Smart IHU: an open source IoT project for wireless sensor network based on the Raspberry PI platform – Physical design and implementation

Karaklas Kostas

SID: 3307160004

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Communications and Cybersecurity

March, 2019

THESSALONIKI – GREECE

INTERNATIONAL
HELLENIC
UNIVERSITY

Smart IHU: an open source IoT project for wireless sensor network based on the Raspberry PI platform – Physical design and implementation

Karaklas Kostas

SID: 3307160004

Supervisor:	Dr. Stavros Stavrinides
Supervising Committee Members:	Prof. G. Evangelidis, Dr. S. Stavrinides, Dr. D. Baltatzis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Communications and Cybersecurity

March, 2019

THESSALONIKI – GREECE

This page intentionally left blank

Abstract

Internet of Things was only a concept a few decades ago, a concept that became fruition in the past couple of years, and has enabled the creation of commercial IoT devices, but more importantly advanced industrial applications in many sectors. Everything of course was possible with the evolvement of the Internet globally and advancement of other technologies, like wireless communications, microelectromechanical systems and auxiliary technologies such as artificial intelligence and blockchain. While technologies that affect Internet of Things are important for the advancement of IoT, so is a community that supports and advances the field, which is the goal of the Raspberry Pi foundation and partly the project of this thesis. By creating an open-source sensor network with the help of the Raspberry Pi platform, we are researching the viability of such a system and hopefully to interest students in the platform and Internet of Things in general.

Acknowledgements

I would like to thank my family for their patience and support over the years. Also, my professor Mr. Stavrinides for supervising this thesis and for his support in the overall project, my friends Giorgos, Panos & Akis for helping me over the years when I am stuck debugging and/or troubleshooting and my friend Persefoni for pushing me to continue evolving my academic prowess. Finally, I would like to thank my fellow student, Giorgos Sarigiannis for his help and support in our collaboration in completing this two-thesis project

Contents

Table of Figures

List of Tables

Introduction

Computer science is unique in the way that knowledge is acquired. There is a common joke on the Internet that says that programmers just google better than other people do and as a programmer and self-proclaimed geek, I can attest to that. Of course, all of that is possible because of people's need to share that knowledge and create communities to support projects and technologies, most of the time without gain, which is the so-called free/open source movement.

Therefore, one part of this thesis is about creating a project by using open source tools and technologies, not because this author is against paid software, on the contrary, but because open source software is essential to the advancement of computer science development. One of the main reasons for the choice of the Raspberry Pi for this project was exactly because the foundation behind the Raspberry Pi had the same goal, to get young people interested in computer science.

The other part is about taking different technologies and creating added value, which essentially is the Internet of Things concept. Wireless, embedded system design, artificial intelligence and other technologies are coming together to monitor, control, automate things in our everyday life.

One of the most important things for Internet of Things applications are wireless networks. Consequently, in the first chapter there will be a small breakdown of wireless technologies to provide some background, on the specifications, that Internet of Thing devices require and which wireless standards provide them. A small historical recursion and information about the actual applications of Internet of Things follows before explain how those two technologies are related to create wireless sensor networks.

Been that Raspberry Pi is the main concept of the project, an extensive analysis of its history, specifications, applications, etc. is warranted. Other than been a small affordable mini-computer, what are the reasons for its growing popularity. In addition, in the second chapter we see the evolvement of some of the technologies involved in the project of this thesis like databases, programming languages, etc. and what is the actual implementation that was picked.

As in every software or systems development in general, one of the most important steps is choosing an engineering method for management design and of course actually use that for developing the system in every phase. It is the measure-twice-cut-once for software engineering basically. While performing a full specification analysis for this project, along with developing the actual system and everything else would be out of scope for this thesis, some parts of the design phase will be performed to demonstrate to the reader the concept of analysis & design.

Moreover, besides the analysis for the software design, analysis for the sensors, electrical parts and other components is needed before actually the construction of the nodes begins. In general,

this third chapter is about providing insight to the phases of system development before the actual implementations begin.

The actual implementation chapter is explaining how to retrieve or set values to sensors and how to control devices. Its complementary material for the actual code of the services that run on the nodes the sensor network is comprised of.

Usually there would be more phases of the system development like beta deployment, adjustments and final release, but again it would increase the project out of bounds. Instead, the final chapter is about conclusions concerning the theoretical part of the science, as well as the practical one. It is about answering questions set through the course of this thesis.

Some of the most important questions are the following. Is Internet of Things a consequential technology or just a gimmick of the era? Will it manage to evolve while interplaying with other technologies that enable it or become fragmented as time goes by? Lastly, will this thesis provide added academic value like the Internet of Things concept tries to do? It all remains to be seen.

1 Wireless sensor networks & Internet of Things

1.1 Wireless Networks

In this author's opinion, there are two kinds of technologies, considering the following concept. Technologies that while useful and maybe groundbreaking apply to a specific area of IT, like for example hard drive advancements. For instance, helium filled magnetic HDD's [1] will probably result in faster, bigger and more reliable drives, while SSD's[2] eliminated some problems that magnetic drives had (getting rid of moving parts) and increased speed while decreasing latency [2]. Welcome improvements but not a life-changing technology that evolves other fields.

Then there are technologies that are capable of changing the way other technologies work and result in further innovation in the IT world. I can say with a high level of certainty that wireless communications is one of these technologies.

1.1.1 History

If ARPANet[1] was the precursor to the internet, then AlohaNet was the precursor to wireless networks. The University of Hawaii created AlohaNet [3] in 1971 and it was an experimental UHF network that was built in order to connect the various universities and colleges in different cities of Hawaii. The network was using 2 random access channels at 407.350 MHz & 413.475 MHz, 100 KHz each. In 1973 the first commercial satellite link was created and connected NASA in California with universities in Hawaii, California, Alaska from the USA and other countries such as Japan, Tokyo and Australia creating a network called Pacnet. Eventually all three of the aforementioned networks (ARPANet, AlohaNet and Pacnet) were connected with funding from ARPA. In 1985 [4] the US Federal Communications Commission allowed unlicensed use of the ISM[2] bands, 902-928, 2400-2483.5 and 5725-5875 MHz thus encouraging development in communications. In 1988 NCR corporation introduced a new wireless network [5] called WaveLAN which was a precursor to the 802.11 standard. In the 1990s, an Australian Engineer Dr. John O'Sullivan [6] led a research group in CSIRO[3] that eventually patented the standard that made wireless LAN a viable alternative to Ethernet. The rest is history as they say.

---

[1] Advanced Research Projects Agency Network. A packet switching network that was funded by the military and created with the help of universities in 1969 [224]

[2] Industrial, Scientific and Medical

[3] Commonwealth Scientific and Industrial Research Organization

1.1.2 Types of Wireless Networks

Wireless networks can be categorized considering type of operation, range, applications etc.

Categorizing by range the most common network groups are the following

Wireless PAN

Personal Area Networks usually capable of covering distances of a few meters to theoretically a hundred or more. Some examples are the following.

Infrared. While it is fair to say that it is an obsolete technology now, it was used extensively in mobile devices in the late 1990s. Infrared is a wireless technology that requires line of site between the two devices, in order to transmit-receive data (Think of the television remote control). Its specifications were developed by IrDA[4], an organization established in 1993 [7] that aimed to develop a protocol for short-ranged wireless communication between devices. In that decade and with the emergence of personal computers and mobile devices such as cellphones and PDA's, infrared was an easy way to perform functions such as sending files between, tethering a device with a connection to another device (e.g. cell phone - > laptop), etc. and because of that 30-60% of devices where equipped with it [8]. Eventually better technologies such as Bluetooth and Wi-Fi superseded Infrared because of its disadvantages, such as the devices needed to have a line of sight with each other, in a specific angle without obstacles in between and low speeds.

Bluetooth. Although few might know that the inventor of the Bluetooth was Jaap Haartsen [9], a Dutch engineer that worked at Ericsson at the time (1994), fewer would know that it is named after a Viking King named Harald "Blatand" Gormsson [10] and that the symbol is his initials merged together as seen in the picture below.

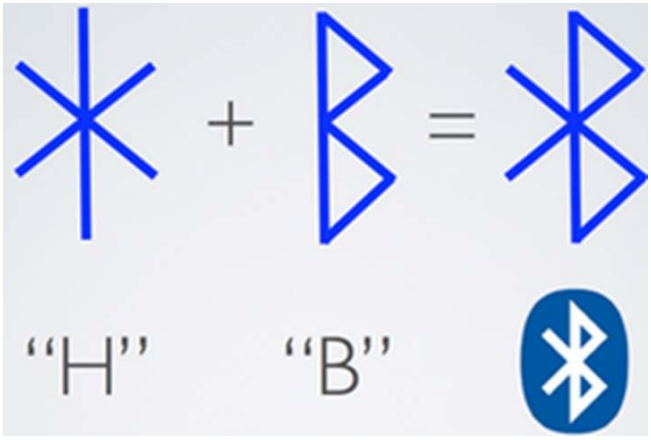


Figure 1 – Scandinavian runes of King Harald "Blatand" Gormsson initials [11]

[4] Infrared Data Association

The first specification for Bluetooth (1.0) was released in 1999 by the Bluetooth Special Interest Group [12] (SIG)[5] and the last version (5.0) in 2016. It was described by the working IEEE group [13] 802.15 in 802.15.1 standard. It is a wireless technology operating at the 2.4 GHZ ISM band like Wi-Fi and uses FHSS[6] modulation [14] to transmit signals in comparison to Wi-Fi (802.11) that uses DSSS[7], CCK[8], OFDM[9] but more on that later. As a technology, Bluetooth was invented to accommodate wireless connections between devices relatively close to each other, whilst not using much energy. The BR/EDR[10] Bluetooth has three classes, while the LE[11] Bluetooth has four [15] as seen in the table below along with information about output power and theoretical range.

Table 1 – Bluetooth power classes [15]

Power class	Maximum Output Power (P max)	Minimum Output Power	Theoretical Range
1	100 mW (+20 dBm)	10 mW (+10 dBm)	100 m
1.5*	10 mW (+10 dBm)	0.01 mW (-20 dBm)	20 m
2	2.5 mW (+4 dBm)	0.01 mW (-20 dBm)	10 m
3	1 mW (0 dBm)	0.01 mW (-20 dBm)	1 m

* Only in LE Bluetooth

ZigBee. A wireless standard first described in the technical standard 802.15.4 by IEEE in 2003. ZigBee as a standard [16] tries to accomplish low levels of cost, power, complexity and data-rate in wireless personal area networks. It is maintained by a group of companies that formed an organization called ZigBee Alliance [17] that maintains the standard, similar to what Wi-Fi Alliance [18] does for the 802.11 standard. Like Bluetooth and Wi-Fi, it operates on the 2.4 GHZ ISM band but also on the 868 MHz band in Europe and 925 MHz band in America and specifically in 16, 1 and 10 channels respectively. Also, originally the standard used DSSS modulation but in later revisions it supports BPSK, O-QPSK depending on the band. The fact that the standard has the aforementioned low requirements on cost, power, etc. makes it desirable for large number of applications [19] including home automation, control systems and suitable in general [20] for wireless ad hoc[12] mesh networks.

[5] A standards organization that maintains the Bluetooth standard
[6] Frequency Hopping Spread Spectrum
[7] Direct Sequence Spread Spectrum
[8] Complementary Code Keying
[9] Orthogonal Frequency Division Multiplexing
[10] Basic Rate/Enhanced Data Rate
[11] Low Energy
[12] A Latin phrase [219] meaning for a particular purpose or need. In wireless networking it's used to describe networks that don't use pre-existing infrastructure but were made for a particular purpose

Wireless LAN

Wireless Local Area Networks usually cover areas of up to a hundred meters. They refer to the 802.11 family of networks, mostly known to the general public as Wi-Fi, which despite what some people think, does not actually stand for something [21], it was meant as a pun to Hi-Fi[13]. The initial draft by IEEE for the 802.11 legacy protocol [22] in 1997, described a protocol that would use either IR or 2.4 GHZ ISM band and DSSS or FHSS modulation respectively, with a max data rate of two Mbps. It refers on issues on the effectiveness of the throughput, interference, security, power consumption, etc. In 1999 two amendments for the initial draft were published [23], the 802.11b & 802.11a standards, that were using the 2.4 & 5 GHz frequencies respectively and were the first standards that were incorporated broadly in devices under the Wi-Fi Alliance supervision. The standards are evolving every few years by increasing the channel width, from 22 MHz to 20-160 MHz, data rates from 11 Mbps to almost 10 Gbps (theoretical) and other improvements from the initial 802.11 to expected [24] 802.11ax. These standards represent the main branch as seen in the table below. Other branches of the 802.11 family [25] focus on specific parts of communication like quality of service in 802.11e, replace deprecated security protocols like WEP with WPA2[14] in 802.11i, or regulatory issues in Europe & Japan with 802.11h & 802.11j respectively.

Table 2 – Common IEEE 802.11 network standards

Protocol	Release Date	Frequency (GHz)	Channel width (MHz)	Max Data rate (Mbps)[15]	Modulation
802.11	Jun 1997	2.4	22	2	DSSS, FHSS
802.11a	Sep 1999	5	20	54	OFDM
802.11b	Sep 1999	2.4	22	11	DSSS
802.11g	Jun 2003	2.4	20	54	OFDM, DSSS
802.11n	Oct 2009	2.4 / 5	20, 40	600	MIMO-OFDM
802.11ac	Dec 2013	5	20, 40, 80, 160	6.93 Gbps	MIMO-OFDM
802.11ax	-	2.4 / 5	20, 40, 80, 160	9.61 Gbps	OFDMA

[13] High Fidelity
[14] Wired Equivalent Privacy - Wi-Fi Protected Access II
[15] For example, the 802.11n has a max theoretical data rate of 600 Mbps by using 4 spatial streams in the 40 MHz channel [221]

As mentioned in the beginning while the 802.11 standard is meant for LAN, there are cases which by using directional antennas or by "hacking" the protocol, large distances have been achieved like the record set [26] at the time in Venezuela which was 237 miles or 381 kilometers.

Wireless MAN

Wireless Metropolitan Area Networks capable of covering a few kilometers.

This category is the 802.16 family of networks, usually known as WiMAX[16]. Along with the initial standards of the 802.11 family in 1999, the IEEE also started the 802.16 working group to explore and improve standards for MAN. The standard [27] identifies specifications for only the first two layers of the OSI[17] model (as seen in the picture below) and not the upper levels.

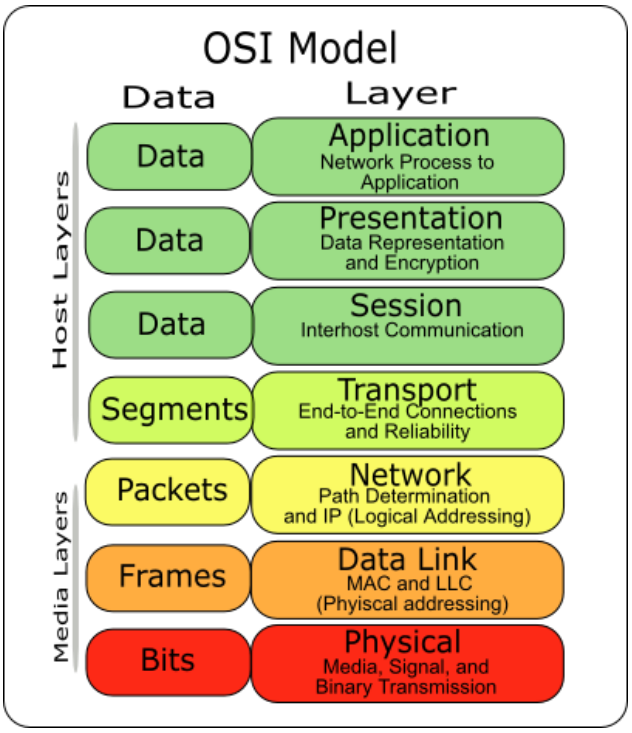


Figure 2 – Seven-layer OSI model [28]

The latest specification 802.16.1a published in 2013 [29] works at frequencies of 2-11 GHz with OFDMA modulation and supports maximum data rate of 130Mbps in distance of upward 50 km. As a standard, it stands in the middle between Wi-Fi and mobile cellular technologies, with better distance and mobility than Wi-Fi but worst speeds.

[16] Worldwide Interoperability for Microwave Access
[17] Open Systems Interconnection

Wireless WAN

Wireless Wide Area Networks use some of the aforementioned technologies like Wi-Fi and WiMAX, along with cellular technologies like LTE[18], to create bigger networks spanning along cities or countries. One part of the WWAN family are LPWAN or low-power WAN networks which are meant to connect low powered devices [30] which falls into either IoT[19] applications or M2M[20] communications. LPWAN's characteristics are long range, low power, lost cost, etc. because of the nature of IoT devices and fall into two categories. Spread Spectrum, which is a technique used by some 802.11 protocols and Ultra Narrow Band. Ones of the protocols in the Spread Spectrum category is named LoRa [31] and more specifically uses Chirp Spread Spectrum. It uses unlicensed frequency bands of 863-870 MHz (Europe) or 902-928 MHz (USA) to transmit data with low data rate (50 kbps) over long distances (up to 10km). In the other category, one of the most known standards is the NB-IoT. NB-IoT uses some LTE specifications and operates along with LTE or in a single channel of 200 MHz bandwidth [32] and in contrast with LoRa uses the 700, 800, 800 MHz licensed bands. It maintains low power while achieving maximum data rates of 150 Kbps over distances of 10-15 km.

Low-power wireless networks are been developed alongside Internet of Things since it is the most common application for them and are expected to evolve over the next few years.

1.2 Internet of Things

In the past years a few technologies have been on emerging [33] and trending lists [34], which include blockchain with the bitcoin craze, artificial intelligence subsets like machine or deep learning, augmented or visual reality and others. One common denominator in those lists for years has been Internet of Things or IoT, either as a standalone technology or used in conjunction with another technology. Why that is though, is it that IoT is such an evolutionary technology or it is something else? About what it is, it's pretty obvious at first, its devices connected to the Internet allowing us get information about something, for example temperature in our home and possibly control that temperature (smart home). In the last years of course, it has become intertwined with other technologies like Artificial Intelligence so it could perform actions intuitively and not just communicate with the user, but also with other devices (machine-to-machine communication).

[18] Long Term Evolution
[19] Internet of Things
[20] Machine to machine

1.2.1 History

Internet of Things in first glance is a relative new technology, at least with its current name. Back in 1991, a scientist at Xerox PARC[21] was explaining in an article [35] how computers would evolve in the next years from desktop computing to ubiquitous computing and he did that in a time when graphical operating systems were at their infancy and Wi-Fi did not even exist yet. Ubiquitous computing is describing many forms of computing we have today, mobile, IoT, sensor networks, etc. The first IoT device connected to the Internet could be a Coca-Cola vending machine [36] in Carnegie-Mellon University back in early 1980's. Students at the university worked on installing switches in the vending machine columns and writing a program, so anyone in the University could query the program to see if there are cold beverages in the vending machine. In the early past, Kevin Ashton is thought as the first person [37] who coined with the term Internet of Things in 1999, when working for Procter & Gamble and used it in a presentation for a RFID[22] implementation in P&G's supply chain.

In the past years the evolvement of IoT has skyrocketed for the reason that the technologies IoT relies on, have evolved as well, in addition to that other emerging complementary technologies have big interest and development as mentioned earlier. For the first part, wireless connections have advanced rapidly since Wi-Fi first came out twenty years ago, Low energy WAN networks and mobile broadband technologies like LTE have helped make IoT devices more sufficient. Furthermore, continues development on computing has made processing units more efficient and powerful in the same time. Most of these technologies has advanced in part because of needs of devices like smartphones and tablets, thus enhancing IoT devices at the same time. For the second part many emerging technologies like AI, blockchain and others work very well in addition with IoT technologies. For example, while RFID is standard in supply chains now, blockchain seems to be a perfect supplementary technology [38] to make supply chain management systems more transparent and efficient.

1.2.2 Auxiliary technologies & Applications

As mentioned earlier there is a number of technologies that facilitate IoT, first and foremost wireless communications. There is a large number of protocols like NB-IoT, ZigBee, Z-wave, etc. that were developed or evolved especially because of IoT applications. Common qualifications for such communication technologies are low cost, low power and in some uses long range.

[21] Palo Alto Research Center
[22] Radio-Frequency Identification

One more relevant improvement in computing and a continues one, is the upgrade of processing units and memory modules that evolve and improve every year keeping Moore's law relevant [39] even after almost forty-five years, a fact that obviously has a direct impact on IoT devices.

Lastly, the increasing number of smart connected devices means that there are large data sets of information to be stored and analyzed. Traditional SQL databases are not exactly equipped to facilitate such large amounts of data and as it will be discussed in a later chapter, it gave rise to different kinds of databases.

At the same time, the pool of applications that revolve around IoT is surprisingly large. An obvious area of application is of course smart home systems and building automation. Automated thermostats, door locks and lights amongst others, are few of the available applications. Even in this example technologies like virtual assistants [40] like Siri by Apple or Alexa by Amazon facilitate the IoT application.

Another obvious application is systems that monitor environmental & agricultural data, [41] like temperature, soil moisture, precipitation, etc. provide useful data meant to improve productivity or prevent climate change and a large number of other industrial applications [42]

Furthermore, and as mentioned repeatedly, the biggest advantage of IoT is integration and interconnection with other technologies. A small list of these technologies is:

- Artificial Intelligence
- Machine Learning
- Big data
- Blockchain
- Augmented & Predictive Analytics
- Digital Twins [43]

1.2.3 Problems & Future of IoT

Of course, not everything is perfect with Internet of things. As with most new technologies and applications, problems arise from their use. Even Internet has negative effects [44].

One the problems that arise with IoT technology is security [45]. As anything that uses wireless protocols to connect with the Internet, is vulnerable to various risks and attacks. IoT applications specifically, because of the need to keep low overhead and the fact that devices are not always connected to preserve power, can't get regular updates and they don't have the same robustness in security protocols. A connected problem with security is privacy [46] and again, one that is prevalent in all technologies today. The problem as in security is that by having many different connected devices [47], there as many ways for a malevolent actor to gain untheorized access or data leaks to happen. The positive side of the security & privacy concerns is that an area that

enjoys as much attention as IoT is Cybersecurity, so new research and resulting suggestions [48] & regulations come out frequently.

A very important disadvantage that IoT has, is derived by one of its advantages. By having many interconnected technologies, IoT suffers from fragmentation [49]. While many of the related technologies like wireless communications follow specific standards, IoT does not, at least not always. That has impact in the growth of adoption by the industry [50] and makes development harder. Every interconnected technology adds more problems and tries to impose its standards in IoT making standardization more difficult, for Internet of Things applications.

A similar problem is Interoperability [51]. Like every technology that evolves rapidly, in different fields and by different manufactures, without, as mentioned, having robust standards, eventually reaches a point of many systems that may not be totally compatible with each other. As the technology matures, research [52] will find way to combat those problems.

Whatever the problems and hindrance, IoT seems like it is here to stay. Research [53] suggests that the number of IoT devices will quadruple or even grow 100 times [54] in the next few years and with time and development, there will be better standardization and bigger adoption by the industry.

1.3 Wireless Sensor Networks

A Wireless sensor network (WSN) is a network of sensor nodes configured in a number of the possible topology configurations like simple star topology where all the nodes communicate with a main node or gate way or multi-hop networks were nodes communicate with each other, but eventually reaching some main nodes or gateways in order to send their data for observation or further analysis as seen in the picture in the next page. Those sensor nodes or motes[23] are basically simple autonomous IoT devices, that are used [55] to monitor, gather and relay data about a number of things, like environmental data or for industrial uses like in supply chains or in factories to assist with automation.

1.3.1 History

Historically WSN predates IoT as a technology and as many technologies before that, the first WSN that resemblances the modern ones, was created for military needs. Back in early 1950s the US Army, after WW2 and before the cold war was looking for a way to monitor underwater and act as an early warning system for missiles and detect enemy submarines. The system called

[23] Literally mean a small particle or speck. In this case describes a sensor node.

SOSUS[24] uses hydrophones [56] in order to detect sound waves in the water, in order to detect foreign objects. Other than been the first sensor node network, it is also the first underwater WSN.

A few decades later, around 1980, DARPA was researching [57] Distributed Sensor Networks (DSN) in order to implement WSNs with the help of the newly created and evolving ARPAnet. In 1993, research was starting in UCLA[25] about a new type of sensor node architecture called WINS [58] or Wireless Integrated Network Sensors. WINS architecture was the first to support the multi-hop self-created network type we mentioned earlier. These types of networks try to take advantage of the topology, with nodes being close together, they need less energy to transmit data but need to adhere to a specific schedule of when to turn on and send data.

In the 21st century and more specifically late 2000's, NASA was set to create a new type of topology, a global one with every sense of the word. The Sensor Webs [59] project that started with the launch of the Earth Observing One (EO-1) satellite, was the start of a network capable of communicating with ground sensors and monitoring large areas of land from satellite orbit, in order to provide data [60] for various phenomena.

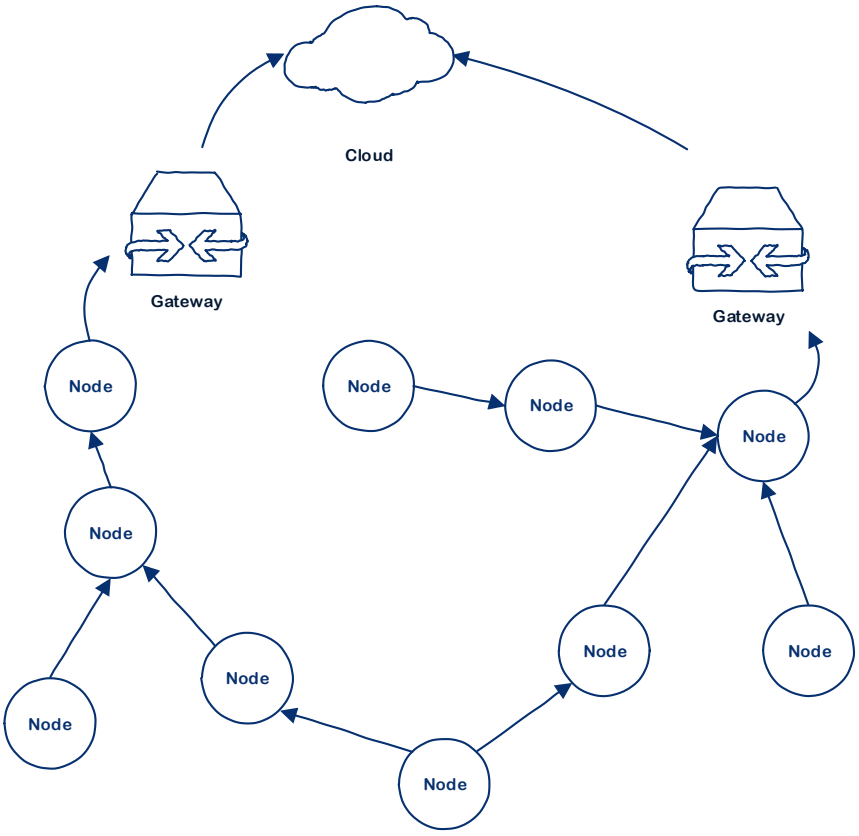


Figure 3 – Example of multi hop mesh WSN

[24] SOund SUrveillance System
[25] University of California, Los Angeles

1.3.2 WSN applications, problems & future

As in Internet of Things, WSNs have similar applications but are especially suitable hard-to-reach areas like forests, where sensors are deployed by dropping them from an airplane [61], so the topology is been made by the nodes themselves (of course with planning) but nevertheless ad-hoc. In this type of applications, WSNs are used to monitor environmental data and alert in the case of forest fires [62]. From what we saw in the history of WSNs, it becomes apparent that monitoring environmental data like seismic activity, water tides, etc. are the most used field of application and the qualities that are needed for this kind of activity (low cost, low power, long range) is the cornerstone for technologies like WSN and IoT.

Another area of application, is of course industrial applications. One of the most common ones is supply chains and warehouse management. Since 1999 when Mr. Ashton proposed an RFID system to P&G, RFID has been intertwined with IoT and by extension sensor networks [63]. With wireless communications evolving it is becoming easier for companies to follow the raw materials or products from the suppliers to the factories or final consumers. Also, with the emergence of blockchain [64], all this information becomes attached to an immutable chain, thus certifying the information. Furthermore, productions lines and manufacturing [65] in factories in general, have increased productivity in the past decades, at least in part because of sensors. WSN's can and have played a big role in increasing productivity by allowing machinery to work in a more automated fashion.

Other areas of applications include agricultural & livestock [66], where WSNs can provide weather information to help with crop cultivation and animal tracking-movements, and how that affects production respectively.

Finally, WSNs have applications in the military, which makes sense, since the first ever sensors network application SOSUS, was for military use. Modern day uses [67] include personnel tracking, wearable sensors to enhance the soldier's abilities, like smart helmets, chemical & ballistics detections systems and others.

As far as the problems WSNs encounter are on par with that of IoT with some differences. WSNs especially for industrial use, require a higher ceiling of reliability compared to their IoT counterparts. That of course means that the cost can be higher.

Moving forward wireless sensor networks will be very important in advancing other fields like manufacturing and automation by working in close association with other emerging technologies like AI, blockchain and of course any technology innovation that manages to make nodes faster, smaller, cheaper and less power-hungry.

2 Client - Server platform analysis

In this chapter we will elaborate on the platform that will be used for the sensor network, the reasons behind choosing the particular platform along with its history and specifications. Furthermore, the server side will be discussed briefly in the lines of operating system, database, programming language of the server that will be handling the traffic, etc.

2.1 Sensor network platform

First and foremost, choosing the platform of the sensor network was probably the most important step towards shaping the configuration of the project.

At first, a collaboration was pursued with Greek companies that are involved in IoT technologies like Cosmote [68], that uses a NB-IoT[26] platform using the company's 4G[27] network. The company had already started a collaboration with the Democritus University of Thrace [69], in order to implement a sensor network that will measure air quality and manage fuel and lighting in the university to lower costs.

The company was not interested in another collaboration, so there was a choice to be made regarding the platform. As mentioned before, there are a number of different sensor network technologies but the choice made was the Raspberry Pi, for reasons that will be explained in the following chapter.

2.2 Client Platform - Raspberry Pi

The main reasoning behind choosing the Raspberry Pi was that it was initially built as a learning platform, which was a nice fit for a project used in a Master's degree thesis, but more on that later.

