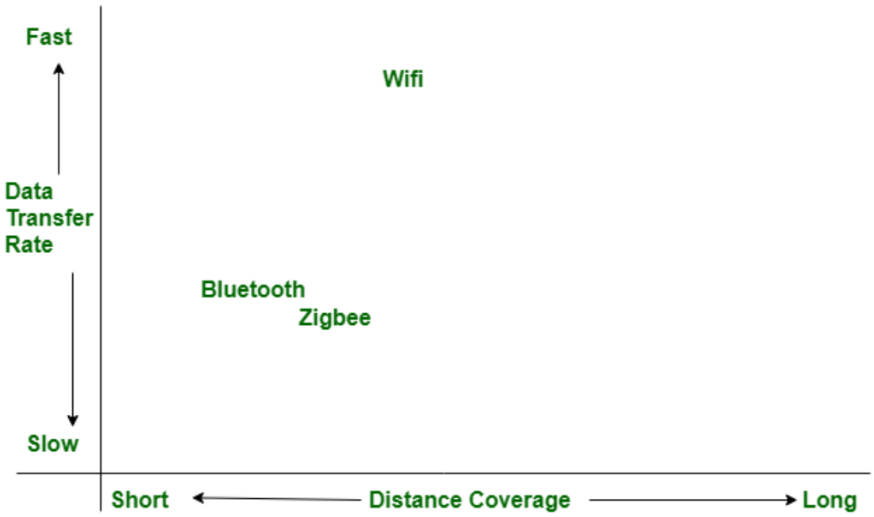
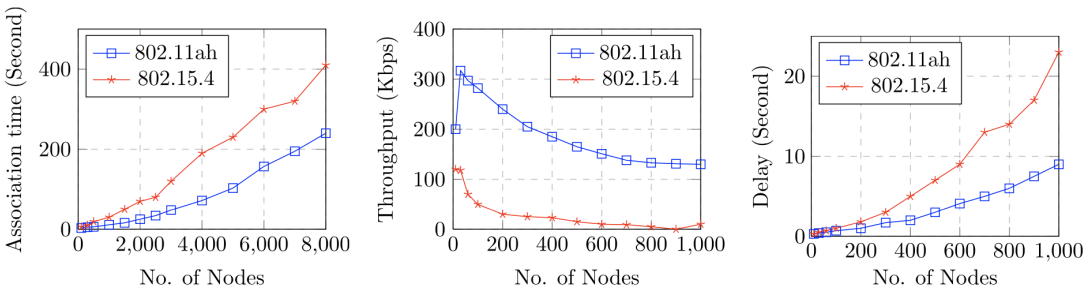


Figure 2.5: Data rate and coverage of Bluetooth, Zigbee and Wi-Fi [5].

Another recent technology worth referencing is Wi-Fi HaLow - 802.11ah standard - operating in spectrum below 1 gigahertz (GHz), consumes less power than a traditional Wi-Fi device and also has a longer range, nearly twice that of traditional Wi-Fi given it's powerful penetration through walls. Because of the relatively lower frequency, the range is longer since higher frequency waves suffer from higher attenuation. We can extend the range by reducing the frequency further, however, the data rate will also be lower and thus the trade-off is not justified. IEEE 802.11ah is also designed to support large star shaped networks where a lot of stations are connected to a single access point. When compared to Zigbee protocol as a function of the number of users/nodes, with regards to Access Point association time, throughput, delay and coverage range, simulation studies by N. Ahmed and H. Rahman and Md.I. Hussain [6] demonstrate how the Wi-Fi HaLow protocol performs better as shown in figure 2.6.



(a) Association time. (b) Throughput achieved. (c) End-to-end delay.

Figure 2.6: HaLow and Zigbee performance[6].

This IP based based technology, uses Wi-Fi security and management capabilities while decreasing energy consumption, at the expense of a lower data rate (0.6 to 8 Mbps) [53]. It achieves better performance than ZigBee, Bluetooth and despite providing kilometer range coverage it is

far from matching long range wireless technologies such as LPWANs (Low Power Wide Area Networks).

2.3.2 LPWANs

Contrariwise to the previous approach, assuming there's a fault within the establishment and the power is down our IoT module will make use of a long range communication technology. Low Power Wide-Area-Networks or LPWANs represent our wireless networks of interest. This is a power efficient technology that allows for a long battery life of up to 10 years, as opposed to standard cellular technologies like 3G, 4G and 5G which given their high power consumption as illustrated in figure 2.7 can't consolidate a sustainable IoT system.
LPWAN technologies can be split into unlicenced spectrum services, like Sigfox or LoRa used mainly in high density areas such as cities, for example, and cellular-based licenced spectrum services, such as EC-GSM-IoT, LTE-M and NB-IoT.

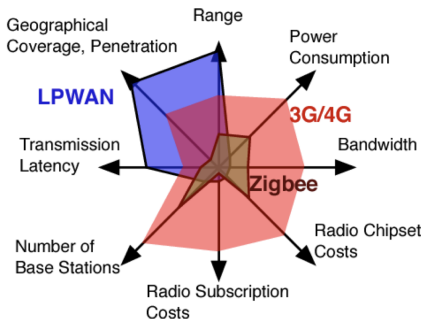


Figure 2.7: Characteristics of cellular, LPWAN and Zigbee technologies[7].

The 3GPP specifications regulate the section of frequencies that are licensed to private companies by regional or even national authorities [54]. A Cellular IoT solution benefits from being cost-efficient since there is already in place infrastructure and consistent standards that provide great coverage. Another aspect in favor of these technologies regards network management, since it becomes easier for a service engineer to monitor any device given the fact that they can be notified of any abnormality.

Extended Capabilites-GSM-IoT (EC-GSM-IoT)

EC-GSM-IoT is a solution that can be installed on existing Global System for Mobile Communications (GSM) deployments, which is the world's largest and most widespread cellular technology. The EC-GSM-IoT protocol was designed to provide connectivity to devices connected to the Internet of Things under difficult radio coverage conditions and it could be considered to be the equivalent of LTE-M (Long-Term Evolution Cat-M1) in the GSM spectrum, as it mainly brings range and power efficiency improvements. Bandwidth per channel is of 200 kHz, for a total bandwidth of 2.4MHz.

Notwithstanding the useful features for IoT applications, EC-GSM-IoT is getting much less attention in the context of IoT solutions. The reason behind this is the planned decommission of GSM networks by telecommunication companies [55] and the fact that no functional EC-GSM-IoT network is ready for usage right now.

LTE for Machine-type Communication(LTE-M)

LTE-M (Long-Term Evolution Cat-M1) technology which is also named LTE CAT-M1 and eMTC (enhanced Machine-type Communication) can be implemented for wide-ranging, power efficient and scalable connections. Seamlessly coexisting with other LTE services, together with Narrow band IoT (NB-IoT), expands LTE technology. It is able of serving higher bandwidth and low latency IoT applications when compared to other cellular-based licenced spectrum services. When compared to Long-Term Evolution (LTE), LTE-M cuts the bandwidth from 20 MHz to 1.4 MHz and lowers the data throughput to a tenth of that of LTE (up to 1 Mbps)[56].

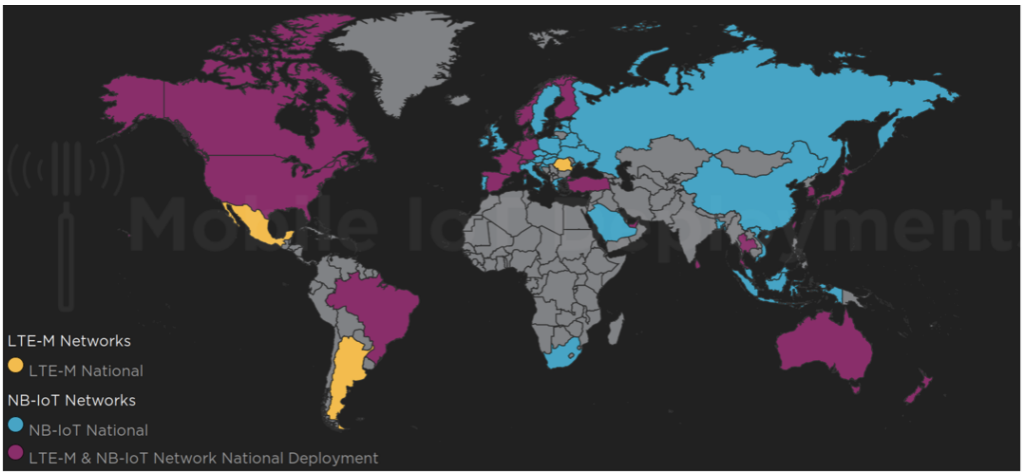


Figure 2.8: Mobile IoT Deployment Map [8]

NB-IoT

Narrow band IoT (NB-IoT) is a fairly recent technology within the licensed spectrum of cellular services for multiple devices, that just like LTE-M it is optimized for a greater coverage, lower complexity/power, and higher device density. requires low energy consumption, supports a high data rate with low latency as opposed to LoRaPWAN, and a coverage of 1 km or 20 km in urban areas and in rural areas, respectively [7]. This protocol refers to a standard developed by the Third Generation Partnership Project (3GPP) and can be deployed with both GSM and LTE cellular devices. It is best suited for applications that focus on quality of service as interference and noise are significantly reduced in licensed spectrum frequencies.

In cellular devices, IoT applications might not need high data transmission and therefore a licensed LPWAN technology, like NB-IoT for low power and low bandwidth is adequate.

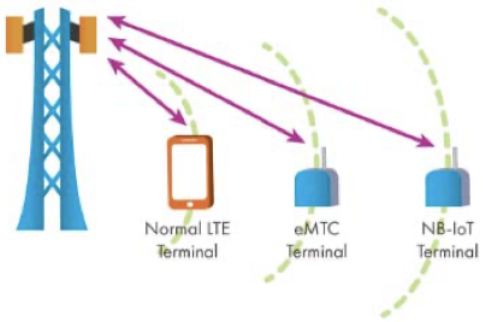


Figure 2.9: Cellular IoT range[9]

Sigfox

Another popular technology that uses narrow band communication is Sigfox. Unlike NB-IoT, it is a well matured technology that relies on very low power transmission through great wavelengths and therefore allows for a coverage of up to 1000 km. Despite having a range advantage and a great noise reduction due it's Ultra-Narrow-Band (UNB) feature it has a lower bandwidth when compared to NB-IoT. A single payload is limited to 12 bytes of information which may not be reasonable for monitoring remote locations [56].

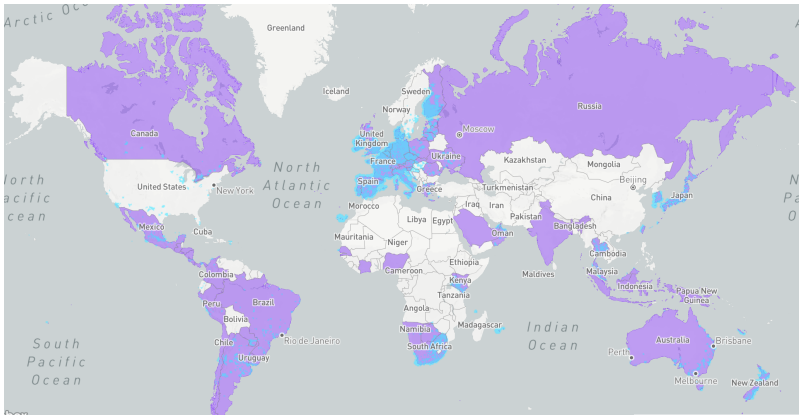


Figure 2.10: Blue and purple represent live and future Sigfox coverage, respectively[10].

LoRaWAN

LoRa Alliance introduced LoRaWAN technology, which similarly to Sigfox it uses the unlicensed spectrum. It is best suited for wide area network applications and it was implemented to work at low power consumption, with a coverage range of 2-5km. As it is designed for long range IoT protocols, LoRaWAN presents features like secure and reliable communication, as well as a long battery life and cost-effective structure. Both LoRa and Sigfox lead to less power consumption from end devices since they use asynchronous communications, in contrast with NB-IoT which beyond synchronous communications incorporates high energy modulation techniques(figure 2.12).

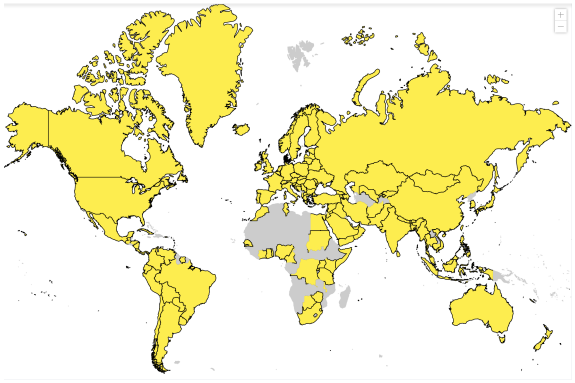


Figure 2.11: LoRaWAN global coverage [11]

As specified by NIBBLE, our module will encapsulate a SIM (Subscriber Identity Module) card for stored subscriber-related information and implementation of security functions concerning the ciphering and authentication on the user side in UMTS and LTE (3G and 4G) devices. Compared with LTE-M, the low peak rate and narrower band of NB-IoT denotes it's higher performance when it comes to ultra-low power consumption and low data rate services.

NB-IoT is more resilient to congested networks and it also requires less power consumption when compared to EC-GSM-IoT [54]. It is also predicted that around 75 billion NB-IoT devices will be in use by 2025, which represents a 400% increase over the approximately 15 billion devices nowadays [57].

Gathering this information and comparisons from [9] it was decided that NB-IoT represents a promising technology with an overall better performance desired for our IoT system.

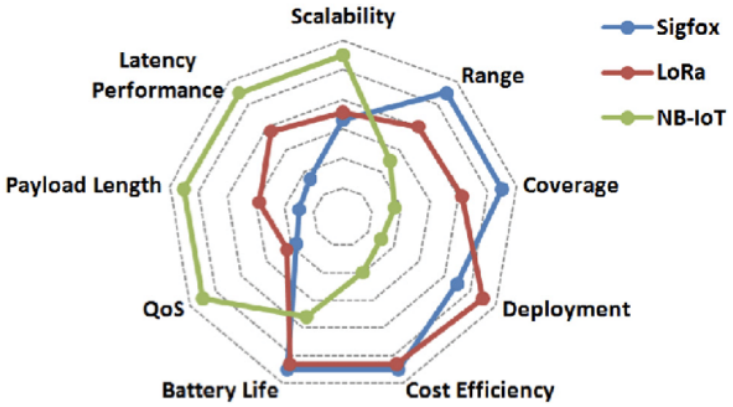


Figure 2.12: Characteristics of LPWANs technologies[7]

2.4 Hardware Platform

2.4.1 Microcontrollers

It is now common knowledge that computers, calculators, mobile phones, microcontrollers and everything digital works in binary. But few people know what binary technology is and how it works. In digital devices, binary means that all data is stored as zeros and units, known as bits. Images, sounds or text are converted into numbers and stored in bits.

To facilitate processing, bits are organised into groups of octets called bytes. The binary number 01001010 is one byte. The bits are numbered right to left from 0 to 7, the leftmost being the most significant bit or MSB and the right most being the least significant bit or LSB. To convert a byte to a decimal number, multiply each bit by the value 2 corresponding to its position, and then add the values. Any number between 0 and 255 can be written in bytes. It does take some effort to convert from binary to decimal, but this is not a big issue because when computers are programmed for calculations that require high precision, 32, 64 or even 128 bits are often used to represent each number [58]. Calculators, microcontrollers and microcomputers use this mechanism to interpret each instruction. A microcontroller performs calculations in logical and numerical blocks through it's core. When configured, it sends signals with simple instructions such as, for example, "high" to send data and "low" to stop.

For the IoT module implementation, with regards to the hardware platform there's the possibility to use an FPGA (Field-Programmable Gate Array) which is a type of integrated circuit that is composed by programmable logic gates within interconnected logic blocks, allowing for a more versatile and flexible approach since there is no fixed hardware implementation. Some features of this particular board include storage elements like RAM and ROM, but also switches, LEDs, pin expansion headers, flip-flops, and necessary I/O ports. FPGAs can be found in numerous systems like data centers, medical equipment, vehicles and so on [59]. However, rewiring logic gates through programming in descriptive languages like VHDL or Verilog to carry out a different task is not a feature that outweighs a cost effective and scalable solution needed for our module, since FPGAs are in fact a significantly expensive alternative when compared to other options like microcontrollers.

