Aalto University
School of Science
Master’s Programme in Computer Science and Engineering

Charles Wasswa Sewagudde

The design and implementation of a smart-parking system for Helsinki Area

Master’s Thesis

Espoo, June 3, 2016

Supervisor: Professor Kary Främling, Aalto University

Instructors: Momir Beljic, M.Sc. (Tech.) and Sampsa Siitonen

Aalto University
School of Science

Aalto University School of Science Degree Programme in Computer Science and Engineering Master's Programme in Service Design and Engineering	ABSTRACT OF THE MASTER'S THESIS
Author: Charles Wasswa Sewagudde	
Title: The design and implementation of a smart-parking system for Helsinki Area	

Number of pages: 63	Date: June 3, 2016	Language: English

Professorship: Systems of Systems	Code: CSE-E4675
Supervisor: Professor Kary Främling	
Instructors: Momir Beljic M.Sc. (Tech.), Sampsa Siitonen	

Abstract:
The strain on the parking infrastructure for the general public has significantly grown as a result of the ever rising number of vehicles geared by the rapid population growth in urban areas. Consequently, finding a vacant parking space has become quite a challenging task, especially at peak hours. Drivers have to cycle back and forth a number of times before they finally find where to park. This leads to increased fuel consumption, air pollution, increased likelihood of causing accidents, to mention but a few. Paying for the parking is not straight forward either, as the ticket machines, on top of being located at a distance, in many times, they have several payment methods drivers must prepare for. A system therefore, that would allow drivers to check for the vacant parking places before driving to a busy city, takes care of the parking fee for exact time they have used, manages electronic parking permit, is right direction towards to toppling these difficulties.

The main objective of this project was to design and implement a system that would provide parking occupancy estimation, parking fee payment method, parking permit management and parking analytics for the city authorities. The project had three phases. The first and the second phases used qualitative approaches to validate our hypotheses about parking shortcoming in Helsinki area and recruitment of participants to the pilot of the project, respectively. The third phase involved the design, implementation and installation of the system. The other objective was to study the challenges a smart parking system would face at different stages of its life cycle.

The objectives of the project were achieved and the assumption about the challenges associated with parking in a busy city were validated. A smart parking system will allow drivers to check for available parking spaces beforehand, they are able to pay for the parking fee, they can get electronic parking permits, and the city authority can get parking analytics for the city planning.

Keywords: Smart parking system, LoRa, LoRaWAN, Internet of Things, IoT, Reverse SSH tunneling, Machine learning

Acknowledgements

On this project I worked as both systems and electronics engineer. Under the systems engineer role, I worked on the design of the Smart Parking System architecture together with the CIO (Krister Holmstrom). With the Electronic Engineer hood on, I was responsible for all the hardware used on the project, right from schematic design up to soldering the components to the printed circuit board, testing and installation. Other duties included brain storming on technology selection and algorithms, site surveys, and system installation and maintenance.

The interview part of this work was done with help of my colleague Aftab Ansari during the course *Methods for Software Engineering and Business Research P*. The objective of the course was to introduce us to research methods, approaches and processes in software engineering and business both in academy and industry. The course also provided us with practice on formulating research questions and planning empirical studies for our master's theses.

I would like to thank my supervisor professor Kary Främling for his guidance and advise on many issues including the technical correctness of this thesis. I wish also to take this opportunity to thank my colleagues at Witrafi Oy without whose support and hard work, the project would not have been completed. First I thank my instructor Sampsa Siitonen, for allowing me to write my thesis on their project, and my instructors for guiding me whenever I went wrong. Secondly, i wish to thank the following teams for their great work towards this project: The backend team – Krister Holmstrom and Nelson Makubuya; the frontend team - Doan Minh Hieu; Industrial design team – Thomas Wahl; Data analysis/machine learning team-Momir Beljic; Management team - Sampsa Siitonen and the embedded software team Jonathan Malangoni and Toni Piira. Thank you guys, I learned a lot from you and please continue that good spirit.

Lastly, I wish to thank my family especially Rosette and my son Ethan for enduring my absence during the project and thesis writing time. And the family members and friends, forgive me for not returning your calls, I was packed up but now I am free again.

“*Try Not to Become a Man of Success, But Rather Become a Man of Value.*”

–Albert Einstein

Table of Contents

CHAPTER 1

1. INTRODUCTION

Due to the increasing number of vehicles caused by rapid population growth in urban areas, the demand on parking infrastructure for the general public has increased significantly (Polycarpou et al., 2013). As Polycarpou et al. (2013) further points out, finding a free parking space in urban cities especially during peak hours is more or less impossible, in many cases.

Looking for a vacant parking lot in a busy city like Helsinki, at a peak hour is a nightmare to drivers. They have to drive around looking for a free parking spot, something that is believed to increase traffic congestion (Chinrungrueng et al., 2007; Srikanth et al., 2009; Geng et al., 2012). The cycling around not only frustrates drivers but also increases the average consumption of gas and hence the air pollution (Chinrungrueng et al., 2007) that affect the environment. It is also believed that, as drivers' attention is partly on looking for a free spot in a busy city, the likelihood of causing an accident is higher. Another challenge is associated with paying for the parking, most of the time, the machines from which to get the payment ticket are located some distance from the car. The ticket machines too, present some complications since they are not the same; some machines take only coins, or coins plus credit/debit cards or only credit/debit cards. The situation looks like drivers have to prepare for the possibility of any of the above type of ticket machines.

Smart parking is a vehicle parking system that aid drivers to identify empty parking lots (Pcmag, 2014). The smart parking system also includes the means of calculating and paying for the time spent in the parking lot. The idea behind such arrangement is that, the system allocates a parking space and automates the payment for the parked time (Revathi et al. 2009). In addition to establishing the location of vacant parking space and payment for the parking, the system could also include the management of parking permits. Currently, the parking permits are hard copies and digitizing them would be a value addition and a further utilization of the system.

1.1 Objectives of the project

This work aims to study how challenging it is to find parking space in Helsinki area in comparison to what research works have pointed out for other urban cities. This research investigates whether there is need for smart parking systems in Helsinki area basing on driver's point of view. A smart parking system would be designed, implemented and deployed at Katajanokka Helsinki, with the aim of identifying some of the challenges a smart parking system would face. The challenges to learn includes design, implementation, deployment and operations challenges.

The main objective of this paper is to highlight the steps and decisions that were taken during the conception, design and implementation of a smart parking system, concentrating mainly on the architecture and the hardware required. The technologies and standards that were relevant towards the completion of the project are briefly introduced.

1.2 Research questions

The objectives of the project were compressed into three research questions. The idea was, if these questions are answered, then the objectives of the project would be achieved.

1.2.1 What are the challenges associated with parking in Helsinki Area?

To answer this question, related work in this area will be evaluated to know and study some of the known challenges. In addition, interviews will be conducted with the target group relevant to this work to capture existing challenges for the Helsinki area.

1.2.2 What are the advantages of smart Parking in Helsinki Area?

To answer this question, insights about the challenges in parking collected from the conducted interview will be used to analyze the kind of benefits that can be realized if those challenges were solved in the best way possible. The possible solutions will be looked at during the analysis.

1.2.3 What are the challenges a Smart Parking system for Helsinki Area would face?

To answer this question, a prototype will be designed, implemented and deployed, and along the way, the challenges and benefits of a smart parking system will be documented and analyzed.

1.3 The different roles I played during the project

The project was presented in a form of what the client wanted to achieve. A system to fulfill the following needs was required.