2.2.1 Raspberry Pi history

The Raspberry Pi or RPi is a platform maintained by the Raspberry Pi foundation, which is a UK charity [70] that was established in 2008 with help from the Computer Laboratory, by founders such as Eben Upton who currently works as Broadcom ASIC[28] architect and was the CEO of both

[26] Narrow Band Internet of Things
[27] 4G or IMT Advanced is a cellular technology aimed to provide advanced mobile services as described in M.2134 by ITU-R [232]
[28] Application-Specific Integrated Circuit

the Raspberry Pi (Trading) & the Raspberry Pi Foundation until Lance Howarth [71] took over of the latter as CEO, followed by Philip Colligan [72]. It was conceived, when Eben Upton [73] who was director of studies in computer science in University of Cambridge's St John's College, Jack Lang [74] who was an affiliated lecturer at the Computer Laboratory of Cambridge, Alan Mycroft [75] a professor in Computing, also at the Computer Laboratory of Cambridge University and a few others realized that interest in computer studies was falling and few of the dwindling applicants had programming skills. That's when the Raspberry Pi foundation was formed, with a goal to create a small and cheap microcomputer that could interest young people to computer science [76].

The original founders [77] in no particular order are:

- Eben Upton
- Jack Lang
- Alan Mycroft
- Robert Mullins
- David Braben
- Pete Lomas

Continuing on the Raspberry Pi, the initial concept of the foundation was a low-cost microcomputer that would run on Linux based OS and mainly support the Python programming language, hence the name Pi as seen at the official logo below.

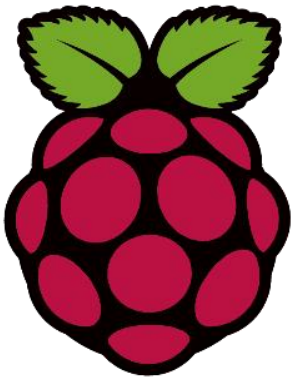


Figure 4 – Raspberry Pi Logo designed by Paul Beech [78]

So, continuing with the production of the Raspberry Pi, after designing the prototypes, the Alpha boards arrived and started to be tested by a team of the Raspberry foundation, on August 2011 [79]. They featured an ARM processor based on a Broadcom chipset, 80 MB of RAM (although that would change in the final models), USB 2.0, 10/100 Ethernet controller and it used an SD card for the operating system. All of the above would essentially fit in a microcomputer in the size of a credit card with dimensions 85.60mm x 53.98mm (not measuring extruding ports) but

for the time being was 20-30% larger. You can see a picture of one of the first alpha boards in the next page.

Figure 5 – Raspberry Pi alpha board [79]

Soon after that, they had produced the beta boards and were testing them. Also, they managed to get them to the size of credit card as promised, as seen in the picture below

Figure 6 – Beta board with credit card for comparison [80]

Although initially a release of the first boards was expected in the last quarter for 2011 – early 2012, manufacturing issues such as not being able to use UK manufacturers [81] (although later manufacturing would move to the UK [82]) , not finding a component in China [83], since manufacturing was sourced there and then a problem with the Ethernet connector [84]. Finally, in April of 2012 Raspberry Pi was released and started shipping out to customers [85] and the rest is history.

2.2.2 Raspberry Pi versions & specifications

Over the following years, a number of different versions of the Raspberry Pi came out, either improving on the specifications or fulfilling different needs depending on the use. There is an analytical table with specifications for the latest models of every model family in appendix C, but there will be a brief presentation of the different models below.

The Raspberry Pi foundation has released four different families of Pi's.

Model B. The original Raspberry and the main line of the family

As said, the Model B was the first version released and is the most common model, with six releases from March-April 2012 to March 2018 with the latest being Raspberry Pi 3 Model B+, which is on of models used in the current project (the other being zero v 1.3).

Before discussing the latest RPi 3 Model B+, an honorable mention is the first RPi Model B+ that also had improvements over the original RPi, but one of the most important ones was the GPIO[29] port that changed to 40-pin instead of the original 26-pin [86], which gave the RPi more connections [87] and it changed the card socket from an SD to a micro SD [88]. There will be an analysis of the 40-pin port and its importance of the success of the Raspberry Pi later on.

Moving on to the latest model B, RPi 3 Model B+ [89] has a quad core CPU like his predecessors but clocked higher at 1.4 GHZ, along with faster RAM, albeit the same 1GB. It also features upgraded power rails (going up to six). Furthermore, it offers for the first time, gigabit Ethernet, 802.11 ac dual band Wi-Fi and for the first time the possibility to add PoE[30] capability with a PoE HAT[31] [90] as seen in the following picture.

[29] General Purpose Input Output
[30] Power over Ethernet
[31] A HAT, Hardware attached on Top, is a board that adds some additional functionality to the RPi

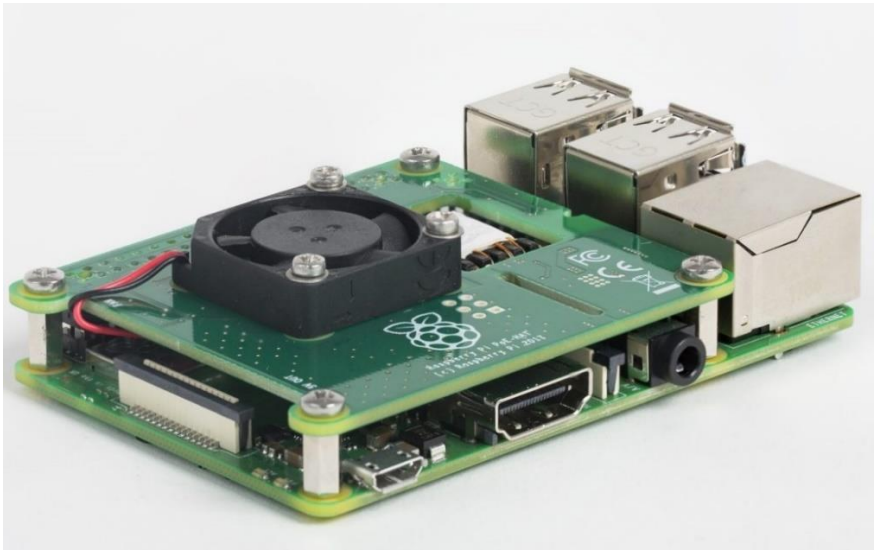


Figure 7 – HAT attached on the RPi 3 Model B+ [91]

Model A. The first "cut down" version of the Raspberry

The model A [92] was the first "cut down" version that initially had less RAM and power requirements, 200 mA vs 500mA of the original Model B [93]) and of course a lower cost. The Model A+ that came 1.5 years later increased on the RAM but came in a smaller form factor reducing the dimensions at 65mm x 56.5mm [94] (compared to the original A and all model B's that measure at 85.6mm x 56.6mm). In the past few years with no new model A releases and with the release of the Zero model the Raspberry foundation seemed to retire the model A version. However, four years later and as it did with the original model A (that followed the original model B), it released [95] the model RPi 3 Model A+ (after the RPi 3 Model B+) in November of 2018.

Figure 8 – Raspberry Pi 3 Model A+ [96]

Zero. The new “cut down” version in both size & cost.

Last but not least, the Raspberry Pi Zero is the newest implementation of a small size - low cost version of the Raspberry Pi [97]. It is safe to say it succeeded in both aspects. The RPi Zero a reduced size almost three times smaller with dimensions 65mm x 35mm and a price cut of almost 700% compared to the RPi 2 model B at the time for just $5. It features the same shipset (BCM2835) as the original RPi with an overclocked speed to 1GHZ and different ports as you can see in the following picture.

There is also a newer version The Raspberry Pi Zero W that can out in February 2017 [98] that added the additional functionality of Wi-Fi 802.11n and Bluetooth 4.0.

Figure 9 – Raspberry Pi Zero [99]

Compute module. The version for embedded designs

Although the Raspberry was initially built for learning purposes, on April 2014, the Raspberry Pi foundation decided to make a module for industrial users [100]. The original compute module had basically the same specifications as the initial Model B, but in a more compact design. As seen in following picture, it has the dimensions of a DDR2 SODIMM RAM and it is meant to be used by IO boards produced by the manufacturers that will use it in their products, in mostly embedded designs or use the module IO board by the Raspberry foundation. That gives freedom to designers to utilize the internals of the RPi, the BCM2835 chipset (or the last version 3, with the BCM2837 [101]) and decide on the number of ports, connections etc., along with providing more GPIO pins.

There is a number of companies that make motherboards for the compute module like Project Fin [102] that provides additional functionality like dual band Wi-Fi, Bluetooth, mini PCI express and on-board eMMC memory instead of a micro SD card. Also, there are companies that utilize

the compute module on a number of applications. Revolution Pi for example is a type of industrial PLC[32] device that has the compute module at its core and runs on Ubuntu Core [103].

A tech company called Argon Design [104] used the compute module and the Raspberry Pi camera [105] to create a project dealing with stereo depth perception[33]. Finally, NEC decided in 2016 [106] to add support for the compute module in a range of its displays to be used in public places such as schools, airports, etc.

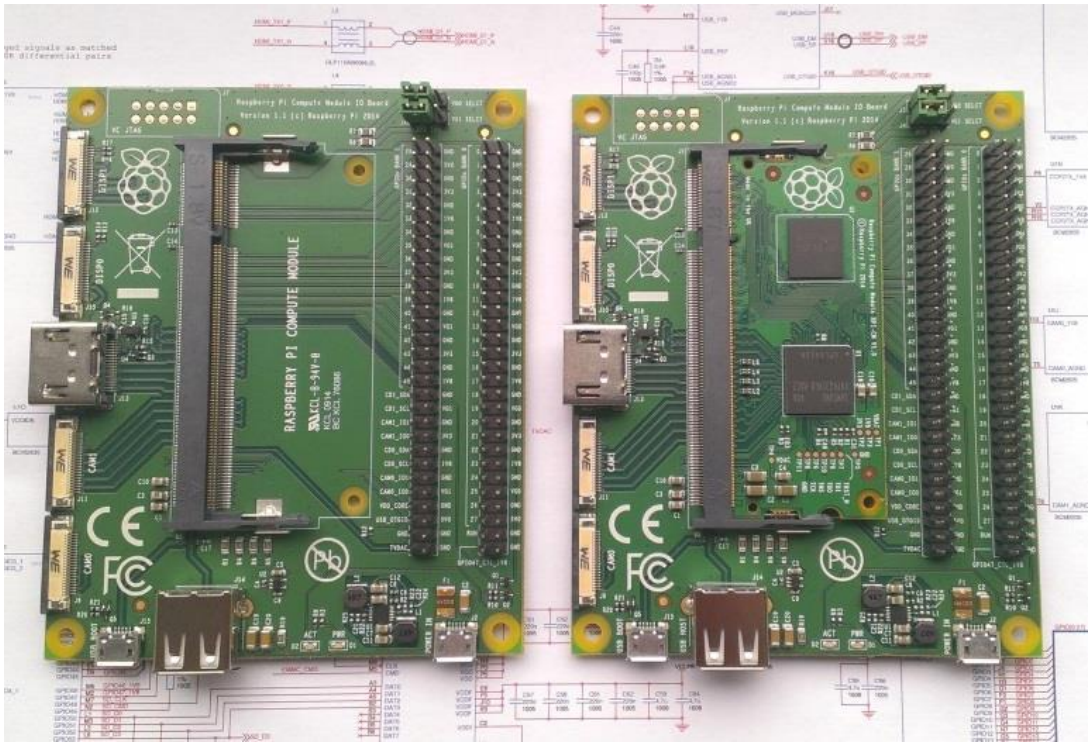


Figure 8 – IO Module board (Unpopulated & populated) /w the Compute Module [100]

2.2.3 Raspberry Pi operating system & programming language

The latest Raspberry Pi [107] is powered by a Broadcom using an ARM[34] chipset, specifically the ARM Cortex-A53 that utilizes the ARMv8-A (64/32-bit) instruction set. ARM chips use a RISC[35] architecture (as one can easily distinguish from their name) instead of the CISC[36] architecture used in personal computer x86 processors such as Intel and AMD. In turn that means that operating systems such as the Windows NT family (Windows NT to Windows 10 versions) are incompatible with RISC architecture or at least that was the case until recently [108], when ARM announced a processor capable of running full Windows (instead of the now discontinued

[32] Programmable Logic Controller
[33] Using cameras closed to each other in order to determine distance between objects
[34] Advanced RISC Machine
[35] Reduced Instruction Set Computing
[36] Complex Instruction Set Computing

Windows RT [109]). Of course, in order to run native x86 (but not x64) applications [110], Windows uses an emulation layer [111] and that translates to limitations and in sluggish performance [112], and that is with chips much faster than the one on the Raspberry Pi.

Supported operating systems

Because of the reasons mentioned before, Raspberry, from the beginning, supported almost completely UNIX-like operating systems, most frequently Linux distributions configured for ARM processors. There is a plethora of distributions available for the Raspberry Pi since its inception in 2012, mostly operating systems but also specialized distributions like media center software, that started with Arch Linux [113], FreeBSD [114], NetBSD [115] and others.

The official operating system maintained by the Raspberry foundation is the Raspbian. A Debian based operating system created [116] by Mike Thompson and Peter Green specifically for the RPi in order to address "hard float ABI[37]" problem [117] that Raspberry had in the early versions of the ARM chipset. The initial version of the Debian used a "soft float ABI" which basically is working with Integer registers instead of floating points registers compared to the "hard float ABI", which basically means that every time a floating-point value needs to be calculated it passes through both registers, first the Integer, then the floating point one, which translates to lower performance.

Aside from Raspbian, other operating systems-distributions mentioned [118] on the official page are Ubuntu Mate & Core, Windows 10 IoT Core (since 2015) [119], RISC OS, OSMC (media center), etc.

On this project, the obvious choice was made to pick Raspbian as the operating system of the nodes, mainly because it is the official operating system and thus has more documentation and support compared to the other distributions.

Programming language

Since the beginning of the development of the Raspberry Pi, one language had the edge of being the office language of the RPi and that was Python [120]. Python is a high-level programming language made for general purpose programming, that was created by Guido van Rossum in the late 80s [121].

Python is one of the most loved programming [122] and fastest growing programming languages out there today. There are many reasons for that, such as:

- It is a multi-paradigm language, meaning it can support many paradigms including objected-oriented, structured, functional etc.

[37] Application Binary Interface

- Continuing from above that means that it is well suited for a number of applications [123], some of which are scripting, machine learning, web programming etc.
- Its use of whitespace indentation makes for a better readable code.
- Great support with open source frameworks, libraries and great documentation
- Lastly, the main reason is that it is easier to learn & write code that similar high-level languages and gives it a great advantage for a platform like the Raspberry Pi, whose initial goal was to introduce more people to programming.

So, while Python holds the crown as the "official" programming language, there is a large number of others supported by default as well such as:

- Java [124], HTML, JavaScript
- Scratch [125] (learning language for kids)
- Ruby, Perl, Lisp
- C, C++, C# family

Basically, any programming language that is supported by the instruction set of the ARM chipset can run on the Raspberry Pi.

Once again, the choice for this project was again the official language, which is Python, mainly because it has great support on the platform, along with a plethora of documentation, both for the board and the sensors used in this project.

2.2.4 Raspberry Pi GPIO

It finally came to the part that differentiates this microcomputer, which is the 40-pin GPIO. The GPIO pins are versatile in their usage and allow a vast number of sensors and devices in general to be used with them, but first we need to explain the different functions of some of the pins as seen in the picture below, which can be split in four distinct categories as also seen in the next picture

- *Power pins*. Pin #1 for 3.3v and pins #2, #4 for 5v
- *Ground pins*. Pins #6, #9, #14, #20, #25, #30, #34, #39
- *Reserved pins*. Pins #27 & #28
- *General-purpose pins*. The rest.

The general-purpose pins have some useful functions & properties [126]. They can function as both input & output. As an output they have two levels, 0v and 3.3v. As an input they can read voltage from 0 to 3.3v and that's made possible with the use of the internal pull-up or pull-down resistors that all the pins have. While the I^2C pins have fixed resistors, the rest of GPIO pins can be configured programmatically.

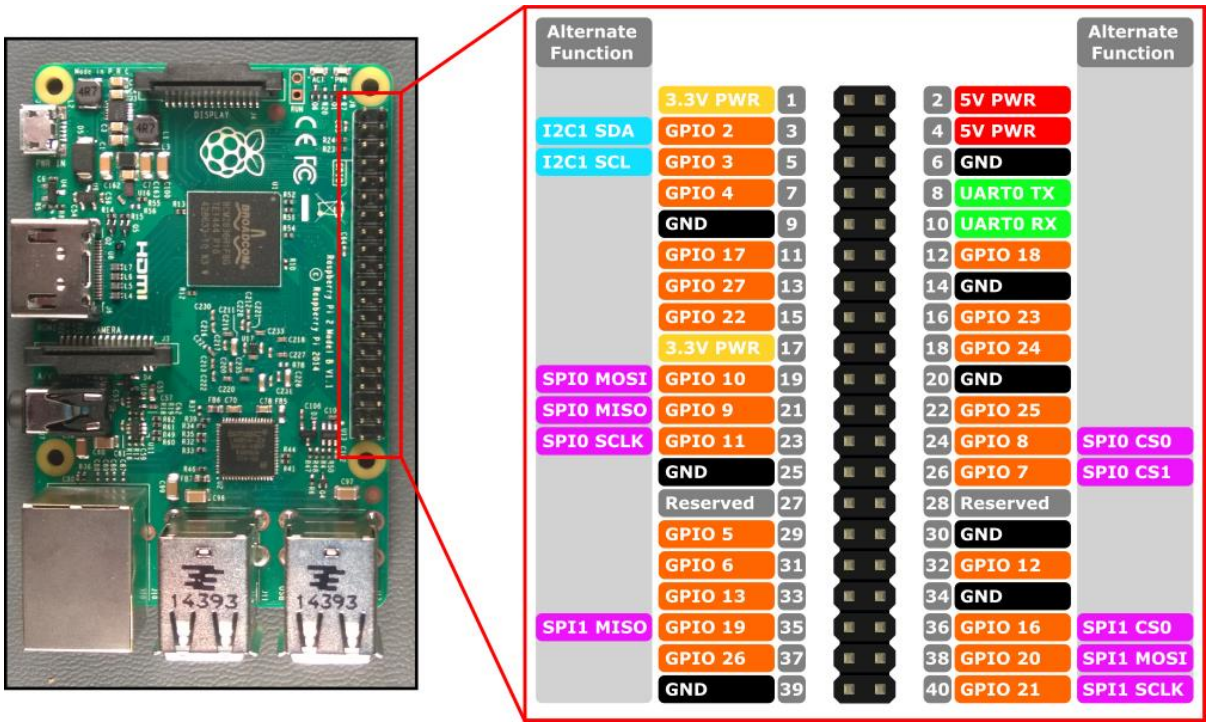


Figure 10 – Raspberry Pi GPIO [127]

While the general-purpose pins are interchangeable, there are specific functions only specific pins can perform.

- *Hardware PWM* [38]. While all pins can perform software PWM, only the pins below support hardware PWM
  - GPIO12, GPIO13, GPIO18, GPIO19
- *SPI* [39]
  - SPI0: GPIO7 (CE1), GPIO8 (CE0), GPIO9 (MISO), GPIO10 (MOSI), GPIO11 (SCLK)
  - SPI1: GPIO18 (CE0), GPIO17 (CE1), GPIO19 (MISO), GPIO20 (MOSI), GPIO21 (SCLK)
- *I^2C* [40]
  - GPIO2 (Data), GPIO3 (Clock)
  - Reserved: GPIO0 (EEPROM Data), GPIO1 (EEPROM Clock)
- *UART* [41]
  - GPIO14 (TX), GPIO15 (RX)

[38] Pulse-Width Modulation
[39] Serial Peripheral Interface
[40] Inter-Integrated Circuit
[41] Universal Asynchronous Receiver-Transmitter

Therefore, continuing an explanation is required for the types of specialized pins.

Hardware PWM

PWM is used when there is a need to generate analog signals through a digital source [128]. With PWM that entails controlling the frequency & duty cycle, in order to produce alternate voltage for example, from a set starting voltage [129]. In this context it is used to control devices or sensors that need alternate voltage when the Raspberry pins usually provide static 3.3v or 5v.

UART

UART is a physical circuit made to perform basic serial communication [130] and like every serial communication, it needs two connections TX (Transmitter) & RX (Receiver). As stated in the name it is asynchronous, does not use a clock, but instead uses start-end bits in the packets. It can be used by any device requiring a serial interface, but it cannot share it with multiple devices.

SPI

SPI is a more complicated protocol because it adds the functions missing from basic serial communication such as the UART, which are a clock for synchronous communication and the ability to have master-slave devices. In order to accomplish that [131] SPI uses four connections [132] instead of two, which are the following:

- MOSI[42]. Communication line from the master device to the slave devices
- MISO[43]. Communication line from the slave device to the master device
- SCLK[44]. Clock for synchronous communication
- SS (or CE) [45]. The slave select line

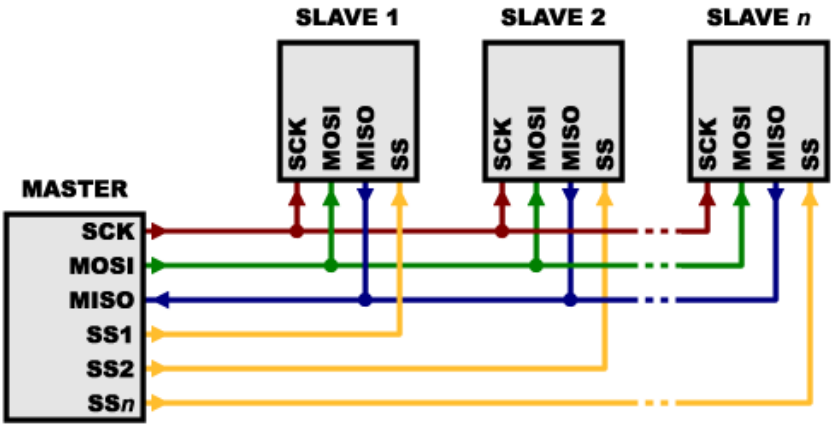


Figure 11 – SPI example communication master-to-multiple slave devices [132]

[42] Master-Out, Slave-In
[43] Master-In, Slave-Out
[44] Serial Clock
[45] Slave Select or Chip Enable / Chip Select

I²C

Last but not least, the I²C protocol, which has the advantage [133] of using only two signals like the UART, but still keeps the synchronous communications of the SPI. Originally developed by Philips, I²C allows up to 1008 slave devices [134], given the 10-bit address space, with those two signals called SCL & SDA, the clock & data signal respectively. Since only two signals are used, this bidirectional communication requires a unique feature called "open drain". What that means is that the devices trying to communicate can only lower the voltage on the line and not increase it, which as a result that precludes collisions from happening as seen in picture below. When the channel is not used, a pull-up resistor brings the voltage up to its starting value.

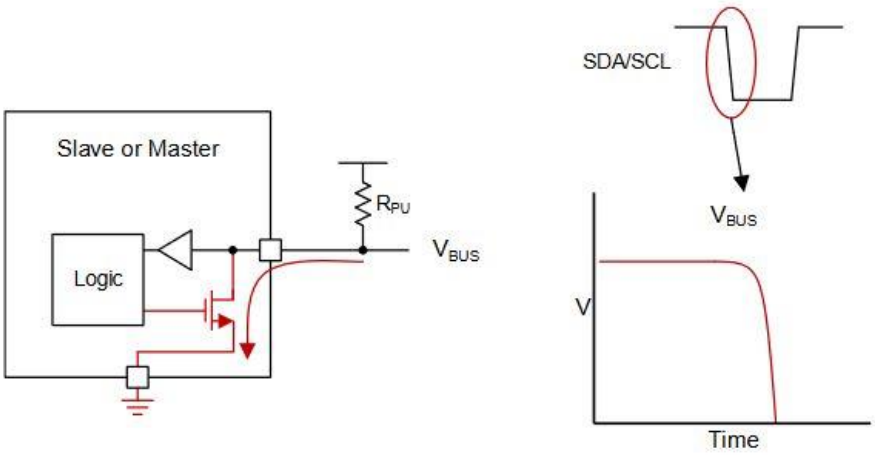


Figure 12 – Open drain on low voltage [133]

2.3 Server platform

While some choices regarding the IoT platform for the nodes were straightforward at times, on the server side it is different because there are no longer restrictions based on instruction sets and problems with lack of documentation.

While IoT platforms are still growing and evolving, operating systems, programming languages, databases, etc. are in a very mature stage in normal computing. By no means they are stagnant, new programming languages are been developed every few years like Go[46] in 2012, Rust[47] in 2015, Kotlin[48] in 2016, etc. and new databases like MongoDB[49] in 2009, MariaDB[50] in 2009,

[46] A programming language developed by Google [233]
[47] A System programming language developed by Mozilla employee Graydon Hoare [228]
[48] A pragmatic programming language developed by JetBrains [230]
[49] A NoSQL database developed by MongoDB Inc. [227]
[50] A database developed by Michael Widenius (original author of MySQL) as fork to MySQL [229]

Cassandra in 2010[51], etc. and other tools are gaining popularity, when a specific type of job is needed, but for the most part there are fully developed, fully tested environments for decades.

2.3.1 Server hardware – Virtual Machine

As said before, the actual specifications of the hardware don't really matter since every commercially available server out there uses more or less the same architecture. To go a step further, software doesn't really care or doesn't need to have exclusive access to the hardware of a server, so thus came the virtual machine (VM). Virtual machines are not actually a new technology, it has existed since the early 1960s when IBM was trying to make the first usable VM's with full virtualization with the IBM CP-40 and CP-67 [135]. In 1972 it pioneered the first virtual machine with hardware-assisted virtualization with the VM/370 [136], which was a step in the right direction, but it would take another almost thirty years to get x86 virtualization from VMware in 1999 [137]. Furthermore, it would take a few more years and the needed involvement of the two leading x86 microprocessor manufacturing companies, Intel and AMD, in order to achieve full hardware-assisted virtualization, instead of software virtualization, which was common until then. Finally, the past few years, servers and virtualization in general has changed with the emergence of cloud computing. The actual Virtual Machine provided by the University for this project is VMware ESXi.

2.3.2 Operating system

Considering operating systems to run on the VM there are really two branches of operating systems, the Windows NT family and the various Linux distributions in existence. There are other operating systems as well but they are either not suitable for servers (Mac OS) or UNIX operating systems (like Solaris) which are beyond of the scope of this project.

Linux systems usually have more advantages over their Windows counterparts when considering a server operating system, especially a web server. They usually offer better stability, flexibility, security and require fewer resources at a usually lower price or with no cost at all. Therefore, it is no surprise that they have more than 65% of the market share (in web servers at least) [138]. Finally, this being an academic environment, an open source OS like Linux seems to be the more appropriate choice.

The actual Linux distribution chosen was Ubuntu. It is a Debian based distribution [139] that is upgraded regularly and has great support and documentation and it is currently the leading Linux distribution in websites [140]. While there are other distributions like Red Hat or CentOS (which

[51] A NoSQL database developed by Facebook [226]

is basically a fork of Red Hat), that one might argue are more stable and better for business use. Nevertheless, Ubuntu is better universally supported and is closer to the Raspbian used by the RPi. For example, it uses the same package management system in APT[52] that all Debian distributions use compared with YUM[53] that RHEL[54] type distributions like Red Hat, Cent OS and Fedora use.

2.3.3 Database management system (DBMS)

Database is another software that has been around for decades but continues to improve and even evolve in the past years. There is number of ways to classify databases like by type of operation, type of content, type of application area, etc. Chronologically and by type of operation, we have the following groups and a small explanation for each one.

- 1960s – 1970s: Navigational databases

Early development of databases started in the 1960s [141] with General Electric's employee Charles Bachman who designed the network model DBMS[55] called IDS[56] and IBM introducing a hierarchical model with IMS[57], both of which are representations of navigational databases. A hierarchical model is basically a tree-like shape when an entity can have one or more sub-entities but not the opposite [142], so a one-to-many relationship we see in today's relational databases. The network model is more flexible allowing for more complicated relations between the entities and not necessarily hierarchical.

- 1970s – Present: Relational databases

In the early 1970s, an IBM employee called Edgar F. Codd proposed a new database model called the Relational model [143] mainly because the data in the database were defined by mathematical relations between them. Although met with resistance at first, at 1975 IBM put a team together that implemented a RDBMS[58] called System R and was the first implementation of the language SQL[59]. In the late 1970s, Larry Ellison, the president of a company called Relational Software Inc. (later renamed to Oracle) wanted to make the company's database product called Oracle compatible with that of IBM, but IBM relented. In 1979, RSI released the first commercial RDBMS while the IBM's database DB2 was released in 1983 [144]. While there was further

[52] Advanced Packaging Tool [220]
[53] Yellowdog Updater, Modified [225]
[54] Red Hat Enterprise Linux
[55] Database Management System
[56] Integrated Data Store
[57] Information Management System
[58] Relational Database Management System
[59] Structured Query Language

evolvement in the next decades, relational databases are the norm today with the top four databases most commonly used are based on the relational model as seen in the next graph.