Microcontrollers, similarly to computers, come with RAM, ROM, timers, I/O ports and other external features, software programs to execute commands like C, C++, python and others. There's four main components that make a microcontroller: a CPU, program memory, data memory and peripherals.

Peripherals represent a task the CPU may do but it is either too busy or is simply unable to preform it. The most common peripheral would be the timer, as a way to time events frequency and allocate in memory their time occurrences, by counting up or down.

Program memory can be volatile meaning all stored data is lost when the power supply is disconnected, or non-volatile like in most embedded systems, where data persists in memory after turning off the power supply mainly by electrically addressed devices or mechanically addressed devices [60].

There are some feasible options to retain data when a microcontroller is switched off. A common example is the EPROM (Erasable Programmable Read Only Memory) commonly used within computers to store BIOS and Toolbox routines and initiate software in Mac and IBM computers, these devices are erasable by ultraviolet (UV). A more scalable and popular alternative is the OTP (One Time Programmable) EPROM [61] since it is a lower cost plastic package device. EEPROM (Eletrically Erasable Programmable Read Only Memory) is another lower cost non-volatile memory that can be reprogrammed many times and erased by applying a suitable electrical voltage to the device, but the erase and write cycles are slow when compared to previous alternatives.

Flash memory, usually referred as RAM memory, holds an unmatched popularity within microcontrollers or embedded systems which usually require remote software updating. The embedded software in modems, for example, is stored in FLASH and may be updated by downloading the current version via the Internet or a bulletin board using the modem itself. The new version can then be sent from the computer to the modem via the serial link once it has been downloaded. FeRAM or FRAM (Ferroelectric RAM) is a significantly faster and low power alternative to FLASH [62].

Microcontrollers also consume way less power than FPGAs and for that reason they can be easily integrated in IoT solutions for constrained resources. However, microcontrollers are task limited given their instruction based programming and fixed circuitry which restricts the systems overall complexity [59].

Choosing a microcontroller can be a result of decisive factors like energy consumption, data addressing, data processing and external features. For IoT applications there are some popular development boards such as Raspberry Pi, Arduino and ESP, that can be a good starting point for an overview of the hardware platform of our IoT module.

Arduino

The Arduino microcontroller, like many others, is able to control objects by connecting them to sensors and actuators. For example, it can connect sensors to measure temperature and relays to turn heaters on and off[63]. One of the distinctive features of Arduino is that the hardware and software are open source, which means that anyone can freely modify the code, design and circuitry and contribute to the platform's global community of users [64].

Arduino boards consist of various components such as microcontrollers, digital and analogue connectors, digital and analogue contacts, power connectors, resistors, diodes, capacitors and LEDs [65]. Arduino boards have an Atmel AVR microcontroller and an oscillator (crystal) that emits pulses and operates at a specific frequency. Most of them have a 5 V voltage regulator and a USB port that connects it to a computer [66].

The voltage regulator maintains a voltage between 7 and 12 V and the DC port converts this voltage to 5 V. Arduino can also receive power from a USB port where configuration is done [63]. The Arduino board has input, output and analogue (ADC) pins that must be specified separately in the project code. Each of these pins can have digital inputs and outputs. Depending on the project, the ADC pins can also be used as analogue inputs [67].

In short, if we say that a pin has a digital output, it means that it can receive either 0 V or 5 V. If we say it has an analogue output, it can mean that it can take any value between 0 V and 5 V [68]. In order for the Arduino to work, it must be programmed using IDE software. It is based on the C/C++ programming language and contains instructions to interact with external devices connected to it [66].

The development environment can run on different platforms (Windows, Macintosh OSX and Linux) and is easy to program for beginners and advanced users [64]. All that is needed is a USB cable and a computer to install the application. The code is saved in the Arduino and remains there until another sketch is loaded. Once the program is loaded, the Arduino can be disconnected from the computer. The sketch works normally when connected to another power source [65].

Arduino is a development committee that has been used in hundreds of projects over the years, from household items to complex scientific instruments. This open source platform is used by a global group of users, from students to professionals, amateurs to programmers, and their contributions have built an incredible body of knowledge that is accessible to everyone, beginners and experts alike.

This development platform was created by the Ivrea Institute of Interaction Design to create simple rapid prototyping tools. Arduino has managed to reach a wider community by adapting to new needs and expanding from 8-bit boards to products with IoT applications [64].

The platform offers many possibilities. The most popular Arduino Uno has 14 digital ports (6 of these 14 ports can be used for PWM waveforms) and 6 analogue ports, the ports are about 40 mA and have 32 Kb of flash memory, 2 Kb RAM and 1 Kb EEPROM. Two of the six analogue ports can be used for I2C communication, namely the SDA (Serial Data Line) port and the SCL (Serial Clock Line) port [69].

It also has TX and RX serial ports. One of the main problems with this platform is the lack of wireless communication such as Bluetooth or Wi-Fi, and this problem has been solved with the development of the Nano 33 IoT. The Nano 33 IoT has 14 digital pins, a much larger flash memory than Arduino Uno, about 256 KB, but more importantly for operation, Bluetooth, Wi-Fi and BLE communication, it is only 18 x 45 mm in size [69]. It is, however, slower than other microcontrollers like Raspberry Pi and ESP with a clock speed of 48MHz.

Raspberry Pi

The Raspberry Pi is focused on web applications and is built on the standard Linux operating system, but the platform is also compatible with other flash-based systems and offers a great variety of features. The Raspberry Pi 3 A+ board, for example, has four USB ports, one RJ45 port for Ethernet, audio and video output, camera input, Full HD display output, power supply, touchscreen input, quad-core processor, Bluetooth and Wi-Fi and Micro SD input, connection to other microcontrollers via serial port, BCM2837B0 ARM Cortex-A53 microprocessor with 1.4 GHz 64-bit clock frequency [58].

Developed by Raspberry Pi Foundation and Broadcom controller, this microcontroller is very powerful and can run high performance applications, with modules of up to 1.8 GHz clock speed

that connect to other devices or prototype platforms. This platform is well known in the market as it is often used in set-top boxes that turn normal TVs into smart TVs. It is also used in minigames, low-cost computers and many other applications that sell well on the Internet [58].

Raspberry Pi is one of the most popular development kits for complex applications due to it's flexibility given the many features it encapsulates. It may run on Raspbian, Windows and also Android, allows for a high-level programming with C, C++, Java and many others, thus suiting multiple web-based services. Depending on the many versions available (figure 3.7), Raspberry Pi has multiple ports like USB, HDMI for display, ethernet or Wi-Fi through a given adapter and GPIO pins for external features.

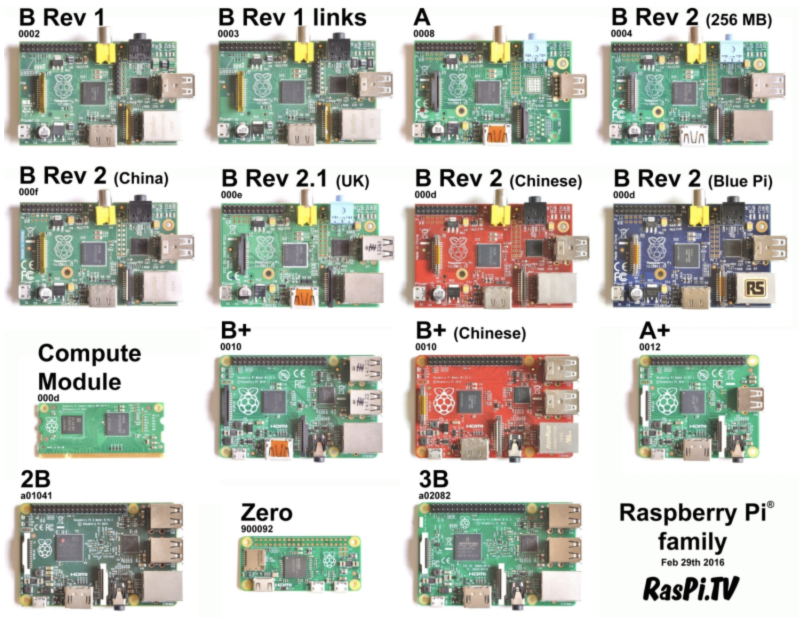


Figure 2.13: Raspberry Pi boards [12].

Because it's performance equates to one of a computer it is generally priced as a higher cost solution, not meant for restrained environments with small batteries, since many of it's resources such as the process power and available memory fall out of the scope of IoT managed systems. Nevertheless, it should be noted that there are interesting Raspberry Pi modules that can be applied to IoT systems such as the Raspberry Pi Zero W 2.14. This module offers a cheaper and smaller board when compared to a common Raspberry Pi, by reducing it's processing power and graphics. Most connectors are unsoldered or available in the micro version, allowing them to be integrated into medium-sized battery-powered devices. However, the Raspberry Pi Zero W is not very scalable since it's availability is scarce and there is also not a lot of well documented support with regards to it's functionality when compared to ESP boards.

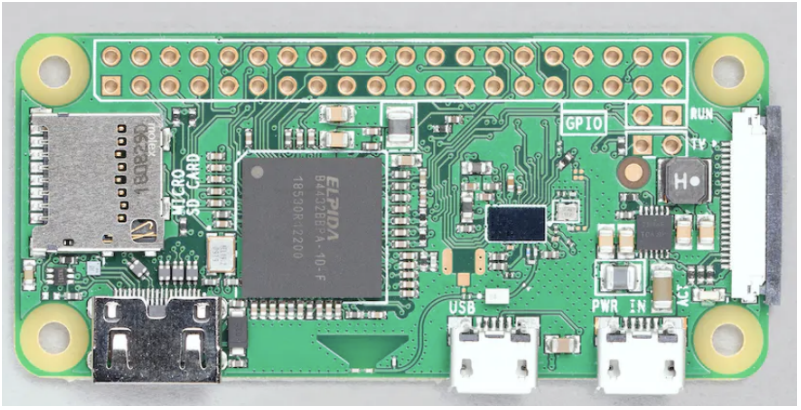


Figure 2.14: Raspberry Pi Zero W board [13].

ESP01

The vast family of ESP development boards are cheaper and popular solutions within IoT systems. ESP-IDF is the development framework for Espressif SoCs supported on Windows, Linux and macOS, powering millions of devices in diverse fields[70].

As opposed to the Raspberry Pi, the ESP runs RTOS (Real Time Operating Systems) instead of a familiar OS. Comparatively, the operating systems on smartphones and personal computers are overstuffed with programs and functionality because they need to accommodate whatever a user would want to do right now. On the other hand, an RTOS is streamlined and designed to carry out its functions in a timely and efficient manner. It is much smaller—sometimes only a few megabytes (instead of more than 20 gigabytes)—has a basic graphical user interface, and lacks many common functions like a web browser.

The ESP8266 ESP01 Wi-Fi module is specially designed to connect the microcontroller to Wi-Fi easily and efficiently. The ESP8266 Wi-Fi module supports 802.11 b/g/n networks commonly used today and can function as an access point or a sending and receiving data station. The ESP01 Wi-Fi adapter is now equipped with a level reducer and can be connected directly to the prototype board. This allows data to be transferred from one component to another via Wi-Fi. The ESP01, because of its miniaturised size, at around 14.3mm and 24.8mm, and under 10μ A, makes it a microcontroller of extreme prominence and interest. Its input voltage is 3V, it has two digital pins, a 516kb flash memory and wireless communication via Wi-Fi, as already referenced [69].

ESP32

Espressif System has introduced the ESP32 in September 2016 and it remains, to this date, one of the best supported by Espressif chips. The ESP32 integrates Wi-Fi, Bluetooth and BLE conectivity and also a higher performance with 80 MHz clock speed and slightly higher price when compared to it's predecessor ESP8266.

The integration of Bluetooth, Bluetooth LE (Low Energy) and Wi-Fi ensures that a wide range of applications can be developed, and that the module is all-around: using Wi-Fi allows a large

physical range and direct connection to the Internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5μ A, making it suitable for battery powered and wearable electronics applications. The module supports a maximum data rate of 150 Mbps, and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity[71].

ESP32 has up to eighteen 12-bit analog to digital converters (ADC), two 8-bit digital to analog converters (DAC) and ten capacitive touch-sensor inputs. As for serial communication, there are two I2C and I2S bus connections as well as three UARTs. The module ESP32-WROOM-32 has a total of 38 pins but not all of them are recommended for use and although most pins have multiple functionalities some pins can't be used simultaneously, for example, the ADC2 pins can't be used if the Wi-Fi feature is being used [72].

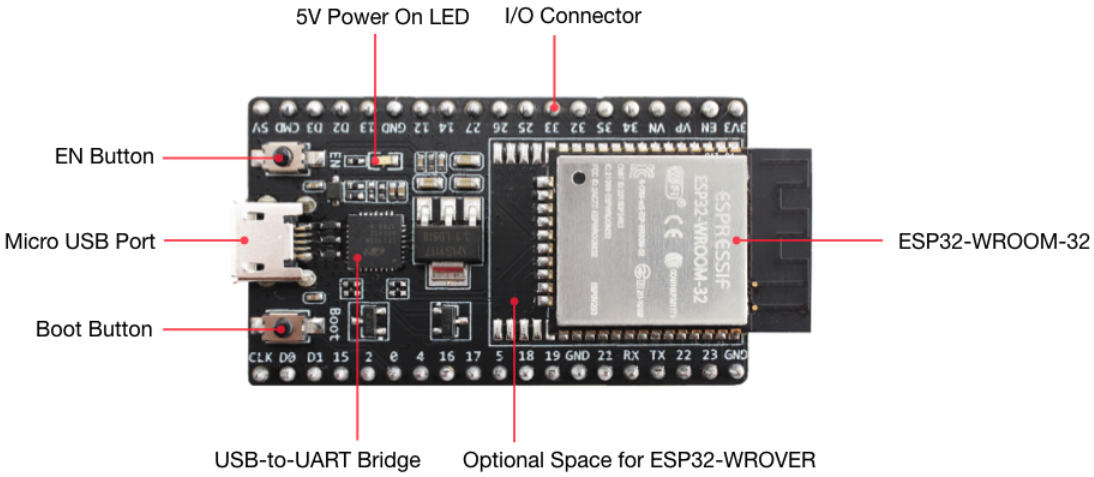


Figure 2.15: ESP32-DevKitC V4 with ESP32-WROOM-32 [14].

2.4.2 Chip Shortage

In the midst of 2020 the COVID-19 crisis introduced fluctuations and uncertainty within the supply chain that reflected on the microncontroller world.

During the pandemic many items went out of stock and at the same time stores were closed, people stayed at home and began to work or learn remotely. For this shift to happen the semiconductor industry has provided computer and electronic devices and has seen a rise in revenue in the latest years with Asian economies leading the market[15]. Consequentially, Euro area imports of semiconductors have declined significantly. The recovery of semiconductor imports has not been at the same rate as the recovery of total imports which then affected the delivery time of suppliers.

For NIBBLE and the rest of the technology equipment and automotive industries that use semiconductor for production, this increased delivery time caused an unusual increase in the ratio of PMI (Purchasing Managers' Index) new orders to suppliers' delivery time for manufacturers as shown in figure 2.16. The semiconductor scarcity is projected to continue in the foreseeable future despite the global chip manufacturing growth [15].