1. A system that could help to learn the parking behaviors of drivers in the city
2. A system that could help manage parking in the city (Parking management system)
3. A system that could support paying for the parking in the city (Parking fee payment system)
4. A system that could provide the parking facility occupancy information to drivers and other systems. The required system was actually a system of systems.

A three-month pilot was allowed to test the system, to verify its feasibility and also gain more insights on what benefits such a system would bring to its users.

From the given requirements, I started by identifying some of the problems that the system was likely to solve, but I was not sure whether I was not creating just problems of my own. To get a clear picture of what challenges drivers in Helsinki area face, I decided to conduct interviews during a class project for the course, *Methods for Software Engineering and Business Research P*, that in so doing would help me validate my hypotheses. I chose to adopt the name *Smart Parking System* for the name of the system and the class project title was, *Usefulness of Smart Parking System in Helsinki Area*. I worked with my colleague Aftab Ansari during the class project when conducting interviews and analysis of data. We collected a great deal of useful information and the hypotheses were validated.

Equipped with this information, it was time to get to the design table and start planning how to proceed.The first step was to identify the key components the system should be comprised of. To learn the parking behaviors of drivers in the city resonated with the idea of tracking the cars. The needed system was a system that could monitor how and where cars are parked. There I knew an active RFID tag could get that task sorted out. My instructors had some ideas on the technology to use. They had tried out some positioning technologies that operated in 2.4 GHZ band and used angle of arrival to estimate the position, but the results were not satisfying. That helped me as I was discouraged from driving down that lane. At least they knew one upcoming wireless technology, which they mentioned to me that I should put on my table as one of the candidates to choose from. The technology was LoRa, the technology I had never heard of, but here it was for me to learn and figure out if it was a fit for our application.

Having worked with indoor positioning systems using active RFID tags and with some knowledge on mobile cloud computing, the system's architecture was clear to me that it was to be similar, with a star topology. What was needed was the means to collect that data about cars' identity and parking location, send that to the server where it is analyzed together with data from other sources. At this stage I knew what to do but how to do it, in a very limited time was out of my reach. I came to know that GPS technology was black listed when I proposed it as one of the options, so it was off my menu. I was also persuaded to believe that time-of-flight (TOF) could be used to estimate the position of an RFID tag. I didn't have enough knowledge and experience on that to give an affirmative answer, so I started researching for me to be able to validate or invalidate that claim. An obvious choice was to use received signal strength indicator (RSSI) to estimate position of a transmitter, I had used it before and I knew its limitations.

I bought a number of different development boards for prototyping, but the timing requirement for using TOF to estimate position was less unlikely to be achieved within our budget and time brackets. My instructor brought in different consultants to assist on the system implementation. But I disagreed with these consultants, most of the time, as it looked like they were pulling me, mostly, towards their comfort zones which in many times looked unsuitable for our application. The confidence I had was knowing what to do, but my challenge was about how do it in reasonable time, this lead to me rejecting the consultant and advised to hire interns instead. "The advantage of utilizing interns is that when they learn, they could be hired as employees unlike consultants", I explained. So I recruited one student to help me develop embedded software for the RFID tag. At this point I decided that we concentrate on LoRa technology as it was promising long range communication even in built up areas.

The sourcing problem then struck. Where to get the LoRa tags was not obvious and those from Semitech – the LoRa chip maker, were quite expensive. Other tags that were available then, were preprogramed and could only work with the vendor's gateway. They were costing more because they had sensors on board, but I needed no sensors and I never wanted to pay for what I would not use. The next option was to buy a LoRa radio module to build my own tag with a microcontroller of my preference. The first LoRa radio modules I bought for prototyping, even though they were expensive, but again the vendor never put available the data sheets, unless if I used their development boards and ecosystem. They had encoded and hidden the module pins to make them difficult to reverse engineer, time was not my ally to try that.

During my search I came across a company in German that sold LoRa modules relatively cheaper. These modules though pre-programmed, it was possible to reprogram them and good documentation was available teaching how to go about all that we wanted. But again the module had only the radio and MCU, and nothing else. Therefore, there was a need for an adapter board to host the module and other components including a button, LEDs, antenna, and power supply, or to buy it from the vendor. The adaptor board from the vendor was a bit expensive for our budget and also had many extras we didn't need, so I chose to have those designed in-house.

With tags looking like they had been taken care of, it was time to look for the gateway. There was not so many vendors in gateway business, something that made the gateways not only expensive but also scarce.

Many of the available gateways then, had either WiFi or Ethernet for the IP network interface. I wanted a gateway with 3G interface. I had tested GSM/GPRS modules but I was not impressed with their performance. Another limitation was that many of the LoRa gateways were none re-programmable, they were already linked to the vendor's cloud or the ecosystem's network server. This meant that I had to use their cloud service if I was to use their gateways. Some of the gateways that were open, were quite expensive to fit well into our budget. So I decided to assemble my own gateway. To assemble a gateway, I needed a LoRa concentrator, a raspberry pi, a 3G dongle, and there was my relatively cheap gateway. The concentrators were later put available by the German company at a reasonable price.

There were two tasks on the table to execute; the embedded software development for the gateway and the assembly of the gateway, and the making of the LoRa tags. The making of the LoRa tags required some knowledge in radio electronics, power electronics and printed circuit board design. So, I needed someone to take care of the hardware and i handle the embedded software development for the gateway. What turned out was a failure to find a hardware person but instead I spotted a person for embedded software development. I decided to recruit him and myself took head-on the task of developing the hardware for the project.

After the first version of the LoRa tags were out, it became apparent that the tags would need enclosures and it was time to get started designing them. This task required an industrial designer with some experience in this area. I made calls to a number of them in my contacts and fortunately one of them accepted to take up the challenge, and so I recruited him. At this stage all looked progressing on well under the tight schedule we had.

At first I thought I would build a backend where the contents of the gateways could be off-loaded. The plan was to use Django (python) to build a server with APIs to be used by the other company server to fetch the tags data for processing and analysis. I actually quickly built a test server to visualize how the gateway utilizes the RESTful APIs to post and get data from the server. But after a discussion with the company's CIO, we realized that building another server would be a duplication of work. So we agreed that he should provide endpoints from the server he had developed, to handle the traffic from the gateways. Under this arrangement, the backend development was transferred to his command. But still I participated in formulating some logics and development of some algorithms that were used in the backend. When it came to employing machine learning algorithms to estimate the location of the LoRa tags based on RSSI, I was available to give the domain knowledge needed for the data scientist to perform his tasks effectively.

On top of the above mentioned roles, I was also out in the field surveying the area to figure out the system requirements and locations fit for the installation of the system. I was also I key resource during the installation of the system itself.

1.4 The Use Case diagram for the Smart Parking System

The smart parking system to be developed has four actors whose needs the system must satisfy. The primary actors of the system include car drivers, system administrators, parking enforcement officers, and the city planning authority. Figure 1 shows the use case diagram of the system highlighting the actors and their actions.