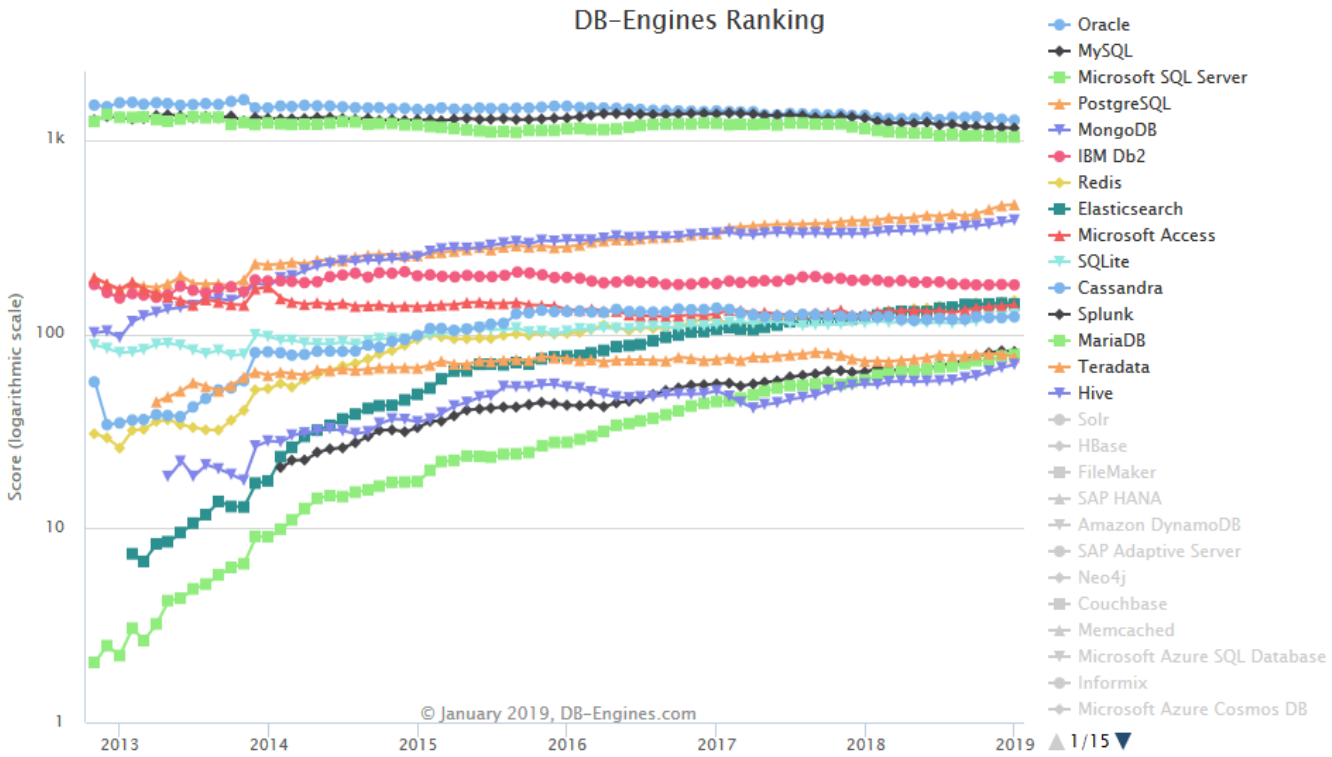


Figure 13 – DBMS ranking [145]

- 1990s – Present: Object-oriented databases

In the early 1990s came the shift to object-oriented programming and with it the idea that relational databases were not built to relay information about objects and their properties, such as polymorphism, encapsulation, inheritance, etc. so a new model was needed, an object oriented one [146]. There was some development done and a few implementations of object-oriented DBMSs appeared but eventually they did not gain enough traction because RDBMS were easier to implement, had years of support and a large number of active users, among other reasons.

- 2000s – Present: NoSQL databases

While the object-oriented database model wasn't the answer to the limitations of the relational model it was obvious that something different was needed with the emergence of Web 2.0 and the new technologies that came with it like Big Data[60]. Relational databases weren't really designed to accommodate files in a large scale [147], either by number or size. The term was first used by Carlo Strozzi [148] in his implementation of a DBMS called Strozzi NoSQL, that as the name suggests, didn't use SQL queries but is based on the relational model. The term was

[60] A term used to describe large volumes of complex data, a term coined by a computer scientist named John Mashey in early 1990s [222]

mentioned again almost a decade later by Johan Oskarsson when he organized a meet to discuss non-relational databases [149]. Unlike the initial Strozzi database, modern NoSQL databases are usually non-relational and the "No" means Not only SQL, in the sense that they be compatible with some SQL-like queries. Furthermore in the early development of NoSQL databases they didn't full support the ACID[61] model, a term first mentioned by Jim Gray [150] in 1981 and Andreas Reuter and Theo Härder on their paper in 1983 [151]. The ACID model is followed by traditional Relational DBMSs and it describes the characteristics needed by those databases in order for the data to be valid in case of system, network and other type of failures. In contract, NoSQL databases usually compromise one characteristic of ACID, consistency in favor of availability. In 2000, Eric Brewer proposed a conjecture [152] that was later proved in 2002 by Seth Gilbert & Nancy Lynch [153], the CAP[62] theorem as seen visually below.

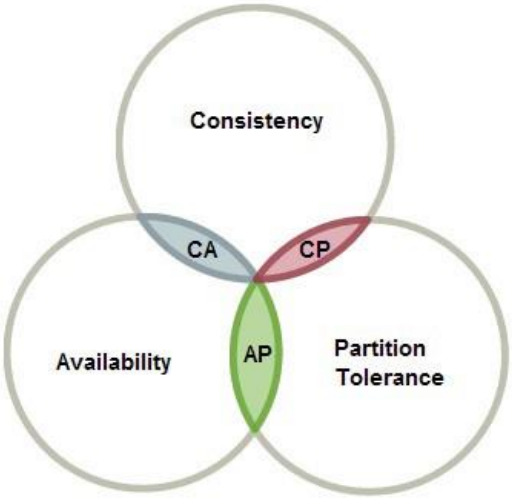


Figure 14 – CAP theorem visualized

Brewer suggested that distributed web systems could not guarantee all three properties, although in a later paper he clarified some misconceptions [154], in that, designers should not take as granted that availability and consistency could not be achieved together but rather try to maximize both. The NoSQL movement tried to replace the ACID model with the BASE[63] model which by "sacrificing" consistency for the shake of availability, eventually achieves consistency.

While a lot has been said about NoSQL database models, another important part is how the data is actually stored. The database types [155] considering the method used to store data are:

[61] Atomicity, Consistency, Isolation, Durability
[62] Consistency, Availability, Partition-tolerance
[63] Basically Available, Soft state, Eventually consistent

- Key-value. This is the simplest form of a NoSQL database model. It stores pair of keys and values and thus is really fast.
- Document. In this model every key is paired with a data structure (like XML[64] or JSON[65]) called document, each containing more pairs of keys-values or more complex structures.
- Graph. In this type, the database stores the relationship between nodes with the help of properties of the nodes and the edges between them.
- Wide-column. In this instance data are stored in cells and grouped by columns instead of rows.

From the above it can be inferred that one of the biggest differences between SQL and NoSQL database models is that SQL databases are usually schema-oriented while NoSQL databases usually aren't. That means that the data stored have a high impact on the designing phase of a traditional RDBMS while for example anything can be stored in a document model database because you can have a different structure in every document.

Another difference is that Relational databases are scalable vertically scalable [156], meaning by increasing the server hardware power (computational power, memory etc.,) you can in increase its ability to serve more requests and users in a faster fashion, while NoSQL databases are horizontally scalable, in that you just add more servers in order to increase the collectively server power.

Furthermore, one of the biggest differences is the language used for queries. Traditional databases use SQL while NoSQL databases have a plethora of languages used like JavaScript, Python and others for MongoDB, custom languages like CQL[66] for CassandraDB and N1QL [157] for Couchbase.

While the fact that traditional databases have to follow specific rules based on the data they are designed to store and their schema, a fact that sometimes makes them slower than the NoSQL counterparts, it is also what makes them robust and more staple. That and the fact that 50 years of development have made RDBMS a mature and well-known product with support from the community.

- 2011 – Present: NewSQL databases

NewSQL was a term first mentioned by researcher Matthew Aslett [158] at a post in blog 451group used to describe products known until then as ScalableSQL, that is relational model databases that are better scalable than traditional RDBMS. The goal of this model is to provide

[64] eXtensible Markup Language
[65] JavaScript Object Notation
[66] Cassandra Query Language [231]

the same scalability as NoSQL databases, particularly in OLTP[67] workloads [159] while keeping intact the ACID properties and of course use the same relational model and SQL language that is familiar to people for the last 50 years.

NewSQL databases use different methods [160] for achieving the aforementioned improvements, for example they can use memory storage for some or all data in comparison to traditional databases that use in disk storage, thus rendering them faster. Also, NewSQL databases are using sharding/partitioning, which is a technique to divide a database in many smaller DBMSs. Another important part is the concurrency control offered by NewSQL databases, that is to try and eliminate conflicts when users simultaneously access the same data hence increasing data integrity. Furthermore, NewSQL provides important qualities such as availability and durability of data with replication. Of course, all modern databases provide some kinds of replication but in the era of cloud computing it is important to have database as a service or DBaaS, and NewSQL databases provide that. In replication it is important to have data consistency (because the same data is replicated in many servers) but it was hard to achieve. NewSQL database achieve that by processing requests on a single node and then replicating that change to the other servers.

This new model is developing fast, a number of companies already have NewSQL products, like Google with Cloud Spanner [161], SAP HANA [162] by SAP[68], VoltDB [163] by VoltDB Inc. etc. On the next page we can see a taxonomy of databases based on different architecture overall.

Closing this part and considering this project, in that, IoT generates large amounts of data, one can easily come to the conclusion that a NoSQL database would be chosen and it would be a fair assumption. Unfortunately, after conversing with my fellow student Giorgos Sarigiannis that has the second part of this thesis, we concluded that the platform for the website does not support NoSQL type databases without pain staking development and it goes out of the scope of this thesis. Furthermore, the amount of data of this generated by this project isn't that overwhelming for a traditional RDBMS and there are ways to store data in Pseudo-NoSQL way in a RDBMS, but more on that later.

[67] On-line transaction processing
[68] A European company that specializes in enterprise software

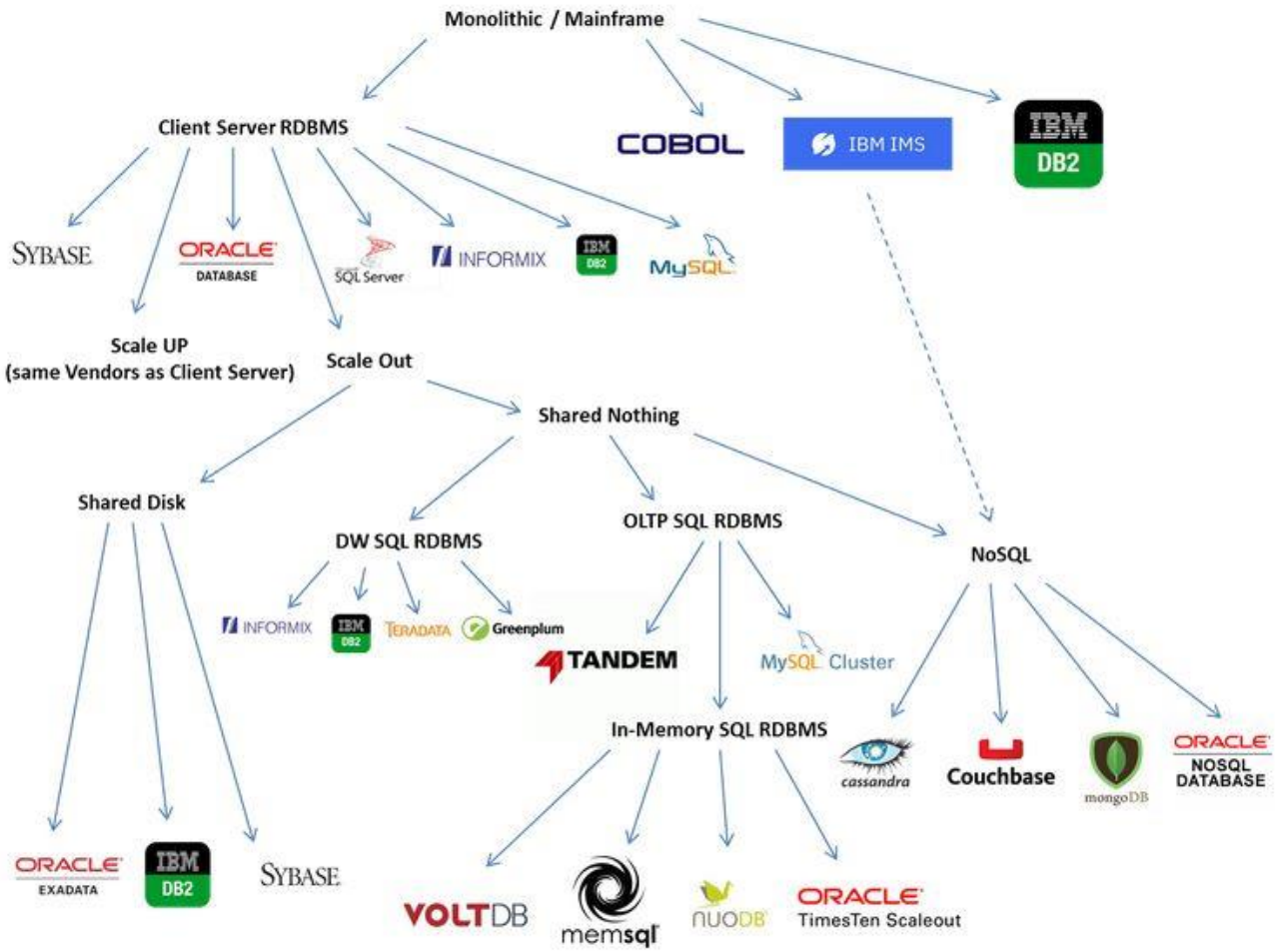


Figure 15 – Taxonomy of databases based on architecture [223]

2.3.4 Web page & services

Finally, the virtual server hosts the web page that displays the node information and the service that the nodes use to send data to the server. Their implementation is part of the "sister" thesis that is prepared by Giorgos Sarigiannis.

At the beginning of the World Wide Web history, the man that was instrumental in combining existing technologies such as HTTP[69], TCP[70] and others was Tim Berners-Lee [164]. While working at CERN[71] he presented the initial idea of the WWW to his superiors in a document in 1989 [165] that led to him building the first website at CERN. Of course, in the first years, web

[69] Hypertext Transfer Protocol
[70] Transmission Control Protocol
[71] European Organization for Nuclear Research

pages did not look like today, it was static mostly static text instead of dynamic content. It was what today is known as Web 1.0. It wasn't until 2004 that Tim O'Reilly or O'Reilly Media made the term Web 2.0 popular at a conference [166] he organized, although it is known that an employee mentioned it in a brainstorming meeting first [167], named Dale Dougherty. However, the term was actually mentioned was in 1999 by Darcy DiNucci in an article [168]. To continue, web 2.0 actually is what we have today, which is dynamic content, much of which is user generated. The past few years is what drives the metamorphosis of the web, which is social media, wiki pages, blogs etc.

Now the actual technologies that support the web pages today are split into two categories. The frontend and backend as they are called in software engineering.

Basically, everything the user is seeing is the frontend [169] and that is comprised most if not all of the time by three technologies, HTML[72], CSS[73] and JavaScript. Without getting into too much detail, the three technologies are responsible for the structure of the web page, the presentation of the web page and the tool to actually make the page dynamic, respectively.

The backend is everything else the page needs to function and that includes the connections to servers, databases, services, etc. In this part there is a large number of frameworks and languages that developers use to create the server-side website. That includes some of the most popular like:

- PHP. Arguably the most well-known programming language for web pages. I was developed in 1995 by Rasmus Lerdorf [170]. It is the language Mark Zuckerberg initially used [171] to create Facebook. It is used by web management systems such as WordPress, Drupal, Joomla and others.
- The Microsoft ASP.NET framework [172] that usually uses the MVC[74] architecture and support various programming languages such as C#, F# etc.
- The Ruby on Rails framework that also uses MVC and obviously uses Ruby as the programming language.
- Most recently, the trend has been JavaScript frameworks like Angular [173] maintained by Google and the community and libraries like React [174] maintained by Facebook and the community.

Regarding web services they share much of the frameworks and languages of the pages but are usually implement based on one of two technologies, SOAP[75] and REST[76].

---

[72] Hyper Text Markup Language
[73] Cascading Style Sheets
[74] Model-View-Controller
[75] Simple Object Access Protocol
[76] Representational state transfer

Starting with SOAP, it is a protocol specification designed [175] by a team in 2000, to provide a mechanism for exchanging information across the internet. It uses HTTP as application communications protocol and XML[77] as the markup language for the structure of the messages.

In contract, REST is not a protocol, but an architectural style. REST was defined by Roy Fielding in his doctoral dissertation [176] in 2000. As described in his dissertation, his blog [177] and in articles, in order for a service to be RESTful it has to follow specific rules some of which are

- A REST API[78] is not contingent on any specific communication protocol
- A REST API should not alter protocols except in the cases of bugs
- A REST API should be as descriptive as possible with the use of URI[79]'s
- A REST API should be state-less, meaning the client will provide any information needed for the request.

REST as an architecture uses both XML and JSON[80] for document structure, although most often JSON because it carries less overhead and does not require a schema.

Now considering the choices of many of the components in our system design until now, one notices a trend. Linux, PHP, MySQL type database. All of these along with the Apache web server are part of what is called the LAMP[81] stack [178]. Stack refers to a solution stack for the technologies needed for web development. For example, the respective windows stack is WINS that includes Windows Server for the operating system, Internet Information Services for the web server, .NET as the software framework and SQL Server for the database.

That is usually the case with all technologies concerning software development, some are usually affiliated with each other for a number of reasons such as one company develops all of them, such as C#, .NET, SQL server, etc. for Microsoft. We can see from the stack overflow survey, the common correlated technologies in the next picture.

In this project, Giorgos will use the LAMP stack technologies along with WordPress as the CMS to develop and manage the web page and services.

[77] XML
[78] Application programming interface
[79] Uniform Resource Identifier
[80] JavaScript Object Notation
[81] Linux, Apache, MySQL and PHP

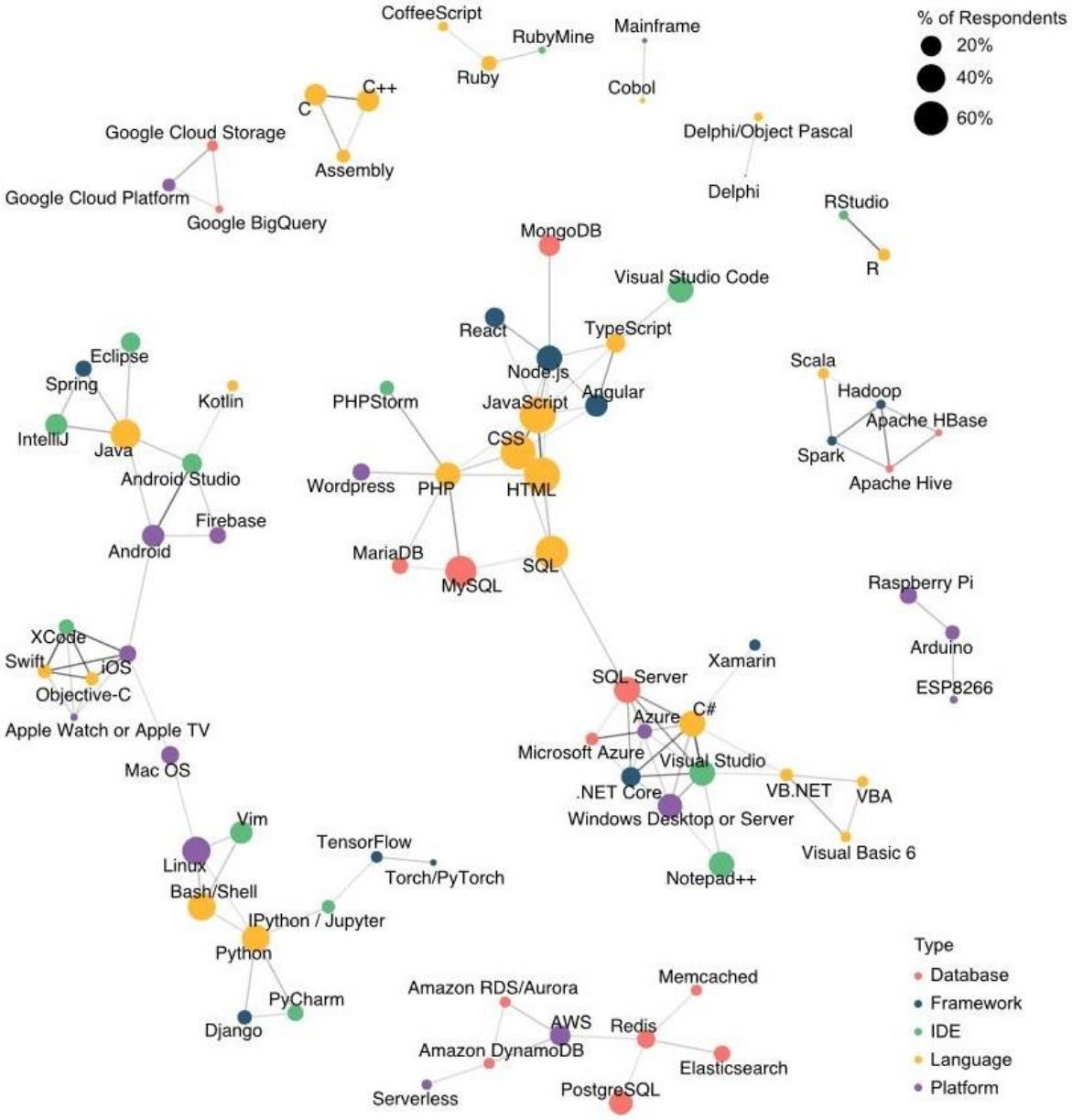


Figure 16 – Correlated technologies – Stack overflow survey 2018 [179]

3 System Analysis

The initial phases in every system development [180] is defining the requirements and perform a system analysis and design. In this system there are two parts that need analysis, the software and hardware. Initially we will explain the node design, connections, housing, etc. and then the services the nodes and the server need for the system to work.

3.1 Nodes analysis & characteristics

3.1.1 Nodes housing – sensors

At this part, there will be a summary of every node, along with analysis on housing, components and their functions. In addition, some information and visual material will be provided about the sensors and the available datasheets will be provided in appendix A

Node #1

Summary

The first node, also known as the main node for gathering outside weather data, is the most complicated node of the project with multiple sensors gathering a variety of data, as well as having additional functions such as opening electronically using an RFID card.

Housing

- Stevenson screen. The majority of the components are housed in an enclosure called Stevenson screen [181]. It is used to house and protect meteorological equipment against weather phenomena such as rain, direct sun, etc. Usually it is from wood and painted white, in order to reduce heat absorption and most often employs a design with angled slats so that it keeps precipitation out and direct heat of the instruments, but allows air to flow freely. The WMO[82] has numerous reports over the years [182] from scientists to ascertain if there are better designs and how much the housing affects the measurements, considering it is a 150 years old design by an engineer named Thomas Stevenson [183].
- Plastic boxes (different sizes). Sensitive components of the node are housed inside an IP65 rated (IEC[83] standard 60529) plastic box, in order to protect them from moisture and dust. Standard 60529 [184] is used to determine degrees of protection that an enclosure offers from intrusions such as dust, water, etc. against its electronic parts. We can see further information on the table below

[82] World Meteorological Organization
[83] International Electrotechnical Commission standard

Table 3 – IEC standard 60529 (digits explanation) [185]

First digit	Solid particles protection	Second digit	Liquid protection
0	No protection	0	No protection
1	Objects greater than 50 mm	1	Dripping water
2	Objects greater than 12.5 mm	2	Dripping water when tilted up to 15 degrees
3	Objects greater than 2.5 mm	3	Spraying water up to 60 degrees
4	Objects greater than 1 mm	4	Splashing water from any direction
5	Dust protected	5	Water jets from a nozzle of 6.3mm
6	Dust tight	6	Powerful water jets from a nozzle of 12.5mm
-	-	7	Submersion up to 1m
-	-	8	Submersion beyond 1m

Components – Functions

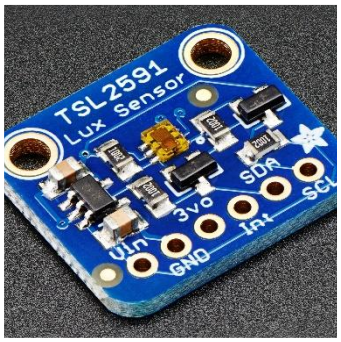
Below are the main parts of the first node, along with a picture of most them in the next page

- *Raspberry Pi 3 Model B+* [186].
- *Power supply for the Raspberry Pi.* Rated at 5V – 2.5A
- *Micro SD card*. A 64GB card used for the operating system & programs of the RPi
- *Adafruit Perma-Proto Full-sized Breadboard* [187]. PCB used for the connections
- *Adafruit Perma-Proto Half-sized Breadboard* [188]. PCB used for the connections
- *Adafruit BME680* [189]. A sensor made by Bosch, capable of measuring temperature, humidity, barometric pressure and VOC gas.
- *Adafruit TSL2591* [190]. A sensor capable of measuring light in lux, visible and infrared light as 32-bit and 16-bit unsigned values with no unit, respectively and full spectrum, which includes both of the last two values mentioned, again in 32-bit unsigned value without a unit.
- *Anemometer* [191]. An anemometer capable of measuring up to 32.4m/s wind speed
- *RGB LCD screen* [192]. An RGB LCD screen 16x2 (characters x rows) to provide information
- *Lock-style Solenoid* [193]. An electronic lock that is controlled by the keypad or the RFID sensor.
- *Power supply for the solenoid.* Rated at 12V – 5A
- *MFRC522 module* [194]. A RFID reader-antenna providing a way to control the solenoid.
- *Magnetic contact switch* [195]. A simple reed switch to provide information in case of the opening of the second door of the Stevenson screen.

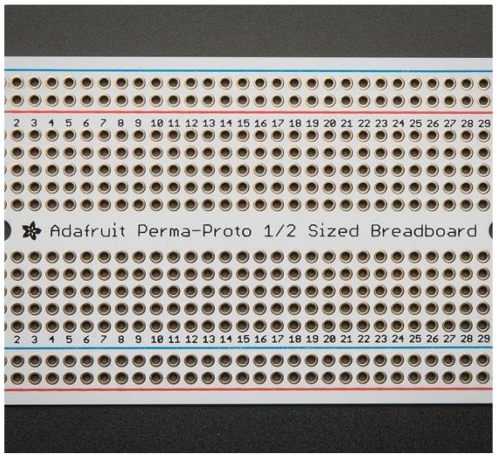
Anemometer

BME 680

TSL2591



Lock-style Solenoid



Door Contact

RGB LCD display

MFRC522

Figure 17 – Node #1 components

Node #2

Summary

This second node is the internal weather node used to measure data such as temperature, humidity, air quality in a classroom

Housing

Custom open style wooden & plexiglass box

Components – Functions

Below are the main parts of the second node, along with a picture of most them

- *Raspberry Pi 3 Model B+.*
- *Power supply for the Raspberry Pi*. Rated at 5V – 2.5A
- *Micro SD card*. A 64GB card used for the operating system & programs of the RPi
- *Adafruit Perma-Proto Half-sized Breadboard*. PCB used for the connections
- *Adafruit BME680*. A sensor made by Bosch, capable of measuring temperature, humidity, barometric pressure and VOC gas.
- *Adafruit SGP30* [196]. An air quality sensor that measures VOCs[84] & eCO^2 [85].
- *RGB LCD screen*. An RGB LCD screen 16x2 (characters x rows) to provide information

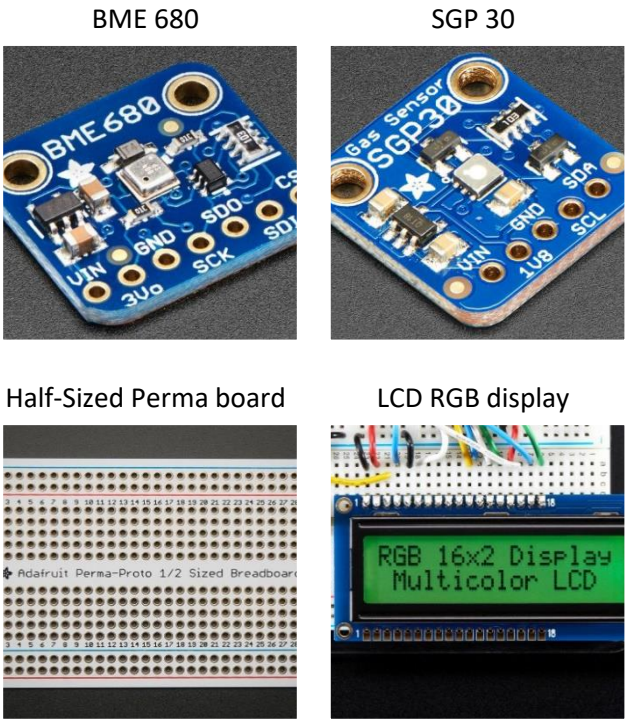


Figure 18 – Node #2 components

[84] Volatile Organic Compound
[85] Equivalent Calculated Carbon-Dioxide

Node #3

Summary

The third node located on the campus of the university, next to the main gate that provides two main functions. It takes photographs of the horizon and counts the numbers of cars entering and exiting the campus.

Housing

Plastic boxes (different sizes). Similar IP65 rated plastic boxes as the first node.

Components – Functions

Below are the main parts of the third node, along with a picture of most them in the next page

- *Raspberry Pi 3 Model B+*.
- *Power supply for the Raspberry Pi*. Rated at 5V – 2.5A
- *Micro SD card*. A 64GB card used for the operating system & programs of the RPi
- *Adafruit Perma-Proto Quarter-sized Breadboard* [197]. PCB used for the connections
- *Raspberry Pi Camera* [198]. An 8MP camera for the photographs
- *IR Distance Sensor x 2* [199]. Two Sharp sensors capable of detecting objects 1 – 5m away, that are used to distinguish incoming and outgoing cars.

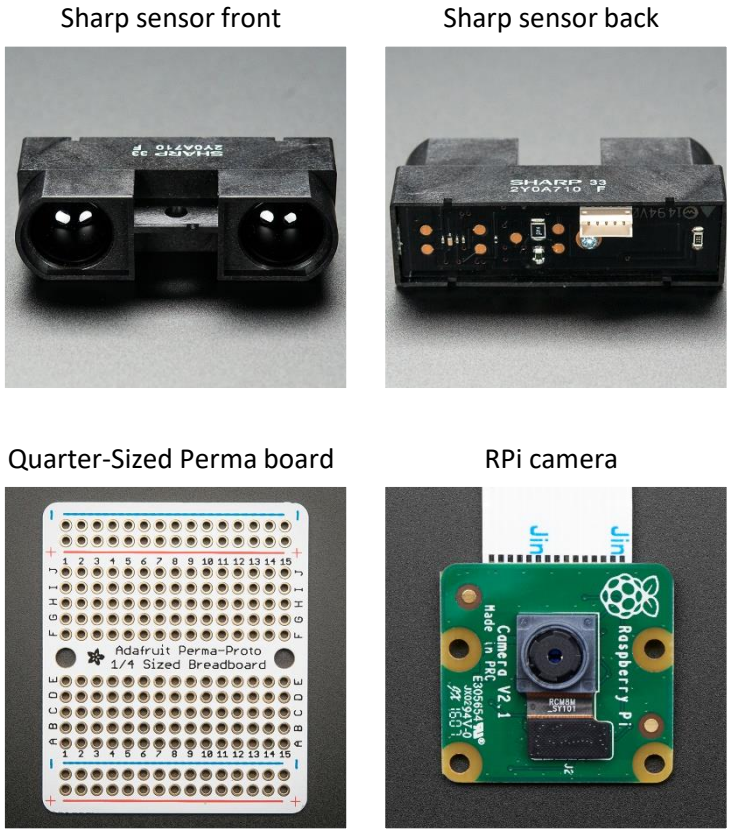


Figure 19 – Node #3 components

Node #4

Summary

The forth node is an internal node meant to provide information about the other nodes. It activates when it detects movement. The main design of the node is that of a "magic mirror", which is a two-way mirror in front of a computer monitor. The way two-way mirrors work is based on how much light they reflect in contrast to a regular mirror. While a regular mirror will reflect most of the light, two-way mirrors reflect part of the light thus creating the effect were the side of the mirror with the greatest lightning intensity acts as a mirror, while the darker side acts as a see-through glass. In this specific scenario, most of the screen remains dark while some parts displaying data about the nodes are bright, thus achieving the desired effect.