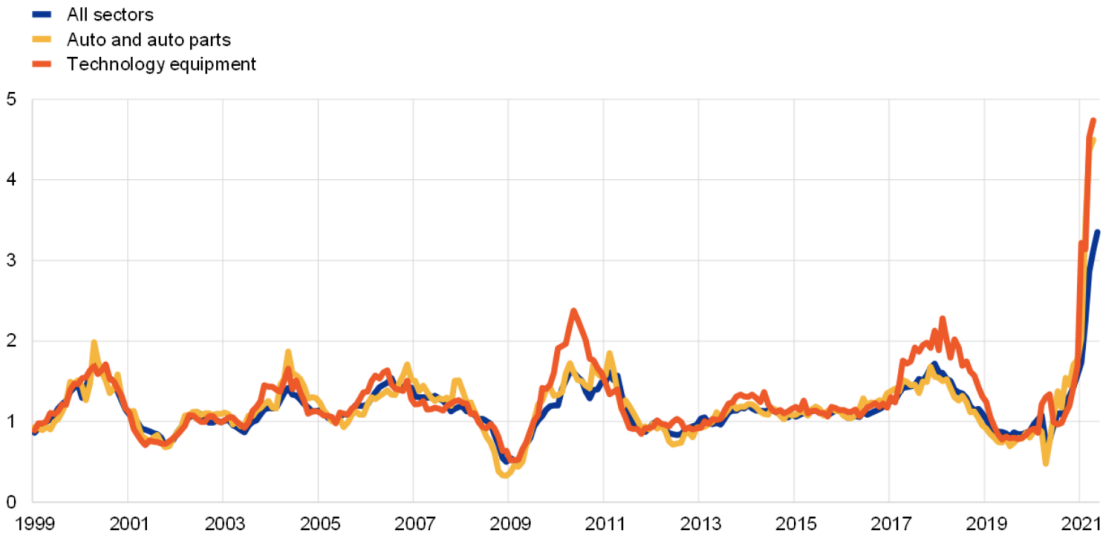


Figure 2.16: Euro area ratio of PMI new orders to suppliers' delivery times.[15]

2.4.3 Communication Interface

The hardware interface between the IoT module and fire detection system depends on how the connection will be defined. Every fire detection has a fire and fault output. The module is connected with a fire detection central which will then receive inputs either by relays or voltage readings to manage the fire detection system.

The ESP32 peripherals offer serial communication, such as I2C, I2S bus connections as well UART. For the NB-IoT communication a serial interface is needed between the microcontroller and the modem. The BC660K-GL NB-IoT modem provides two UART ports [73].

The UART(Universal Asynchronous Receiver-Transmitter) protocol is a popular solution to transmit data asynchronously, which in turn means that there is no clock signal that synchronizes the output bits from the transmitting UART and the sampling of bits by the receiving UART. The transmitting UART expands the data packet being sent with start and stop bits in place of a clock signal. In order for the receiving UART to know when to begin reading the bits, these bits specify the start and stop of the data packet. The actual data being sent is included in the data frame. Parity bits indicate whether an integer is even or odd. The parity bit allows the UART receiving the data to determine whether any of the data has changed since it was last sent. ESP32 serial communication could also be done using a USB-to-UART bridge for programming, updating and testing firmware.

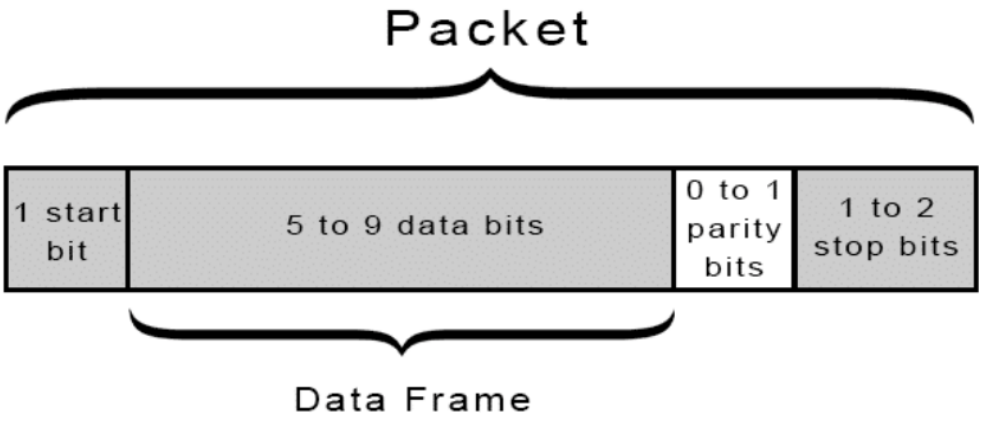


Figure 2.17: UART packet [16]

Universal Serial Bus (USB) is an input/output port standard for computers and digital equipment that allows easy high-speed data transfer through a direct connection or cable and can be interfaced with microcontrollers in a simple way [74].

USB is intended to be an extensible high-speed, fully standardised, plug-and-play interface between a computing device using a single port and a number of different peripherals using one port each, all control being performed by signals within the data stream [75].

2.5 Server

As previously discussed the server communication is able to use an external modem to establish an isolated local network connected to the internet, or a no modem solution where the module communicates with the server through a router or access point and shares the same network as other devices in the establishment where the fire detection system will be implemented .

Both solutions will make use of an application protocol well suited for the IoT communication. The Node-RED software will then interpret the module data sent through the internet and display it to the user in a intuitive dashboard.

Since we may adopt a non IP network to communicate with our server, we're able to have both the Wi-Fi network and non IP network fully functional within the establishment where our IoT module is implemented. Each IoT module is easily identified through an unique MAC address and it does not follow fundamental pillars of IoT sensing applications for dynamic environments that require Radio Frequency Identification (RFID) and a Wireless Sensor Network (WSN).

2.5.1 Application layer

This subsection describes the protocols based on their main characteristics. In short, it summarizes the standardization status, interaction model, quality of service options, transport protocol and security mechanisms. MQTT, AMQP and XMPP are designed to run on networks using TCP, while CoAP uses UDP as the underlying transport.

MQTT and AMQP implements a publish/subscribe model, while CoAP implements a request/response interaction model. A basice basic QoS is provided by the AMQP, MQTT and CoAP protocols for message delivery. MQTT and AMQP implement three different levels of QoS, while in CoAP the request and response messages are limited to two [76]. The majority of such protocols select the TLS or DTLS protocol for security arrangements.

AMQP

Defined by OASIS, AMQP (Advanced Message Queuing Protocol) was designed to allow interoperability between a wide range of different applications and systems, regardless of their internal design, as an open standard protocol that follows the publish-subscribe paradigm. At first, it was developed for commercial messaging with the intent of providing a non-proprietary solution that is able to manage a great amount of message exchanges that may happen in a short period of time in a system [77]. This AMQP interoperability feature allows for message exchange across different platforms, implemented in many languages, which is perhaps significant in heterogeneous systems [78].

AMQP has been implemented in two very different versions, AMQP 0.9.1 and AMQP 1.0, each with a completely different message exchange paradigm. AMQP 0.9.1 implements the publish-subscribe paradigm, which revolves around two main AMQP entities, both part of an AMQP broker: exchanges and message queues.

Exchanges represent a part of the broker that is used to direct messages received from publishers. Publishing messages to an exchange entity is the first step in the process, and after that messages are routed to one or more appropriate queues. This depends on whether there are more subscribers interested in a particular message, in which case the broker may duplicate messages and send their copies to several queues. A message will remain in the queue until it is received by a subscriber [77].

Exchanges and queues are linked by the routing process, which relies on bindings, that are predefined conditions and rules for message distribution. On the other hand, the latest version of the AMQP protocol, AMQP 1.0 is not tied to any specific messaging mechanism.

While the older versions of the protocol specifically used the publish-subscribe approach, with an architecture consisting of message exchanges and queues, the newer implementations of AMQP follow a peer-to-peer paradigm, and can be utilized only when there is no broker in the middle. The broker is only present in communication that needs to provide the storage and forwarding mechanism, while in other cases direct messaging is possible. This option to support different topologies increases the flexibility for possible AMQP-based solutions, allowing different communication patterns, such as client-to-client, client-to-broker, and broker-to-broker [79].

It should be noted that a significant amount of infrastructures still use the old version of AMQP 0.9. AMQP uses TCP for reliable transport, and in addition provides three different levels of QoS, the same as MQTT. Finally, the AMQP protocol provides complementary security mechanisms, for data protection through the TLS protocol for encryption, and for authentication through SASL (Simple Authentication and Security Layer) [77].

Given the numerous features it offers, AMQP has relatively high requirements in terms of processing, power, and memory, making it a quite heavy protocol. This has been its biggest drawback in IoT-based environments. This protocol is more suitable in the non-bandwidth and latency constrained parts of the system with higher processing power.

XMPP

XMPP (Extensible Messaging and Presence Protocol) is an open standard messaging protocol formalized by IETF, and was initially designed for instant messaging and inter-application messaging. It is a text-based protocol, based on Extensible Markup Language (XML), which implements client-server and publish-subscribe interaction, running over TCP. In IoT solutions it is designed to allow users to send messages in real time, in addition to managing user presence. XMPP enables instant messaging applications to achieve all the basic features including authentication, end-to-end encryption and compatibility with other protocols [80].

XMPP supports the client-server interaction model, but there are new extensions that also allow the use of a generic publish-subscribe model. These extensions allow XMPP entities to create topics and publish information; an event notification is then broadcast to all entities that have subscribed to a specific node. This functionality is quite important for IoT-fog-cloud scenarios, and is the basis for a wide variety of applications that require event notifications. Clients and servers in XMPP communicate with each other using XML streams to exchange data in the form of XML stanzas (semantic structured data units) [36]. There are three kinds of stanzas which are defined: <presence/>, <message/> and <iq/> (info/query). One message stanza define a header and message contention and is used to submit data among XMPP organizations. Message stanzas do not get an acknowledgement from the accepting party, neither the server nor the client. A presence stanza displays and notifies entities of status updates, playing the role of a signature in XMPP.

If there is interest in the presence of some JID (Jaber ID - a node address in XMPP), a client subscribes to its presence and each time a node sends a presence update, the client will be notified. An iq stanza pairs senders and receivers of messages. The message stanza is used to get some server background information, for example some information on the server or its recorded customers, or to enforce some configuration settings on the server. Its function is similar to the HTTP protocol GET and POST methods [77].

Since XMPP works on top of a persistent TCP connection and lacks efficient binary encoding, it has not been practical for use in lossy, low-power wireless networks often associated with IoT technologies. However, there has been much effort to make XMPP more suitable for IoT [81] [82] [83] .

HTTP

Originally introduced to reduce the inefficiencies of the FTP (File Transfer Protocol) [84], in the context of Web applications in the Internet data formation and presentation is generally based on HTTP (Hypertext Transfer Protocol). This protocol is widely used in the web and it works as a half-duplex, on a TCP connection with TLS security. The TCP handshake and HTTP header overhead make this kind of protocol inefficient in real-time data transfer or small messages.

This protocol is also not adequate in resource constrained environments as it is verbose in nature, meaning it causes an unnecessary overhead. For less amounts of network packets, power-optimized performance, less storage consumption and longer battery power there are popular application protocol alternatives like CoAP and MQTT.

CoAP

CoAP (Constrained Application Protocol) was designed by the IETF Constrained RESTful Environments (CoRE) working group for use on constrained devices with limited processing capabilities. Similar to HTTP, one of its most defining features is the use of the tested and well-accepted REST architecture [85].

With this feature, CoAP supports the request/response paradigm (figure 2.18) just like REST HTTP, and especially for constrained environments. CoAP is considered a lightweight protocol, so the headers, methods and state codes are all binary encoded, thus reducing the overhead of the protocol compared to many protocols.

It also runs over a less complex UDP transport protocol instead of TCP, further reducing the protocol overhead. When a CoAP client sends one or several CoAP requests to the server and receives the response, this response is not sent over a previously established connection, but exchanged asynchronously through CoAP messages. This reduction in security comes at a price in terms of trustworthiness. Nonetheless, it should be noted that due to the characteristics of reduced reliability, which is known when using UDP, the IETF created an additional standard document, opening the possibility for CoAP to run over TCP [86].

However, at the moment, this feature is still in its early stages. CoAP is based on a framework that is divided into two logically different layers. One of the layers, the so-called request/response layer, implements the RESTful paradigm and allows CoAP clients to use HTTP-like methods when sending requests. In other words, clients can use GET, PUT, POST or DELETE methods to manage the resources identified by the URI in the network [87].

As in HTTP, for its requests to obtain data from the server, for example, when obtaining the sensor values, the client will use the GET method with a URL of the server, and as a response it will receive a packet with this data. The request and responses are combined by means of a token; a token in the response has to be the same as the one defined in the request. It is also possible for a client to push data, for example updated sensor data, to a device using the POST method for its URL [77].

As we can see, at this layer CoAP uses the same methods as REST HTTP. The second layer is the one that distinguishes CoAP from HTTP. Because UDP does not ensure reliable connections, CoAP relies on its second structural layer for reliability, called the message layer, designed for retransmission of lost packets. This layer defines four types of messages: CON (Confirmable), NON (non-confirmable), ACK (Acknowledgement), and RST (reset) [77].

CON messages are used to ensure reliable communication, and require an acknowledgement from the receiving side with an ACK message. Precisely this feature that marks whether messages need acknowledgement is what allows QoS differentiation in CoAP, albeit in a limited way.

CoAP has an optional feature that can improve the request/response model by allowing clients to continue to receive changes to a requested resource from the server by adding an observe option to a GET request. With this option, the server adds the client to the list of observers of the specific resource, which will allow the client to receive notifications when the state of the resource changes [88] .

Rather than relying on repetitive polling to check for changes in the state of the resource, placing an observer flag in a CoAP client's GET request allows for a much closer interaction to a publish-subscribe paradigm with a server alerting a client when there are changes.

In an attempt to move even closer to a publish-subscribe paradigm, the IETF recently launched the Publish-Subscribe Broker project that extends the capabilities of CoAP to support nodes with long connectivity outages and/or uptime0, with preliminary performance evaluations showing promising results [89].

MQTT

MQTT(Message Queue Telemetry Transport) is one of the lightweight messaging protocols that also follows the publish-subscribe paradigm, which makes it quite suitable for resource-constrained devices and for non-ideal network connectivity conditions, such as with low bandwidth and high latency.

MQTT (Message Queue Telemetry Transport) runs on top of the TCP transport protocol, which guarantees it's reliability. It is established as a M2M (machine to machine) protocol meaning that connections between IoT devices are independent and can synchronize on their own [90]. This protocol implements a typical method with message update structure with a Publisher, Broker and Subscriber devices. As mentioned before, publishers send messages regarding a certain topic to the broker, which then sends the message to devices subscribed to that topic automatically so they can execute a specific function. A device can be a publisher and/or a subscriber.

Arvind Kumar and Suchi Jobari [91] have empirically proven how MQTT can be implemented as a lightweight protocol. Despite following a client/server approach like other reliable protocols such as HTTP, it reveals itself as more suitable solution for IoT applications when it comes to power consumption and bandwidth for devices with limited resources such as memory, processing power and others.

In the MQTT protocol clients are topic publishers/subscribers that communicate through TCP with the server that represents a broker which manages and authenticates device subscriptions. The

connection oriented nature of the TCP protocol introduces a large overhead. Moreover, MQTT uses text for topic names, which further increases it's overhead. However, when compared to HTTP it still holds a lighter header, which means MQTT comes with much lower power requirements, making it one of the most prominent protocol solutions in constrained environments [77].

MQTT was released by IBM, with its latest version MQTT v3.1 adopted for IoT by OASIS. Due to its simplicity, and a very small message header compared to other messaging protocols, it is often recommended as the communication solution of choice in IoT.

As previously mentioned, there are two communication parties in the MQTT architecture that usually assume the roles of publishers and subscribers, clients and servers/brokers. The clients will be the machines that can advertise messages, can subscribe to host services, or both. The client must know the broker it is connecting to, and for its subscriber role it must know the subject it is subscribing to. A customer subscribes to a specific subject in order to receive the corresponding messages. However, other customers can also subscribe to the same topic and receive the broker's updates with the arrival of new messages. The broker serves as a central component that accepts messages posted by clients and, with the help of topic and filtering, delivers them to the subscribed clients [77].

In MQTT, the communication usually happens between an actual broker and two MQTT clients, a publisher and a subscriber. For a device to play a broker role, it is necessary to install the MQTT broker library, for example Mosquitto broker, which is one of the best known open source MQTT brokers. Nevertheless, it should be noted that there are several other MQTT protocol brokers open for use, which differ through their implementation of the MQTT protocol. Some of them are Emqttd, ActiveMQ, HiveMQ, IBM MessageSight, JoramMQ, RabbitMQ, and VerneMQ [77].

Clients are implemented by installing MQTT client libraries. The publisher creates tagged topics in the Broker. Topics in MQTT are treated as a hierarchy, with strings separated by slashes indicating the level of the topic [92].