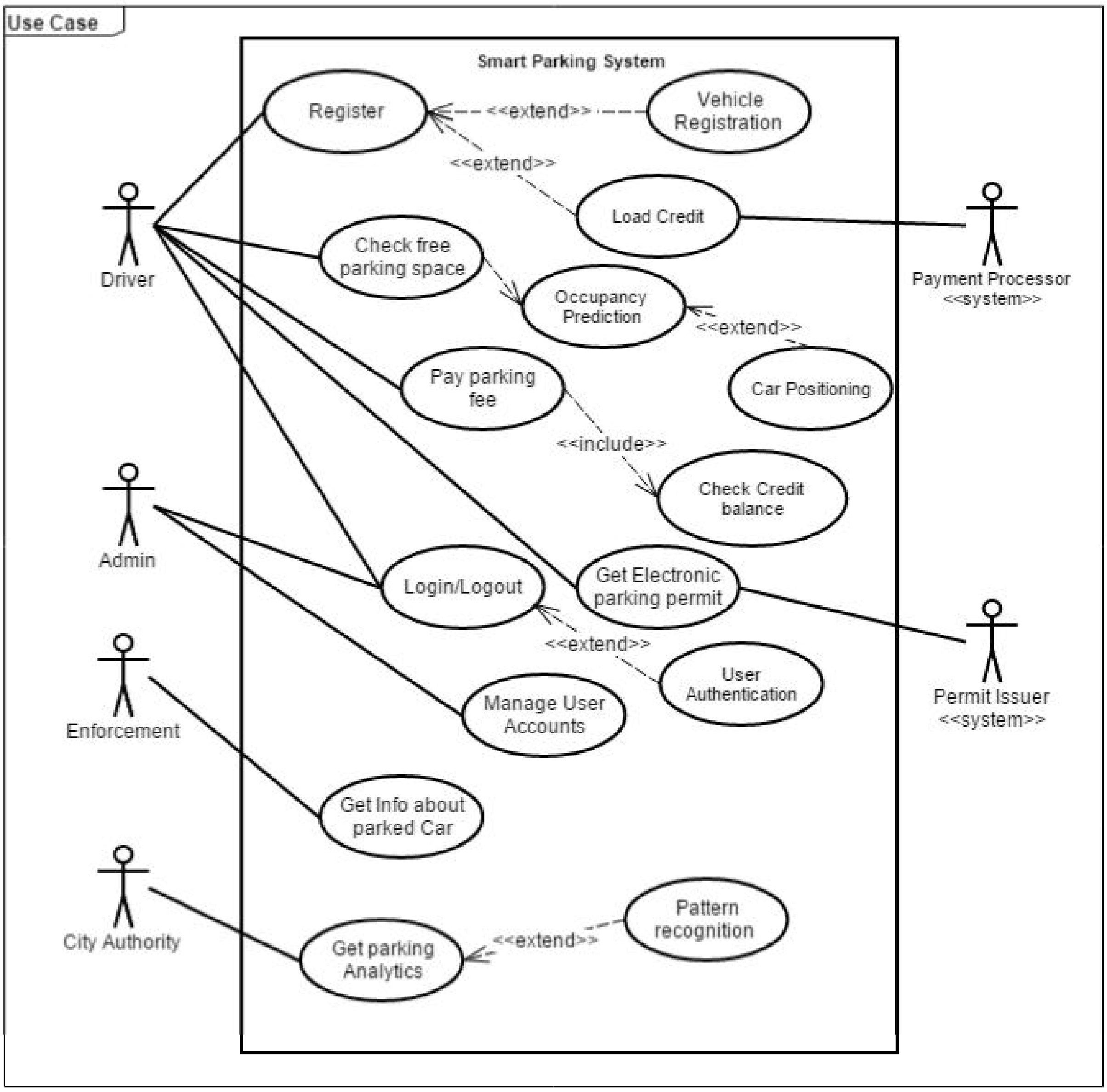

Figure 1: The use case diagram for the smart parking system

The secondary actors include the payment processing system and the parking permit issuing system. The **Driver** has to **Register**, during that process the vehicle's details are asked and some credit has to be load to the created account. The **Payment Processor** – the external system is used to process the payment. The driver checks for available empty parking spaces in the part of the city before driving there. The **check free**(empty) **parking space** depends on the **Occupancy Prediction** engine which includes the execution of

Car positioning algorithms. When a driver finds a parking space, the driver wants to **Pay parking fee** which action requires the system to **Check Credit balance** to cover the parking fee due. The driver furthermore wants to **Get Electronic parking permit**, this action invokes the **Permit Issuer** – an external system that manages the issuance of parking permit.

The **Admin**, just like the driver, both have to **Login/Logout** and the system must verify that they are who they claim to be using **User Authentication** engine. The system admin also wants to **Manage User Accounts** in case of a problem or during routine maintenance of the system. The **Enforcement** officer wants to **Get Info about parked Car**s which tells if a car has a parking permit or if it has the parking event running at the end of which the parking fee would be paid. The **City Authority** want to **Get Parking Analytics** to predict the parking behaviors of the drivers in the city to aid their city planning. This action involving the **Pattern recognition** engine that looks for patterns and correlations between different data set of parking events and occupancy.

1.5 Structure of the thesis

This thesis is made up of six chapters. Chapter 1 introduces the problem to be solved. Chapter 2 gives the background information of architectures, standards, radio and web technologies that could support the realization of the project. Chapter 3 gives the methods/approaches employed to collect information used to implement the system. Chapter 4 explains how the system was implemented. Chapter 5 presents the results from the research, measurements, and tests that were done. Chapter 6 gives the discussions, conclusions and future research areas that were not looked at during this project.

CHAPTER 2

2. Background

The problems this project is addressing needs an aggregation of data and information from different sources, but these sources must be connected to the internet to make that data accessible. The sharing and utilization of data between interconnected data sources, through the internet, shares similar characteristics to the concept known as *The Internet of Things (IoT)*. This chapter presents some contributions towards the IoT, with anticipations that inspirations towards this project could be gained from what is already done for IoT. Some of the IoT architectures, standards, radio technologies and web technologies that are considered relevant to the execution of this project are briefly introduced.

The Internet of Things is a paradigm referring to the interconnection of physical objects (Things), people, virtual data and environment that allows them to interact at will (Framling et al., 2014; Al-Fagih et al., 2013). The physical objects are equipped with hardware and software that permits them to collect and exchange data and/or execute control commands to drive actuators.

2.1 IoT Architectures

The architecture, on the other hand is a blueprint describing how a physical structure would be built. Putting it together, IoT architecture describes how a system of interconnected physical objects, people and environment would be built. It should be noted that, to be able to develop a blue print for a system, the expectations out of such a system must be known first.

The architecture of an IoT system has to take into consideration the diverse and distinct technologies spanning from computing (e.g. cloud computing), communication (e.g. 3 or 4G) to semantics (e.g. data mining) (Vasco et al, 2014). The IoT architecture of an IoT system is also influenced by the requirements of a given application/project and its location in the IoT domain (Krco et al., 2014). A good number of project-based IoT architectures such as IoT-A, SENSEI and SPITFIRE, were developed to address specific requirements of those projects (Krco et al., 2014). The difference of the focus to which an IoT system could be designed has led to the development of several different IoT architectures with different components and protocols.

Among the challenges faced by these IoT architectures is the difference in the terminology, and limited interoperability between systems (Krco et al., 2014). The IoT community agrees, according to a survey (Krco et al., 2014), that a common IoT reference model would be useful if the lack of interoperability is to be addressed. A good number of projects have been carried out including those supported by European Union, with the aim of harmonizing the IoT architectures and hence, systems. Global initiatives such as oneM2M were created to see to it that a common platform has been specified and a reference model developed.

The European Telecommunications Standards Institute (ETSI) defined a machine to machine communication (M2M) top-level architecture that is made up of two layers; The Device and gateway domain, and the network domain (Krco et al., 2014). The role of the gateway is to create a communication channel between devices and the rest of the IoT/M2M system aided by a communication network such as GSM (Global System for Mobile communication) network. The IoT-A project was inaugurated with an objective of availing a generic Architectural Reference Model (ARM) from which IoT architectures could derived (Krco et al., 2014). With the same technical base, architects of IoT systems would use similar terminologies and best practices that would simplify their efforts towards optimizing the interoperability of their systems.