Housing

- Custom Wooden enclosure designed as a mirror with a frame.

Components – Functions

- *Raspberry Pi 3 Model B+*.
- *Power supply for the Raspberry Pi*. Rated at 5V – 2.5A
- *Micro SD card*. A 64GB card used for the software of the RPi
- *PIR*[86] *sensor*. A motion sensor to detect movement in front of the node
- *Adafruit Perma-Proto HAT* [200]. A HAT PCB used for the connections
- 2-way mirror.
- *Monitor*. A 23-inch monitor to display the information behind the mirror

Perma-prot HAT breadboard

PIR sensor

Figure 20 – Node #4 components

[86] Passive Infrared

3.1.2 Nodes topology & communication

With the functions of every node in mind, their placement at the university will be, as seen in the picture below. Every node is connected to the nearest access point, in order to get access to the university network

Figure 21 – IHU top view with the node positions annotated

The network topology obviously is based on the pre-existing infrastructure of the university and not on an ad-hoc network. As mentioned before more often than not, wireless sensor networks employ multi-hop mesh networks, where nodes communicate with each other in order to send data to the gateway nodes. This is not the case here. By using the university network, essentially the nodes are connected like in star network topology. In this scenario, nodes are not connected with each other but with a central gateway or in our case with different access points that lead to the same gateway.

Therefore, in this stage of the project the concept of the system design can be seen in the diagram in the next page.

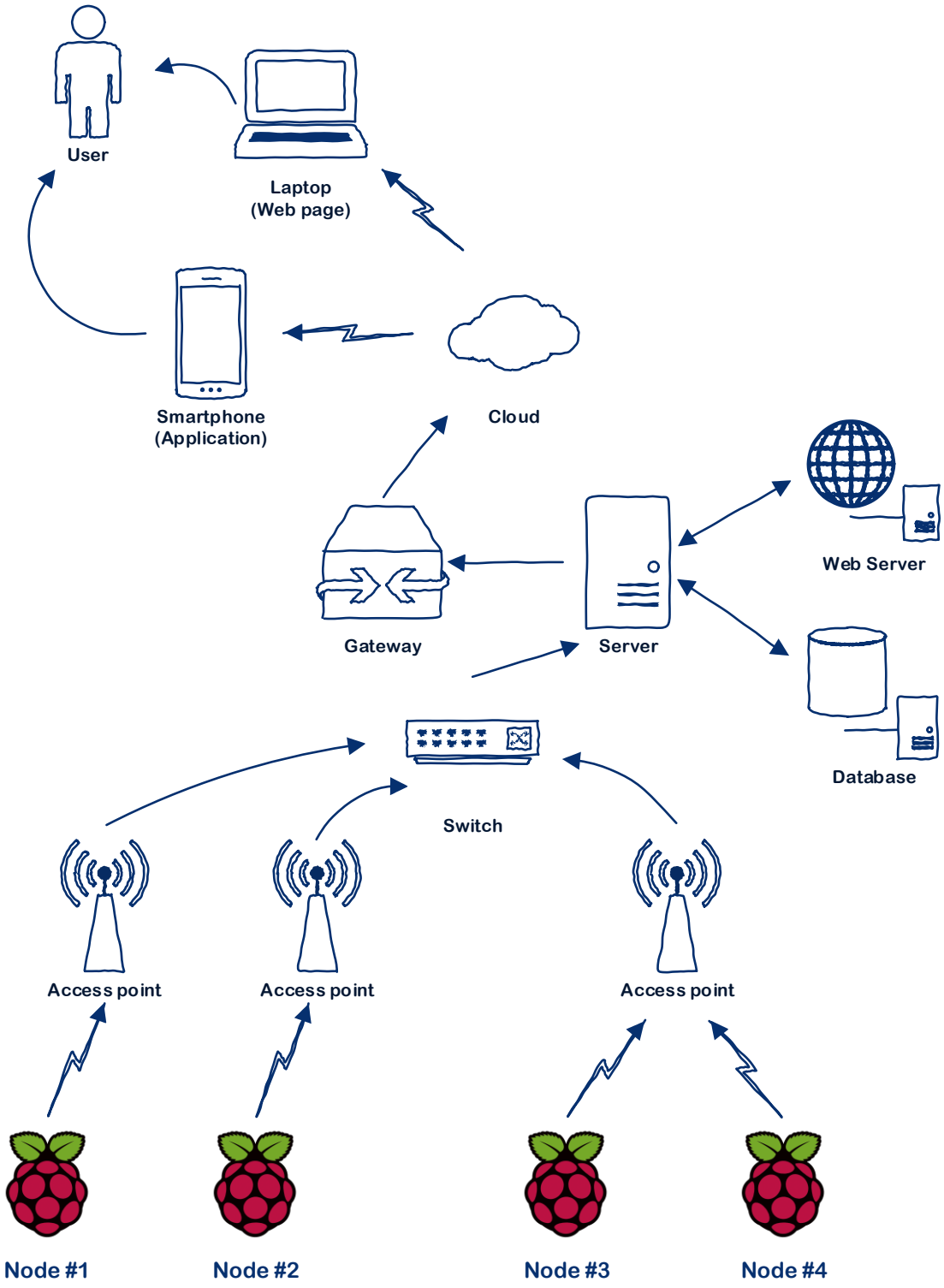


Figure 22 – System diagram

3.1.3 Sensors characteristics - connections

In this chapter we will see some of the sensors are connected to the RPi & the protocols they use.

BME680, TSL2591, SGP30

Characteristics. All three sensors support the I²C protocol for communication (the BME680 also supports SPI).

The I²C addresses are:

- BME680 – **0x77**
- TSL2591 – **0x29**
- SGP30 – **0x58**

Connections. Since the sensors have difference addresses, we can connect different sensors on the same I²C GPIO pins without problems as is the case with BME680 & TSL2591 in node #1 and BME680 & SGP30 in node #2. The connection with the Raspberry Pi is as seen in the table and drawing below.

Table 4 – I²C connections

Pin number	GPIO number	GPIO function	BME680 pins	TSL2591 pins	SGP30 pins
1	-	3.3V	VIN	VIN	VIN
3	#02	I²C SDA	SDI	SDA	SDA
5	#03	I²C SCL	SCK	SCK	SCL
6	-	GND	GND	GND	GND

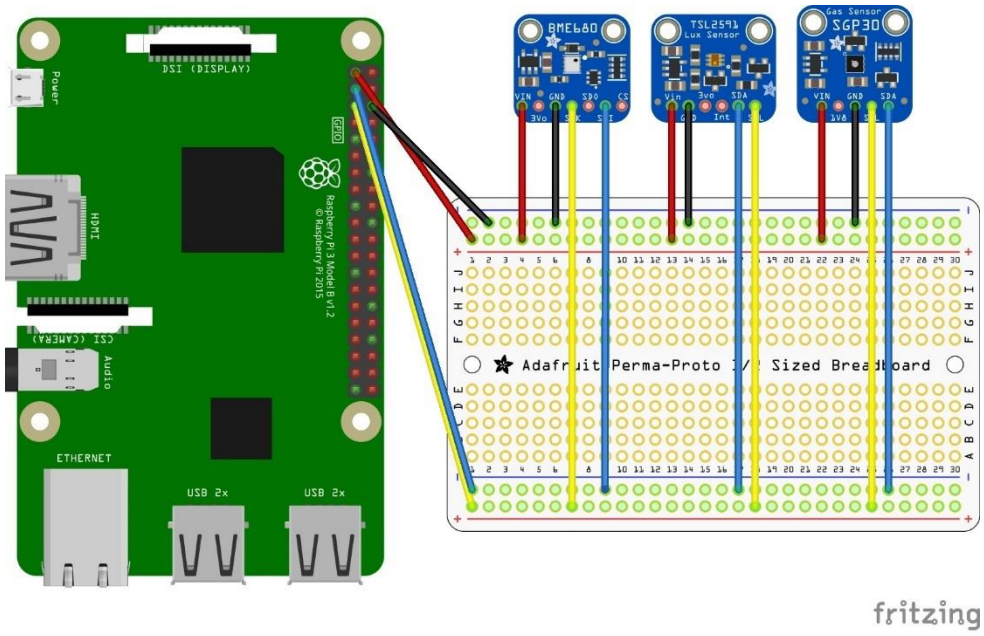


Figure 23 – I²C connections

Anemometer, Distance sensor

Characteristics. This sensor has an analog output. The problem with the Raspberry Pi is that it does not natively support analog inputs. The solution is using an ADC[87] like the MCP3008. The MCP3008 [201] is an 8-channel 10-bit ADC that provides an SPI interface to connect to the RPi as seen in the picture below.

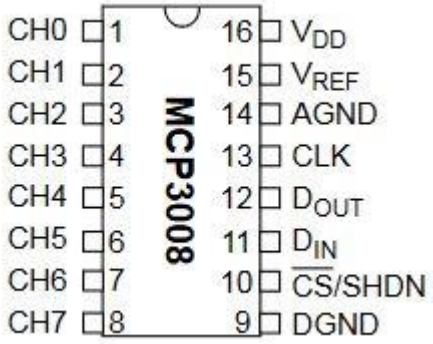


Figure 24 – MCP3008 pins

Connections. So, while the Raspberry Pi provides a hardware SPI, in the first node it is already used by the MFRC522 sensor. For that reason, we will use a software SPI for the MCP3008 as seen in the table and drawing below

Table 5 – MCP3008 connections

Pin number	GPIO number	GPIO function	MCP3008 pins
1	-	3.3V	VDD, VREF
6	-	GND	AGND, DGND
12	#18	-	CLK
15	#22	-	CS / SHDN
16	#23	-	DOUT
18	#24	-	DIN

[87] Analog-to-Digital Converter

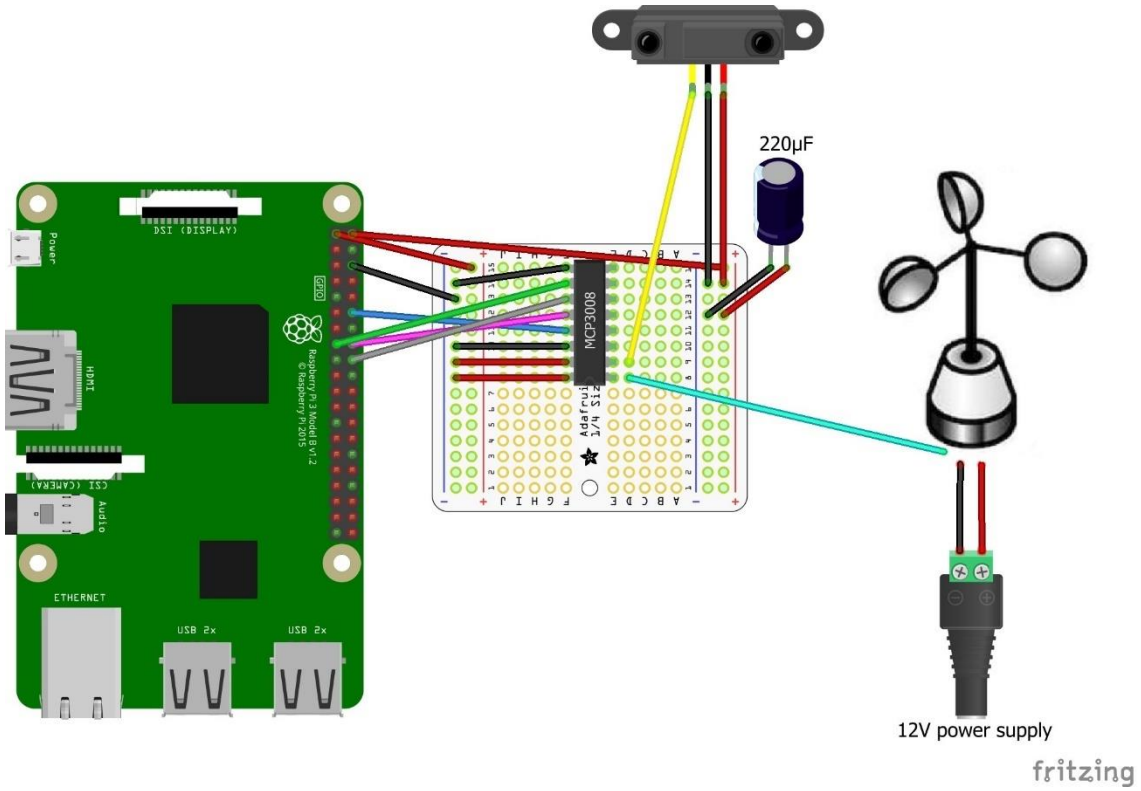


Figure 25 – Anemometer, Distance sensor - MCP3008 connections

PIR sensor

The PIR sensor has just three connections, power, ground and an output connected to any GPIO pin of the RPi.

Door Contact

The door contact is a simple reed switch, when the two parts are closed to each other the switch is closed and when they are apart it's open. It requires two cables, one on any GPIO pin of the RPi and the other on ground.

Camera

The Raspberry Pi camera is connected directly to the RPi using a flex cable.

MFRC522

Characteristics. The MFRC522 is a RFID reader/writer that works at 13.56 MHz

Connections. As mentioned before the MFRC522 uses the SPI protocol to communicate with the Raspberry Pi and specifically the hardware SPI and is connected as seen in the table and drawing below

Table 6 – MFRC522 connections

Pin number	GPIO number	GPIO function	MFRC522
1	-	3.3V	3.3V
6	-	GND	GND
19	#10	SPI0 MOSI	MOSI
21	#09	SPI0 MISO	MISO
22	#25	-	RST
23	#11	SPI0 CLK	SCK
24	#08	SPI0 CE0	SDA

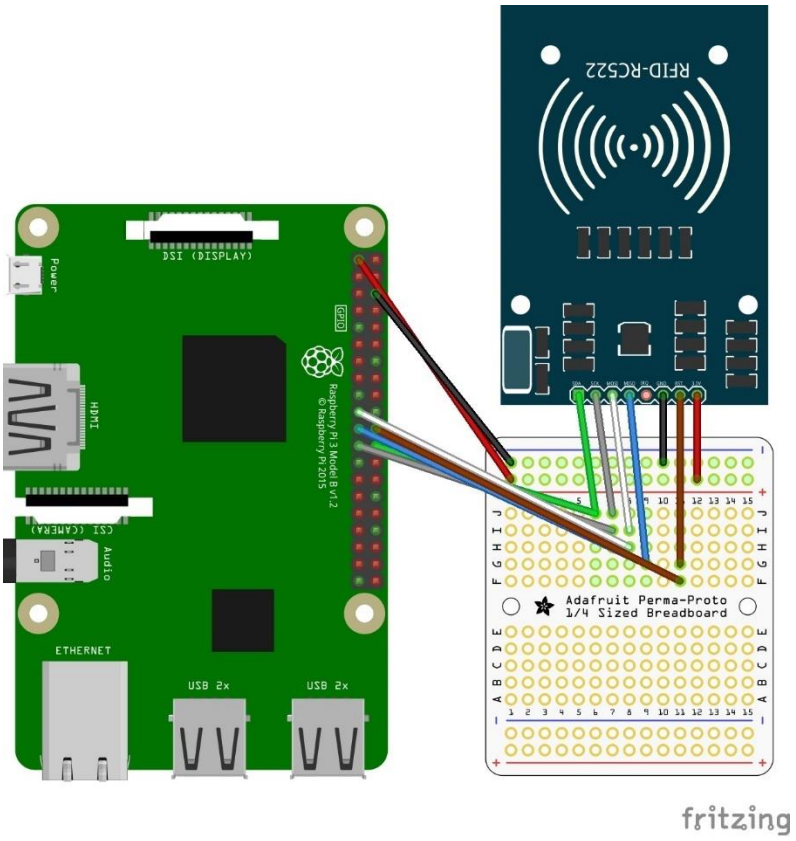


Figure 26 – MFRC522 connections

Lock-Style Solenoid

Characteristics. The solenoid is basically an electromagnet that operates at 9-12VDC, 500-650mA.

Connections. The desired effect is to control the lock-style solenoid with a GPIO pin. In order to accomplish that we will use a TIP120 Darlington transistor to control the power that reaches the solenoid. Additionally, we will need a 2.2KΩ resistor and a 1A 1N4001 diode.

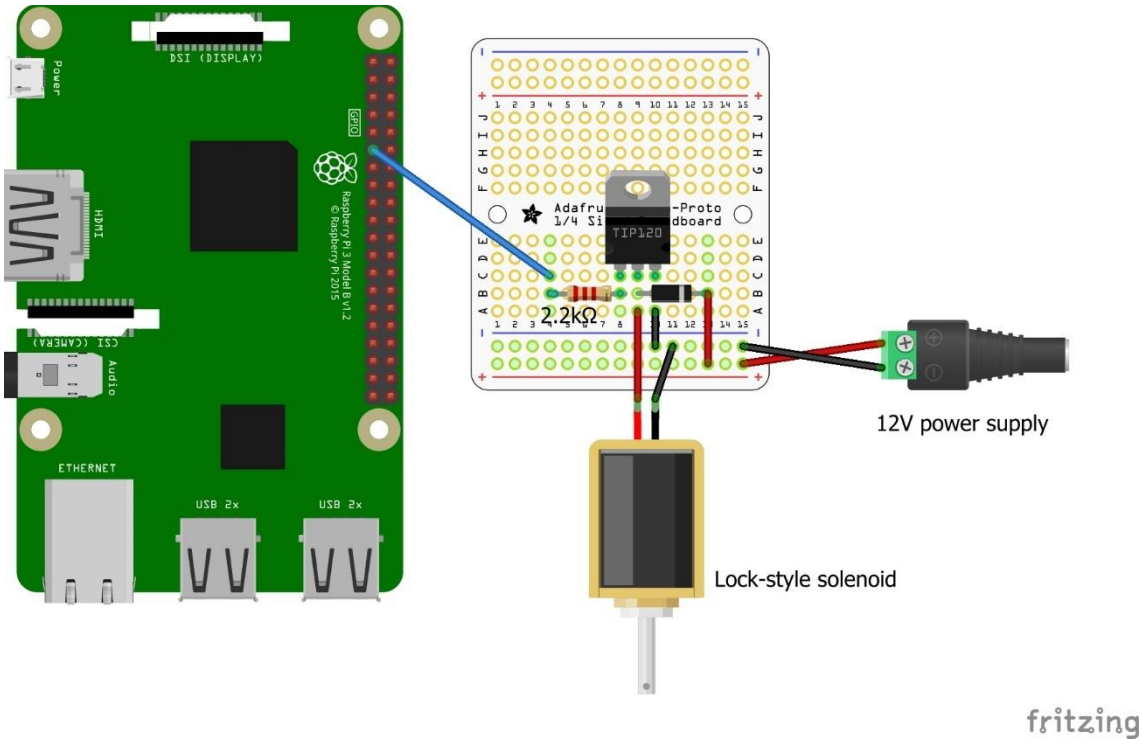


Figure 27 – Lock-style solenoid connections

LCD RGB Display

Characteristics. It is 2-row by 16-character RGB display that uses the common Hitachi HD44780 controller.

Connections. Display based on this controller always have similar connections as seen in the diagram and drawing in the next page.

Table 7 – LCD RGB display connections

Pin number	GPIO number	GPIO function	RGB LCD
2	-	5v	VDD, LED+
6	-	GND	VSS, RW
29	#05	-	RS
31	#06	-	E
32	#12	-	DB4
33	#13	-	RED
35	#19	-	GREEN
36	#16	-	DB5
38	#26	-	BLUE
38	#20	-	DB6
40	#21	-	DB57

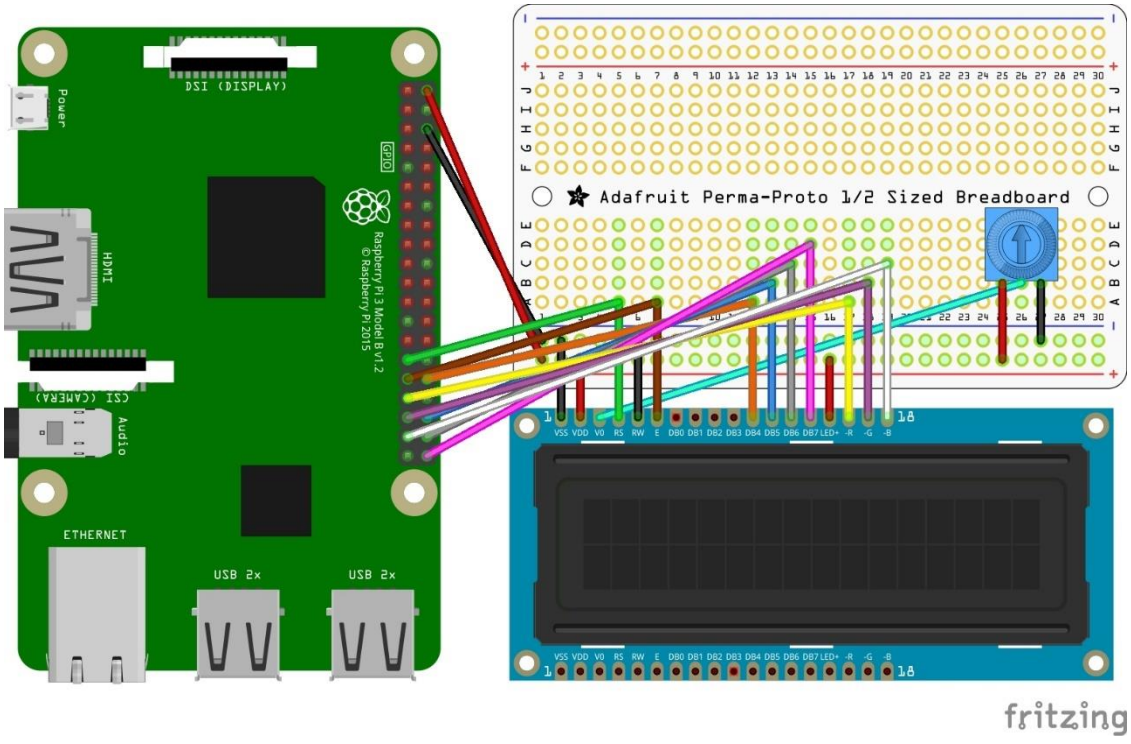


Figure 28 – LCD RGB display connections

3.2 Software design

Therefore, before the actual implementation of the programs, scripts, services, etc. that this system will consist from, we need to analyze and visualize the system by breaking down the components and their functions. The design & analysis stage is part of many major system life-cycle software design processes like the waterfall model [202] or the agile software development [203]. Usually life cycles will have most of the stages shown at the diagram below.

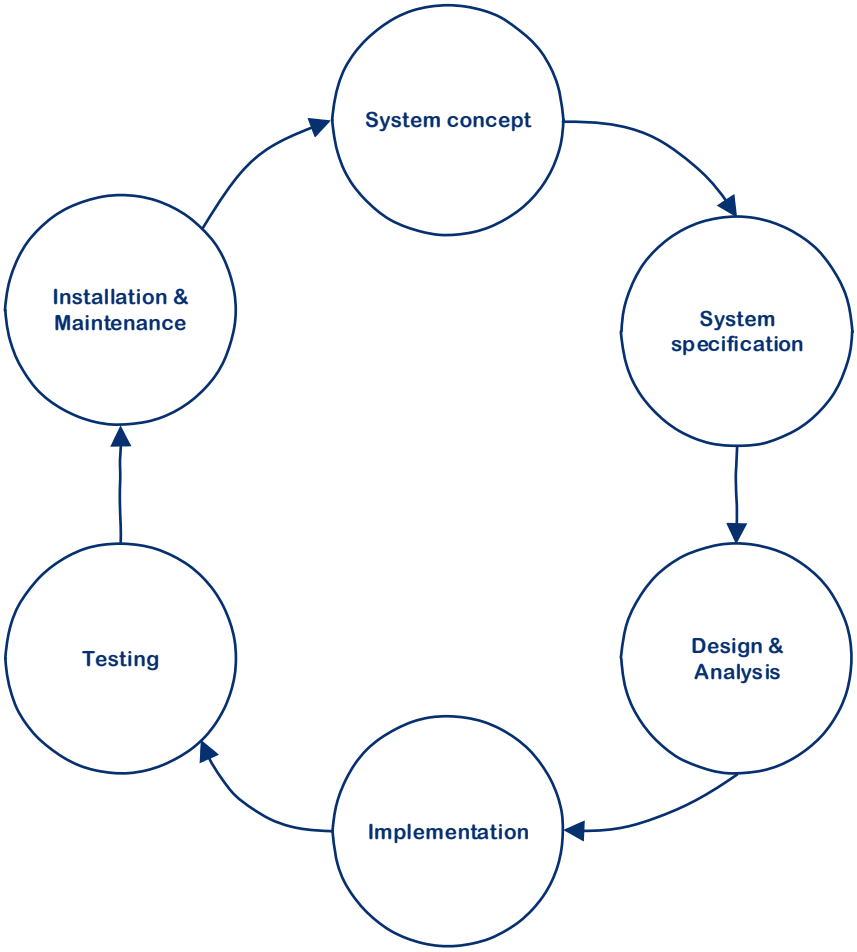


Figure 29 – System development common life-cycle stages

Hence, before implementation begins, we need to break down system processes in the design & analysis phase and a great way to do that is by using UML[88] and specifically UML activity & sequence diagrams. UML is a modeling language developed [204] by Grady Booch, James Rumbaugh and Ivar Jacobson back in late 1990's as a way to model object-oriented programs which had become the dominant programming method by then. UML is used to represent static

[88] Unified Modeling Language

and dynamic data in the software and as such, there are two categories of diagrams responsible for the different data. Structural diagrams that include class, object, etc. diagrams and behavioral diagrams that include use case, activity, sequence, etc. diagrams. In this project, we will use activity and sequence diagrams to model the processes of the system.

3.2.1 Node design

The current system design consists of four nodes. Three of the nodes mainly collect data and send them to the server and the fourth is displaying them. Additionally, the first node has sensors that are dealing with access to the enclosure housing the components. Therefore, we can discern between the four following services:

- Reading data
- Access data
- Sending data
- Displaying data

While the system concept phase is not mentioned in this thesis, the idea was that the outdoor and indoor weather nodes would gather data and send them to the server to be displayed in the fourth node and of course on the website and mobile application. But also, that the nodes themselves will have an LCD display to provide live information about their operation. That made the design different in the sense that the reading data service had to be in somewhat sequential order so that the user could properly see live updates. Additionally, in order for the access part of the system to work live as well it could not be in sequential order but running in parallel. Finally, for the nodes that need to send data to the server. It could be a method of the reading data service but ultimately, I decided against it with the reasoning that the gathering and transmitting of the data can and should be two distinct services, because in case of a network failure for example the node would continue to work but send the data then the problem was resolved.