An MQTT publisher can publish messages for a defined set of topics. Here, the customer publishes the topic: topic/1. This information will be published to the broker who may temporarily store it in a local database. The subscriber interested in this topic sends a subscription message to a broker, specifying the same topic. For QoS, MQTT defines three QoS levels, QoS 0, 1, and 2 [93].

The choice of level can be defined either in the publication or in the body of the subscription message. QoS 0 delivers on a best-effort basis, without confirmation on receipt of the message. This is a choice in cases where some sensors collect telemetry information over a longer period of time, and where sensor values do not change significantly. It is then acceptable if sometimes messages are missing, because the overall sensor value is still known since most of the message updates have been received.

The next level of assurance is QoS 1, which ensures that messages will arrive, so an acknowledgement of the message is required. This means that the receiver must send an acknowledgement

of receipt, and if it does not arrive within a defined time period, the publisher will send a publication message again. The third option, QoS 2, guarantees that the message will be delivered exactly once, without duplication. The amount of resources required to process the MQTT packet increases with the higher QoS level chosen, so it is important to adjust the QoS choice to the specific network conditions [77].

Another important feature that MQTT offers is the possibility to store some messages for new subscribers by setting a retain flag on published messages. If there is no one interested in a topic about which the publisher sends updates, the broker will discard the published messages. But in some situations, especially when the state of the followed topic does not change frequently, it is useful to allow new subscribers to receive the information on that topic. In this default case, new subscribers would have to wait for the state to change to receive a message about the topic. By setting a retain flag to value: true, the broker is told to store the published message so that it can be delivered to new subscribers [77].

As already mentioned, TCP represents one of the pitfalls of the MQTT protocol since it can be critical for restricted devices. For this purpose, a solution has been proposed as MQTT for Sensor Networks (MQTT-SN) version that uses UDP and supports topic name indexing. This solution does not rely on TCP, but uses UDP as a faster, simpler and more efficient transport option over a wireless connection [94], [95].

The other important improved feature is the reduced size of payloads. This is done by numbering the data packets with numerical topic IDs instead of long topic names. The major drawback is that at the moment MQTT-SN is only supported by a few platforms, and there is only one known broker-free implementation, called Really Small Message Broker [96].

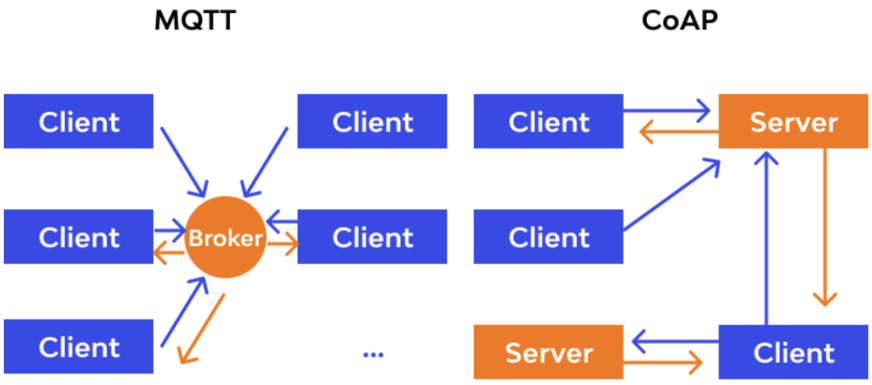


Figure 2.18: MQTT and CoAP architectures [17].

Our IoT system will feature the MQTT protocol since the chosen NB-IoT modem for the server communication BC660K-GL only supports MQTT commands, while the CoAP protocol commands are still under development.

Other modems have been analyzed by NIBBLE, such as the modem BC66 [97]. However, it has many protocols incorporated and is generally more advanced with functionalities far be-

yond our project's scope. For this reason, in our module, the BC660k-GL represents a simpler and straightforward solution as it only supports the preferred NB-IoT network and native MQTT support.

Eclipse Mosquitto

The message broker for the MQTT protocol will be the Eclipse Mosquitto broker. It is part of the Eclipse Foundation and it is a very versatile solution for devices ranging from low powered computers to more complex servers. This public broker is an open source implementation of a server that we'll be using which includes MQTT protocol versions such as 5.0, 3.1.1, and 3.1. It also includes a C and C++ client library, and the very popular mosquitto_pub and mosquitto_sub command line MQTT utilities for publishing and subscribing data between our module and the user interface, respectively [98].

2.5.2 Monitoring application with GUI

The concept of Human Computer Interaction (HCI) in the early 1960s brought out the idea of Graphical User Interface (GUI). And now it is at such a stage that in all our daily life parts we are expecting some graphical windows which will make our life easier [99].

Nowadays, GUI's are used in applications such as home security, in different industrial and energy management and in laboratory instruments. Some GUI's are also developed for the blind or visually impaired class [100] [101] [102] [103].

There have been several popular graphical software applications in the market like Visual Basic, C, Java applets, Adobe Flash CS4 etc; which integrate graphical user interface and script programming functions all-in-one for the interface designer and software developer to work seamlessly for a software development job [104].

Node-RED is a simple yet powerful tool for creating visual programming Internet of Things (IoT) applications that connect code blocks (or "nodes") to carry out tasks that may be developed in JavaScript. In order to communicate with IoT devices to control outputs and monitor sensors, Node-RED offers a dashboard (Node-RED dashboard) that may be used as a home automation platform.

Node-RED was developed by IBM's Emerging Technology Services for visualising and manipulating mappings between MQTT topics before being a more extended tool. The nodes may instantiate functions, buttons, gauges, sliders, charts beyond many other things. The different nodes will then follow a flow-based program, created by J. Paul Morrison in the 1970s[105] showcasing the graphical behavior of our application.

2.5.3 IoT Security

Giving how sensible consumers' privacy is, security and privacy play a crucial role in all markets around the world. Unfortunately in IoT systems security vulnerabilities are fairly common [106][107]. Information about an IoT device is collected regularly. This information describes

device details and travels through multiple hops in a network. Due to a diverse integration of services, these IoT devices are bound to attacks that can compromise the existing nodes in an IoT network.

When a large number of devices are infected, hackers can execute a wide range of attacks to cause immediate disruption to various systems, therefore, protection in multiple layers is paramount for an IoT solution, specially in emergency systems.

A secure IoT Application is also dependent on the security of lower layers of the network architecture. Finding what actually triggers security meltdowns is usually harder than fixing the problem[108].

SSL (Secure Socket Layer) and TLS (Transport Layer Security) are popular cryptographic protocols that are used to embed web communications with integrity, security, and resilience against unauthorized tampering. DTLS is a standardized security protocol designed to provide end-to-end secure communication among two peers in the presence of unreliable datagram transport protocols such as UDP.

For a more reliable IoT system, the main concepts that have to be considered and implemented are as follows [109].

- Confidentiality: Information confidentiality is an important objective, since unauthorized parties can't have access to stored data.
- Integrity: Data integrity needs to be checked in case of intentional or unintentional corruption of information;
- Availability: Access control or data availability can be compromised when there's an attempt of a DDoS (Distributed Denial of Service) attack;
- Authentication: Transmission and stored data should always be available for authorised IoT devices in the same network. Each device needs to authenticate and identify other devices. Authentication may be complex in constrained environments where many entities are present in the same network;

Cellular IoT devices follow GSMA standards for security measures and can be configured to use private network technologies like VPN to enhance security. IoT devices that use Wi-Fi follow the 802.11 standard encryption scheme called Wi-Fi Protected Access, WPA or more recently WPA2 and WP3. These methods fall into Security by design solutions, as they're implemented along with the development of the wireless technologies, but they're far from perfect and other solutions may be used for a more effective secure system like Machine Learning and Polymorphic Security[90].

AMQP applications use TLS for a secure communication and many other security-related properties making it the application protocol with strongest security [110].

One of the most important features of XMPP is its security features, which also makes it one of the most secure messaging protocols researched. Unlike the other protocols surveyed, for example MQTT and CoAP, where TLS and DTLS encryptions are not incorporated in the protocol

specifications, the XMPP specification already incorporates TLS mechanisms, which provides a reliable mechanism to ensure confidentiality and data integrity.

Further extensions to the XMPP standard specification include enhancements related to authentication, security, confidentiality, privacy and security access control. In addition to TLS, XMPP implements SASL, which ensures server validation through an XMPP-specific profile [111]. Since XMPP was originally conceived for delivering instant messaging, there are a few notable possible shortcomings. By using XML, the sheer size of the games makes it disadvantageous in bandwidth-constrained environments. Another disadvantage is the absence of reliable QoS guarantees.

Being a security engine, CoAP employs DTLS over its transport protocol UDP. It is based on the TLS protocol with the necessary changes to pass through an untrusted connection. The result is a secure version of the CoAP protocol. Most of the modifications compared to TLS include features that stop the termination of the connection in case of lost or out of order packets. As an example, there is the possibility to retransmit handshake messages [112].

The handshake process is very similar to that of TLS, with client and server exchanging 'hello' messages, but with the additional possibility of a server sending a verification query to ensure that the client was sending its 'hello' message from the authentic source address. However, this helps to protect against Denial of Service type attacks. Through these messages, the client and server also exchange supported encryption facts and keys, and agree on those that both parties support, which will be further used for data exchange protection during communication. Since DTLS was not originally designed for IoT devices and constrained devices, new versions optimised for lightweight devices have recently emerged [113][114].

Some of the optimization mechanisms of DTLS aimed at making it lighter include IPv6 over low power wireless network header compression mechanisms (6LoWPAN) to compress DTLS headers. Due to its limitations, the optimization of DTLS for IoT is still an open question [115].

Since it is designed to be a very lightweight protocol, MQTT does not provide encryption, and data is exchanged as plain text, which is clearly a problem from a security point of view. Therefore, encryption needs to be implemented as a separate feature, for example via TLS(Transport Layer Security) or SSL(Secure Sockets Layer) security, which on the other hand increases the overhead.

Authentication is implemented by many MQTT brokers, through one of the MQTT control type message packets, called CONNECT. Brokers require clients, when sending the CONNECT message, to set the username/password combination before validating the connection, or to reject it in case authentication was unsuccessful [77].

Overall, security is an ongoing effort for MQTT, and probably the most important, as MQTT is one of the most widely adopted and mature communication protocol solutions. Resolving the security issue would create an important and major advantage for MQTT compared to other available solutions [116].

Chapter 3

IoT System - Implementation and components

This chapter is intended as a summary of the conceivable work development in the context of the fire detection company. We'll overview requirements and components for the IoT module implementation and focus this project into three main stages, where the first stage will address the PCB design and assembly of hardware, the second stage is dedicated to the firmware functionality description while the third and final stage is meant to discuss our server communication and dashboard.

3.1 PCB Design

The printed circuit board (PCB) was developed through the electronic design automation (EDA) software tool EAGLE which stands for Easily Applicable Graphical Layout Editor.

For our module, the PCB is encapsulated within an area of 80 cm^2 in order to fit inside the standard NIBBLE FIREWALL along other components. The chosen microcontroller ESP32-WROOM-32E and the modem BC660K-GL as referenced in the Chapter 2 are suitable solutions for our IoT system. They are placed in the centre of the board for easier routing and all connectors are lined up so as to display a more practical implementation when tracing each board pins.

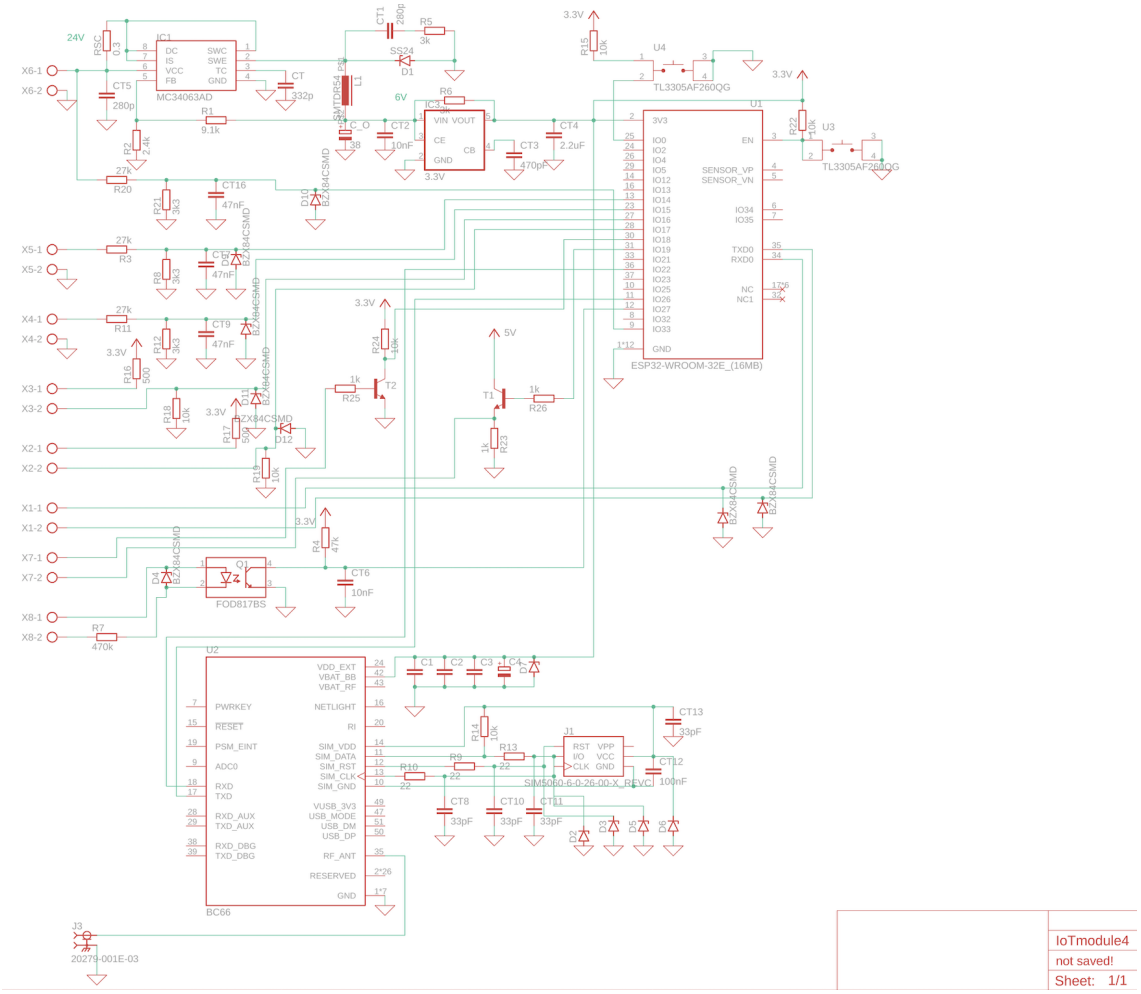


Figure 3.1: Schematic of our PCB module.

Protective Measures

To achieve a better performance, high frequency filters were implemented, as well as RC snubbers that reduce ringing and voltage spikes caused by inductors. The PCB noise is usually present due to changes in the load current, floating grounds or incorrect ground connections. The main noise sources are classified as ground bounce errors, triggered by fast switching in digital circuits, crosstalk errors that result in influenced signals that are too close in the PCB and EMI (Electro Magnetic Interference) errors which can be the most unpredictable type of errors since electromagnetic interference may have various sources [117].