A good number of IoT architectures have been proposed and the proposed architectures seem to fall under the M2M top-level architecture, defined by the ETSI mentioned in previous paragraph. Some of the proposed architectures are briefly explained in sections: - 2.1.1 (Architectural Model of IoT Public Sensing), 2.1.2 (IoT Architecture Proposal for Disabled People), and 2.1.3 (An IoT Gateway Centric Architecture). The Long Range Wide Area Network in section 2.1.4 combines three-in-on; the architecture, the standard and the protocol.

2.1.1 Architectural Model of IoT Public Sensing

Al-Fagih et al., (2013) proposed an Architectural model of IoT public sensing consisting of Data collectors (mobile and static), Gateway and Access Point for the main components. Data is collected from locations where they are generated and passed on to the Gateway, which in turn hands it over to the Access Point from where users accesses it. The gateways are responsible for managing the data within their corresponding peripheral networks and the access points are mainly owned by various service providers. Figure 2 shows the architectural model proposed by Al-Fagih et al., (2013).

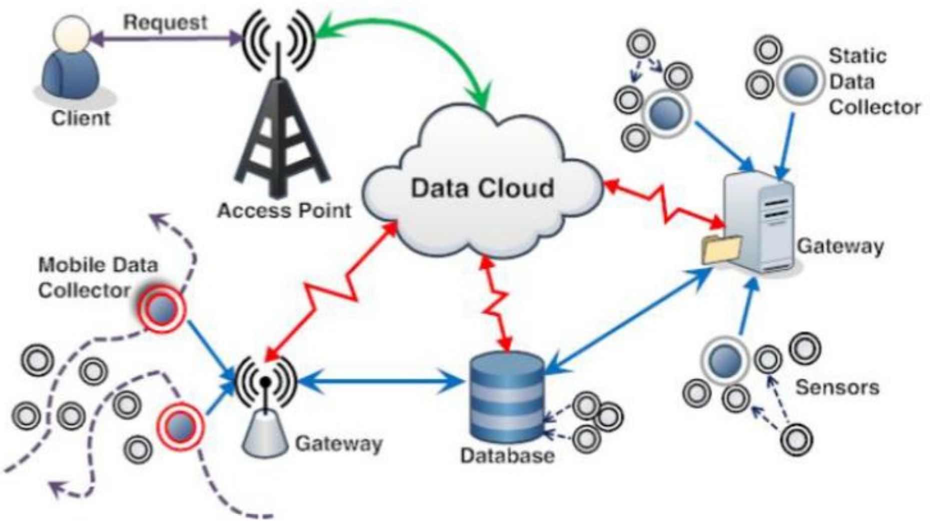

***Figure 2**: Architectural model of IoT public sensing (Reprint from (Al-Fagih et al., 2013)*

Data is collected from sensor networks together with that freely shared by mobile devices users. Publically available data such as data from environmental monitoring stations, public transport schedules and alike, from databases is also collected. The deliverable is a set of services derived from combining different available data to achieve an improved residential experience and quality of living in smart cities.

2.1.2 IoT Architecture Proposal for Disabled People

The IoT Architecture for disabled people was proposed by Vasco et al. (2014) targeting to address the needs of people with disabilities, the needs that available architectures never considered. The architecture is made up of four layers; the Device, Network, Service and Application layer. Figure 3 illustrates the proposed architecture and the corresponding mapping to the TCP/IP stack.

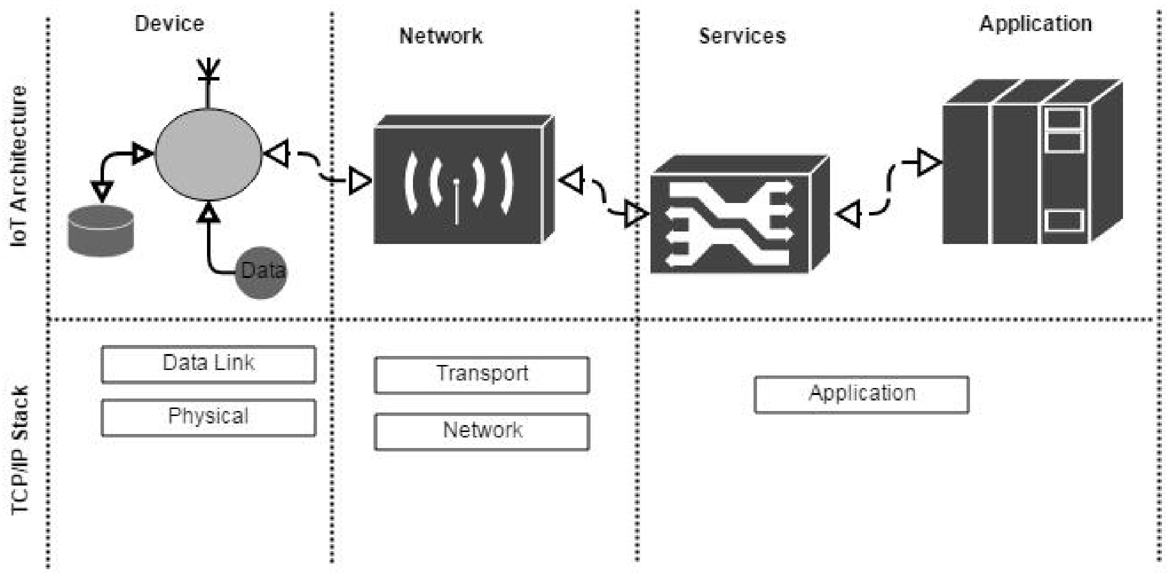

Figure 3: IoT Architecture for People with Disabilities (Vasco et al., 2014)

It is proposed that the Device layer would collect data/event from the physical world and then pass it to the network such as WiMax or 3GPP. The Device layer maps to the Physical and Data link layers of the TCP/IP stack. The Network layer handles the end-to-end communication, addressing, and routing of messages to specified destinations. It maps to Transport and Network layers of the TCP/IP stack. The service layer is introduced to, literary, hide the vertical silos caused by the heterogeneous nature of IoT environments from the application layer and hence, facilitating high application level interoperability between IoT systems. Application layer is the interface between the users and the system, it provides the user interface. (Vasco et al., 2014)

The Device layer is made up of two components; sensing devices and wireless sensor networks. The sensing device includes the detecting and sensing (perception) technologies such as sensors, cameras, e.t.c, that captures the state of the environment. The wireless sensor networks are self-organized wireless sensor networks that collects the physical status of the environment. An IoT gateway can only link devices that are not IPv6 compliant to the core network. The IoT technologies from which the device layer could benefit includes the identification technology, the sensor technology and the assistive devices. (Vasco et al., 2014)

2.1.3 An IoT Gateway Centric Architecture

Datta et al. (2014) claims that the proposed IoT Gateway Centric Architecture would allow real time interaction between mobile clients and sensors and/or actuators through a wireless gateway. The architecture is composed of mainly three layers; the M2M Devices and Endpoints layer, the M2M Sensor gateway layer, the wireless gateway layer and the Client, as depicted in figure 4.