So, to recap, the services of the nodes are the following for each node

Table 8 – Node services

	Reading	Access	Sending	Displaying
Node #1	x	x	x	
Node #2	x		x	
Node #3	x		x	
Node #4				x

Node #1

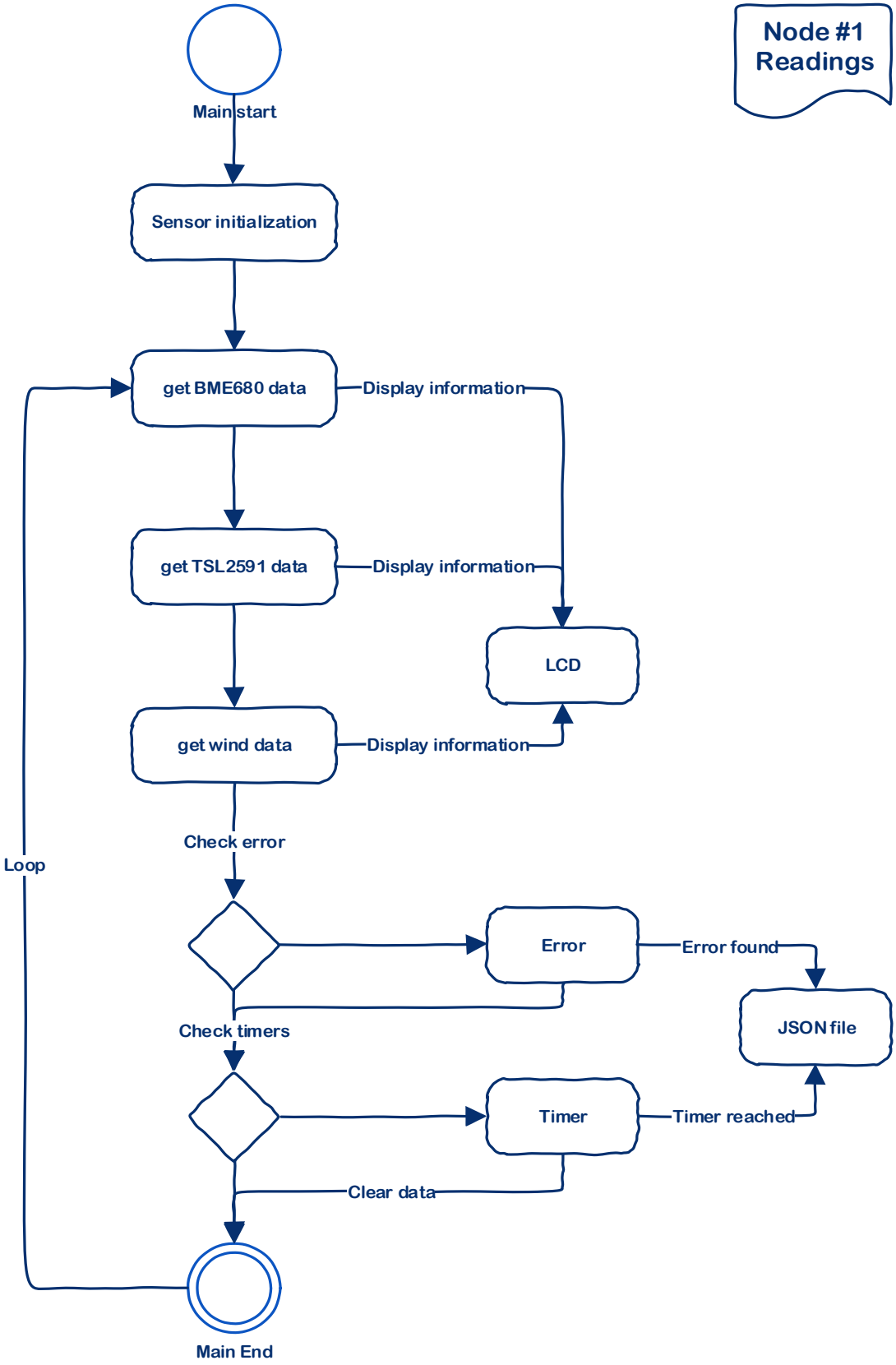


Figure 30 – Node #1 Readings activity diagram

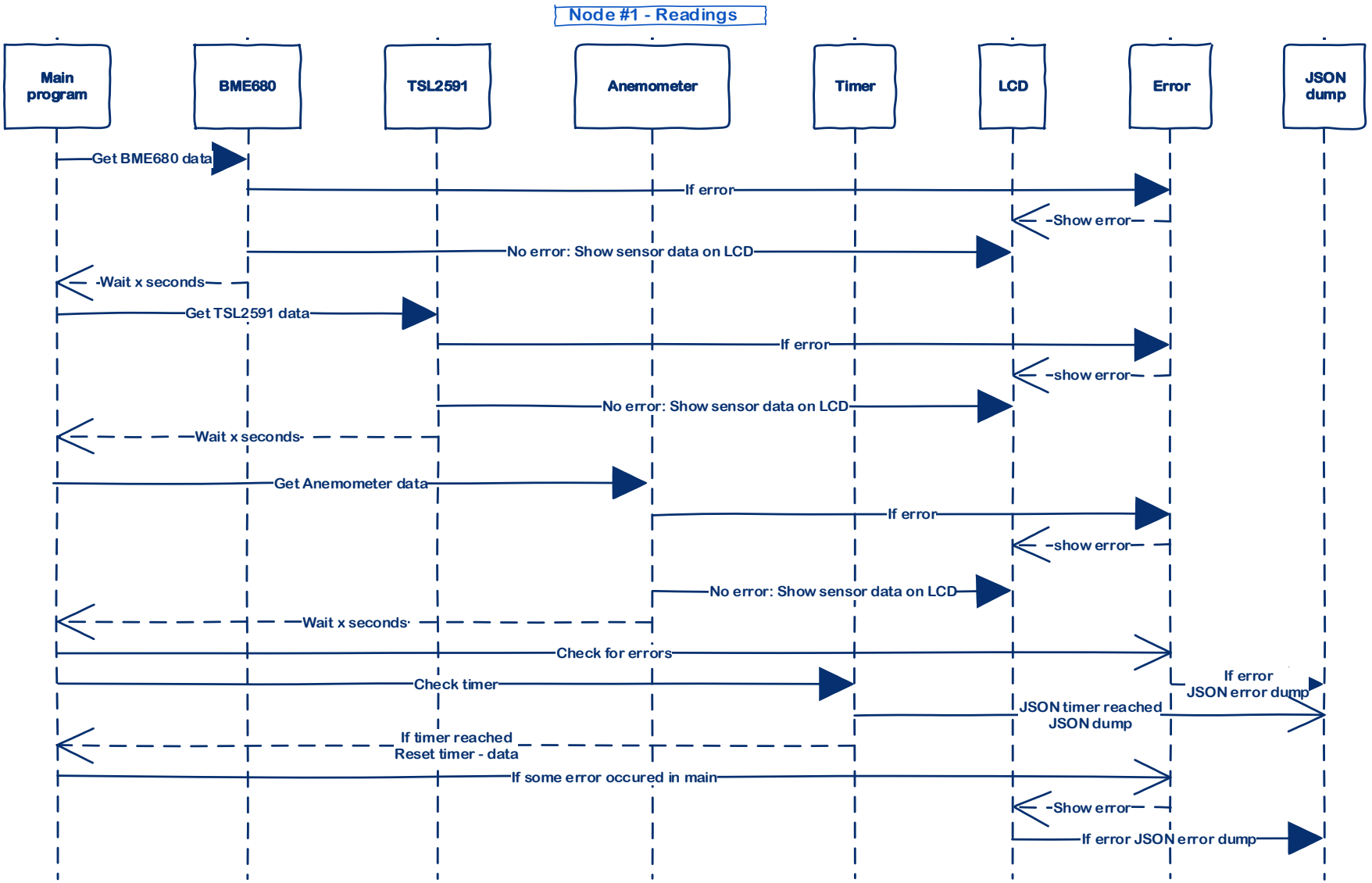


Figure 31 – Node #1 Readings sequence diagram

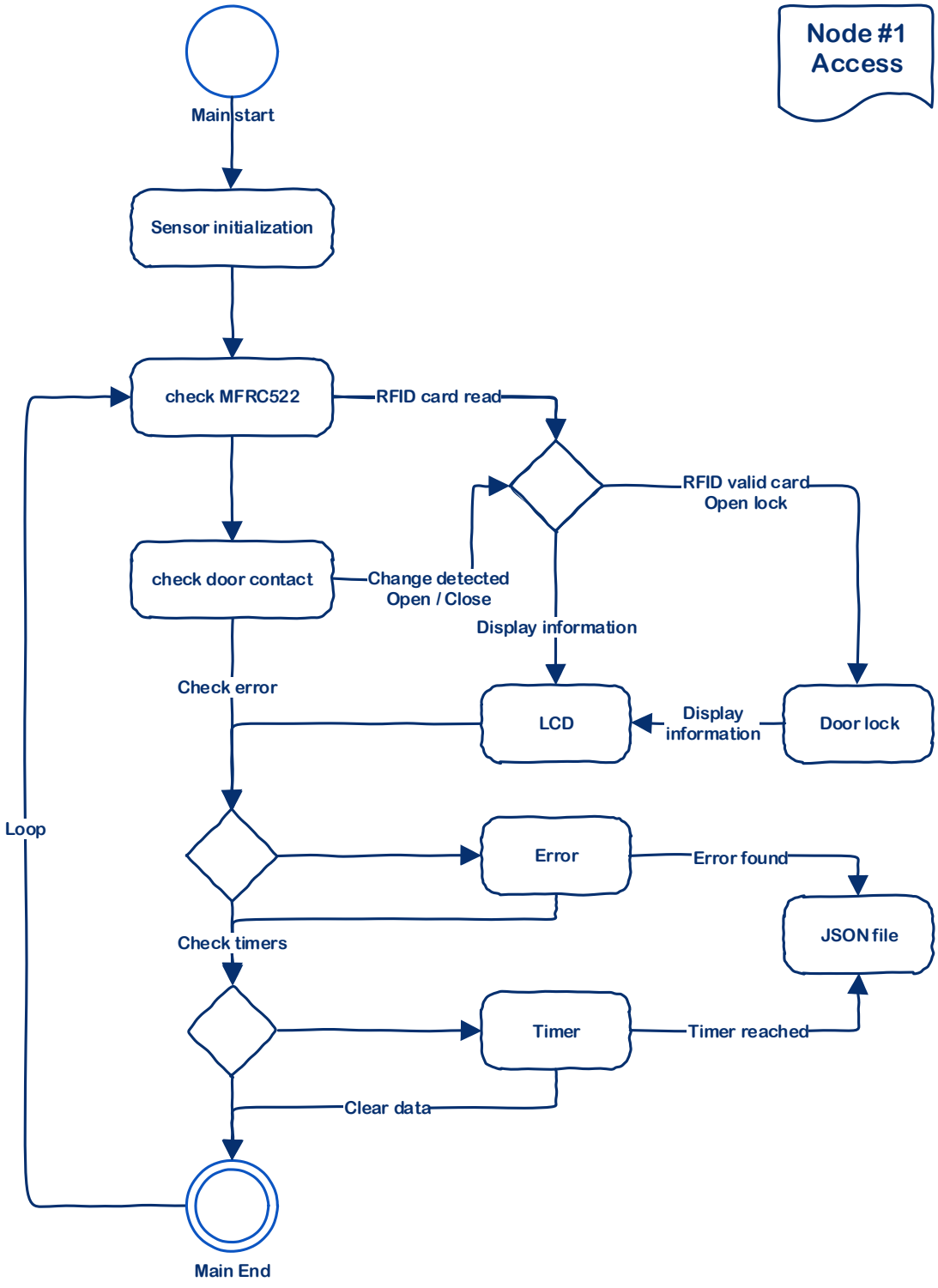


Figure 32 – Node # 1 Access activity diagram

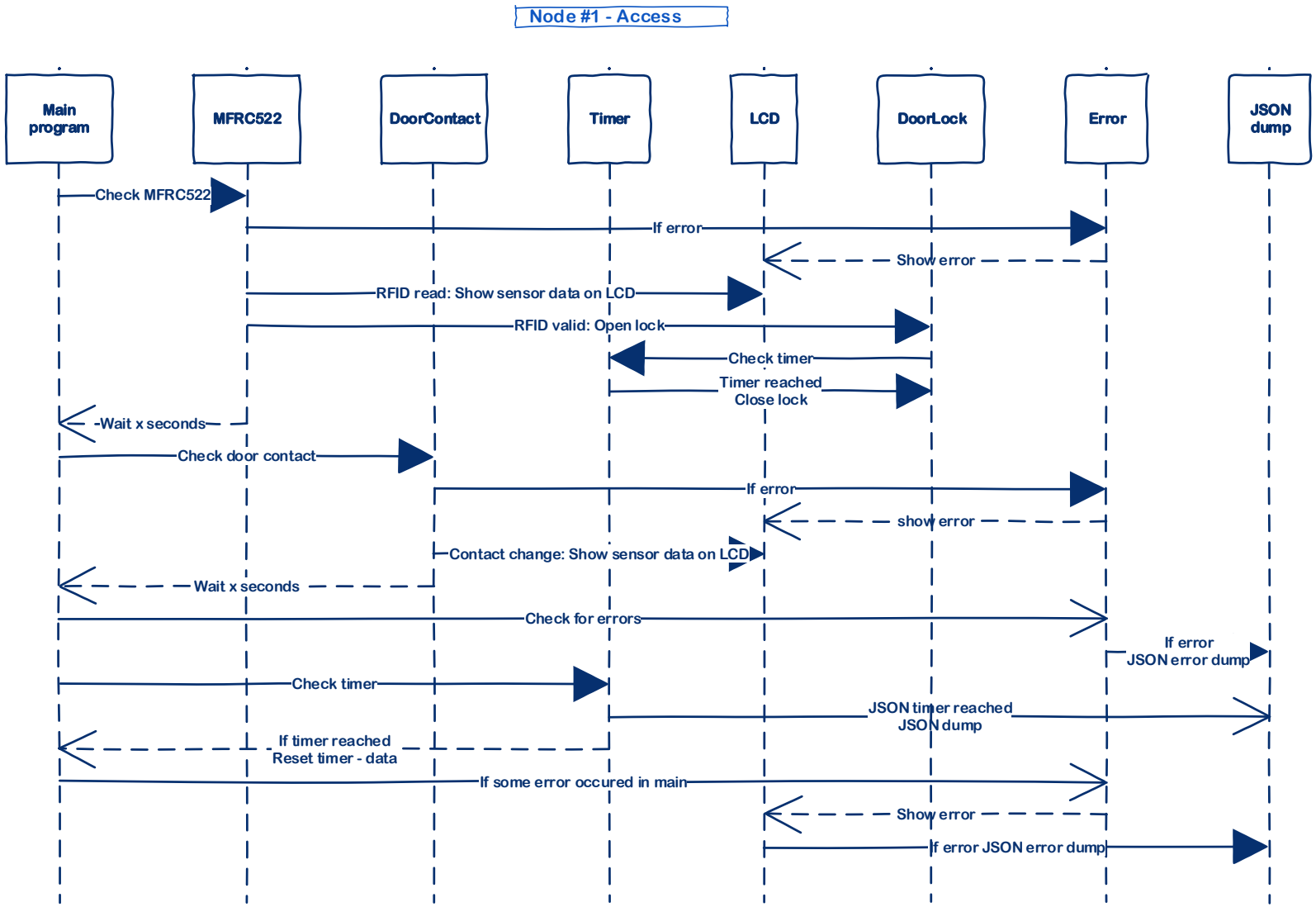


Figure 33 – Node #1 Access sequence diagram

Node #2

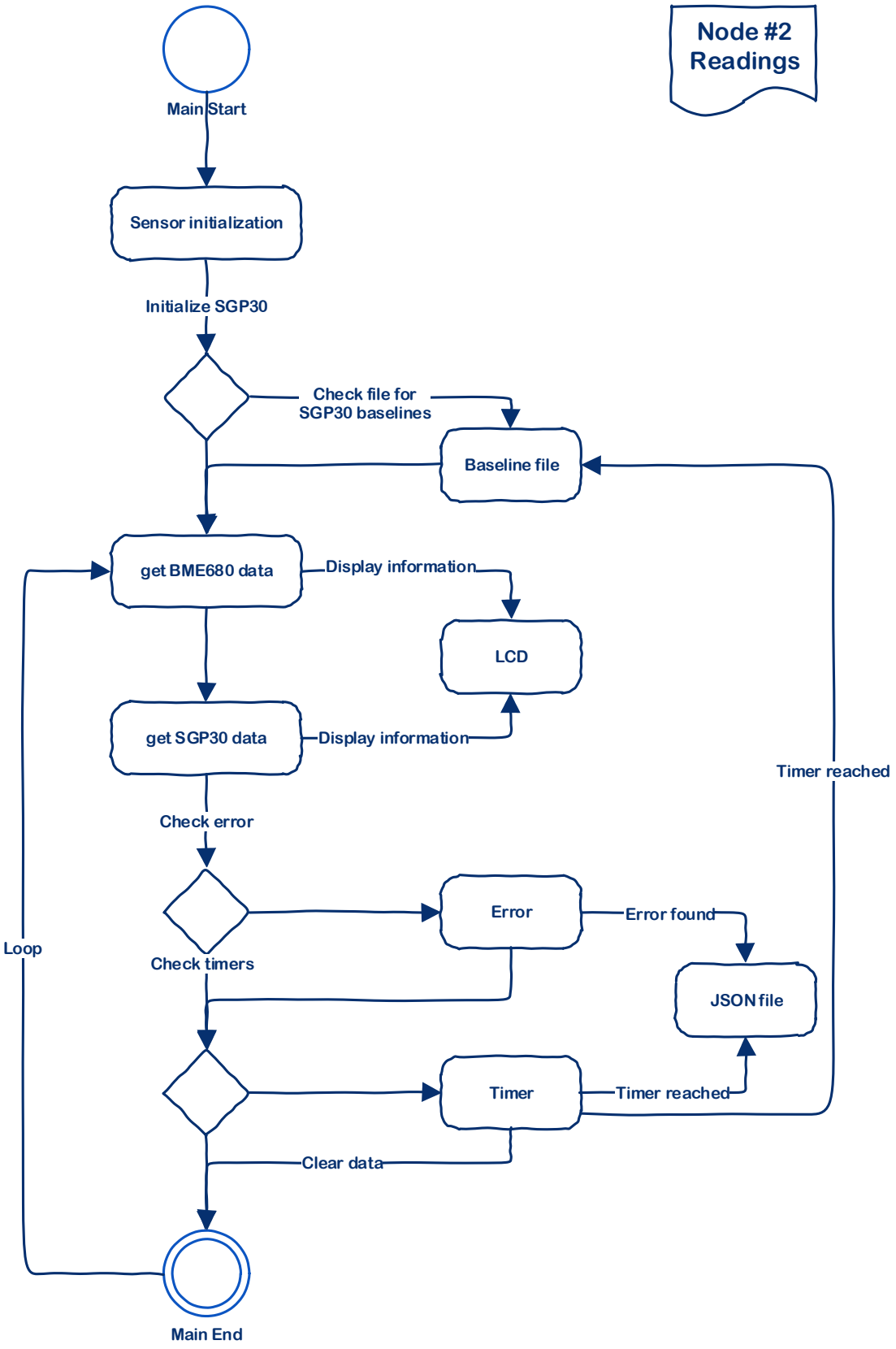


Figure 34 – Node #2 Readings activity diagram

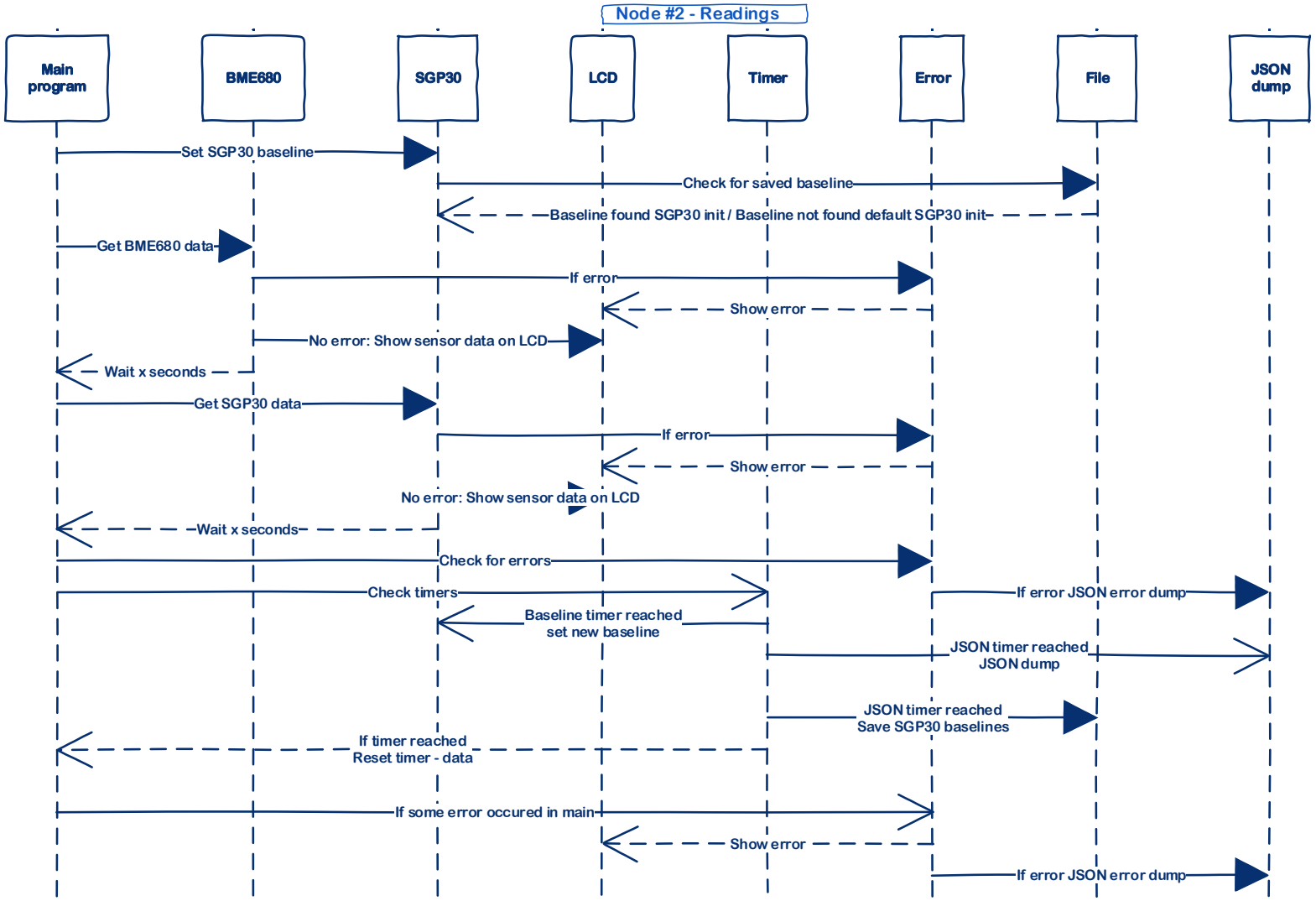


Figure 35 – Node #2 Readings sequence diagram

Node #3

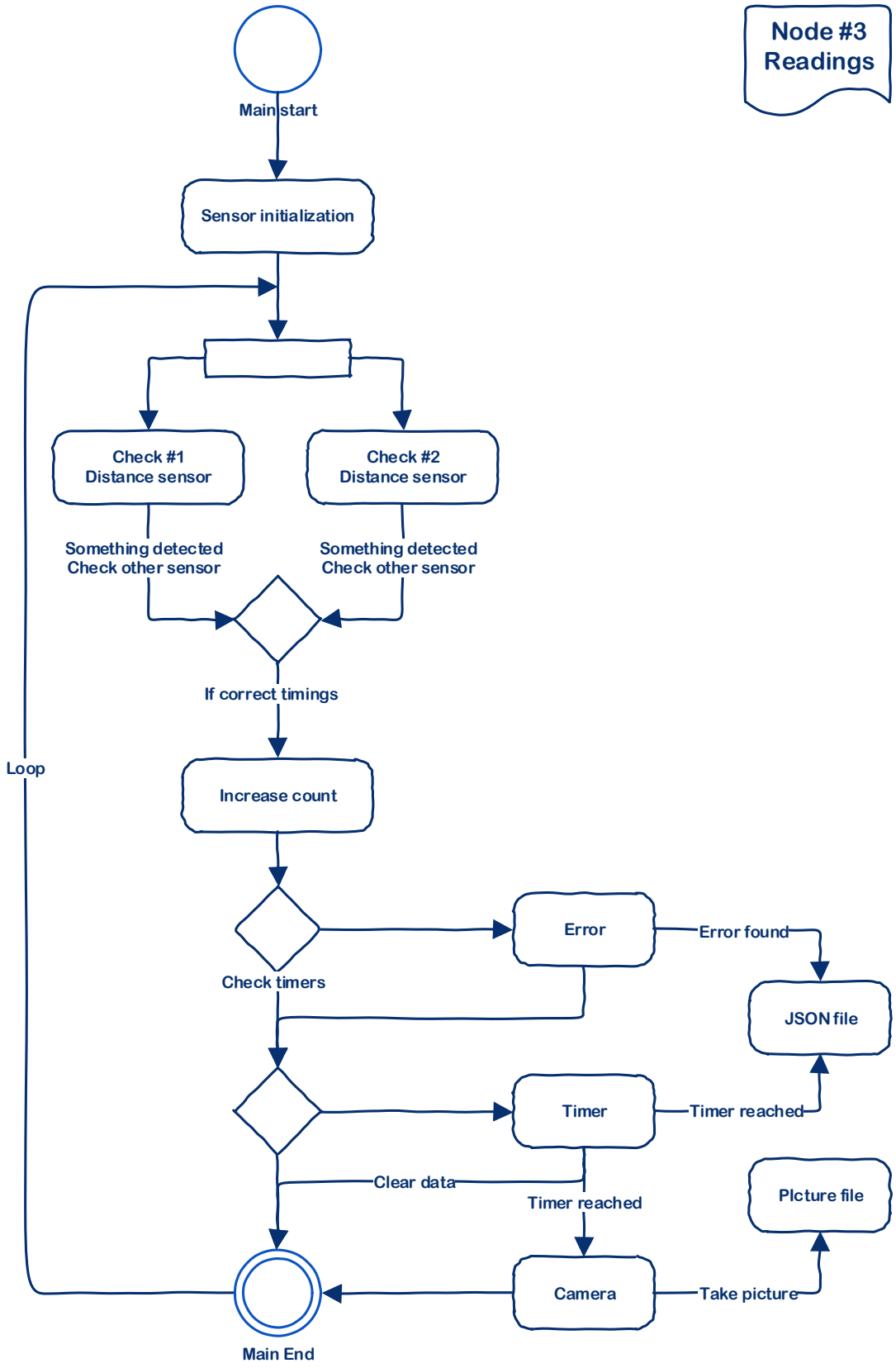


Figure 36 – Node #3 Readings activity diagram

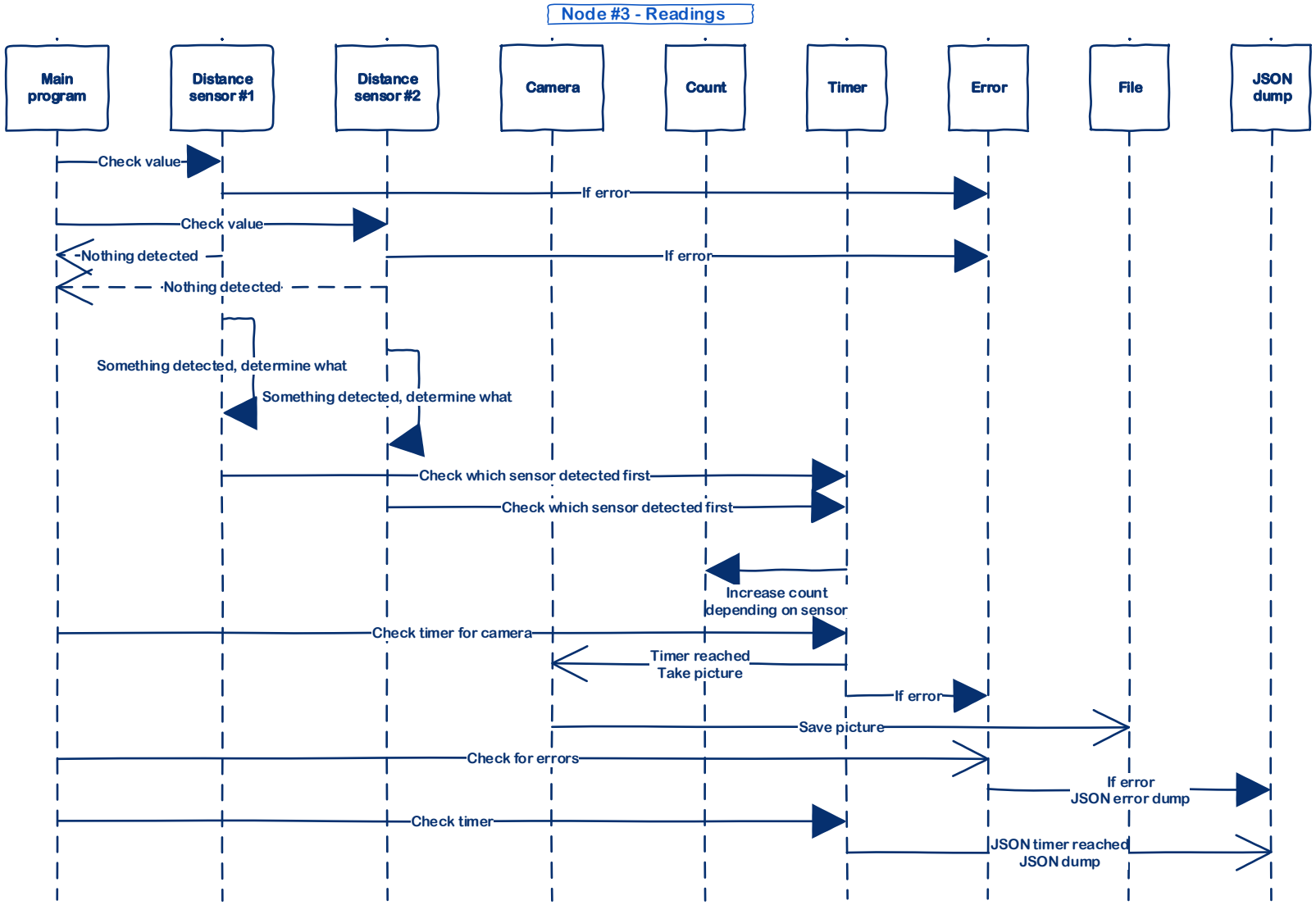


Figure 37 – Node #3 Readings sequence diagram

Node #4

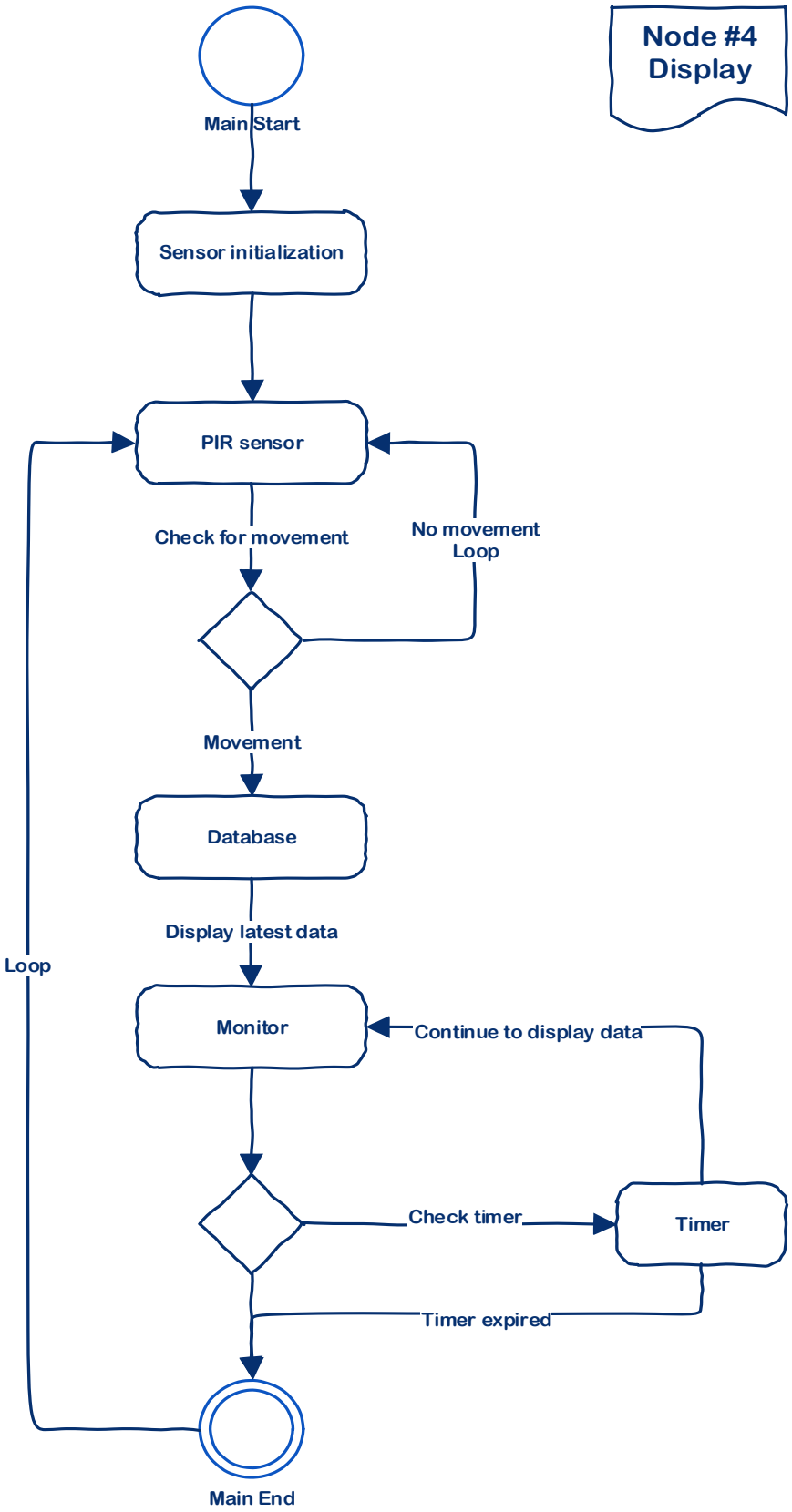


Figure 38 – Node #4 Display activity diagram

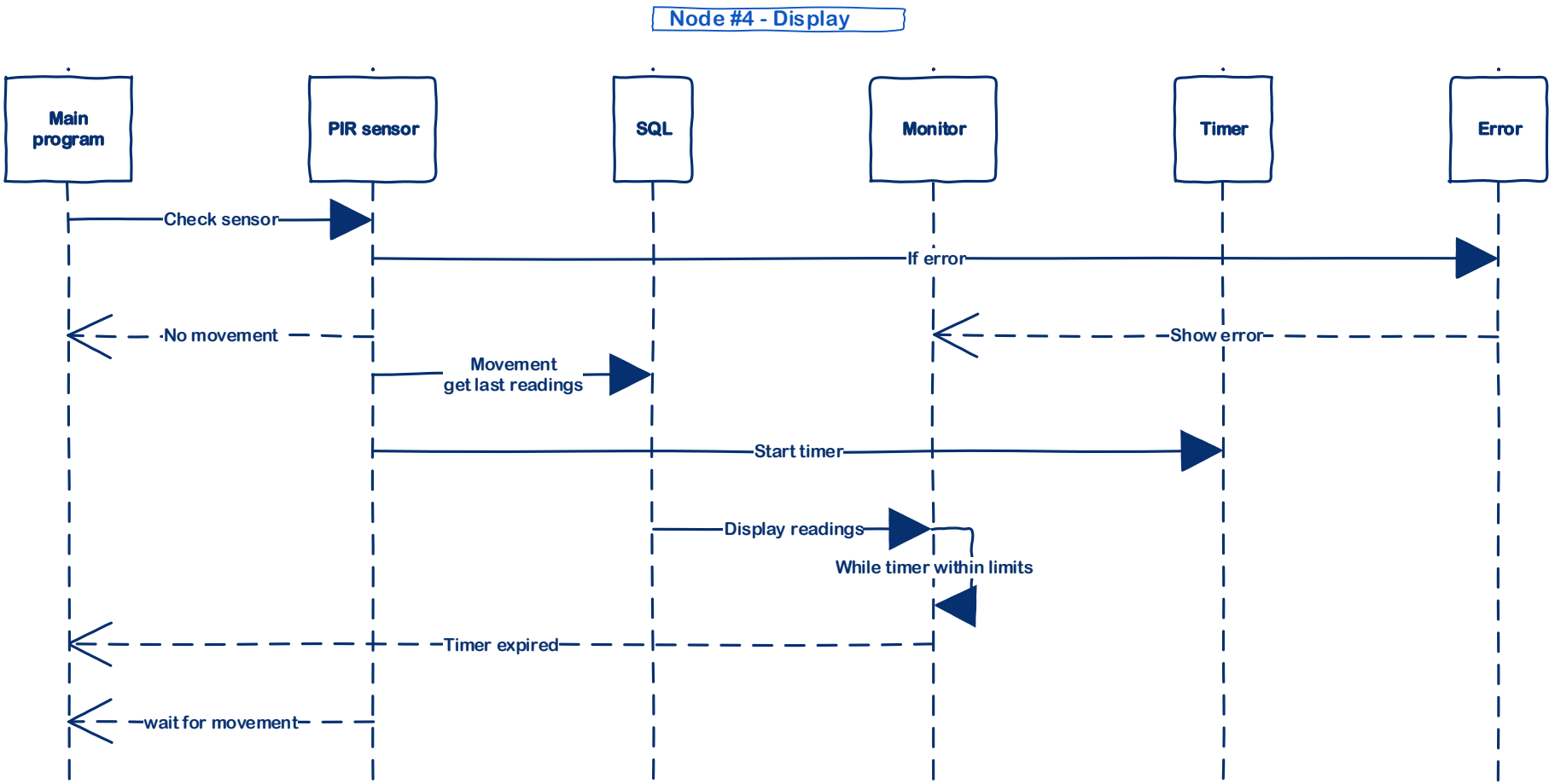


Figure 39 – Node #4 Display sequence diagram

Nodes #1-3

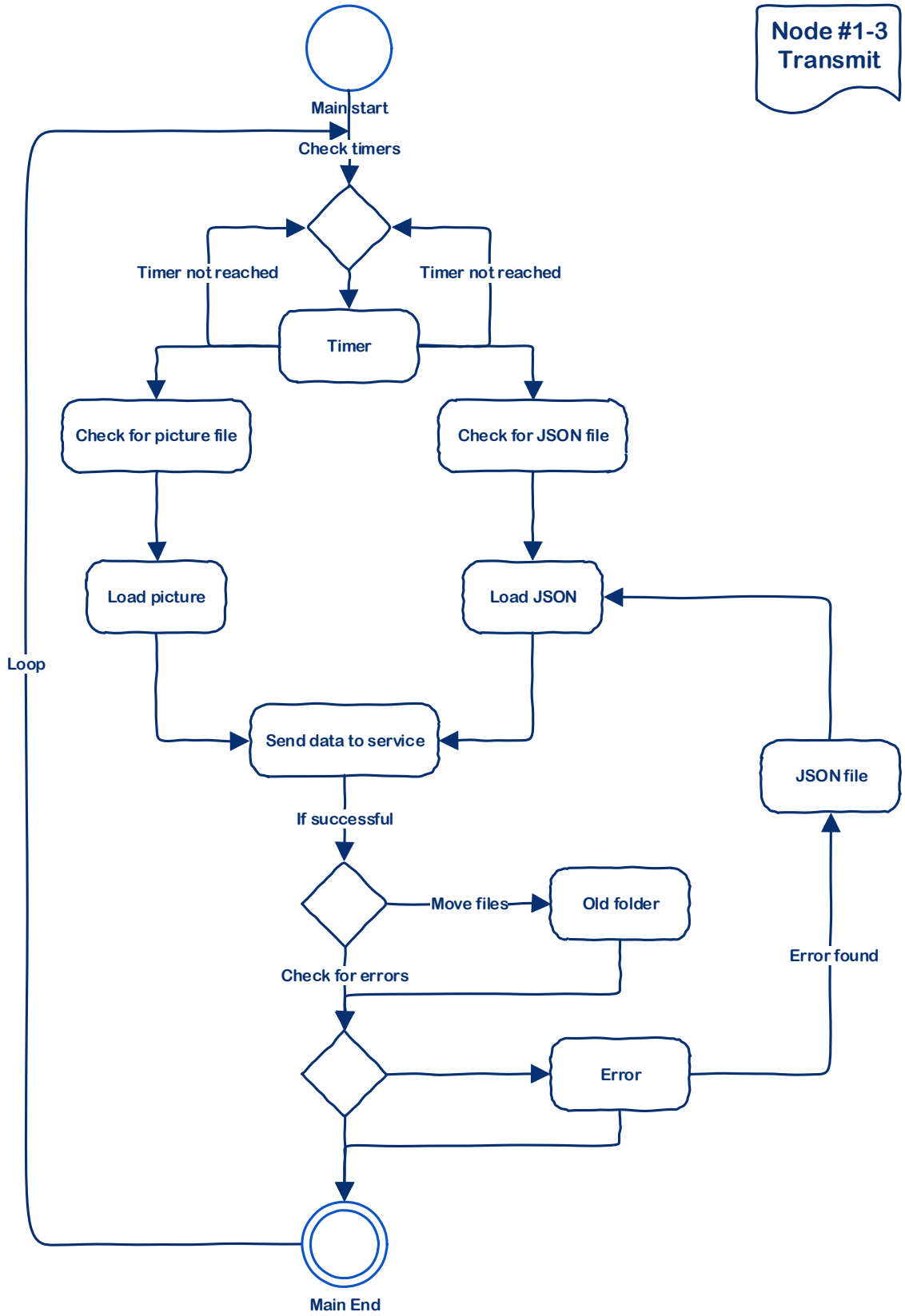


Figure 40 – Node #1-3 Transmit activity diagram

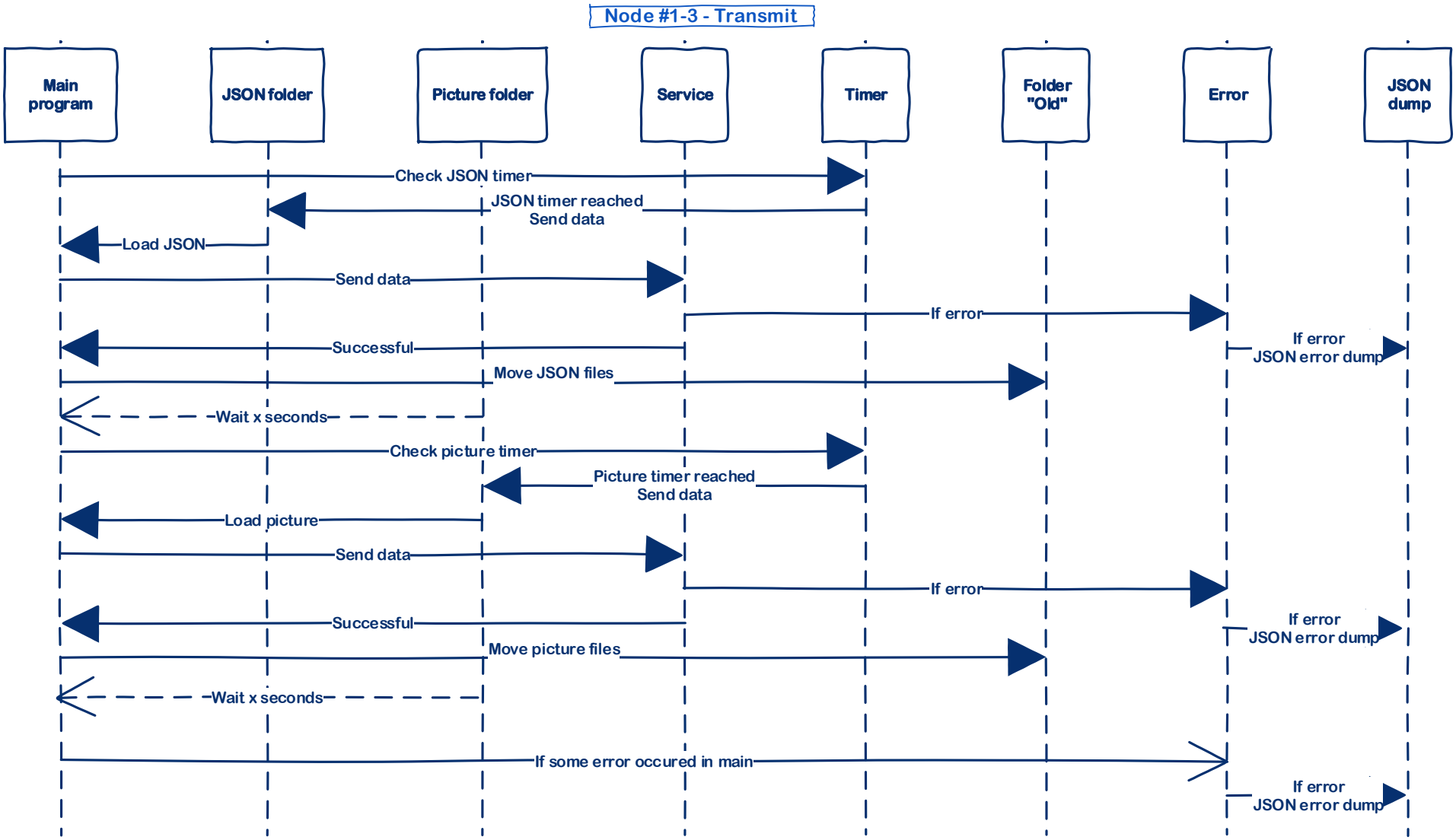


Figure 41 – Node #1-3 Transmit sequence diagram

4 Implementation

In this chapter we will try to implement the design analysis of the previous chapter and of course adjust it as needed, in order to achieve the desired result. As mentioned earlier the programming language chosen for the overall project, was Python, or at least for the client-side programming of the Raspberry Pi. Nevertheless, as mentioned on the design phase, in order for node #4 to display the data, most of monitor should be dark and thus it requires some kind of visual programming like Java. Python maybe the most popular language on the RPi platform but Java [124] was supported from the beginning, is open-source and of course it runs fine on Raspbian as the original Sun Microsystems moto claim [205], "Write once, run anywhere" thanks to Java's JVM[89].

4.1 Client-side programming

As mentioned in the design phase, there are four type of services that run on the nodes, reading, access, transmitting and displaying data. In this part we will see parts of the code that compose the different scripts of each type.

4.1.1 Reading data

In every python script and actually in programming in general, initially there is an import of the libraries needed and this is especially true in this project because of the sensors used. Each sensor requires a library in order to interface it with Python. In this case we mostly use the manufacturer's libraries, Adafruit.

Importing libraries

Adafruit libraries

```
import adafruit_bme680
import adafruit_tsl2591
import Adafruit_MCP3008
import digitalio
import busio
import board
import adafruit_character_lcd. character_lcd as characterlcd
```

[89] Java Virtual Machine

Each of these libraries provide methods in order to initialize the sensors or devices and retrieve or display data.

System libraries

```
import time
import datetime
import json
import os
import logging
```

These libraries are in Python standard library and are needed for many things like retrieving the current date-time, creating json or logging.

Sensor initialization

```
lcd_columns = 16
lcd_rows = 2
lcd_rs = digitalio. DigitalInOut (board. D5)
lcd_en = digitalio. DigitalInOut (board. D6)
lcd_d4 = digitalio. DigitalInOut (board. D12)
.
.
i2c = busio.I2C (board.SCL, board.SDA)
bme680 = adafruit_bme680.Adafruit_BME680_I2C (i2c, debug=False)
.
.
CLK = 18
MISO = 23
MOSI = 24
CS = 27
mcp = Adafruit_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)
```

Next, we need to initialize the sensors and devices by setting which GPIO pins they use

Methods

Every method, including the main program is enclosed in a try-catch-finally statement as is common practice in programming in order to catch errors and exceptions

```
try:
.
.
except Exception as e:
 .
 .
finally:
.
.
```

Almost every node uses a method for creating the JSON files containing the readings or errors of the sensors-devices in the node.

```
Data = {
        "Node_Name": Node_Name,
        "Date": time.strftime("%d-%m-%Y %H:%M:%S"),
```

```
        "General_Error": General_Error,
        "General_Error_Message": General_Error_MSG if General_Error is
True else None,
        "Readings": Readings,
        "BME680": [
            {"BME680_Temperature": BME680_Temperature} if BME680_Read
is True else None,
            {"BME680_Humidity": BME680_Humidity} if BME680_Read is
True else None,
.
.
  ] if BME680_Read is True else None,
elif not Readings:
    Data = {
        "Node_Name": Node_Name,
        "Date": time.strftime("%d-%m-%Y %H:%M:%S"),
        "General_Error": General_Error,
        "General_Error_Message": General_Error_MSG if General_Error is
True else None,
        "Error Data": [
            {"BME680_Error": BME680_Error} if BME680_Error is True
else None,
            {"BME680_Error_MSG": BME680_Error_MSG} if BME680_Error is,
.
.
  ] if BME680_Error is True else None
  }
```

After which, the JSON object gets exported in a text file

```
JSONFilename = JSONDirectoryNew + Node_Name + "_ReadingsJSON_ " +\
    time. strftime("%d-%m-%Y_%H-%M-%S"). replace (" ", "-") + ".txt"

with open (JSONFilename, 'w') as outputFile:
    json. dump (Data, outputFile)
```

Methods for sensors gathering or reading data are pretty straight forward because of the use of the manufacturers libraries we mentioned earlier. So, for example getting the measurements from sensor BME 680 is as follows

```
BME680_Temperature = str (round (bme680.temperature, 5))
BME680_Humidity = str (round (bme680.humidity, 5))
BME680_Gas = str(bme680.gas)
BME680_Pressure = str (round (bme680.pressure, 5))
```

We just round them in 5 decimals and convert them to string type for use later.

In some cases, such as the anemometer method, we need to perform some calculations, in order to get usable data. At referred in the design phase, the anemometer returns an analog signal for the voltage based on how fast it is spinning.

In our design we get this output with the help of the ADC MCP3008 which outputs a value that corresponds to a voltage value. According to the manufacturer's datasheet (equation 4-2) the output is calculated with the equation shown below.

$$Digital\ Output = \frac{1024\ x\ Vin}{VRef}$$

Also, in the prototyping phase we took measurements with a multimeter and saw that while the anemometer was not moving, the output value from the MCP3008 channel was around 124-127.

According to the anemometer manufacturer's datasheet, the voltage correlation with wind speed in meters per seconds is the following

$$\frac{Windspeed - 0.4}{1.6} * 32.4$$

In the program it looks like this

```
MCP3008_output = mcp. read_adc (0)
WindSpeedVoltage = MCP3008_output / 310
MCP3008_output2 = str (MCP3008_output)
WindSpeed = ((WindSpeedVoltage - 0.4) / 1.6) * 32.4
```

Main program

In the main program there is a continuous loop that makes calls to the methods getting data from the sensors and when conditions are met, either and occurred error or a set time limit was reached, dump JSON data into a file.

```
while True:
    if Readings_start is None:
        Readings_start = time.time()
    if SGP30_Baseline_Timer is None:
        SGP30_Baseline_Timer = time.time()

    print(Node_Name + " : " + time.strftime(
        "%d-%m-%Y %H:%M:%S") + " : " + "Main" + " : " + "Getting
BME680 readings!!")
    getBME680Data()
    time.sleep(2)
.
.
.
if General_Error or BME680_Error or SGP30_Error:
    lcd.color = [100, 0, 0]  # red color
    lcd.message = "JSON error dump\n" + time.strftime("%d-%m-%Y
%H:%M:%S")
    print(Node_Name + " : " + time.strftime("%d-%m-%Y %H:%M:%S") + " :
" + "Main" + " : " +
          "Json error dump!!")
    writeJSON()
.
.
.
if (Readings_now - Readings_start > 90) and Readings:
    print(Node_Name + " : " + time.strftime("%d-%m-%Y %H:%M:%S") + " :
" + "Main" + " : " + "Json dump!!")
    lcd.color = [50, 0, 50]  # purple color
    lcd.message = "JSON dump\n" + time.strftime("%d-%m-%Y %H:%M:%S")

    writeJSON()
    Data.clear()
```

4.1.2 **Access data**

This script is specific to the first node for controlling and registering access in the enclosure of the node and while it is similar to the readings script, it is also the only script that manipulates a device, the lock solenoid.

The two devices responsible for access are the door contact and the lock solenoid

In the case of the door contact the method just checks if the GPIO input has changed

```
if GPIO.input(7) == 1 and not Door_Open:
    Door_Open = True
    Access = True
    Door_Contact = True
    print (Node_Name + ": " + time. strftime ("%d-%m-%Y %H: %M: %S ")
+ ": " + "checkDoorContact" + ": " + "Door #2 open! - Magnetic con-
tact")
    logger.info (Node_Name + ": " + time. strftime ("%d-%m-%Y %H: %M:
%S ") + ": " + "checkDoorContact" + ": " + "Door #2 open! - Magnetic
contact")
```

In the case of the lock solenoid, by changing the GPIO pin output, it manipulates voltage reaching the lock with the help of the TIP120 transistor.

```
id, text = reader. readInterval ()
.
.
if (id) is not None:
    textremovespaces = text. rstrip ()

    for x in Access_list_id:
        if x == id:
            Access = True
.
.
RFID_Read = True
if Access:
    RFID_Closed = not RFID_Closed
.
.
if Access and not RFID_Closed:
    print(Node_Name + " : " + time.strftime("%d-%m-%Y %H:%M:%S ") + "
: " + "checkMFRC522" + " : " + "User: " + RFID_Card_User + " with ID:
" + RFID_Card_No)
    logger.info(Node_Name + " : " + time.strftime("%d-%m-%Y %H:%M:%S
") + " : " + "checkMFRC522" + " : " + "User: " + RFID_Card_User + "
with ID: " + RFID_Card_No)
    print(Node_Name + " : " + time.strftime("%d-%m-%Y %H:%M:%S ") + "
: " + "checkMFRC522" + " : " + "Lock #1 open for 5 seconds!")
    logger.info(Node_Name + " : " + time.strftime("%d-%m-%Y %H:%M:%S
") + " : " + "checkMFRC522" + " : " + "Lock #1 open for 5 seconds!")
    GPIO.output(4,GPIO.HIGH)
.
.
```

4.1.3 Sending data

The Transmit script has two methods for sending data that work in similar fashion. For example, for sending JSON data it searches for files in a specific path, calls the web service of the site to send the data and if it's successful, moves the JSON files in a different folder

```
JSONDirectoryNew = "/home/pi/NodePrograms/JSON_NEW_FILES/"
if not os. path. Exists (JSONDirectoryNew):
    os. makedirs (JSONDirectoryNew)