Although it is not possible to completely clear our PCB board from any kind of noise, it is relevant to apply some protective measures in our board since it may interfere with the system's functionality. These measures include:

- Added ground and power planes as well-defined paths for both signals;

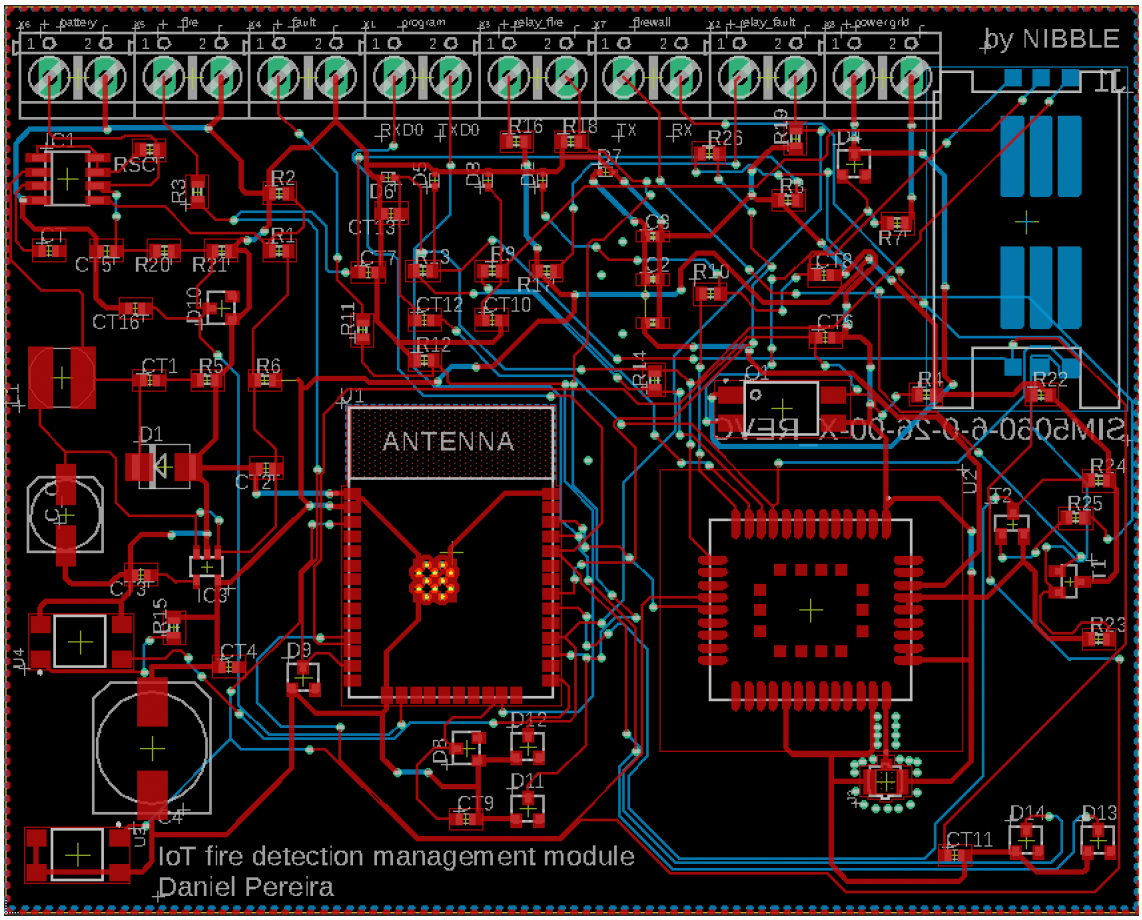


Figure 3.2: Layout of our PCB module.

- For reduced capacitive coupling which is a source of noise, implementing thin traces typically lower than 8 millimeters is needed. In our PCB they are 6 mils wide or 0.1524 mm.
- The separation of different functionality circuits within the board for easier routing and reduced electromagnetic interference.
- Power components should be next to each other on the same layer to avoid high inductance through via holes.
- The NB-IoT antenna connector must be also closer to the modem BC660K-GL for proper functionality and reduced noise.
- In order to minimise current spikes during signal switching and prevent bounce back to ground, decoupling (or bypass) capacitors can be implemented and positioned as close as feasible to each power pin of the active components.

After all the required measures applied to the board in order to enable an appropriate performance and protect our module from noise interference, the design data was sent to a PCB manufacture.

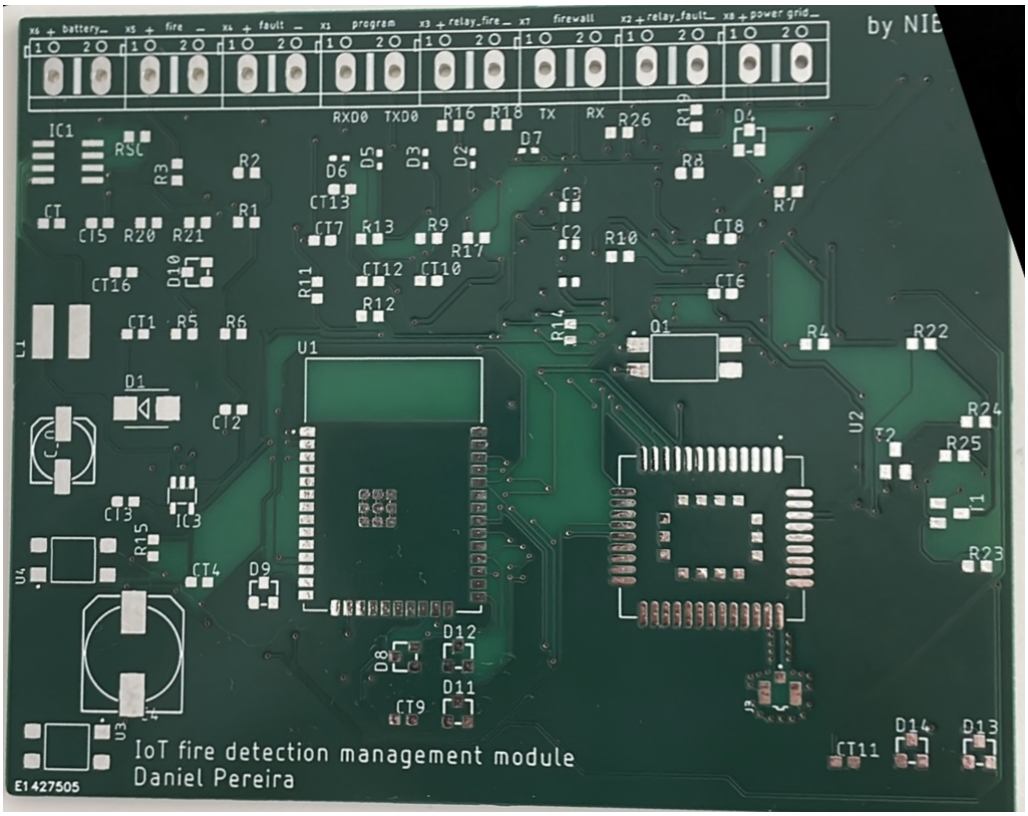


Figure 3.3: PCB board.

3.2 Firmware

Programming the hardware platform for our module can be done through different programming languages, however, for computationally intensive programs the low overhead and the layer of hardware abstraction are decisive factors to take into account when choosing how the firmware will be implemented. The C language allows for more efficient implementations of data structures and algorithms since the layer of abstraction from hardware is very slim. C is typically used in embedded systems which are composed by computer processors, input and output peripheral devices and memory. From a practical perspective the compiled C language code may be invoked from a simple boot code making it simple to execute [118]. The demand for system resources in the processor is low given the fast mapping of expressions and statements into sequenced instructions. The flexibility of C language allows for data structures and file systems to be written, modified, easily navigated and understood. Other programming languages like Perl, Ruby, Python and PHP showcase the availability and efficiency of the C language since they implement compilers, libraries and interpreters in C.

The firmware development will be done in C with the ESP-IDF, which is Espressif's IoT Development framework for the ESP32. The Espressif API reference for the ESP-IDF includes a wide range of libraries, functions and required drivers for our program peripherals and network functions.

Digital Reading

The digital reading is set as a direct reading of rising and falling edge signals from the monitored pins of either a voltage input or a relay input that indicate a fire, alarm, fault and power signal. Each level change on a signal will then be interpreted as a change of pin state, by routinely getting the level of each pin within a while loop.

Upon a level change, a *mqtt_publish()* function is set to send a message to the dashboard under a specific topic. The dashboard will then send an acknowledgement of the received message to better track any issue occurring in the public mqtt eclipse broker.

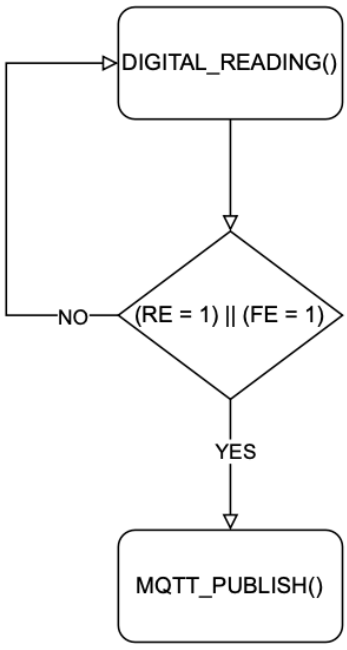


Figure 3.4: Digital reading flowchart.

Battery

To monitor the state of a battery of either 12V or 24V it was implemented an analog reading process by continuously sampling through the 12-bit ADC1 channel where the raw values for every reading range from value 0 to 4095. These values will then be converted to voltage values within the voltage range multiplied by a defined variable *BAT_NUM* that holds the number of batteries, a factor of 1 or 2 for a 12V or 24V battery, respectively.

The voltage values are updated in the same loop as the digital readings, from which the BATTERY pin may trigger the variables *low* or *dead* and publish an MQTT LOW or DEAD message under the topic *battery_state* if the voltage values are bellow *BAT_NUM* * *VIN_LOW* or *BAT_NUM* * *VIN_DEAD* . If both variables hold a 0 value than an MQTT FUNCTIONAL message is transmitted under the same topic.

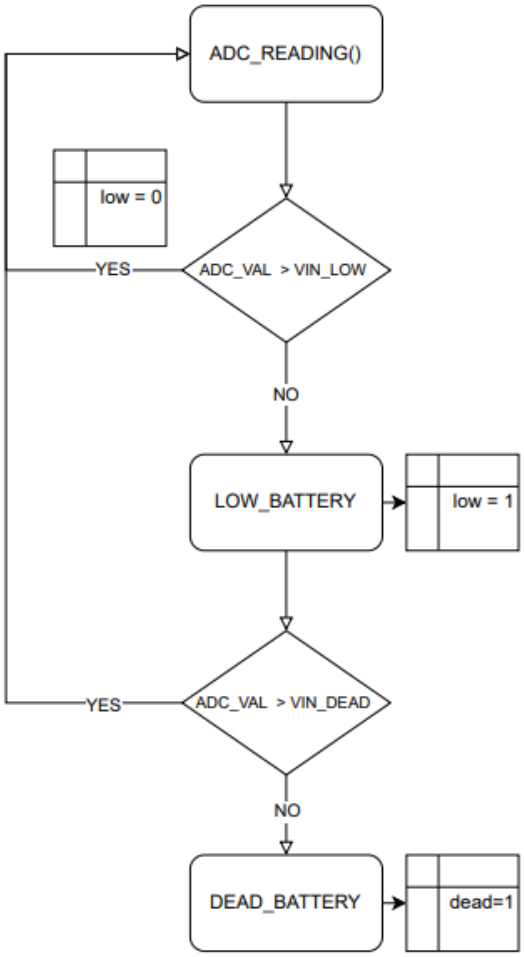


Figure 3.5: ADC reading flowchart.

The ADC values are then converted into a percentage value *current_battery_percentage* with the equation 3.1 and sent to a gauge chart in the intuitive dashboard for the IoT fire detection system.

$$current\_battery\_percentage = \frac{(current\_voltage - (BAT\_NUM * VIN\_DEAD)) * 100)}{(BAT\_NUM * VIN\_MAX) - (BAT\_NUM * VIN\_DEAD)} \quad (3.1)$$

Wi-Fi

When calling our *wifi_connection()* function, configuration variables are declared and the Wi-Fi credentials are attributed. These credentials include an SSID, a password and a number of connection retries, all defined in the ESP-IDF configuration menu as a pre-compilation process. Setting all the configurations for the Wi-Fi will then allow for the connection to start.

When faced with a failed attempt to connect to a router or access point, the system will retry to start the connection until reaching a maximum number of retries *retry_num* also defined in the ESP-IDF configuration menu.

Having reached the maximum number of retries the connection attempt is yielded and the NB-IoT connection takes place.

The Espressif's API MQTT functions are used for Wi-Fi connection on the IoT module for publishing to the eclipse broker/server a message with a specified topic, data length, QoS and retain flag, or to subscribe the IoT module to a defined topic and QoS.

Data length is set to 0 so therefor the length of each message is calculated from the payload string, the retain flag is attributed to 0 while the QoS is set to 1 which ensures that messages will arrive, given a message acknowledgement sent by the receiver. In this case, the Node-RED dashboard will be sending the acknowledgment again if a defined time period is reached.

Forwarding the connection to the NB-IoT network may be introduced upon the system initiation. An attempt for Wi-Fi communication may be successful after powering the IoT module. If this happens, a *wifi_connected* flag is set to 1 and a *NB_IoT* flag is set to 0.

However, if the system is faced with a power outage for example, after the successful Wi-Fi connection, then *wifi_connected* is set to 0 by a *wifi_event_handler()* function signalling a lost connection to the access point. Consequently, *NB_IoT* flag is set to 1 and the NB-IoT network and UART configuration for the BC660K-GL modem communication takes place.

Modem communication and NB-IoT

In the context of a failed or lost Wi-Fi connection the serial UART protocol is set to be configured at the pins GPIO22 and GPIO26 for communication between the ESP32-WROOM-32E and the BC660K-GL modem. Similarly to the MQTT publish functions for the Wi-Fi connection, the *nGSM_manage()* function is called routinely in the while loop to manage a state machine that initiates the NB-IoT communication through AT commands sent to the modem.

The function *GSM_ExecuteCmdResponse()* executes the AT commands and waits for the answer which is a string started with 'response_start' and ended with 'response_end' for every state.

- GSM_ST_INIT: It is the initial or idle state which will forward to GSM_ST_AT.
- GSM_ST_AT: Generates an "OK" response to check for the modem's activity.
- GSM_ST_QSCLK: Disables the sleep mode of the modem.
- GSM_ST_CPIN: Queries the modem for readiness to receive data.
- GSM_ST_CEREG: Registers the IoT module to the NB-IoT network.
- GSM_ST_OPEN: After registration a connection with a specified broker and port may be opened.
- GSM_ST_CONNECT: Enables the connection between the modem and the broker/server.
- GSM_ST_SUB : Subscribes to a specific topic and QoS.
- GSM_ST_WAIT : Forwards to GSM_ST_PUB only if *data_flag* is set to 1 by a *GSM_publish* function that is called in the while loop whenever a pin state is updated and no Wi-Fi connection is taking place.
- GSM_ST_PUB : Publishes to the eclipse broker/server a message with a specified topic, data length of 0, QoS set to 1 and a disabled retain flag.
- GSM_ST_AT_CMD_TIMEOUT : When the time period between any command sent and it's response is exceeded a timeout occurs leading to the GSM_ST_RESTART state.
- GSM_ST_AT_CMD_ERROR : Given any failed communication it will forward to the state GSM_ST_RESTART.
- GSM_ST_RESTART : Upon any error or timeout the RESTART state will reset the NB-IoT connection by forwarding to GSM_ST_AT.

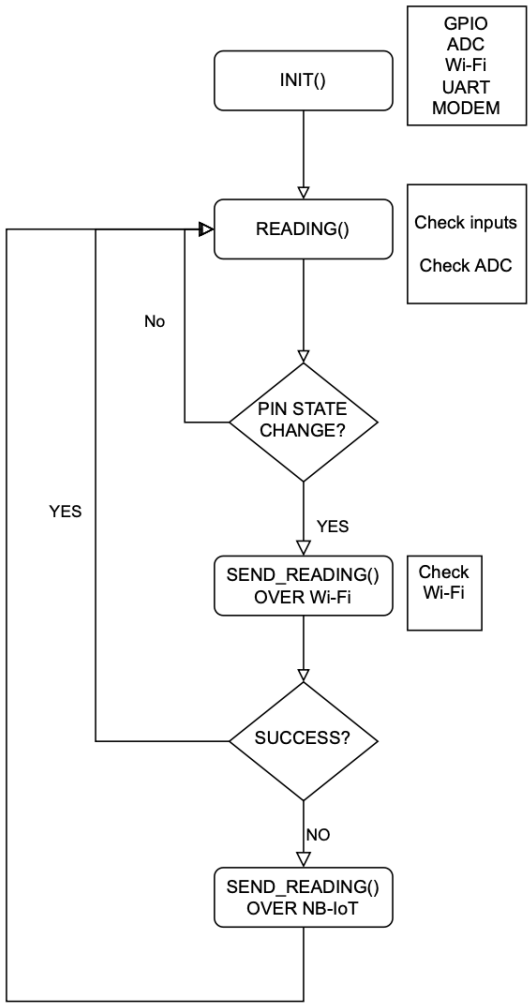


Figure 3.6: IoT system flowchart

3.3 Dashboard

The Node-RED dashboard is generated through a node-flow based program which can be exported as a JSON file. For the MQTT communication, the Eclipse Mosquitto MQTT broker is used as a server mediator for messages exchanged between the server's dashboard and our IoT module.