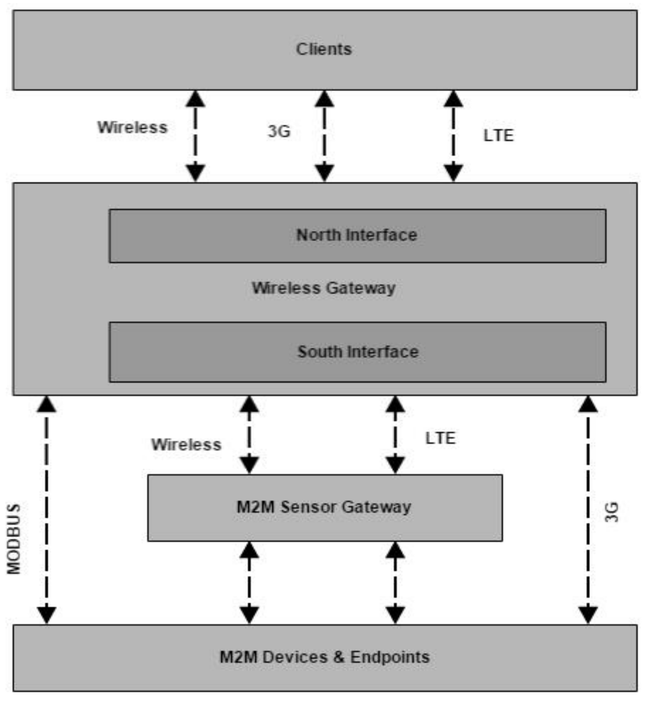

***Figure 4**: An IoT Gateway Centric Architecture (Datta et al., 2014)*

The wireless gateway is mad up two interfaces; the north and south interface. The north interface handles the communication between the mobile clients as well as the discovery phase of M2M devices and endpoints. Whereas the interconnection with and the management of the M2M devices is left to the south interface. The M2M Devices and Endpoints layer is responsible for collecting physical quantities such as temperature and pressure from the environment, aided by appropriate sensor endpoint. The smart M2M devices have the capability, resources and support for the Sensor Markup Language (SenML) to encode the measurements into the JSON format, pass the data directly to the South interface of the wireless gateway. But for the non-smart devices, they have to go through the MODBUS or the M2M sensor gateway which creates the desired SenML compliant metadata and encodes it to a format acceptable to the wireless gateway.

The mobile clients the architecture assumed, included smartphones and tablets running Android and iOS. These clients are running a “Connect and Control Things” (CCT) application whose role is to receive the measurements from sensors and can control actuators via a cloud service such as Google App Engine (GAE).

Other layered IoT architectures listed by (Vasco et al, 2014) that has not been covered here, are given by their layers:

1 Perception, Network, Middleware, Application, Business
2 Network, Element Management, Network Management, Service Management, Business Management
3 Edge Technology, Access Gateway, Internet, Middleware, Applications
4 Perception, Network, Application and
5 Perception, Transmission, Application

2.1.4 Long Range Wide Area Network (LoRaWAN) Architecture

LoRaWAN is a low power wireless networking architecture and protocol developed for low-cost secure two-way communication in IoT. The LoRa network, which is LoRaWAN due its ability to provide wide area network, is made up of four main components: Endpoints, LoRa gateway, NetServer and Remote computer (Ian Poole ,2014; Vangelista et al., 2015). The LoRaWAN architecture is commonly deployed in a star-of-stars topology with the gateway interconnecting the endpoints and network server (LoRa Alliance, 2016). Figure 5 shows the illustration of the interconnection of components in a LoRa network.

Endpoints (End-devices): These are components of the LoRa network that are responsible for collecting data or execution of control commands, usually from remote locations.

LoRa gateway: The gateway links the LoRa network end-devices to the backhaul system. The link between the gateway and the NetServer can be through Ethernet, cellular or any other telecommunications link wired or wireless that supports standard Internet Protocol (IP) connections. (Centenaro et al., 2015; Ian Poole ,2014)

NetServer: The Network Server handles the cores of the network. It is possible in some cases for the end-device to be in range to more than one gateway, in which case the Netserver has the task of filtering out duplicates and outlying packets (Ian Poole ,2014; Centenaro et al., 2015). It also schedules acknowledgements on top of adapting data rates (Ian Poole ,2014). The NetSever plays a key role in cases when the network is public or shared by different integrators. It must receive and differentiate packets from different companies' nodes and route them to their respective backend systems.

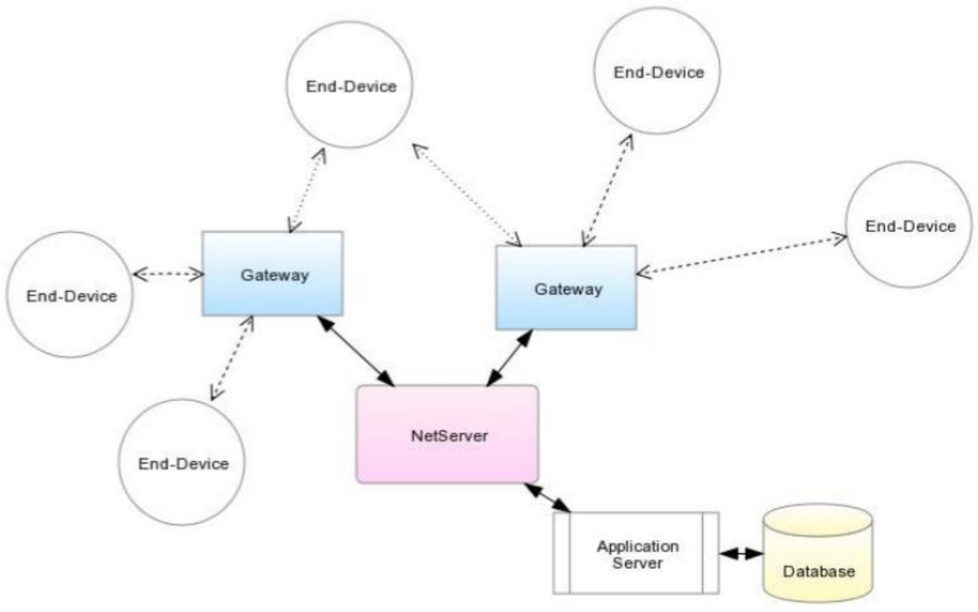

Figure 5: LoRa network architecture

Remote computer/Application Server: The remote computer represents different systems in a shared LoRa network. It is responsible for controlling the actions of the end-devices or for the collection of data from them (Ian Poole ,2014).

2.2 IoT standards that would support Smart Parking

The need for a standard is no question any more if the interoperability of IoT systems is to be achieved. The question is whether a single standard can cover the enormous number of technologies required to get IoT work, covering Layers including Application, Network/Transport and Physical/Link layer. In the following sections, a few of the standards covering one or more layers are briefly presented.

2.2.1 The Open Data Format Standard

The Open Data Format (O-DF) is an Open Group IoT Standard whose cardinal objective is to addresses the application layer requirements of the IoT and foster the interoperability between systems. The O-DF allows data sources to publish the presence of that data, and provides easy access to the data with a possibility to attribute/rule based filtering of such data.

The publication of available data using O-DF can be achieved using already familiar Uniform Resource Locator (URL) addresses. The same structure of O-DF supports for exchanging data between system. O-DF makes it possible for information systems to manage IoT-related data due to its generalized data representation standard. The O-DF uses object identifier to make it possible to trace data related to a particular IoT item from different information systems. An Object identifier is one of the most important data structure in all general IoT standards, as it might be the only way to make the identification of parties requesting for data, context of request, and the visibility and access to data could be dependent on the same.

The specification of O-DF is done using XML Schema. The O-DF is comparable to object-oriented programing in relation to the style used to create information structures. An O-DF structure is a hierarchy with an Objects element as its top element, as illustrated in figure 6. There is no limit on the number of Object sub-elements to make up an Objects element.