JSONDirectoryOld = "/home/pi/NodePrograms/JSON_OLD_FILES/"
if not os. path. exists (JSONDirectoryOld):
    os. makedirs (JSONDirectoryOld)

for r, d, f in os. walk (JSONDirectoryNew):
    for files in f:
        j_file_names. append (os. path. join (r, files))

for f in j_file_names:
    with open (f, "r") as filescontent:
        temp = filescontent. readlines ()
        .
        .
        .
        if service_success:
                shutil. move (f, JSONDirectoryOld)
```

4.1.4 Displaying data

Initially it was thought that this node would use a PIR sensor to detect movement and then display information on the monitor. While implementing this seemed a bad design for two reasons. Initially the movement detection didn't offer any additional value because by the time the sensor detected movement, the information had been displayed in the monitor and the person watching it didn't distinguish any difference. The second reason had to do with the way the 2-way mirror works. Since it needs to be darker in some parts in order to behave like a mirror, somehow the brightness of the screen needs to be adjusted. So, the solution for that would be using a graphic environment like Java.

The operation part of the program is pretty simple, it's essentially an SQL query in the database to retrieve information about the nodes and display it in specific part of the screen. For example

```
Connection conn = DriverManager.getConnection(WebURL, "root", "");
String query = "SELECT * FROM Nodes";
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery(query);
while (rs.next())
    {
        node_id = rs.getInt("node_id");
        node_active = rs.getBoolean("online");
        node_data = rs.getString("node_data");
    }
    st.close();
```

5 Conclusions

In this final chapter, there will be a summary on the Internet of Things microcosm, the specific platform chosen in the Raspberry Pi and the outcome of this project, along with problems tha emerged while trying to complete it.

5.1 Internet of things

It is clear by now, I think, that IoT is a very consequential concept in this technologic era. While maybe not in the way, pioneers thought of it a few decades ago, but adjusted in the reality today.

Because of the rise of complementary concepts and technologies, like AI and Blockchain, that help advance IoT, but also stand on their own, IoT isn't just about every device or sub-device of a machine connected to the Internet. The future of IoT is about combining technologies to increase the value of each technology in an array of applications, either in everyday life consumer devices, industrial uses to increase productivity or just for monitoring academic purposes.

The downside of relying to other technologies to provide added value, infers that IoT is bound to the standards and problems of these technologies. Moreover, it suffers from a lack of standardization. That and platform fragmentation are the main reasons, that hammer the efforts for further development.

Ultimately one of the biggest problems that IoT faces, as many technologies in this era, is security and privacy concerns. It seems counterintuitive that in the golden era of social media and smart devices in people's homes and cars, security and privacy has become the hampering effect to advancement, or as other people see it, including this author, the aftermath of what happens when security isn't part of the design process. As any issue in software engineering, security is better addressed in the early stages of development, rather than dealt with after release patches.

Thus, it stands to reason that ineffective design and the fact that every new technology is bound to the Internet, that Cybersecurity is one of the top emerging technologies and rightly so. With the emergence of revelations [206] [207] by Edward Snowden and WikiLeaks about surveillance, the Cambridge Analytica scandal [208] with Facebook and others, there have been people that are vocally resistant to technologies that they regard to be a threat to their privacy. Also, governments show through regulations [209], like GDPR[90] in Europe that they take the threat of security & privacy breaches very seriously.

[90] General Data Protection Regulation

5.2 Raspberry Pi sensor network

When one of the founders of Raspberry Pi, Eben Upton, launched it seven years ago, he didn't expect, as stated in interviews [210], that it would turn out this way. In March 2019, the company shipped its 25th million unit.

I can personally attest, that the Raspberry Pi foundation, have achieved much of what it intended to do. It is an excellent platform for learning, in many areas, like electronics & programming and can be suitable for a great number of applications, albeit none of the non-crucial kind as it will be explained promptly.

5.2.1 Platform quality & sensor consistency

The Raspberry Pi foundation by trying to make the RPi an affordable product, has of course inadvertently reduced how and in which applications it can be used. So, while I would trust the platform for monitoring or for controlling non-essential functions, I would never presume to think it could be used for applications of the serious variety.

Another problem encountered while tackling with this project was sensors that were incompatible with the Raspberry Pi or required additional units, like an analog to digital converter chip, in order to interface them with the RPi. Furthermore, most sensors were as accurate as they could be in their price range and time will tell on how long they will operate.

A sensor that has both of the problems mentioned above is the distance sensor of the second node. It suffers from both inconsistent readings thanks to interference for weather effects such as the sun and rain and the fact that it requires an ADC to work with the RPi.

5.2.2 Project problems and future development

By far the biggest problem with the project the number of parts required to construct the nodes, from specific spaces, cable glands, etc. to the actual acrylic 2-way mirror. Also, while somewhat adept with electronics and programming, this author's knowledge of woodworking hampered and delayed the construction of the housing enclosures for the nodes that required them. Which meant that while the prototyping and testing of most of the nodes had been completed with the submission of this thesis, the remaining nodes would be installed after.

One successful aspect of this project is that it can be evolved later on with the addition of other nodes, with different functions. For example, in the planning stages of this project, there was an idea for node that it would be interactive in nature, maybe some application with face-recognition properties or a social media node.

The most important aspect of Raspberry Pi and every other open source platform is that when it has support from the community, it can evolve exponentially.

Bibliography

[1] C. Ludeman, "The Rise of Helium," 20 July 2017. [Online]. Available: https://blog.westerndigital.com/rise-helium-drives/. [Accessed 20 October 2018].

[2] "Hard Disk Drive (HDD) vs Solid State Drive (SSD): What's the Diff?," Roderick Bauer, 20 September 2018. [Online]. Available: https://www.backblaze.com/blog/hdd-versus-ssd-whats-the-diff/. [Accessed 20 October 2018].

[3] N. A. Missa Schwartz, "The Alohanet - surfing for wireless data," *IEEE,* vol. 47, no. 12, pp. 21-25, 11 December 2009.

[4] FCC, "Authorization of Spread Spectrum Systems Under Parts 15 and 90 of the FCC Rules," 24 May 1985. [Online]. Available: http://www.ieee802.org/11/Documents/DocumentArchives/4L_docs/1987/4l_87_013_R_and_O_Docket_81-413_Spread_Spectrum.pdf. [Accessed 20 December 2018].

[5] B. Tuch, "Development of WaveLAN®, an ISM band wireless LAN," *AT&T Technical Journal,* vol. 72, no. 4, pp. 27-37, Jul-Aug 1993.

[6] "High-speed wireless networking," 2012. [Online]. Available: https://www.epo.org/learning-events/european-inventor/finalists/2012/osullivan.html. [Accessed 20 December 2018].

[7] S. Williams, "IrDA: Past, Present and Future," *IEEE Personal Communications*, vol. 7, no. 1, pp. 11-19, 2000.

[8] "Mobile phone evolution," 9 August 2010. [Online]. Available: https://www.gsmarena.com/mobile_phones_evolution_features-review-501p3.php. [Accessed 22 October 2018].

[9] "Bluetooth technology," 2012. [Online]. Available: https://www.epo.org/learning-events/european-inventor/finalists/2012/haartsen.html. [Accessed 21 December 2018].

[10] R. Nuwer, "Why is Bluetooth Called Bluetooth?," 27 August 2012. [Online]. Available: https://www.smithsonianmag.com/smart-news/why-is-bluetooth-called-bluetooth-hint-vikings-16270647/. [Accessed 21 December 2018].

[11] "Bluetooth: Why Modern Tech is Named After Powerful King of Denmark and Norway," 20 January 2017. [Online]. Available: https://www.ancient-origins.net/history-famous-people/bluetooth-why-modern-tech-named-after-powerful-king-denmark-and-norway-007398. [Accessed 21 December 2018].

[12] "Bluetooth history," [Online]. Available: https://www.bluetooth.com/about-us/our-history. [Accessed 21 December 2018].

[13] "Methods for communicating devices in a Personal Area Network," IEEE, [Online]. Available: https://standards.ieee.org/standard/802_15_1-2005.html. [Accessed 21 December 2018].

[14] F. P. Erina Ferro, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12-26, February 2005.

[15] "Bluetooth Core Specification 5.1," 21 January 2019. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification. [Accessed 1 February 2019].

[16] I. R. ,. A. H. K. N. Salman, "Overview of the IEEE 802.15.4 standards family for Low Rate Wireless Personal Area Networks," in *2010 7th International Symposium on Wireless Communication Systems*, York, UK, 2010.

[17] "Zigbee Alliance," [Online]. Available: https://www.zigbee.org/zigbee-for-developers/about-us/. [Accessed 20 December 2018].

[18] "Wi-Fi Alliance," [Online]. Available: https://www.wi-fi.org/who-we-are. [Accessed 20 December 2018].

[19] "Zigbee Products," [Online]. Available: https://www.zigbee.org/zigbee-products-2/. [Accessed 20 December 2018].

[20] J.-S. Lee, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," in *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, Taipei, Taiwan, 2007.

[21] C. Doctorow, "WiFi isn't short for "Wireless Fidelity"," 8 November 2005. [Online]. Available: https://boingboing.net/2005/11/08/wifi-isnt-short-for.html. [Accessed 10 January 2019].

[22] I. W. ,. J. K. ,. P. S. B.P. Crow, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116-126, September 1997.