The dashboard is accessed through an internet connection of any kind, after locally initiating the mqtt://mqtt.eclipseprojects.io broker, through a default configuration listening on port 1883.

The MAC address of the IoT module is sent under the "Nibbleclient" topic as the client's identification. Each message published by the IoT module corresponds to a subtopic for FIRE, FAULT, BATTERY and POWER updates under the topic "emergency/#" subscribed by the Node-RED dashboard in the mqtt_subscriber nodes. The wildcard character '#' allows the dashboard to receive any subtopic message under the "emergency" topic.

The function nodes act as a filter for different publish topics upon receiving the message at the mqtt_subscriber nodes. The first function outputs 6 messages. Four messages regarding the fire and fault signaling as well as the power and module's battery, while the last two messages are destined for a gauge chart for the battery state and server acknowledgement message. The latter is filtered in the second function with regards to the payload and subtopic received and then sends the right acknowledgement message in the mqtt_publisher node for the topic "server_feedback", subscribed by the IoT module.

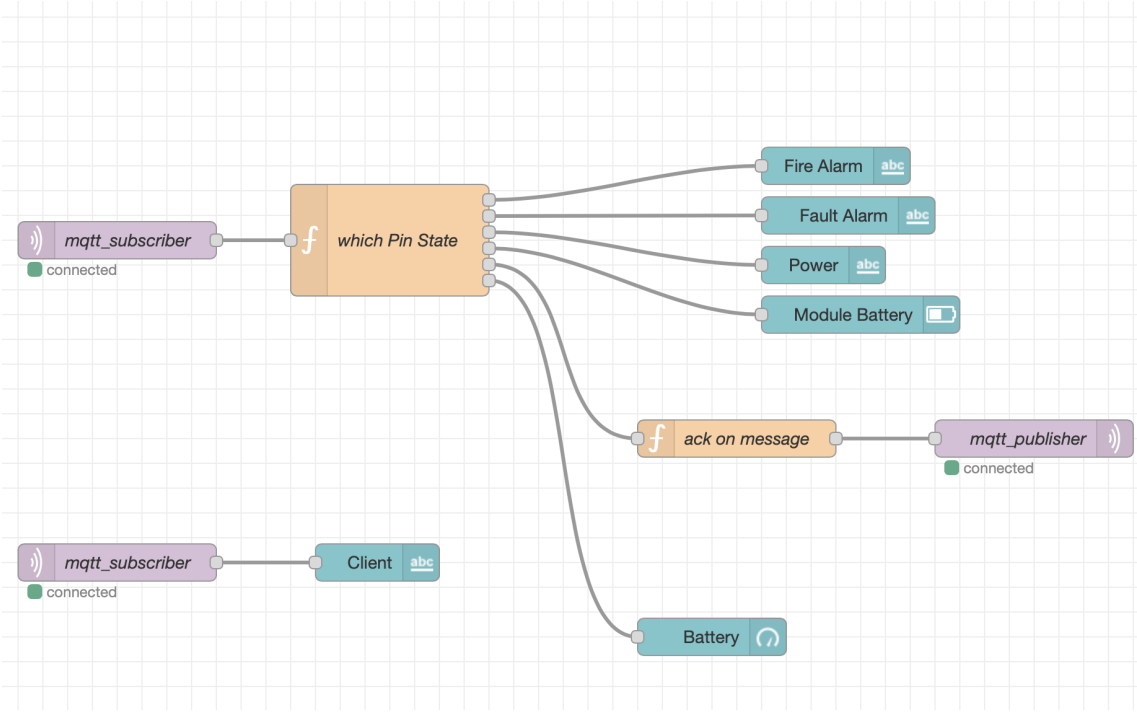


Figure 3.7: Node-RED data node flow

The gauge node displays the battery at 100% to 30% as functional, low between 30% to 10% and bellow that value the module's battery is considered dead.

Chapter 4

Validation and Performance

This chapter outlines a number of tests that were carried out to ensure that the system was functioning properly. All of the system's functions that were detailed in this article have been tested.

4.0.1 Data Visualization

Upon deployment, the Node-RED node flow demonstrates an intuitive UI (User Interface) for the normal function of the system. Each one of the fire detection system pins changed will be displayed in the UI, updating new data, published to the eclipse server by the available network.

The state display of the fire and fault alarms, as well as the power grid of the monitored establishment (230V) and the battery showcase the idle operation of the IoT module 4.1.

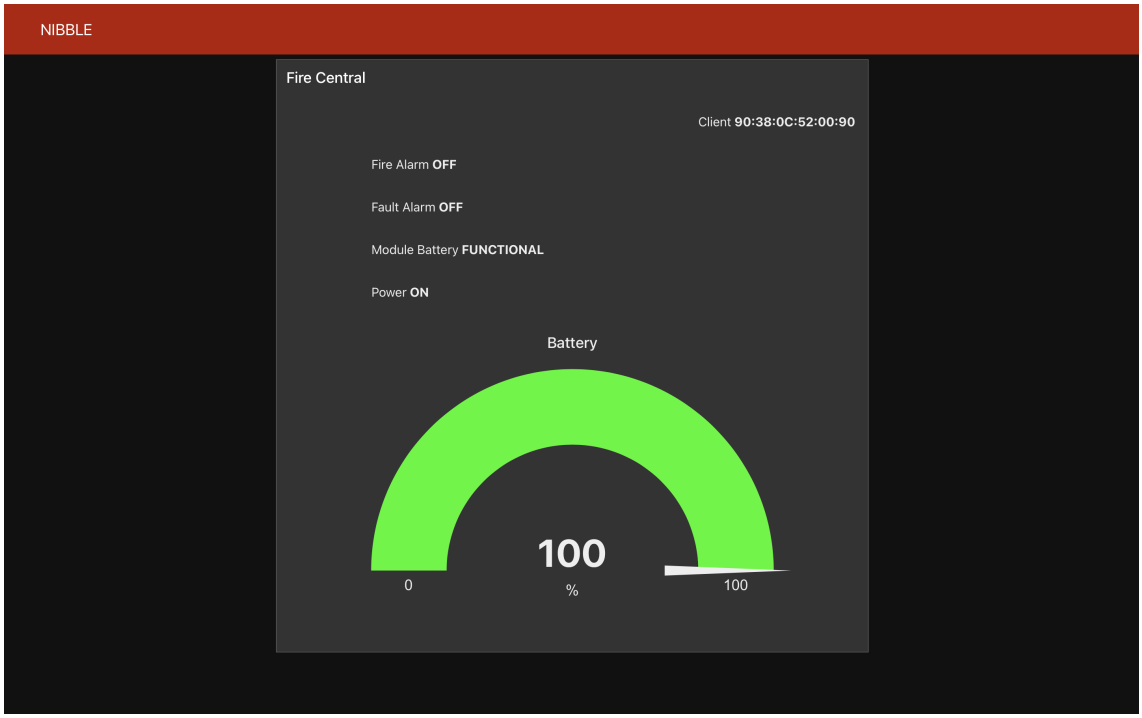


Figure 4.1: IoT module UI.

4.0.2 Battery

A fully charged 12V lead-acid battery may deliver up to 12.7V and 10.5V at 0% of it's capacity [119]. To simulate the battery behavior it has been applied through a power source values for 100% 4.1, 30% 4.2 and 0% 4.3 of a regular 12V lead-acid battery.

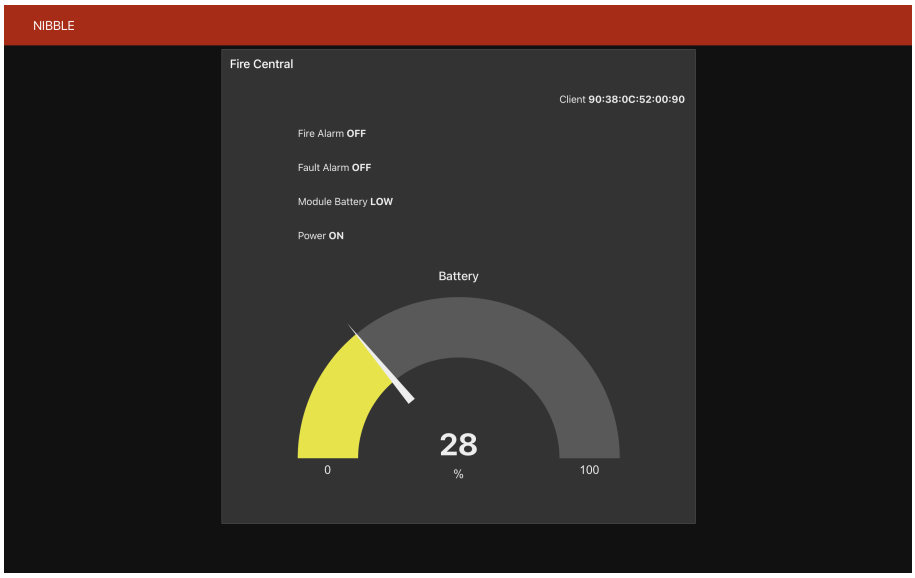


Figure 4.2: Low battery UI test.

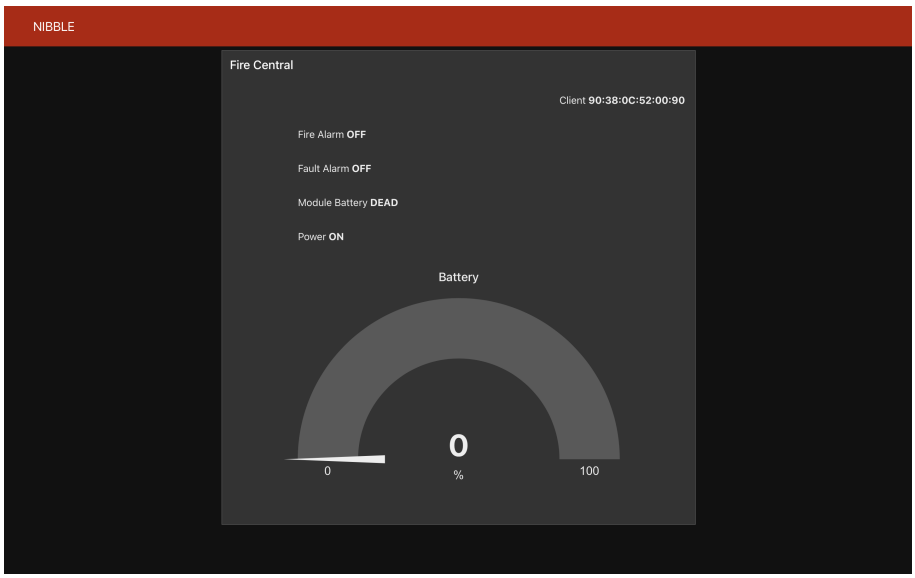


Figure 4.3: Dead battery UI test.

After experiencing a considerable oscillation in the chart, for an improved gauge response a software low-pass filter 4.1 is implemented. This way any sporadic deviations of the measured values are disregarded. The new *voltage* readings will then have a 10% influence in the updated battery voltage (*result*).

$$result = (result * 0.9) + (voltage * 0.1); \tag{4.1}$$

4.0.3 Wi-Fi and NB-IoT latency

After the set-up for each network, a latency test was conducted to empirically verify how Wi-Fi is indeed faster than the NB-IoT communication and how the latency might impact the fire detection system.

For an accurate measure a Pyhton program was set to listen to the serial port of the IoT module and print in computer terminal the current time after triggering a fire alarm. From the Node-RED dashboard a JavaScript code snippet printed to a debug console the current time upon receiving a message. The tabled 4.1 15 pair samples collected from each terminal hold the milliseconds within the same time reference a message is sent and received by the IoT module and the Node-RED dashboard, respectively.

Sent and Received time	
IoT module	Dashboard
30	240
242	404
472	656
663	827
934	1063
386	530
687	914
209	322
459	636
661	856
173	394
343	471
534	744
705	929
885	1048

Table 4.1: Wi-Fi Latency Test (ms)

These delays average at 177 milliseconds, which does not represent a relevant delay in the fire detection system since in the context of fire emergencies a millisecond delay can't change the outcome of fire destruction.

Another test was made to the NB-IoT connection with the same conditions as the previous Wi-Fi latency test. A NOS IoT SIM card was used, which provided communications of sufficient speeds. The tabled 4.2 values represent sent and received millisecond time intervals for a fire message from the IoT module to the Dashboard.

Sent and Received time	
IoT module	Dashboard
987	1716
288	1191
229	1084
930	1670
311	897
212	1085
380	912
520	1066
781	1296
542	1047
963	1419
943	1507
964	1429
184	718
484	1362

Table 4.2: NB-IoT Latency Test (ms)

These delays average at 645 milliseconds, and, as expected, they are higher than the measured Wi-Fi values. This difference between networks is explained by the low bandwidth of NB-IoT as previously referenced and how the routers, and the well matured Wi-Fi technology has developed reliable, faster communications.

4.0.4 Firewall Test

In order to have a more accurate performance test to the IoT System we've setup the module with a NIBBLE firewall 4.4.

The board is placed next to the 12 volt lead-acid battery and accommodates the structured fire detection central 4.4 as originally projected. The NB-IoT antenna is set inside the box just bellow the IoT module without compromising the signal.

As demonstrated by the connection scheme 4.5 the power (230V) is connected to the FIREWALL which in turn will signal the power state to the IoT module. The battery is connected to the board, the output relay of the FIREWALL is configured to the fault relay input in our module and the siren output of the FIREWALL is configured as a fire voltage input in our module since it's output is 24V. A NIBBLE alarm button serves as an actuator to trigger a fire signal at the output of the FIREWALL 4.7. The fault signaling is then tested by turning off the power connected to the FIREWALL 4.8.

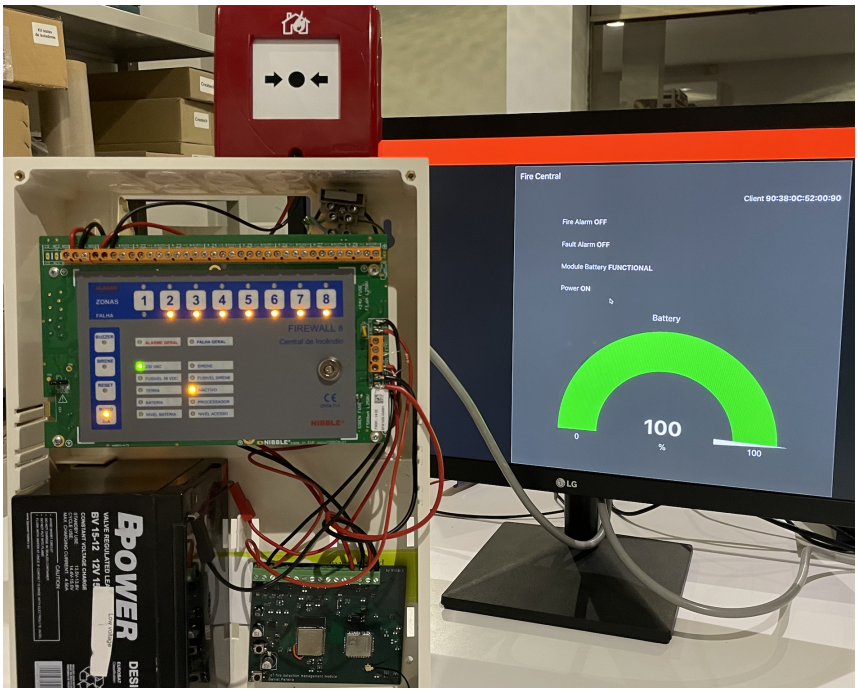


Figure 4.4: Default operation of IoT system.