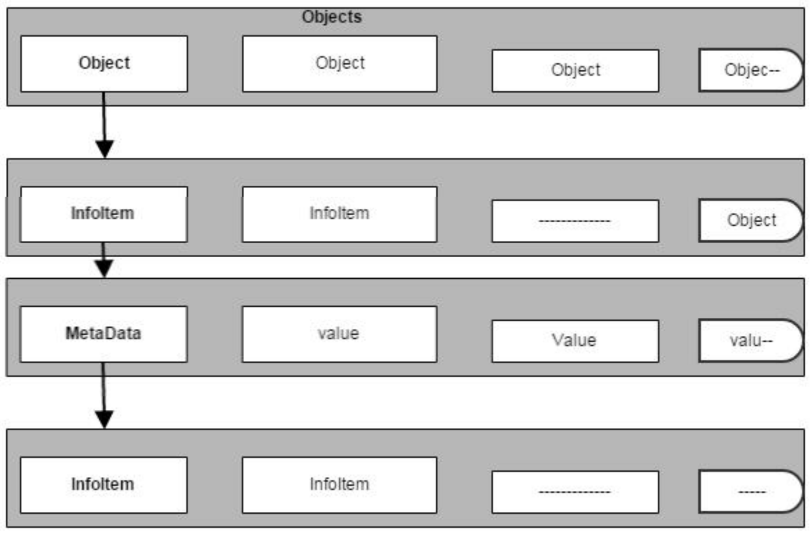

Figure 6*: Illustration of O-DF Element Hierarchy (Source: Open Group, 2014)*

The *InfoItem* are sub-elements for the properties of the *Object* sub-elements, but properties also include *Object* sub-elements and the *Object* tree can be up of any number of levels. The O-DF content describing IoT “Thing” can be sent by any messaging protocol such as RESTful services, Java Message Service

(JMS) e.t.c as both request and responses. For more details on O-DF you may refer to (Open Group, 2014).
Source (Open Group, 2014)

2.2.2 Message Queuing Telemetry Transport (MQTT) for Sensor Networks (MQTT-SN)

The Message Queuing Telemetry Transport (MQTT) is lightweight publish/subscribe messaging protocol, originally developed by IBM but now it is open standard (Lee et al, 2013; Stanford-Clark & Truong, 2013). Its development was intended for application suffering from unreliable networks, low bandwidth and high-latency (Lee et al, 2013; Hunkeler et al, 2008), yet a data-centric communication is needed to be supported. The publish/subscribe messaging system is one of the data-centric communication examples in which the information sent to the recipients is based on the content and interest instead of the network address (Hunkeler et al., 2008, Stanford-Clark & Truong, 2013).

The MQTT protocol fails for applications with resource constrained devices – very simple devices with limited storage, low cost, and low computation capabilities (Stanford-Clark & Truong, 2013), as the support for TCP/IP networking is farfetched and complicated. This lead to the development of the MQTT-SN – a publish/subscribe protocol for wireless sensor networks that is optimized for low-cost, battery operated devices with limited resources (Stanford-Clark & Truong, 2013). The independence of MQTT-SN from the underlying network services permits usage of any 2-way communication between nodes and gateway to be used (Stanford-Clark & Truong, 2013).

The gateways can be classified into two categories; a transparent gateway or an aggregating gateway. This categorization is based on the nature of the connection between the client and the server. With the transparent gateway, there is an exclusive end-to-end connection between each client and a server that is maintained all the time and an almost transparent messages are exchanged between the two. The advantage with this set up is that, all the functions and features from the MQTT server are visible to the client. The drawback however, is on the fact that some servers might not support unlimited number of concurrent connection. The other option is the aggregating gateway in which not all clients connected to the gateway are connected through to the MQTT server, but only a few are selected by the aggregating gateway.

The architectural set up of the MQTT-SN protocol is mainly composed of three components; the MQTT-SN client, gateway and the forwarder. Figure 7 shows the MQTT-SN architecture and the interconnection of the components. Since the MQTT-SN clients have no TCP/IP stack, they connect to the MQTT-SN server through the MQTT-SN gateway using the MQTT-SN protocol. There are two possible configuration of the gateway; it is either an integral part of the server or a stand-alone gateway. In the latter case, the role of the gateway is to translate between MQTT-SN and MQTT protocols. Since MQTT is used between the gateway and the server yet the client and gateway uses MQTT-SN protocol to intercommunicate, translation is necessary.

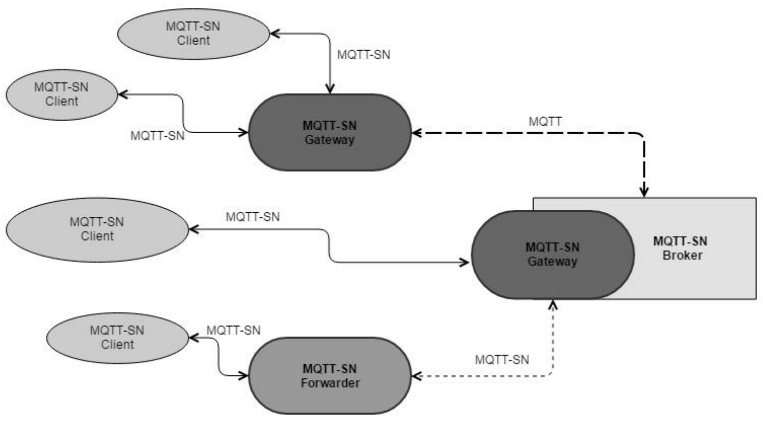

***Figure 7:** MQTT-SN Architecture (source: Stanford-Clark & Truong, 2013)*

The forwarder is another channel through which a client can interact with the server, the forwarder simply encloses the packets from the client into the MQTT frame details and transparently forwards it to the server. The reverse happens for communications from the MQTT server to the MQTT-SN clients. (Stanford-Clark & Truong, 2013)

The packet format of the MQTT-SN protocol is shown in Figure 8. The packet is made up of two parts, the Message header and the Message variable part. The Message header defines the packet length and the message type. The Message variable part may contain one or more parts depending on the type of the message.

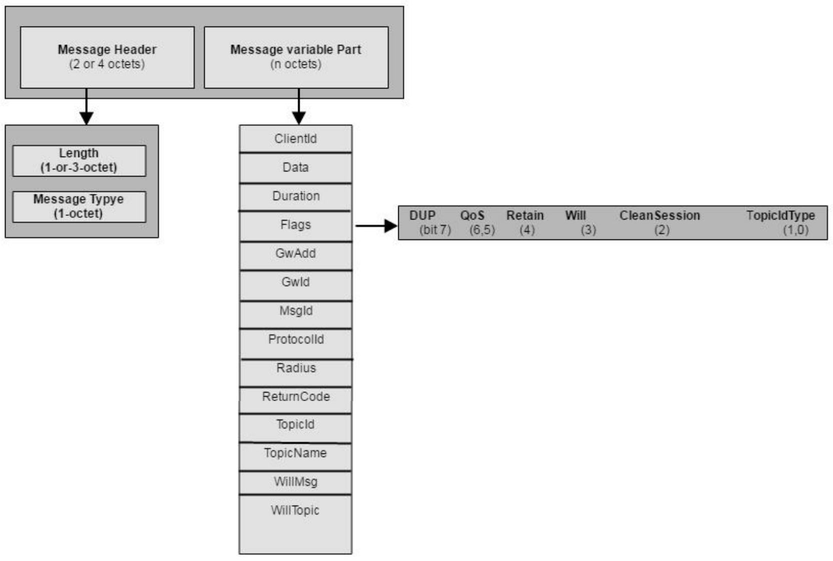

***Figure 8:** MQTT-SN General Message Format (source: Stanford-Clark & Truong, 2013)*

To minimize the packet length, the message type is coded into a one-byte code instead of a whole string. Examples of message types include 0x00 for ADVERTISE, 0x04 for CONNECT, 0x12 for SUBSCRIBE and so on. (Stanford-Clark & Truong, 2013)

2.2.3 The Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) is a specially designed web transfer protocol for nodes and networks with limited capabilities (Shelby et al., 2014; Bormann et al., 2012; Villaverde et al., 2012). The constrained nodes might be as simple as an 8-bit microcontroller based system with limited amount of RAM and ROM. The networks usually have high packet error rates and low throughput such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) (Shelby, et al., 2014). The CoAP was born from the efforts of the Constrained RESTful Environments (CoRE) working group and it was standardized in the IETF as RFC 7252 (Shelby, et al., 2014).