[23] A. B. A. M. A. A. O. Ramia Babiker Mohammed Abdelrahman, "A Comparison between IEEE 802.11a, b, g, n and ac Standards," *IOSR Journal of Computer Engineering*, vol. 17, no. 5, pp. 26-29, September-October 2015.

[24] "Wi-Fi Alliance introduces Wi-Fi 6," 3 October 2018. [Online]. Available: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-wi-fi-6. [Accessed 15 January 2019].

[25] D. D. ,. L. S. Y. Z. X. P. C. B. W. Guido R. Hiertz, "The IEEE 802.11 universe," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 62-70, January 2010.

[26] D. Becker, "New WiFi Record: 237 Miles," 18 June 2007. [Online]. Available: https://www.wired.com/2007/06/w-wifi-record-2/. [Accessed 16 January 2019].

[27] B. Kamali, in *The IEEE 802.16 Standards and the WiMAX Technology*, Wiley-IEEE Standards Association, 2018, pp. 189-258.

[28] "OSI model," [Online]. Available: https://community.cisco.com/t5/security-documents/layer-2-vs-layer-3-addressing/ta-p/3123440. [Accessed 16 January 2019].

[29] IEEE, "IEEE 802.16.1a-2013," 6 March 2013. [Online]. Available: https://standards.ieee.org/standard/802_16_1a-2013.html. [Accessed 5 January 2019].

[30] N. Naik, "LPWAN Technologies for IoT Systems : Choice Between Ultra Narrow Band and Spread Spectrum," in *2018 IEEE International Systems Engineering Symposium (ISSE)*, Rome, Italy, 2018.

[31] V. P. Alexandru Lavric, "Internet of Things and LoRa Low-Power Wide-Area Networks: A survey," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, Lasi, Romania, 2017.

[32] R. K. J. S. J. Sakshi Popli, "A Survey on Energy Efficient Narrowband Internet of Things (NBIoT): Architecture, Application and Challenges," *IEEE Access*, vol. 7, pp. 16739 - 16776, November 2018.

[33] C. Brooks, "Four Emerging Technology Areas That Will Help Define Our World In 2019," 24 December 2018. [Online]. Available: https://www.forbes.com/sites/cognitiveworld/2018/12/24/four-emerging-technology-areas-that-will-help-define-our-world-in-2019/#4051f2ca58dd. [Accessed 25 January 2019].

[34] Gartner, "Gartner Identifies the Top 10 Strategic Technology Trends for 2019," 15 October 2018. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-10-15-gartner-identifies-the-top-10-strategic-technology-trends-for-2019. [Accessed 25 January 2019].

[35] M. Weiser, "The Computer for the 21st Century," September 1991. [Online]. Available: https://www.lri.fr/~mbl/Stanford/CS477/papers/Weiser-SciAm.pdf. [Accessed 25 January 2019].

[36] J. Teicher, "The little-known story of the first IoT device," 7 February 2018. [Online]. Available: https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/. [Accessed 25 January 2019].

[37] K. Ashton, "That 'Internet of Things' Thing," 22 June 2009. [Online]. Available: https://www.rfidjournal.com/articles/view?4986. [Accessed 25 February 2019].

[38] Deloitte, "Continuous interconnected supply chain," 2017. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/technology/lu-blockchain-internet-things-supply-chain-traceability.pdf. [Accessed 25 January 2019].

[39] M. L. a. t. F. o. IoT, "James Halladay," 30 July 2018. [Online]. Available: https://medium.com/mybit-dapp/moores-law-and-the-future-of-iot-d9ed7d725f0a. [Accessed 25 January 2019].

[40] B. Chen, "How to Make Your House a Smart Home," [Online]. Available: https://www.nytimes.com/guides/technology/how-to-make-a-smart-home. [Accessed 25 January 2019].

[41] G. Krisztyián, "Smart agriculture? A huge step towards a truly connected world," 9 November 2018. [Online]. Available: https://www.nokia.com/blog/smart-agriculture-huge-step-towards-truly-connected-world/. [Accessed 25 January 2019].

[42] B. Buntz, "The Top 20 Industrial IoT Applications," 20 September 2017. [Online]. Available: https://www.iotworldtoday.com/2017/09/20/top-20-industrial-iot-applications/. [Accessed 25 January 2019].

[43] W. I. D. T. T. -. A. W. I. I. S. Important, "Bernard Marr," 6 March 2017. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2017/03/06/what-is-digital-twin-technology-and-why-is-it-so-important/#293e0b052e2a. [Accessed 25 January 2019].

[44] N. Carr, "Author Nicholas Carr: The Web Shatters Focus, Rewires Brains," 24 May 2010. [Online]. Available: https://www.wired.com/2010/05/ff-nicholas-carr/. [Accessed 25 January 2019].

[45] E. N. Y. C. Z. C. Arbia Riahi Sfarab, "A roadmap for security challenges in the Internet of Things," vol. 4, no. 2, pp. 118-137, April 2018.

[46] M. Khalili, "IoT, Cryptography, and Privacy," 10 January 2018. [Online]. Available: https://medium.com/beam-mw/iot-cryptography-privacy-210cd9361d41. [Accessed 25 January 2019].

[47] N. Baird, "If Consumer Privacy Isn't Already Dead, IoT Could Kill It," 31 August 2017. [Online]. Available: https://www.forbes.com/sites/nikkibaird/2017/08/31/if-consumer-privacy-isnt-already-dead-iot-could-kill-it/#479b60302cf8. [Accessed 25 January 2019].

[48] "Good Practices for Security of Internet of Things in the context of Smart Manufacturing," 19 November 2018. [Online]. Available: https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot. [Accessed 25 January 2019].

[49] D. Guinard, "The IoT needs a defrag," 12 July 2016. [Online]. Available: https://www.oreilly.com/ideas/the-iot-needs-a-defrag. [Accessed 25 January 2019].

[50] N. Dahad, "Fragmentation, Security Remain Concerns for IoT," 9 November 2018. [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1333751. [Accessed 25 January 2019].

[51] M. A. M. &. G. Noura, "Interoperability in Internet of Things: Taxonomies and Open Challenges," *The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing,* 18 July 2018.

[52] R. L. Michael Blackstock, "IoT interoperability: A hub-based approach," in *2014 International Conference on the Internet of Things (IOT)*, Cambridge, MA, USA, 2014.

[53] S. Symanovich, "The future of IoT: 10 predictions about the Internet of Things," [Online]. Available: https://us.norton.com/internetsecurity-iot-5-predictions-for-the-future-of-iot.html. [Accessed 25 January 2019].

[54] Cisco, "Internet of Things," 2016. [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf. [Accessed 25 January 2019].

[55] M. H. P. v. d. V. M. J. H. J. N. T. N. E. L. Michael Johnson, "A Comparative Review of Wireless Sensor Network Mote Technologies," in *SENSORS, 2009 IEEE*, Christchurch, New Zealand, 2009.

[56] E. C. Whitman, "Sosus," 2005. [Online]. Available: https://www.public.navy.mil/subfor/underseawarfaremagazine/Issues/Archives/issue_25/sosus.htm. [Accessed 28 January 2019].

[57] S. K. Chee-Yee Chong, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE,* vol. 91, no. 8, pp. 1247-1256, 11 August 2003.

[58] W. J. K. G. J. Pottie, "Wireless Integrated Network Sensors," *Communications of the ACM,* vol. 43, no. 5, pp. 51-58, May 2000.

[59] S. F. Daniel Mandl, "Experimenting with an Evolving Ground/Space- based Software Architecture to Enable Sensor Webs," 2005. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20050210130.pdf. [Accessed 29 January 2019].

[60] S. S. G. P. R. S. Karen Moe, "Sensor Web Technologies for NASA Earth Science," in *2008 IEEE Aerospace Conference*, Big Sky, MT, USA, 2008.

[61] F. M. K. Romer, "The design space of wireless sensor networks," *IEEE Wireless Communications,* vol. 11, no. 6, pp. 54-61, December 2004.

[62] W. L. Z. Y. S. L. X. G. Junguo Zhang, "Forest Fire Detection System based on Wireless Sensor Network," in *2009 4th IEEE Conference on Industrial Electronics and Applications*, Xi'an, China, 2009.

[63] P. H. Leon Evers, "Supply Chain Management Automation using Wireless Sensor Networks," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, Pisa, Italy, 2007.

[64] S. Basu, "The Benefits of IoT and Blockchain for the Supply Chain," January 2018. [Online]. Available: https://blogs.oracle.com/profit/the-benefits-of-iot-and-blockchain-for-the-supply-chain.

[65] Z. W. Y. S. Xingfa Shen, "Wireless sensor networks for industrial applications," in *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*, Hangzhou, China, China, 2004.

[66] D. J. M. S. R. Jino Ramson, "Applications of wireless sensor networks — A survey," in *2017 International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICEEIMT)*, Coimbatore, India, 2017.

[67] Z. T. G. D. ,. V. M. Milica Pejanović Đurišić, "A survey of military applications of wireless sensor networks," in *2012 Mediterranean Conference on Embedded Computing (MECO)*, Bar, Montenegro, 2012.

[68] "Cosmote (NB-IoT)," 2 November 2016. [Online]. Available: https://www.cosmote.gr/fixed/en/corporate/details/-/asset_publisher/gLfNzjIgW7PO/content/%CE%BF-%CE%BA%CE%BF%CF%83%CE%BC%CE%BF%CF%82-%CF%84%CE%BF%CF%85-narrow-band-internet-of-things-%CE%B1%CF%80%CE%BF-%CF%84%CE%B7%CE%BD-cosmote#. [Accessed 15 September 2018].

[69] "Cosmote (Smart Campus)," 2 March 2018. [Online]. Available: https://www.cosmote.gr/fixed/en/corporate/details/-/asset_publisher/gLfNzjIgW7PO/content/cosmote-%CF%83%CF%84%CE%B7%CE%BD-%CE%BE%CE%B1%CE%BD%CE%B8%CE%B7-%CE%B7-%CF%80%CF%81%CF%89%CF%84%CE%B7-%C2%AB%CE%B5%CE%BE%CF%85%CF%80%CE%BD%CE%B7%C2%BB-%CF%80%CE%B1%CE. [Accessed 15 September 2018].

[70] "Cambridge (RasPi Foundation)," 2012. [Online]. Available: https://www.cl.cam.ac.uk/projects/raspberrypi/. [Accessed 15 September 2018].

[71] "Raspberry welcomes Lance," 25 September 2013. [Online]. Available: https://www.raspberrypi.org/blog/welcome-lance/. [Accessed 15 September 2018].

[72] "Raspberry welcomes Philip," 29 April 2015. [Online]. Available: https://www.raspberrypi.org/blog/welcome-philip/. [Accessed 15 September 2018].

[73] M. Peplow, "IEEE Spectrum," 28 February 2015. [Online]. Available: https://spectrum.ieee.org/geek-life/profiles/eben-upton-the-raspberry-pi-pioneer. [Accessed 16 September 2018].

[74] M. T. LittleWood, "BusinessofSoftware," 26 November 2015. [Online]. Available: https://businessofsoftware.org/2015/11/the-story-of-raspberry-pi-raspberry-pi-zero-jack-lang-business-of-software-europe-2015-talk/. [Accessed 16 September 2018].

[75] "Cambridge Homepage of Alan Mycroft," [Online]. Available: https://www.cl.cam.ac.uk/~am21/. [Accessed 16 September 2018].

[76] R. Cellan-Jones, "BBC," 5 May 2011. [Online]. Available: http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html. [Accessed 16 September 2018].

[77] "2014 Foundation Review," 2014. [Online]. Available: https://static.raspberrypi.org/files/about/RaspberryPiFoundationReview2014.pdf. [Accessed 15 September 2018].

[78] "Raspberry (Logo)," 7 October 2011. [Online]. Available: https://www.raspberrypi.org/blog/logo-competition-we-have-a-winner/. [Accessed 15 September 2018].

[79] "Raspberry Pi - Alpha boards," 12 August 2011. [Online]. Available: https://www.raspberrypi.org/blog/the-alpha-boards-are-here/. [Accessed 20 September 2018].

[80] "Raspberry Pi - Beta board," 11 December 2011. [Online]. Available: https://www.raspberrypi.org/blog/we-have-pcbs/. [Accessed 20 September 2018].

[81] "Raspberry manufacture started," 10 January 2012. [Online]. Available: https://www.raspberrypi.org/blog/weve-started-manufacture/. [Accessed 21 September 2018].

[82] "Raspberry manufacturing in UK," 6 September 2012. [Online]. Available: https://www.raspberrypi.org/blog/made-in-the-uk/. [Accessed 19 September 2018].

[83] "Raspberry quartz crystal package," 6 February 2012. [Online]. Available: https://www.raspberrypi.org/blog/two-things-you-thought-you-werent-going-to-get/. [Accessed 21 September 2018].

[84] "Raspberry ethernet connector problem," 8 March 2012. [Online]. Available: https://www.raspberrypi.org/blog/manufacturing-hiccup/. [Accessed 21 September 2018].

[85] "Raspberry deliveries started," 14 April 2012. [Online]. Available: https://www.raspberrypi.org/blog/deliveries-have-started/. [Accessed 16 September 2018].

[86] "Raspberry Pi 26-pin GPIO," 28 November 2011. [Online]. Available: https://www.raspberrypi.org/blog/pinout-for-gpio-connectors/. [Accessed 20 September 2018].

[87] "Adafruit Raspbery B+ analysis," 14 July 2014. [Online]. Available: https://learn.adafruit.com/introducing-the-raspberry-pi-model-b-plus-plus-differences-vs-model-b/gpio-port. [Accessed 16 September 2018].

[88] "Raspberry Pi Model B+," 14 July 2014. [Online]. Available: https://www.raspberrypi.org/blog/introducing-raspberry-pi-model-b-plus/. [Accessed 16 September 2018].

[89] "Raspberry Pi 3 Model B+ power supply," 11 April 2018. [Online]. Available: https://www.raspberrypi.org/blog/pi-power-supply-chip/. [Accessed 19 September 2018].

[90] "Raspberry Pi HAT," 31 July 2014. [Online]. Available: https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/. [Accessed 20 September 2018].

[91] "Raspberry Pi HAT attached," 24 August 2018. [Online]. Available: https://www.raspberrypi.org/blog/introducing-power-over-ethernet-poe-hat/. [Accessed 20 September 2018].

[92] "Raspberry Pi Model A," 4 February 2013. [Online]. Available: https://www.raspberrypi.org/blog/model-a-now-for-sale-in-europe-buy-one-today/. [Accessed 20 September 2018].

[93] "Raspberry Pi power draw," [Online]. Available: https://www.raspberrypi.org/documentation/faqs/#pi-power. [Accessed 15 September 2018].

[94] "Raspberry Pi Model A+," 10 November 2014. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-model-a-plus-on-sale/. [Accessed 20 September 2018].

[95] "Raspberry Pi 3 Model A+," 15 November 2018. [Online]. Available: https://www.raspberrypi.org/blog/new-product-raspberry-pi-3-model-a/. [Accessed 10 January 2018].

[96] "Raspberry Pi 3 Model A+ products page," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/. [Accessed 10 January 2019].

[97] "Raspberry Pi Zero," 26 November 2015. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-zero/. [Accessed 22 September 2018].

[98] "Raspberry Pi Zero W," 28 February 2017. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-zero-w-joins-family/. [Accessed 22 September 2018].

[99] "Raspberry Pi Zero picture," 27 November 2015. [Online]. Available: https://www.raspberrypi.org/blog/did-you-get-a-raspberry-pi-zero/. [Accessed 22 September 2018].

[100] "Raspberry Pi compute module," 7 April 2014. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-compute-module-new-product/. [Accessed 21 September 2018].

[101] "Raspberry Pi Compute module 3," 16 January 2017. [Online]. Available: https://www.raspberrypi.org/blog/compute-module-3-launch/. [Accessed 22 September 2018].

[102] «Project Fin,» 20 March 2018. [Ηλεκτρονικό]. Available: https://resin.io/blog/introducing-project-fin-a-board-for-fleet-owners/. [Πρόσβαση 22 September 2018].

[103] "Revolution Pi PLC," 17 January 2017. [Online]. Available: https://www.linkedin.com/pulse/industrial-iot-revolution-raspberry-pi-compute-module-maarten-ectors/?trk=prof-post. [Accessed 22 September 2018].

[104] "Stereo depth perception," 21 October 2014. [Online]. Available: http://www.argondesign.com/case-studies/2014/oct/21/stereo-depth-perception-raspberry-pi/. [Accessed 22 September 2018].

[105] "Raspberry Pi Camera," 18 May 2012. [Online]. Available: https://www.raspberrypi.org/blog/camera-module-first-pictures/. [Accessed 22 September 2018].

[106] "NEC compute module," 10 October 2016. [Online]. Available: https://www.nec-display-solutions.com/p/hq/en/news/dp/Products/Shared/News/2016/PressReleases/Company/RaspberryPi/RaspberryPi.xhtml. [Accessed 22 September 2018].

[107] "Raspberry Pi 3 Model B+," 14 March 2018. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-3-model-bplus-sale-now-35/. [Accessed 20 September 2018].

[108] "Qualcomm announcement for Always-Connected PC," 5 December 2017. [Online]. Available: https://www.qualcomm.com/news/releases/2017/12/05/qualcomm-launches-technology-innovation-advancements-always-connected-pc. [Accessed 22 September 2018].

[109] T. Warren, "Windows RT is dead," 3 February 2015. [Online]. Available: https://www.theverge.com/2015/2/3/7974759/windows-rt-is-dead. [Accessed 22 September 2018].

[110] T. Warren, "Windows 10 on ARM limitations," 19 February 2018. [Online]. Available: https://www.theverge.com/2018/2/19/17027026/microsoft-windows-10-on-arm-apps-games-limitations-support. [Accessed 21 September 2018].

[111] "x86 emulation on ARM chipsets," 15 February 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/uwp/porting/apps-on-arm-x86-emulation. [Accessed 21 September 2018].

[112] J. B. Su, "ARM history of failures with Windows," 22 August 2018. [Online]. Available: https://www.forbes.com/sites/jeanbaptiste/2018/08/22/how-arm-just-ruined-the-launch-of-qualcomms-windows-10-pcs/. [Accessed 22 September 2018].

[113] "Arch Linux for Raspberry," 4 March 2012. [Online]. Available: https://www.raspberrypi.org/blog/arch-linux-arm-available-for-download/. [Accessed 20 September 2018].

[114] "FreeBSD for Raspberry Pi," 21 January 2013. [Online]. Available: https://www.raspberrypi.org/blog/freebsd-is-here/. [Accessed 20 September 2018].

[115] "NetBSD for Raspberry Pi," 27 January 2013. [Online]. Available: https://www.raspberrypi.org/blog/netbsd-is-here/. [Accessed 20 September 2018].

[116] "Raspbian," [Online]. Available: https://www.raspbian.org/RaspbianAbout. [Accessed 22 September 2018].

[117] "Application Binary Interface," [Online]. Available: https://www.raspbian.org/RaspbianFAQ#What_do_you_mean_by_.22soft_float_ABI.22_and_.22hard_float_ABI.22.3F. [Accessed 22 September 2018].

[118] "Raspberry Pi operating systems & distributions," [Online]. Available: https://www.raspberrypi.org/downloads/. [Accessed 15 September 2018].

[119] "Windows 10 for IoT," 30 April 2015. [Online]. Available: https://www.raspberrypi.org/blog/windows-10-for-iot/. [Accessed 20 September 2018].

[120] "Python on Raspberry Pi," 9 May 2013. [Online]. Available: https://www.raspberrypi.org/blog/pypy-on-pi/. [Accessed 22 September 2018].

[121] B. Venners, "A Conversation with Guido van Rossum," 13 January 2003. [Online]. Available: https://www.artima.com/intv/pythonP.html. [Accessed 22 September 2018].

[122] "Stackoverflow survey 2018 - Loved, Dreaded, Wanted," [Online]. Available: https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted. [Accessed 22 September 2018].

[123] "Applications for Python," [Online]. Available: https://www.python.org/about/apps/. [Accessed 22 September 2018].

[124] "Java on Raspberry Pi," 26 September 2013. [Online]. Available: https://www.raspberrypi.org/blog/oracle-java-on-raspberry-pi/. [Accessed 22 September 2018].

[125] "Scratch 2.0 for Raspberry Pi," 27 June 2017. [Online]. Available: https://www.raspberrypi.org/blog/scratch-2-raspberry-pi/. [Accessed 22 September 2018].

[126] "Raspberry Pi GPIO," [Online]. Available: https://www.raspberrypi.org/documentation/usage/gpio/. [Accessed 22 September 2018].

[127] "Raspberry Pi GPIO by microsoft," 28 August 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi. [Accessed 22 September 2018].

[128] "Pulse Width Modulation (PWM) signal," National Instruments, 2 May 2018. [Online]. Available: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019OkFSAU&l=en-GR. [Accessed 22 September 2018].

[129] "Pulse Width Modulation," Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/pulse-width-modulation/all. [Accessed 22 September 2018].

[130] Standford, "Basics of UART Communication," [Online]. Available: https://web.stanford.edu/class/cs140e/notes/lec4/uart-basics.pdf. [Accessed 22 September 2018].

[131] "Raspberry Pi SPI," [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md. [Accessed 22 September 2018].

[132] "Serial Peripheral Interface (SPI)," Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all. [Accessed 22 September 2018].

[133] J. B. Jonathan Valdez, "Understanding the I2C Bus," 2015.

[134] "I2C," Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/i2c. [Accessed 22 September 2018].

[135] M. Varian, "VM and the VM community: Past, Present and Future," August 1997. [Online]. Available: http://www.leeandmelindavarian.com/Melinda/25paper.pdf. [Accessed 23 October 2018].

[136] R. J. Creasy, "The Origin of the VM/370 Time-Sharing System," *IBM Journal of Research and Development,* vol. 25, no. 5, pp. 483-490, September 1981.

[137] S. D. M. R. J. S. E. Y. W. E. Bugnion, "Bringing Virtualization to the x86 Architecture with the Original VMware Workstation," *ACM Transactions on Computer Systems (TOCS),* vol. 30, no. 4, November 2012.

[138] "Usage of operating systems for websites," W3Techs, [Online]. Available: https://w3techs.com/technologies/overview/operating_system/all. [Accessed 20 November 2018].

[139] "Debian is the rock on which Ubuntu is built," [Online]. Available: https://www.ubuntu.com/community/debian. [Accessed 25 October 2018].

[140] "Usage statistics and market share of Linux for websites," W3Techs, [Online]. Available: https://w3techs.com/technologies/details/os-linux/all/all. [Accessed 27 October 2018].

[141] S. B. Navathe, "Evolution of data modeling for databases," *Communications of ACM,* vol. 35, no. 9, pp. 112-123, September 1992.

[142] "Understanding the Hierarchical Database Model," [Online]. Available: https://mariadb.com/kb/en/library/understanding-the-hierarchical-database-model/. [Accessed 23 October 2018].

[143] D. D. C. Bradford W. Wade, "IBM Relational Database Systems: The Early Years," *IEEE Annals of the History of Computing,* vol. 34, no. 4, pp. 38-48, October-December 2012.

[144] T. J. Bergin and T. Haigh, "The Commercialization of Database Management Systems, 1969–1983," *IEEE Annals of the History of Computing,* vol. 31, no. 4, pp. 26-41, October-December 2009.

[145] "DB-Engines ranking," 15 January 2019. [Online]. Available: https://db-engines.com/en/ranking_trend. [Accessed 15 January 2019].

[146] A. Hurson, S. Pakzad and J.-B. Cheng, "Object-oriented database management systems: evolution and performance issues," *Computer,* vol. 26, no. 2, pp. 45-58, February 1993.

[147] J. M. ADAM LITH, "Investigating storage solutions for large data," G oteborg, Sweden, 2010.

[148] "NoSQL - A Relational Database Management System," [Online]. Available: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page. [Accessed 25 October 2018].

[149] "NOSQL meetup," 11 June 2009. [Online]. Available: https://www.eventbrite.com/e/nosql-meetup-tickets-341739151#. [Accessed 20 November 2018].

[150] J. Gray, "The transaction concept: virtues and limitations," in *Proceedings of Seventh International Conference on Very Large Databases*, Cannes, France, 1981.

[151] T. H. Andreas Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys,* vol. 15, no. 4, pp. 287-317, December 1983.

[152] E. Brewer, "Towards robust distributed systems," in *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, Portland, Oregon, 2000.

[153] N. L. Seth Gilbert, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News,* vol. 33, no. 2, pp. 51-59, June 2002.

[154] E. Brewer, "CAP twelve years later: How the "rules" have changed," *Computer,* vol. 45, no. 2, pp. 23-29, February 2012.

[155] "NoSQL Databases Explained," [Online]. Available: https://www.mongodb.com/nosql-explained. [Accessed 15 December 2018].

[156] L. P. Issac, "SQL vs NoSQL Database Differences Explained with few Example DB," 14 January 2014. [Online]. Available: https://www.thegeekstuff.com/2014/01/sql-vs-nosql-db./. [Accessed 15 December 2018].

[157] "N1QL Query," [Online]. Available: https://docs.couchbase.com/server/6.0/getting-started/try-a-query.html. [Accessed 15 December 2018].

[158] M. Aslett, "What we talk about when we talk about NewSQL," 6 April 2011. [Online]. Available: https://blogs.the451group.com/information_management/2011/04/06/what-we-talk-about-when-we-talk-about-newsql/. [Accessed 20 December 2018].

[159] A. J. Archana Raje, "Sql Vs NoSql :NewSql The Solution For Big Data," *Journal of Computer Engineering,* vol. 2, no. 8, pp. 45-51, 2017.

[160] M. A. Andrew Pavlo, "What's Really New with NewSQL?," *ACM SIGMOD Record*, vol. 45, no. 2, pp. 45-55, 2 June 2016.

[161] J. Kestelyn, "From NoSQL to new SQL: How Spanner became a global, mission-critical database," 1 June 2017. [Online]. Available: https://cloud.google.com/blog/products/gcp/from-nosql-to-new-sql-how-spanner-became-a-global-mission-critical-database. [Accessed 20 December 2018].

[162] "SAP HANA," [Online]. Available: https://www.sap.com/greece/products/hana/features/in-memory-database.html. [Accessed 20 December 2018].

[163] J. B. João Oliveira, "NewSQL Databases : MemSQL and VoltDB Experimental Evaluation," 2017.

[164] H. McCracken, "The Web at 25: Revisiting Tim Berners-Lee's Amazing Proposal," 12 March 2014. [Online]. Available: http://time.com/21039/tim-berners-lee-web-proposal-at-25/. [Accessed 11 January 2019].

[165] T. Berners-Lee, "Information Management: A Proposal," March 1989. [Online]. Available: https://www.w3.org/History/1989/proposal.html. [Accessed 11 January 2019].

[166] "Web 2.0 conference," [Online]. Available: https://web.archive.org/web/20080913174125/http://www.web2con.com/web2con/. [Accessed 11 January 2019].

[167] T. O'Reilly, "What Is Web 2.0," 30 09 2005. [Online]. Available: https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html. [Accessed 11 January 2019].

[168] D. DiNucci, "Fragmented Future," 1999. [Online]. Available: http://darcyd.com/fragmented_future.pdf. [Accessed 11 January 2019].

[169] I. Codesido, "What is front-end development?," 29 September 2009. [Online]. Available: https://www.theguardian.com/help/insideguardian/2009/sep/28/blogpost. [Accessed 11 January 2019].

[170] S. Machlis, "Q&A: PHP creator Rasmus Lerdorf," 04 February 2002. [Online]. Available: https://www.computerworld.com/article/2586472/q-a--php-creator-rasmus-lerdorf.html. [Accessed 11 January 2019].

[171] "PHP and Facebook," 3 May 2007. [Online]. Available: https://www.facebook.com/notes/facebook/php-and-facebook/2356432130/. [Accessed 11 January 2019].

[172] "ASP.NET overview," 3 December 2010. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/overview. [Accessed 11 January 2019].

[173] A. Austin, "An Overview of AngularJS for Managers," 14 August 2014. [Online]. Available: https://andrewaustin.com/an-overview-of-angularjs-for-managers/. [Accessed 11 January 2019].

[174] P. Hunt, "Why did we build React?," 5 June 2013. [Online]. Available: https://reactjs.org/blog/2013/06/05/why-react.html. [Accessed 11 January 2019].

[175] D. E. G. K. A. L. N. M. H. F. N. S. T. D. W. Don Box, "Simple Object Access Protocol (SOAP) 1.1," 8 May 2000. [Online]. Available: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. [Accessed 11 January 2019].

[176] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. [Accessed 11 January 2019].

[177] R. T. Fielding, "REST APIs must be hypertext-driven," 20 October 2008. [Online]. Available: http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven. [Accessed 11 January 2019].

[178] IBM, "LAMP stack explained," 14 September 2018. [Online]. Available: https://www.ibm.com/cloud/learn/lamp-stack-explained. [Accessed 11 January 2019].

[179] "Stackoverflow survey 2018 - Correlated Technologies," [Online]. Available: https://insights.stackoverflow.com/survey/2018/#correlated-technologies. [Accessed 11 January 2019].

[180] E. Park, K. Chae and C. Kang, "The structured prototyping life cycle model for systems development management," in *Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert System Programs*, Washington, DC, USA, 1991.

[181] "Stevenson screen," 11 July 2013. [Online]. Available: https://www.canada.ca/en/environment-climate-change/services/sky-watchers/weather-instruments-tour/stevenson-screen.html. [Accessed 15 January 2019].

[182] "A Preliminary Investigation of Temperature Screen Design and Their Impacts on Temperature Measurements," 9 June 1998. [Online]. Available: https://www.wmo.int/pages/prog/www/IMOP/WebPortal-AWS/Tests/ITR649.pdf. [Accessed 15 January 2019].

[183] "Sir Thomas Stevenson (1818 - 1887)," [Online]. Available: https://enschrage.nl/stev/StevensonT.html. [Accessed 15 January 2019].

[184] "Degrees of protection provided by enclosures (IP Code)," [Online]. Available: https://webstore.iec.ch/publication/2452. [Accessed 15 January 2019].

[185] "The two digits of IP ratings explained," [Online]. Available: https://www.gwp.co.uk/guides/ip-ratings-explained/. [Accessed 15 January 2019].

[186] "Raspberry Pi 3 Model B+ - Official page," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed 10 January 2019].

[187] "Adafruit Perma-Proto Full-sized Breadboard PCB - Single," [Online]. Available: https://www.adafruit.com/product/1606. [Accessed 10 January 2019].

[188] "Adafruit Perma-Proto Half-sized Breadboard PCB - Single," [Online]. Available: https://www.adafruit.com/product/1609. [Accessed 10 January 2019].