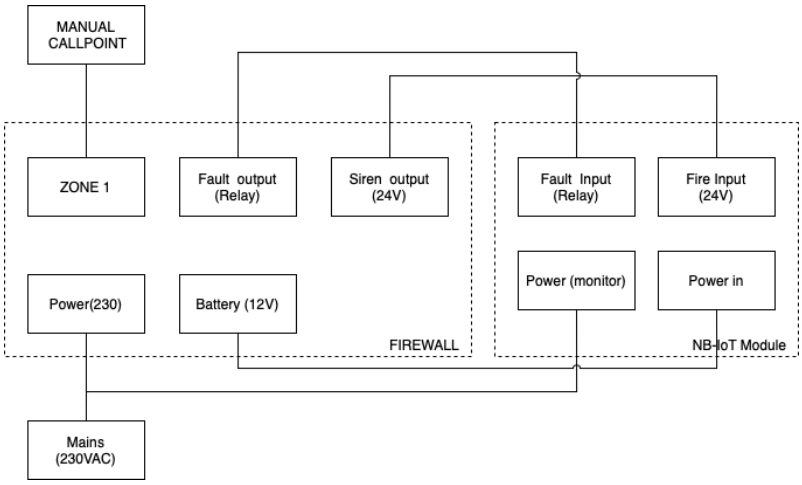


Figure 4.5: Connection scheme.

The testing setup is composed by a FIREWALL 8 [120] following the EN-54 standard, offering 8 different monitored zones with up to 32 fire detectors per zone, two programmable auxiliary outputs (per zone), two relays for fault or fire signaling. The FIREWALL is configured to a single zone only 4.6.

- 8 different monitored zones with up to 32 fire detectors per zone
- Two programmable auxiliary outputs (per zone)
- Two relays for fault or fire signaling

- A short and open circuit monitoring output for the siren of 24VDC with power consumption of up to 500mA.
- Configurable delay between fire detection and signaling allowing the avoidance of fake fire alarms which some detectors may generate during the day.
- A power supply of 24 VDC available by the fire detection central allowing external circuit connections with up to 500mA of power consumption.
- Ability to function with a spear power supply from a 12 VDC / 7Ah battery or two series 6 VDC / 12Ah batteries.
- An intuitive and simple control panel.

Figure 4.6: FIREWALL 8

After monitoring the full capacity of the 12 volt lead-acid battery, a load was applied to simulate a normal and gradual battery discharge.

As expected at 12.50V approximately 90% of the battery capacity is available 4.9 and, at 12.30V the gauge chart displays a battery capacity around 70%4.10.

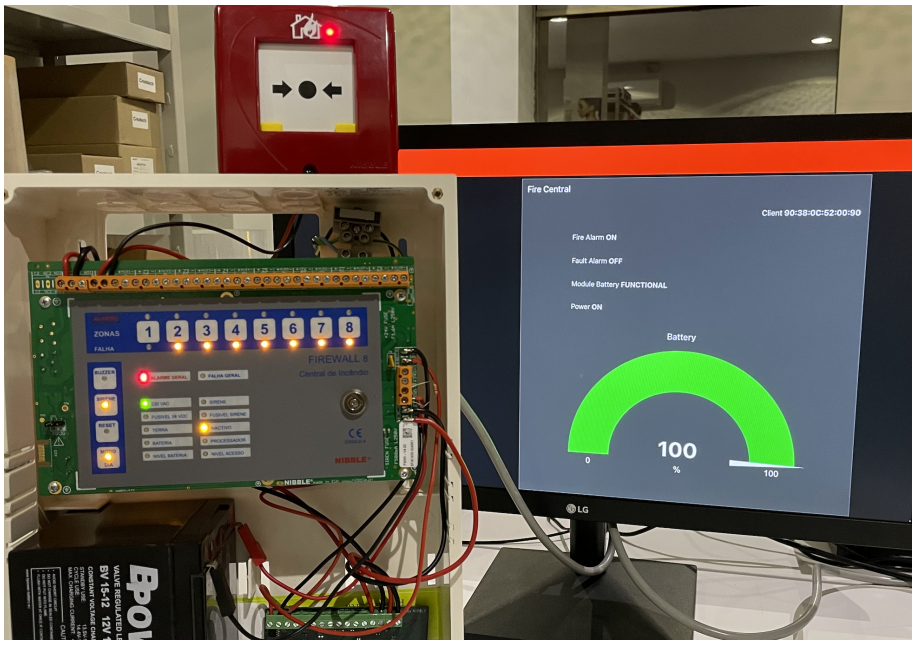


Figure 4.7: Fire signal triggered by the alarm button.

Figure 4.8: Fault signal triggered by power off.

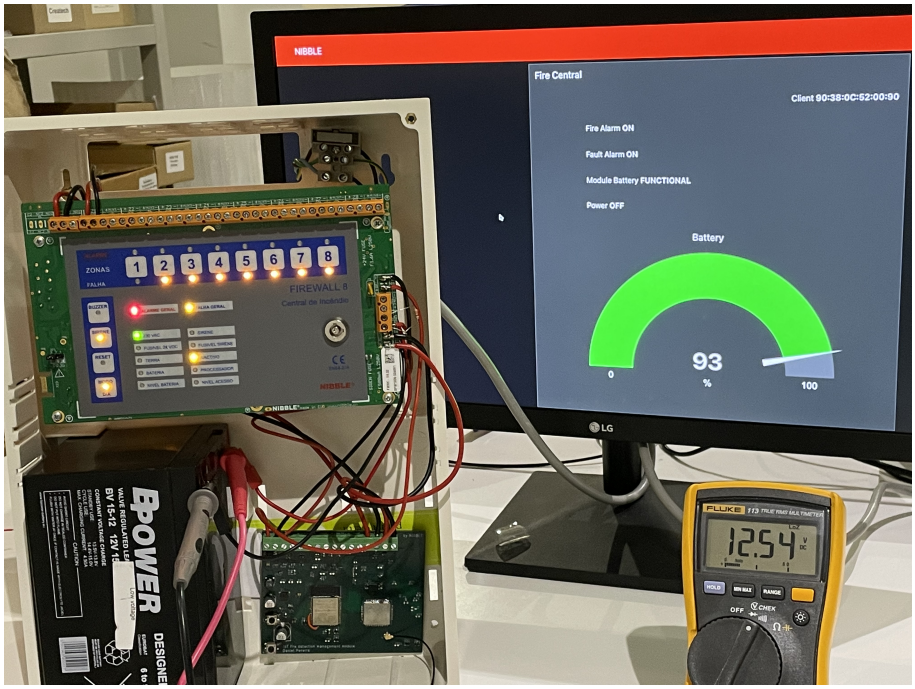


Figure 4.9: Battery discharge of approximately 10%.

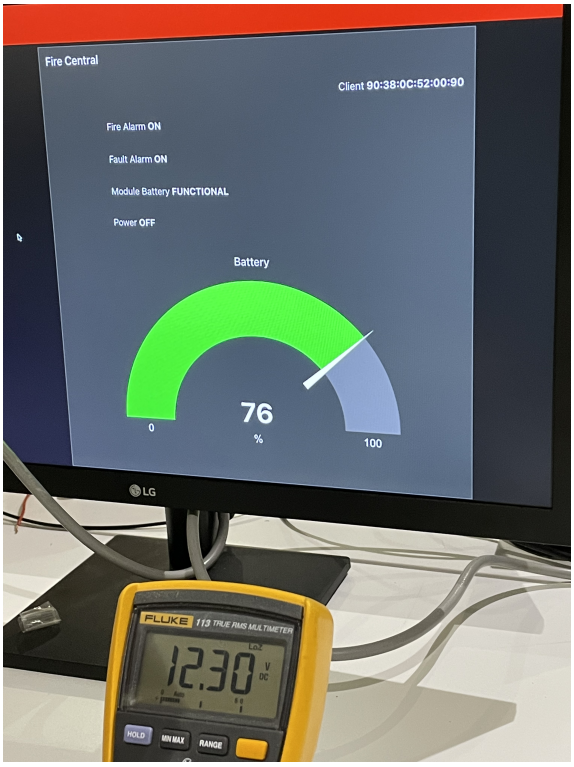


Figure 4.10: Battery discharge of almost 30%.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis report depicts different technologies and possible IoT solutions to implement, regarding specialized networks for many system architectures. Low energy solutions are more relevant for IoT devices and not all communication technologies perform in the best way possible when it comes to emergency systems.

As a modem based approach for a remote management of our fire detection system, NB-IoT appeared to be a communication technology that best suits NIBBLE requirements as a low energy, bandwidth and cost-efficient solution. It did, however, present higher levels of latency when compared to a no modem approach, using the Wi-Fi router to communicate with the IoT module. It has also been done a comparison between some application protocols that need to work on a low power and low memory system. Some development board implementations were presented and compared with regards to their cost efficiency and performance in IoT systems.

After conducting real environment tests with a NIBBLE firewall the results obtained showcase a reliable IoT solution that monitors fire detection systems, with low delay and multi network capabilities enabling the product's functionality in different environments for remote monitoring. The flexible network feature and fast signaling of managed states improve overall safety of a fire detection system.

5.2 Future Work

The IoT system's security may be improved by authentication either by password files, authentication plugins, and/or unauthorized/anonymous access. The MQTT broker can be installed and configured to require authentication enabling SSL/TLS and using a valid username and password before a connection is allowed.

For a faster deployed and scalable product, the dashboard can be accessed anywhere in the world with a cloud hosting provider that offers cloud computing services and a personal mosquitto MQTT broker, configured in a device such as a Raspberry Pi, while also requiring authentication from the client.

As a featured peripheral to the IoT module, serial communication may be integrated as well as different communication protocols for different fire detection centrals to obtain more details about the system that are not possible to acquire through the existent output signals, since they're not too specific. Protocols that allow the module to monitor various states in different zones of an establishment.The dashboard would then hold different tabs for each monitored zone.

Moreover, serial communication is not limited to only read output signals of a firewall, since it is possible to implement any protocol for communication to any other device. One example would be to implement a Modbus client, which could be used to access compatible devices.

References

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015. doi:10.1109/COMST.2015.2444095.

[2] Neeli Prasad and Anand Prasad. 2005.

[3] Zigbee Coordinator, howpublished = https://www.sciencedirect.com/topics/computer-science/zigbee-coordinator, note = Accessed: 2022-06-08.

[4] Wi-Fi CERTIFIED HaLow, howpublished = https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-halow, note = Accessed: 2022-06-08.

[5] Difference between Bluetooth and Zigbee, howpublished = https://www.geeksforgeeks.org/difference-between-bluetooth-and-zigbee/, note = Accessed: 2022-06-08.

[6] N. Ahmed, H. Rahman, and Md.I. Hussain. A comparison of 802.11ah and 802.15.4 for iot. *ICT Express*, 2(3):100–102, 2016. Special Issue on ICT Convergence in the Internet of Things (IoT). URL: https://www.sciencedirect.com/science/article/pii/S2405959516300650, doi:https://doi.org/10.1016/j.icte.2016.07.003.

[7] Imane Cheikh, Rachid Aouami, Essaid Sabir, Mohamed Sadik, and Sébastien Roy. Multi-layered energy efficiency in lora-wan networks: A tutorial. *IEEE Access*, 10:9198–9231, 2022. doi:10.1109/ACCESS.2021.3140107.

[8] Mobile IoT Deployment Map. https://www.gsma.com/iot/deployment-map/. Accessed: 2022-06-08.

[9] Badr Eddine Benhiba, Abdessalam Ait Madi, and Adnane Addaim. Comparative study of the various new cellular iot technologies. In *2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, pages 1–4, 2018. doi:10.1109/ICECOCS.2018.8610508.

[10] Sigfox, COVERAGE. https://www.sigfox.com/en/coverage. Accessed: 2022-06-08.

[11] LoRa Alliance, Coverage Operator Maps, howpublished = https://lora-alliance.org, note = Accessed: 2022-06-08.

[12] How to Choose the Best Development Kit: The Ultimate Guide for Beginners. https://predictabledesigns.com/how-to-choose-the-best-development-kit-the-ultimate-guide-for-beginners/. Accessed: 2022-02-21.

[13] Raspberry Pi Zero W, howpublished = https://www.raspberrypi.com/products/raspberry-pi-zero-w/, note = Accessed: 2022-08-08.

[14] ESP32-DevKitC V4 Getting Started Guide, howpublished = https://github.com/espressif/esp-idf/blob/master/docs/en/hw-reference/esp32/get-started-devkitc.rst, note = Accessed: 2022-08-08.

[15] The semiconductor shortage and its implication for euro area trade, production and prices. https://www.ecb.europa.eu//pub/economic-bulletin/focus/2021/html/ecb.ebbox202104_06~780de2a8fb.en.html. Accessed: 2022-06-06.

[16] BASICS OF UART COMMUNICATION, howpublished = https://www.circuitbasics.com/basics-uart-communication/, note = Accessed: 2022-06-08.

[17] CoAP Protocol Definition. https://www.wallarm.com/what/coap-protocol-definition. Accessed: 2022-02-21.

[18] António Augusto Araújo Gomes. Fundamentos da deteção automática de incêndios em edifícios parte 1. *Neutro à Terra*, (17), Jun. 2016. URL: https://parc.ipp.pt/index.php/neutroaterra/article/view/457, doi:10.26537/neutroaterra.v0i17.457.

[19] Fire Safety: Automatic vs. Manual Fire Alarm Systems, howpublished = https://www.nfpa.org, note = Accessed: 2022-06-08.

[20] Fire Safety: Automatic vs. Manual Fire Alarm Systems, howpublished = https://www.stanleysecurity.com/uk/blog/fire-safety-automatic-vs-manual-fire-alarm-systems, note = Accessed: 2022-06-08.

[21] Rogério Manuel Teixeira Resende et al. Detecção e alarme de incêndio: sistemas actuais. 2009.

[22] A Tanenbaum. Computer networks 4th ed. prentice hall. *Professional Technical Reference*, pages 3–17, 2002.

[23] Larry L Peterson and Bruce S Davie. Computer networks fifth edition: A systems approach. In *Morgan Kaufmann Publishers Inc.* 2011.

[24] James F Kurose and Keith W Ross. Computer networking. *A Top-Down*, 1986.

[25] Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010.

[26] Linnyer Beatrys Ruiz, Luiz Henrique A Correia, Luiz Filipe M Vieira, Daniel F Macedo, Eduardo F Nakamura, Carlos MS Figueiredo, Marcos Augusto M Vieira, Eduardo Habib Bechelane, Daniel Camara, AA Loureiro, et al. Arquiteturas para redes de sensores sem fio. *Tutorial of the simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos(SBRC)*, 2004.

[27] Antonio AF Loureiro, José Marcos S Nogueira, Linnyer Beatrys Ruiz, Raquel Aparecida de Freitas Mini, Eduardo Freire Nakamura, and Carlos Maurıcio Seródio Figueiredo. Redes de sensores sem fio. In *Simpósio Brasileiro de Redes de Computadores (SBRC)*, pages 179–226. sn, 2003.

[28] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[29] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, and Subhas Chandra Mukhopadhyay. Towards the implementation of iot for environmental condition monitoring in homes. *IEEE sensors journal*, 13(10):3846–3853, 2013.

[30] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[31] I Gartner. Gartner's 2015 hype cycle for emerging technologies identifies the computing innovations that organizations should monitor. *Gartner Web site. Available at: https://www. gartner. com/en/newsroom/press-releases/2015-08-18-gartners-2015-hype-cycle-for-emerging-technologies-identifies-the-computing-innovations-that-organizations-should-monitor (accessed 05 January 2021)*, 2015.

[32] Harald Sundmaeker, Patrick Guillemin, Peter Friess, Sylvie Woelfflé, et al. Vision and challenges for realising the internet of things. *Cluster of European research projects on the internet of things, European Commision*, 3(3):34–36, 2010.

[33] Hakima Chaouchi. *The internet of things: Connecting objects to the web*. John Wiley & Sons, 2013.

[34] Gil Press. Internet of things by the numbers: Market estimates and forecasts. *Forbes, available at http://www. forbes. com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts*, 2014.

[35] Feng Wang, Liang Hu, Jin Zhou, and Kuo Zhao. A survey from the perspective of evolutionary process in the internet of things. *International Journal of Distributed Sensor Networks*, 11(3):462752, 2015.

[36] Ala Al-Fuqaha, Abdallah Khreishah, Mohsen Guizani, Ammar Rayes, and Mehdi Mohammadi. Toward better horizontal integration among iot services. *IEEE Communications Magazine*, 53(9):72–79, 2015. doi:10.1109/MCOM.2015.7263375.