The CoAP follows a client/server model in which the client sends a request and the server responds. One of the interesting part of the CoAP clients is that they are able to switch between roles. CoAP can be looked at as an optimized HTTP that offers part of REST methods to constrained M2M/IoT applications. Therefore, the HTTP verbs; GET, PUT, POST and DELETE are supported and devices can use them to request data from the server. CoAP brings on table a couple of good features to the M2M/IoT community including but not limited to built-in discovery, multicast support, and asynchronous message exchange. (Shelby, et al., 2014)

The built-in discovery feature permits clients to learn of the resources located on servers. The subnetwork devices benefit from the same feature as they are able to discover themselves. There is also a possibility to post fully described resources to Resource Directory (RD) entities from where others can know who hosts what resources. The multicast support allows group communication, in which many similar devices are queried simultaneously. (Villaverde et al., 2012)

Since CoAP runs on top of UDP to carry out message exchange between endpoints, it uses four message types to achieve reliable communication and imitate the likes of TCP. These message types are; Confirmable, Non Confirmable, Acknowledgement and Reset. The Confirmable message type stresses to the recipient that an acknowledgement of message receipt is required. The sender repeats sending the same message with the same message ID until an acknowledgement is received from the receiver or a Reset Message to indicate the inability to process the message. The Non Confirmable message is used in situations where a reliable communication is not critical and acknowledgement is unnecessary. (Shelby, et al., 2014; Villaverde et al., 2012)

The adaptation of CoAP has reached a stage that some implementations are already available:

- mbed has CoAP support
- Java CoAP Library
- libCoAP C Library
- OpenCoAP C library

2.2.4 The LoRaWAN protocol

The LoRaWAN protocol stack is open source and it is being developed by the LoRa Alliance, spearheaded by IBM, Actility, Semtech, and Microchip. It fits in the connectivity model known as Low Power Wide Area Network (LPWAN) being promoted to drive IoT deployments. It is developed to support the LoRa network architecture presented in section 2.1.4. The LoRaWAN architecture is composed of end-devices, gateways and a network server, as shown in figure 9. The network server then links to different application servers.

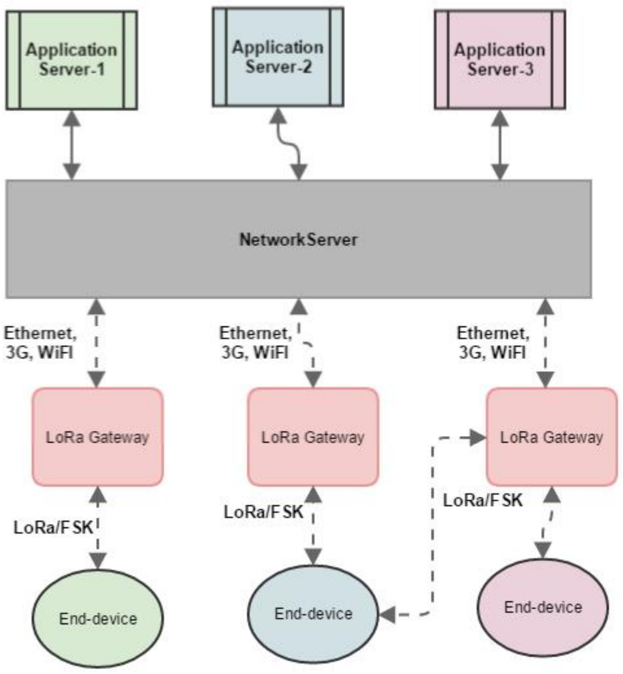

Figure 9: *The LoRa network components and links*

The end-devices can be implemented in three classes that are intended to support the different requirements for given applications. There are classes A, B and C (Ian Poole ,2014). All these classes support bi-directional communication between the end-devices and the gateways, but the implementation is not the same. In class A for instance, there are two listening/receiving instances after every transmission, and the scheduling of which is entirely up to the end-device's needs (Ian Poole ,2014; Vangelista et al., 2015; Vangelista et al., 2015; LoRa Alliance, 2016). The second listening window is scheduled on a different sub-band which must have been agreed upon with the NetServer (Vangelista et al., 2015). This class provides power saving options for devices that must contact the server to receive something. This means that the only way the server can contact the end-device is after the end-device has contacted it.

For Class B end-devices, on top of all the mandatory features offered by class A, they add on the ability to schedule extra receive windows. The scheduling of receive windows by an endpoint creates a need to synchronize the device clock with that of the server, and this is achieved by gateway sending sync beacons (Ian Poole ,2014; Vangelista et al., 2015). Under this arrangement, the server can know when it can contact the end-device instead of waiting for the device to contact the server, through the gateway.

The gateway is transparent to the end-device, that is, it has no knowledge of the contents of the packet, it simply adds extra information about the communication channel such as radio signal strength indicator(RSSI) (Centenaro et al., 2015; Ian Poole ,2014). A gateway built with a concentrator supports parallel processing of up to 9 LoRa channels and channels are differentiate using a sub-band and spreading factor (Centenaro et al., 2015).

The Class C end-devices stay in receive model except at the times when they are transmitting something. This class is suitable for devices where big volumes of data have to be received but less is needed to be transmitted. (Ian Poole ,2014; Vangelista et al., 2015)

Network security: Network security being an important aspect of any network, LoRa networks employed several layers of encryption to protect the LoRa network from the "bad guys". *A Unique Network key (EUI64)* was introduced for network level security. On the application level, a *Unique Application key (EUI64)* handles the end-to-end security on this layer (Ian Poole ,2014). The *Device specific key (EUI128)* is for protecting the devices such that, if one device is attacked, it is only that device that is compromised and none else in the network. The network uses the AES CCM (128-bit) for encryption and authentication. (LoRa Alliance, 2016)

The security of the LoRa network is derived from the already proven working secure protocols such as the IEEE 802.15.4 MAC. For instance, the authentication process of the LoRa network MAC layer was replicated from the IEEE 802.15.4 standard using the 4-octet MIC (Message Integrity Code) (Vangelista et al., 2015), there was no need to re-invent the wheel. Figure 10 show the architecture of the LoRa protocol. The three evident parts translating to the LoRa architecture are the Sensor – the end-device, the gateway and the network server. The gateway in this case is shown to be a packet forwarder and its functionality resembles the one mentioned in section 2.2.2.