[189] "Adafruit BME680 - Temperature, Humidity, Pressure and Gas Sensor," [Online]. Available: https://www.adafruit.com/product/3660. [Accessed 10 January 2019].

[190] "Adafruit TSL2591 High Dynamic Range Digital Light Sensor," [Online]. Available: https://www.adafruit.com/product/1980. [Accessed 10 January 2019].

[191] "Anemometer Wind Speed Sensor w/Analog Voltage Output," [Online]. Available: https://www.adafruit.com/product/1733. [Accessed 10 January 2019].

[192] "RGB backlight positive LCD 16x2 + extras - black on RGB," [Online]. Available: https://www.adafruit.com/product/398. [Accessed 10 January 2019].

[193] "Lock-style Solenoid - 12VDC," [Online]. Available: https://www.adafruit.com/product/1512. [Accessed 10 January 2019].

[194] "MFRC-522 NFC/RFID Controller Breakout Board - 13.56MHz," [Online]. Available: https://grobotronics.com/mfrc-522-nfc-rfid-controller-breakout-board.html?sl=en. [Accessed 10 January 2019].

[195] "Magnetic contact switch (door sensor)," [Online]. Available: https://www.adafruit.com/product/375. [Accessed 10 January 2019].

[196] "Adafruit SGP30 Air Quality Sensor Breakout - VOC and eCO2," [Online]. Available: https://www.adafruit.com/product/3709. [Accessed 10 January 2019].

[197] "Adafruit Perma-Proto Quarter-sized Breadboard PCB - Single," [Online]. Available: https://www.adafruit.com/product/1608. [Accessed 10 January 2019].

[198] "Raspberry Pi Camera Board v2 - 8 Megapixels," [Online]. Available: https://www.adafruit.com/product/3099. [Accessed 10 January 2019].

[199] "IR Distance Sensor - Includes Cable (100cm-500cm) - GP2Y0A710K0F," [Online]. Available: https://www.adafruit.com/product/1568. [Accessed 10 January 2019].

[200] "Adafruit Perma-Proto HAT for Pi Mini Kit - No EEPROM," [Online]. Available: https://www.adafruit.com/product/2310. [Accessed 10 January 2019].

[201] "MCP3008 - 8-Channel 10-Bit ADC With SPI Interface," [Online]. Available: https://www.adafruit.com/product/856. [Accessed 10 January 2019].

[202] W. W. Royce, "Managing the development of large software systems," in *Proceedings of IEEE WESCON*, 1970.

[203] "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 12 January 2019].

[204] J. R. I. J. Grady Booch, The unified software development process, Boston, USA: Addison-Wesley Longman Publishing Co, 1999.

[205] M. Tyson, "What is the JVM? Introducing the Java Virtual Machine," 22 May 2018. [Online]. Available: https://www.javaworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html. [Accessed January 12 2019].

[206] H. L. P. H. S. S. H. Secrets, «Peter Maass,» 13 August 2013. [Ηλεκτρονικό]. Available: https://www.nytimes.com/2013/08/18/magazine/laura-poitras-snowden.html. [Πρόσβαση 5 February].

[207] «US intelligence analyst arrested over security leaks,» 7 June 2010. [Ηλεκτρονικό]. Available: https://www.bbc.com/news/10254072. [Πρόσβαση 5 February 2019].

[208] E. G.-H. Carole Cadwalladr, «Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach,» 17 March 2018. [Ηλεκτρονικό]. Available: https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election. [Πρόσβαση 5 February 2019].

[209] M. Burgess, «What is GDPR? The summary guide to GDPR compliance in the UK,» 21 January 2019. [Ηλεκτρονικό]. Available: https://www.wired.co.uk/article/what-is-gdpr-uk-eu-legislation-compliance-summary-fines-2018. [Πρόσβαση 5 February 2019].

[210] A. Piltch, «Raspberry Pi Founder Shares 10 Things You May Not Have Known,» 11 February 2019. [Ηλεκτρονικό]. Available: https://www.tomshardware.com/news/raspberry-pi-founder-interview,38585.html. [Πρόσβαση 20 February 2019].

[211] "TSL2591," 05 June 2018. [Online]. Available: http://ams.com/documents/20143/36005/TSL2591_DS000338_6-00.pdf. [Accessed 1 February 2019].

[212] "Anemometer," [Online]. Available: https://cdn-shop.adafruit.com/product-files/1733/C2192+datasheet.pdf. [Accessed 1 February 2019].

[213] "BME 680," July 2017. [Online]. Available: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME680-DS001.pdf. [Accessed 1 February 2019].

[214] "HD44780U," [Online]. Available: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf. [Accessed 1 February 2019].

[215] "MCP 3008," 2008. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf. [Accessed 1 February 2019].

[216] "MFRC 522," 27 April 2016. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. [Accessed 1 February 2019].

[217] "SGP 30," August 2017. [Online]. Available: https://www.mouser.com/pdfdocs/Sensirion_Gas_Sensors_SGP30_Datasheet_EN-1148053.pdf. [Accessed 1 February 2019].

[218] "Sharp Distance sensor," [Online]. Available: http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a710k_e.pdf. [Accessed 1 February 2019].

[219] "Ad hoc," [Online]. Available: https://dictionary.cambridge.org/dictionary/english/ad-hoc. [Accessed 20 December 2018].

[220] "Advanced Package Tool," [Online]. Available: https://help.ubuntu.com/lts/serverguide/apt.html.en. [Accessed 22 October 2012].

[221] V. J. G. A. A. V. Z. J. G. G. S. Richard Van nee, "The 802.11n MIMO-OFDM Standard for Wireless LAN and Beyond," *Wireless Personal Communications Journal*, vol. 37, no. 3-4, pp. 445-453, May 2006.

[222] S. Lohr, "The Origins of "Big Data: An etymological Detective Story"," 1 February 2013. [Online]. Available: https://bits.blogs.nytimes.com/2013/02/01/the-origins-of-big-data-an-etymological-detective-story/. [Accessed 20 November 2018].

[223] D. Hood, "The evolution of DB architectures in the quest for scalability," 24 October 2017. [Online]. Available: https://blogs.oracle.com/timesten/the-evolution-of-db-architectures. [Accessed 2 December 2018].

[224] K. Featherly, "ARPANET - United States defense program," [Online]. Available: https://www.britannica.com/topic/ARPANET. [Accessed 20 October 2018].

[225] "Yum Package Manager," [Online]. Available: http://yum.baseurl.org/. [Accessed 25 October 2018].

[226] "The Apache Software Foundation Announces Apache Cassandra Release 0.6," 13 April 2010. [Online]. Available: https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces3. [Accessed 22 September 2018].

[227] "State of MongoDB," 8 March 2010. [Online]. Available: https://www.mongodb.com/blog/post/state-of-mongodb-march-2010. [Accessed 22 September 2018].

[228] "Rust project," [Online]. Available: https://www.rust-lang.org/en-US/faq.html#project. [Accessed 22 September 2018].

[229] "MySQL's creator on why the future belongs to MariaDB," 28 March 2013. [Online]. Available: https://www.computerworld.com.au/article/457551/dead_database_walking_mysql_creator_why_future_belongs_mariadb/. [Accessed 22 September 2018].

[230] "Kotlin 1.0 Released: Pragmatic Language for JVM and Android," 15 February 2016. [Online]. Available: https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/. [Accessed 22 September 2018].

[231] "Introduction to Cassandra Query Language," [Online]. Available: https://docs.datastax.com/en/cql/3.3/cql/cqlIntro.html. [Accessed 15 December 2018].

[232] "International Telecommunication Union M.2134 report," 2008. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2134-2008-PDF-E.pdf. [Accessed 20 September 2018].

[233] "Go at Google: Language Design," 25 October 2012. [Online]. Available: https://talks.golang.org/2012/splash.article#TOC_1. [Accessed 22 September 2018].

Appendix

A – Software – Datasheets - Code

List of software used in preparing this thesis

1. **PyCharm IDE** – Python programming
2. **Thonny IDE** – Python programming
3. **NetBeans** – Java programming
4. **Notepad++** - General Editor
5. **dB Forge Studio** – Database management
6. **Visio 2016** – Diagrams design
7. **Fritzing** – Breadboard & schematic design program
8. **PuTTY** – Connecting – transferring files to Raspberry Pi & the virtual server
9. **RealVNC** - Connecting – transferring files to Raspberry Pi

Datasheets of sensors – devices used in the project

1. AMS TSL2591 [211]
2. Anemometer [212]
3. Bosch BME680 [213]
4. LCD HD4470U [214]
5. MCP3008 [215]
6. MFRC522 (RFID) [216]
7. Parallax PIR (motion) sensor
8. RGB backlight positive LCD 16x2 - black on RGB
9. Sensirion SGP30 [217]
10. Sharp - GP2Y0A710K0F (Distance Measuring Sensor Unit - 100 - 550 cm) [218]

Project code & additional data

Code for the project of this thesis is available on GitHub along with additional information

https://github.com/mentalgr/RPI-IHU-IoT-Project

B – Project Pictures

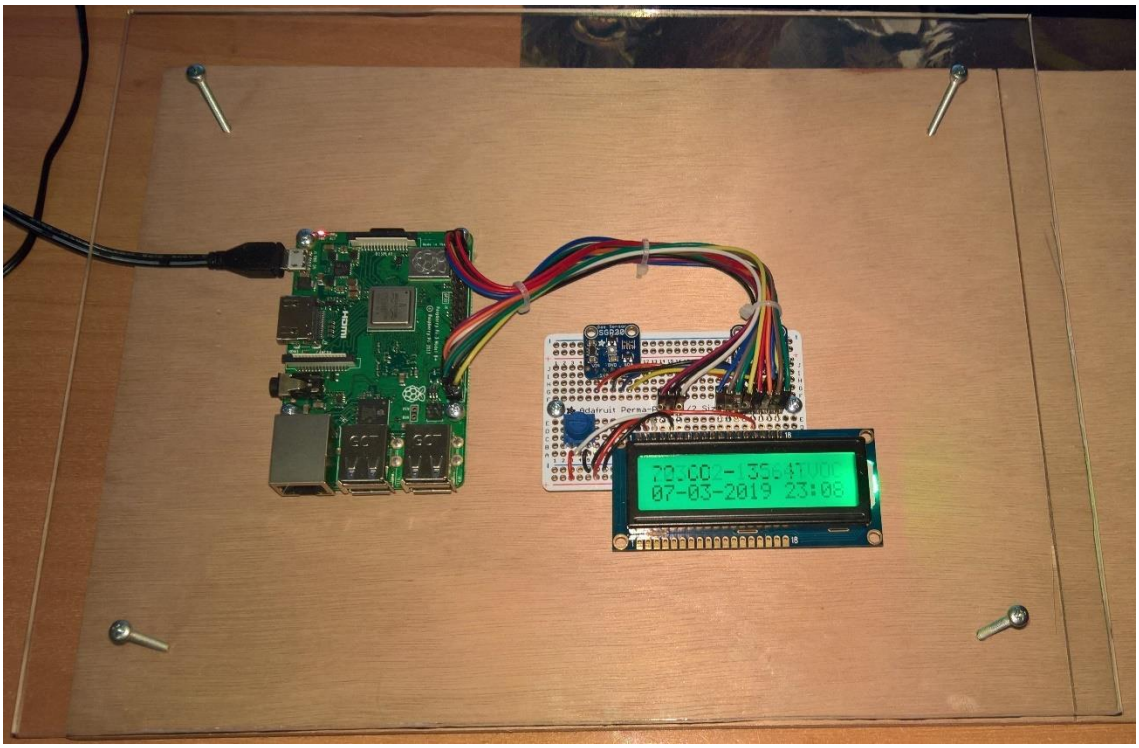


Figure 42 – Node #3 prototyping.

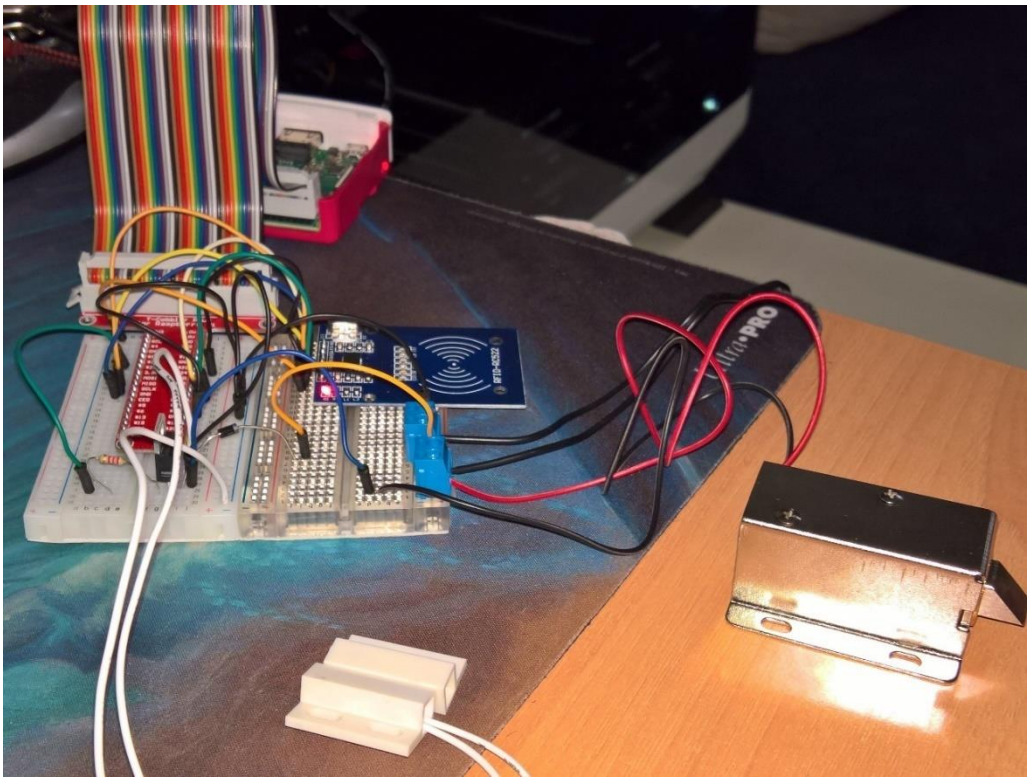


Figure 43 – Testing the MFRC522 – Lock-style solenoid function

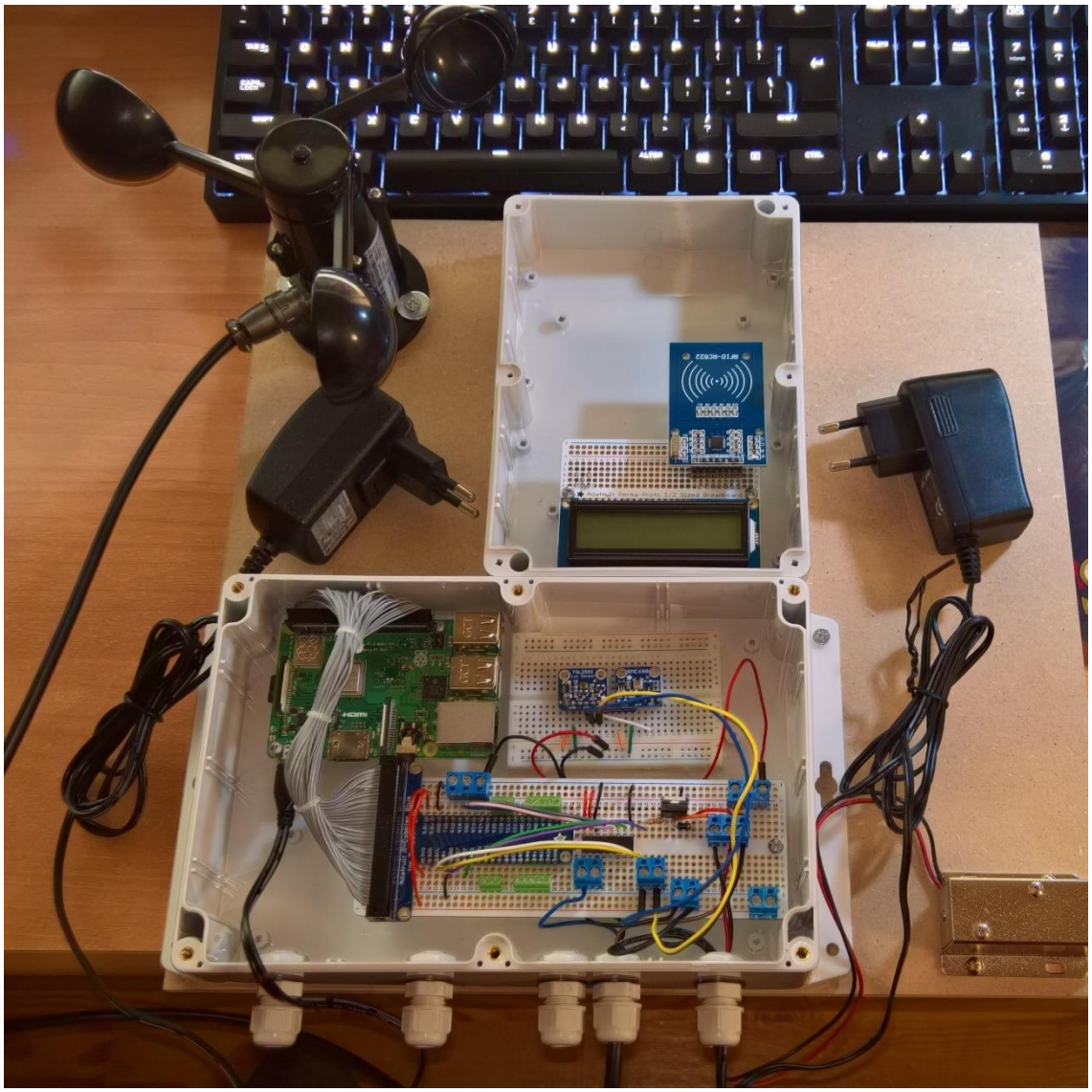


Figure 44 – Node #1 prototyping

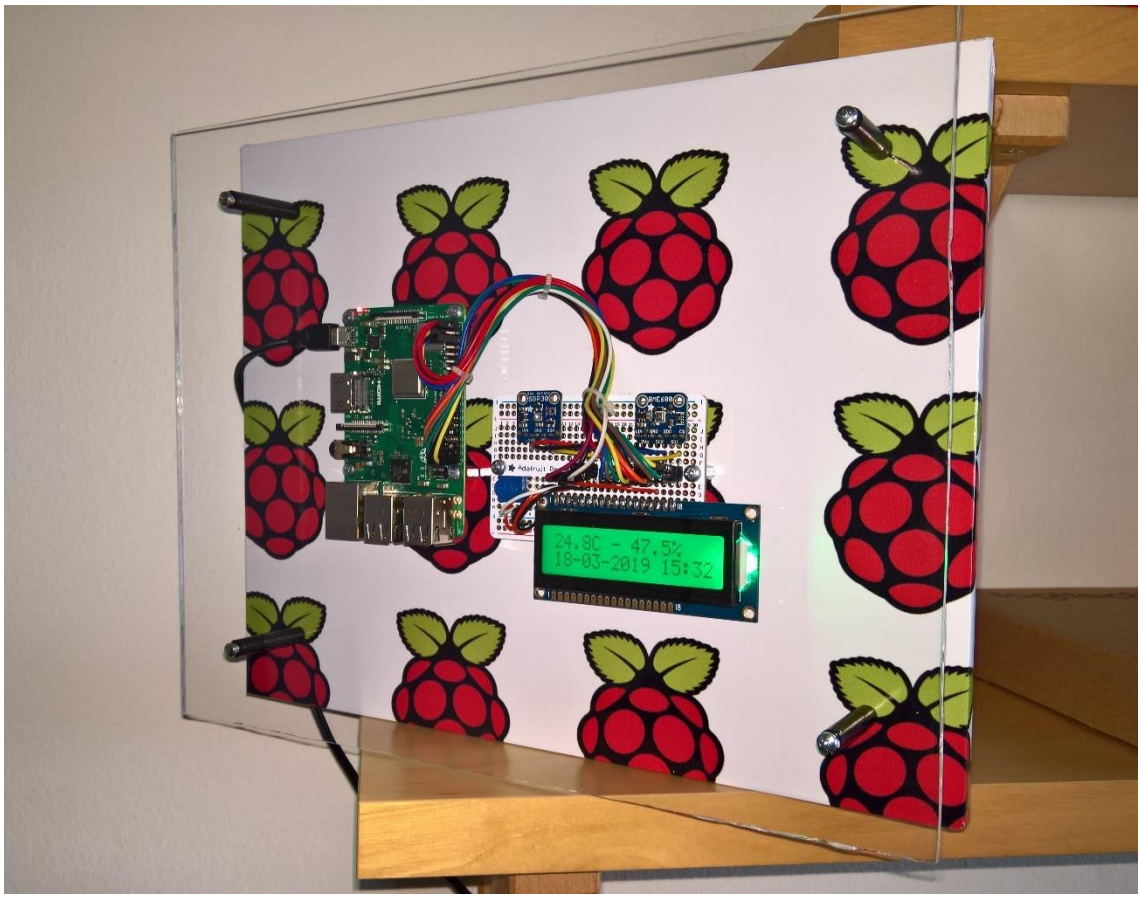


Figure 45 – Node #3 with finished housing

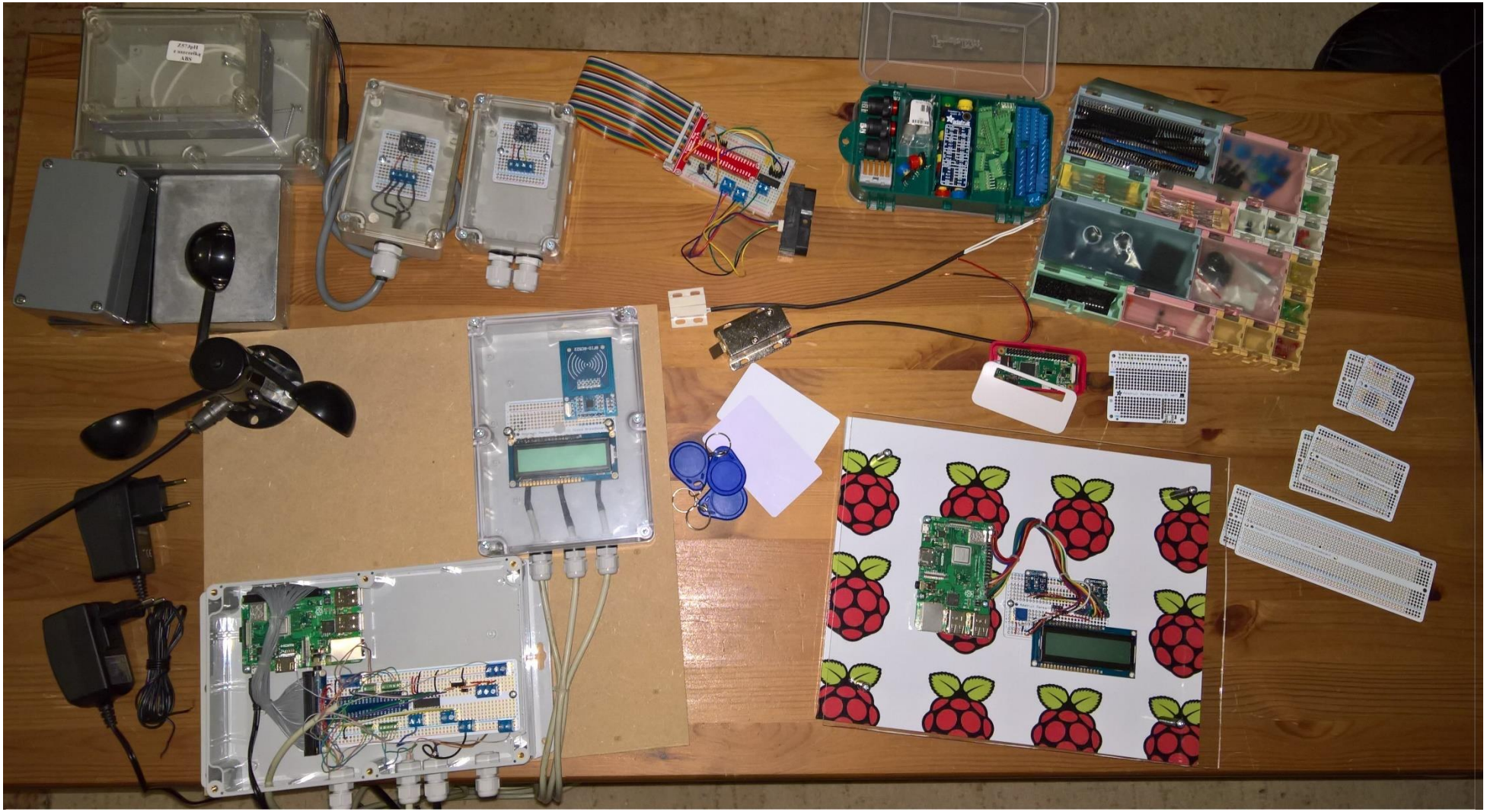


Figure 46 – Sensors & Parts

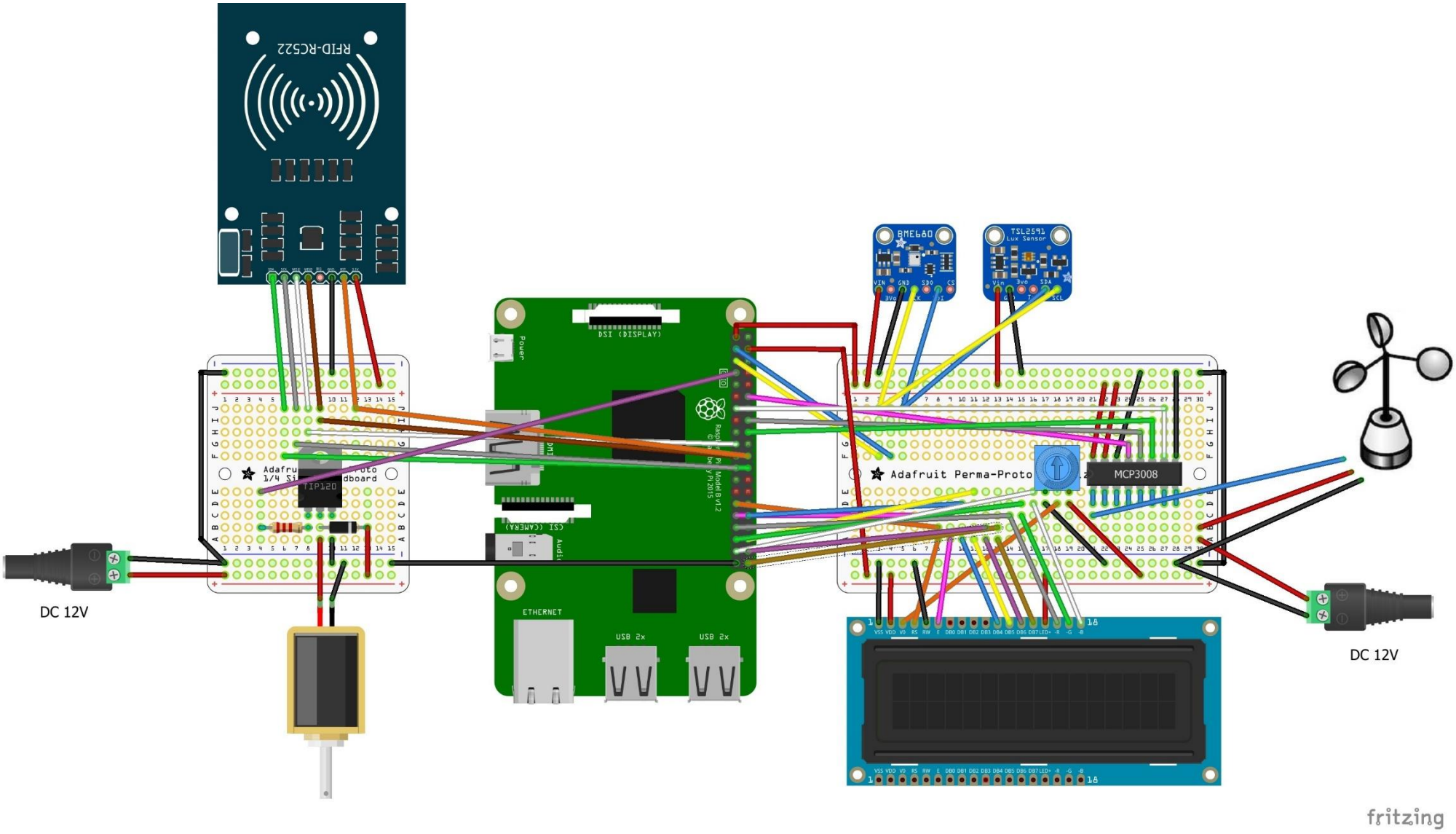


Figure 47 – All the sensors-devices and their connections of node #1

C – Specifications

Model		**Pi – Model B**	**Pi Zero v1.3**	**Compute Module 3+**	**Pi 3 – Model B+**	**Pi 3 – Model A+**
Release Date		April 2012	February 2017	January 2019	March 2018	November 2018
SOC Type		Broadcom BCM2835		Broadcom BCM2837B0		
Architecture		ARMv6Z (32-bit)		ARMv8-A (64/32-bit)		
CPU		700 MHz/1GHz Single Core ARM1176JZF-S		1.2GHz / 1.4GHz Quad Core Arm Cortex-A53		
RAM		512 MB / 1 GB				
GPU		Broadcom Video Core IV 1080p30 / p60				
Connectivity	**USB**	2	1 x Micro USB (OTG)	1	4	1
	Ethernet	-	-	-	10/100/1000Mbit/s	-
	Wi-Fi	-	802.11n	-	802.11ac Dual Band 2.4GHz & 5GHz	
	Bluetooth	-	2.0/4.1	-	2.0/4.1/4.2 LS BLE	
	Video	HDMI, RCA, DSI	Mini HDMI, Composite	HDMI, 2xDSI	HDMI, Composite, DSI	
	Audio	HDMI, 3.5mm Jack	Mini HDMI		HDMI, 3.5mm Composite	
	Camera	15 pin CSI (Camera Serial Interface)				
	GPIO	26	40	46	40	40
Memory		SD	microSD	On-Board eMMC 8/16/32 GB	microSD	
Size	**Mm / (g)**	85.6 x 56.5 / (45)	65 x 35 / (9)	67.6 x 31 / (7)	85.6 x 56.5 / (45)	65 x 56 / (29)

Table 9 – Raspberry Pi different model

This page intentionally left blank