[37] Pallavi Sethi and Smruti R. Sarangi. Internet of things: Architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017:9324035, Jan 2017. URL: https://doi.org/10.1155/2017/9324035, doi:10.1155/2017/9324035.

[38] Nandakishore Kushalnagar, Gabriel Montenegro, and Christian Schumacher. Ipv6 over low-power wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals. 2007.

[39] Bruno P Santos, Lucas AM Silva, Clayson SFS Celes, João B Borges Neto, Bruna S Peres, Marcos Augusto M Vieira, Luiz Filipe M Vieira, Olga N Goussevskaia, and Antonio AF Loureiro. Internet das coisas: da teoria à prática. 2016.

[40] Jean-Philippe Vasseur and Adam Dunkels. *Interconnecting smart objects with ip: The next internet*. Morgan Kaufmann, 2010.

[41] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, page 1–14, New York, NY, USA, 2009. Association for Computing Machinery. URL: https://doi.org/10.1145/1644038.1644040, doi:10.1145/1644038.1644040.

[42] Nadeem Javaid, Akmal Javaid, Imran Ali Khan, and Karim Djouani. Performance study of etx based wireless routing metrics. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–7, 2009. doi:10.1109/IC4.2009.4909163.

[43] Philip Levis, Neil Patel, David Culler, and Scott Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation*, volume 25, pages 37–52, 2004.

[44] Adam Chlipala, Jonathan Hui, and Gilman Tolle. Deluge: data dissemination for network reprogramming at scale. *University of California, Berkeley, Tech. Rep*, 2004.

[45] Bruno P. Santos, Marcos A. M. Vieira, and Luiz F. M. Vieira. extend collection tree protocol. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1512–1517, 2015. doi:10.1109/WCNC.2015.7127692.

[46] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs. *Disruptive technologies: Advances that will transform life, business, and the global economy*, volume 180. McKinsey Global Institute San Francisco, CA, 2013.

[47] History of IEEE. https://www.ieee.org/about/ieee-history.html. Accessed: 2022-06-08.

[48] Blue, howpublished = https://www.androidauthority.com/history-bluetooth-explained-846345/, note = Accessed: 2022-06-08.

[49] Bluetooth, howpublished = https://www.bluetooth.com/learn-about-bluetooth/tech-overview/, note = Accessed: 2022-06-08.

[50] What is zigbee and what devices can it work with within the smart home? Last accessed 01 December 2020., howpublished = https://www.smarthome.news/news/other-systems/what-is-zigbee-and-compatible-smart-home-devices, note = Accessed: 2022-06-08.

[51] Wi-Fi Member Companies. https://www.wi-fi.org/membership/member-companies. Accessed: 2022-06-08.

[52] WiFi Alliance, howpublished = https://www.wi-fi.org, note = Accessed: 2022-06-08.

[53] Wi-Fi CERTIFIED HaLow. https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-halow. Accessed: 2022-06-08.

[54] Johan Bergman Joachim Sachs Olof Liberg, Marten Sundberg and EricWang. *The Cellular Internet of Things*. London, UK, Academic Press, 2017.

[55] A Complete Overview of 2G 3G Sunsets. https://1ot.mobi/resources/blog/a-complete-overview-of-2g-3g-sunsets. Accessed: 2022-06-08.

[56] Brandon Foubert and Nathalie Mitton. Long-range wireless radio technologies: A survey. *Future Internet*, 12(1), 2020. URL: https://www.mdpi.com/1999-5903/12/1/13, doi:10.3390/fi12010013.

[57] Narrowband IoT (NB-IoT), howpublished = https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/resources/innovation-technology/nb-iot, note = Accessed: 2022-06-08.

[58] Bruno Odate Tavares. Otimizando procedimentos com uso da iot no gerenciamento inteligente das salas de aula da esmat. 2020.

[59] FPGA Vs Microcontroller-Which Is Better For Your Needs, howpublished = https://www.ourpcb.com/fpga-vs-microcontroller.html, note = Accessed: 2022-06-08.

[60] non-volatile-memory-market, howpublished = https://www.alliedmarketresearch.com/non-volatile-memory-market, note = Accessed: 2022-06-08.

[61] OTP EPROM, howpublished = https://www.microchip.com/en-us/products/memory/otp-eprom, note = Accessed: 2022-06-08.

[62] FRAMs as alternatives to flash memory in embedded designs, howpublished = https://www.embedded.com/frams-as-alternatives-to-flash-memory-in-embedded-designs/, note = Accessed: 2022-06-08.

[63] Simon Monk. *Programação com Arduino II: Passos avançados com sketches*. Bookman Editora, 2015.

[64] What is Arduino? https://www.arduino.cc/en/Guide/Introduction. Accessed: 2022-02-21.

[65] Indira Knight. *Conectando o Arduino à web: Desenvolvimento de frontend usando JavaScript*. Novatec Editora, 2018.

[66] Michael McRoberts. *Arduino básico*. Novatec Editora, 2018.

[67] Jeremy Blum. *Explorando o Arduino: Técnicas e ferramentas para mágicas de engenharia*. Alta Books Editora, 2018.

[68] Simon Monk. *Programação com Arduino: começando com Sketches*. Bookman Editora, 2013.

[69] Desenvolvimento de um sistema IoT com comunicação via App/Cloud para monitorização de uma cama médica. https://hdl.handle.net/10216/132691. Accessed: 2022-02-21.

[70] Official IoT Development Framework, howpublished = https://www.espressif.com/en/products/sdks/esp-idf, note = Accessed: 2022-06-08.

[71] ESP32-WROOM-32 Wi-Fi Bluetooth Module, howpublished = https://www.digikey.pt/en/product-highlight/s/schtoeta/esp32-wroom-32-wi-fi-bluetooth-module, note = Accessed: 2022-08-08.

[72] API Reference, howpublished = https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html, note = Accessed: 2022-06-08.

[73] BC660K-GL Hardware Design, howpublished = https://www.quectel.com/wp-content/uploads/2021/04/quectel_bc660k-gl_hardware_design_v1.0.pdf, note = Accessed: 2022-06-08.

[74] Gurinder Singh and Lakshmi Munukutla. Interfacing the usb printer interface using vinculum host controller. In *2009 Annual Conference & Exposition*, pages 14–783, 2009.

[75] Soumyajit Datta, Sumana Chowdhuri, and Jitendranath Bera. Development of usb compatible pc based pd motor controller for chemical process. In *2012 1st International Conference on Power and Energy in NERIST (ICPEN)*, pages 1–4, 2012. doi:10.1109/ICPEN.2012.6492340.

[76] Andrew Foster. Messaging technologies for the industrial internet and the internet of things whitepaper. a comparison between dds, amqp, mqtt, jms, rest and coap. *Messaging Technologies Whitepaper*, 2014.

[77] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Comput. Surv.*, 51(6), jan 2019. URL: https://doi.org/10.1145/3292674, doi:10.1145/3292674.

[78] Joel L Fernandes, Ivo C Lopes, Joel JPC Rodrigues, and Sana Ullah. Performance evaluation of restful web services and amqp protocol. In *2013 Fifth international conference on ubiquitous and future networks (ICUFN)*, pages 810–815. IEEE, 2013.

[79] Erik Lindén. A latency comparison of iot protocols in mes, 2017.

[80] Adrian Hornsby and Rod Walsh. From instant messaging to cloud computing, an xmpp review. In *IEEE international symposium on consumer electronics (ISCE 2010)*, pages 1–6. IEEE, 2010.

[81] Daniel Schuster, Philipp Grubitzsch, Dominik Renzel, István Koren, Ronny Klauck, and Michael Kirsche. Global-scale federated access to smart objects using xmpp. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, pages 185–192. IEEE, 2014.

[82] Xiaoping Che and Stephane Maag. A passive testing approach for protocols in internet of things. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 678–684. IEEE, 2013.

[83] Adrian Hornsby and Eloi Bail. μxmpp: Lightweight implementation for low power operating system contiki. In *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pages 1–5. IEEE, 2009.

[84] Joe Touch, John Heidemann, and Katia Obraczka. Analysis of http performance. *URL: http://www. isi. edu/touch/pubs/http-perf96/>, ISI Research Report ISI/RR-98-463,(original report dated Aug. 1996), USC/Information Sciences Institute*, 1998.

[85] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). Technical report, 2014.

[86] Carsten Bormann, Simon Lemay, Hannes Tschofenig, Klaus Hartke, Bilhanan Silverajan, and B Raymor. Coap (constrained application protocol) over tcp, tls, and websockets. Technical report, 2018.

[87] Hoai Viet Nguyen and Luigi Lo Iacono. Rest-ful coap message authentication. In *2015 International Workshop on Secure Internet of Things (SIoT)*, pages 35–43, 2015. doi:10.1109/SIOT.2015.8.

[88] Noélia Correia, David Sacramento, and Gabriela Schütz. Dynamic aggregation and scheduling in coap/observe-based wireless sensor networks. *IEEE Internet of Things Journal*, 3(6):923–936, 2016. doi:10.1109/JIOT.2016.2517120.

[89] Michael Koster, Ari Keränen, and Jaime Jimenez. Publish-Subscribe Broker for the Constrained Application Protocol (CoAP). Internet-Draft draft-ietf-core-coap-pubsub-10, Internet Engineering Task Force, May 2022. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-ietf-core-coap-pubsub/10/.

[90] Derek Johnson and Mohammed Ketel. Iot: Application protocols and security. *International Journal of Computer Network & Information Security*, 11(4), 2019.

[91] Arvind Kumar and Suchi Johari. Push notification as a business enhancement technique for e-commerce. In *2015 Third International Conference on Image Information Processing (ICIIP)*, pages 450–454, 2015. doi:10.1109/ICIIP.2015.7414815.

[92] Nasi Tantitharanukul, Kitisak Osathanunkul, Kittikorn Hantrakul, Part Pramokchon, and Paween Khoenkaw. Mqtt-topics management system for sharing of open data. In *2017 International Conference on Digital Arts, Media and Technology (ICDAMT)*, pages 62–65, 2017. doi:10.1109/ICDAMT.2017.7904935.

[93] Jorge E. Luzuriaga, Juan Carlos Cano, Carlos Calafate, Pietro Manzoni, Miguel Perez, and Pablo Boronat. Handling mobility in iot applications using the mqtt protocol. In *2015 Internet Technologies and Applications (ITA)*, pages 245–250, 2015. doi:10.1109/ITechA.2015.7317403.

[94] Andy Stanford-Clark and Hong Linh Truong. Mqtt for sensor networks (mqtt-sn) protocol specification. *International business machines (IBM) Corporation version*, 1(2):1–28, 2013.

[95] Kannan Govindan and Amar Prakash Azad. End-to-end service assurance in iot mqtt-sn. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 290–296, 2015. doi:10.1109/CCNC.2015.7157991.

[96] Yiming Xu, V. Mahendran, Wei Guo, and Sridhar Radhakrishnan. Fairness in fog networks: Achieving fair throughput performance in mqtt-based iots. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 191–196, 2017. doi:10.1109/CCNC.2017.7983104.

[97] BC66 Hardware Design, howpublished = https://www.quectel.com/wp-content/uploads/2021/03/quectel_bc66_hardware_design_v1.4.pdf, note = Accessed: 2022-06-08.

[98] Roger A. Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265, 2017. URL: https://doi.org/10.21105/joss.00265, doi:10.21105/joss.00265.

[99] Anker Helms Jørgensen and Brad A. Myers. User interface history. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, page 2415–2418, New York, NY, USA, 2008. Association for Computing Machinery. URL: https://doi.org/10.1145/1358628.1358696, doi:10.1145/1358628.1358696.

[100] Harnani Hassan, Raudah Abu Bakar, and Ahmad Thaqib Fawwaz Mokhtar. Face recognition based on auto-switching magnetic door lock system using microcontroller. In *2012 International Conference on System Engineering and Technology (ICSET)*, pages 1–6, 2012. doi:10.1109/ICSEngT.2012.6339345.

[101] M. Lavergne. Graphical user interface for next generation power systems. In *INTELEC. Twenty-Second International Telecommunications Energy Conference (Cat. No.00CH37131)*, pages 109–112, 2000. doi:10.1109/INTLEC.2000.884236.

[102] Yu-Chen Chen, Yuan-Hsiang Chang, and Te-Chuan Wang. A real-time computer graphical user interface for advanced boiling water reactor case comparison using maap software. In *2010 International Computer Symposium (ICS2010)*, pages 274–278, 2010. doi:10.1109/COMPSYM.2010.5685505.

[103] Scott Koziol, Craig Schlottmann, Arindam Basu, Stephen Brink, Csaba Petre, Brian Degnan, Shubha Ramakrishnan, Paul Hasler, and Aurele Balavoine. Hardware and software infrastructure for a family of floating-gate based fpaas. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2794–2797, 2010. doi:10.1109/ISCAS.2010.5536992.

[104] Andisheh Feizi and Chui Yin Wong. Usability of user interface styles for learning a graphical software application. In *2012 International Conference on Computer Information Science (ICCIS)*, volume 2, pages 1089–1094, 2012. doi:10.1109/ICCISci.2012.6297188.

[105] Node Red. https://nodered.org/about/. Accessed: 2022-06-08.

[106] Fire Safety: Automatic vs. Manual Fire Alarm Systems, howpublished = https://www.wired.com/story/namewreck-iot-vulnerabilities-tcpip-millions-devices/, note = Accessed: 2022-06-08.

[107] Fire Safety: Automatic vs. Manual Fire Alarm Systems, howpublished = https://www.wired.com/story/access7-iot-vulnerabilities-medical-devices-atms/, note = Accessed: 2022-06-08.

[108] Fire Safety: Automatic vs. Manual Fire Alarm Systems, howpublished = https://www.wired.com/story/kalay-iot-bug-video-feeds/, note = Accessed: 2022-06-08.

[109] Alaa Ahmed Abbood, Qahtan Makki Shallal, and Mohammed A Fadhel. Internet of things (iot): a technology review, security issues, threats, and open challenges. *Indonesian Journal of Electrical Engineering and Computer Science*, 20(3):1685–1692, 2020.

[110] Jeddou Sidna, Baina Amine, Najid Abdallah, and Hassan El Alami. Analysis and evaluation of communication protocols for iot applications. In *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*, SITA'20, New York, NY, USA, 2020. Association for Computing Machinery. URL: https://doi.org/10.1145/3419604.3419754, doi:10.1145/3419604.3419754.

[111] Davide Conzon, Thomas Bolognesi, Paolo Brizzi, Antonio Lotito, Riccardo Tomasi, and Maurizio A Spirito. The virtus middleware: An xmpp based architecture for secure iot communications. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE, 2012.

[112] Eric Rescorla and Nagendra Modadugu. Rfc 6347: Datagram transport layer security version 1.2. *Internet Engineering Task Force (IETF)*, pages 2070–1721, 2012.

[113] Mukul Panwar and Ajay Kumar. Security for iot: An effective dtls with public certificates. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 163–166, 2015. doi:10.1109/ICACEA.2015.7164688.

[114] Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, and Thiemo Voigt. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal*, 13(10):3711–3720, 2013.

[115] Vishwas Lakkundi and Keval Singh. Lightweight dtls implementation in coap-based internet of things. In *20th Annual International Conference on Advanced Computing and Communications (ADCOM)*, pages 7–11. IEEE, 2014.

[116] Christian Lesjak, Daniel Hein, Michael Hofmann, Martin Maritsch, Andreas Aldrian, Peter Priller, Thomas Ebner, Thomas Ruprechter, and Günther Pregartner. Securing smart maintenance services: Hardware-security and tls for mqtt. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1243–1250, 2015. doi:10.1109/INDIN.2015.7281913.

[117] Using C for CGI Programming. https://www.proto-electronics.com/blog/top-5-rules-pcb-noise-reduction. Accessed: 2022-06-08.

[118] Using C for CGI Programming. https://www.linuxjournal.com/article/6863. Accessed: 2022-06-08.

[119] DISCHARGE AND BATTERY CAPACITY, howpublished = https://www.rebel-cell.com/knowledge-base/the-discharge-and-capacity-of-batteries/, note = Accessed: 2022-06-08.

[120] Manual de utilização, CENTRAL DE DETEÇÃO DE INCÊNDIO, howpublished = https://feelsafesecurity.pt/wp-content/uploads/2019/12/manual_firewall_8z_pt_v4_5_1.pdf, note = Accessed: 2022-08-08.