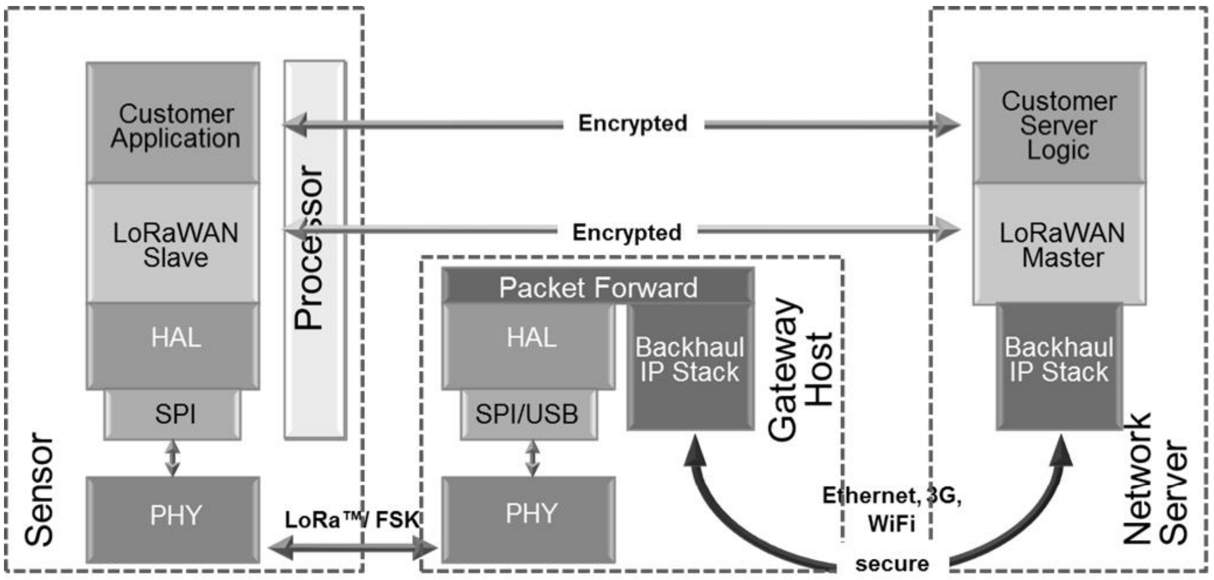

***Figure 10**: LoRaWAN protocol architecture (Reprint from: LoRa Alliance, 2016)*

The physical layer linking the end-device(sensor) to the Gateway can be either through LoRa modulation scheme or the Frequency Shift Keying (FSK), the LoRa gateway supports both. The IP based connection between the gateway and the network server can be anything from a WiFi, Ethernet to 3G.

Message Formats
The LoRaWAN specification defines two message formats, one for the uplink communication and the other for downlink communication. The uplink messages are sent from the end-device to the gateway and

then to the network server. The LoRa physical packet structure consists of a Preamble, LoRa physical header(PHDR), a header CRC (PHDR_CRC), the payload (PHYPayload) and CRC (LoRaWAN Specification 1R0).

```
----------------------------------------------------------------------
| Preamble | PHDR | PHDR_CRC | PHYPayload | CRC |
----------------------------------------------------------------------
```

The downlink messages for the downlink communication are received by the end-devices from the gateway, which passes them over to devices on the behalf of the network server. The downlink physical packet structure is made up of a Preamble, LoRa physical header(PHDR), a header CRC (PHDR_CRC), and a payload (PHYPayload) (LoRaWAN Specification 1R0).

```
--------------------------------------------------------------
| Preamble | PHDR | PHDR_CRC | PHYPayload |
--------------------------------------------------------------
```

Both the downlink and uplink messages use the radio packet explicit mode which includes the LoRa physical header and a header CRC (LoRaWAN Specification 1R0).

2.3 Wireless Technologies and remote connection

Wire-based communication technology is losing the battle to the wireless due to its short comings such as the high cost and the longer time required to have a network set up. When it comes to sensor networks, and now the IoT systems, wireless technologies look to be the obvious choice. It is not only cheaper, but it is also easier to set up a network in quite a reasonable time. In this section, some of the wireless technologies that are considered good candidates for supporting IoT systems are briefly introduced, including some of the techniques useful for remote controlling.

2.3.1 Active RFID based positioning

The Radio frequency identification(RFID) is one of the oldest technologies that has lived the test of time. The RFID technology has been popular in supply chain for mainly item tracking and identification (Tesoriero et al., 2008), but now it is found in other areas such indoor positioning supporting location based services. Unlike the passive RFID tags that are powered by the radiated electromagnetic signal from the transmitter, active RFID tags carry their own power supply that allows them to transmit a strong signal that permits long distance readings (Weinstein, 2005; Retscher and Fu, 2008). Active RFID tags can have more memory and processing power to support relatively complex functions in comparison to the passive tags. The capabilities and flexibility of active RFID tags allows them to be used for data collection and/or actuation in monitoring systems.

The Global Positioning System (GPS) is the de facto standard for outdoor positioning, but it suffers some limitations in areas with tall buildings and inside buildings or in places where signal penetration is poor (Wang et al., 2009; Zhang et al., 2011). Furthermore, GPS systems have higher energy requirements that simple sensor nodes might not afford to satisfy, as it could limit their battery life considerably. Using active RFID for positioning in places where GPS in impractical to use could bridge the gap, but a combination of the two technologies would give a complete solution for indoor and outdoor positioning. When active RFID is used for indoor positioning, signal strength could be used to give about 5 meters of precision (Tesoriero et al., 2008). The location of an item could be obtained by using RFID either as a mobile tag or as a mobile reader (Tesoriero et al., 2008).

Mobile tag is when an RFID tag is attached to the item whose position is to be determined and RFID readers are usually stationary, whereas the reverse is true for a mobile reader. A mobile reader was used by (Zhang et al., 2011) to implement an Active RFID Positioning (ARP) scheme where RFID tags were deployed along each lane of the road to position cars installed with RFID readers. It is possible to use signal strength indication (RSSI) from active RFID tags to estimate the location of the tag. Some of the approaches to do the positioning includes; signal strength to range conversion and location fingerprinting based on RSSI, and Cell-based Positioning (Fu and Retscher, 2009). For the first approach, logarithmic and a simple polynomial model (or trilateration) is use to convert the signal strength to distance. In the fingerprinting approach, there is a training phase where RSSI is measured at known location of interest and stored in a database and later used with current measurements to estimate the current location. (Fu and Retscher, 2009)

The Cell-based Positioning, an RFID tag's location is estimated by choosing the cell in which the tag has the highest RSSI value. It is also known as the Cell of Origin (CoO) (Fu and Retscher, 2009). This approach works in situations where just the knowledge of which cell an item is located is sufficient and accuracy is not very important. In some situations, stating some thresholds outside of which the measurements are rejected is necessary. This is useful if there are some prohibited areas whose location is not needed. The volatility of the RSSI values can lead to misestimating of the location since RSSI values can change, due to slight changes to the environment such as when big vehicles passes by. In this case, machine learning algorithms could be employed to design an adaptive system to changes in the environment. The cell is identified by the stationary RFID component; it can be a tag or a reader.

2.3.2 Long Range IoT Communication Systems in Unlicensed bands

Wireless communication looks to be the most feasible and cost effective means to have the end devices connected to the internet for most of the applications and services. In fact, the IoT paradigm does not impose any restriction on how end devices should connect to internet. The most popular long range communication technologies for IoT include SIGFOX, Ingenu and LoRa. (Centenaro et al., 2015)

SIGFOX

SIGFOX technology is the oldest among these three technologies and the first to be suggested for IoT systems (Centenaro et al., 2015; Vangelista et al., 2015)). It was founded as early as 2009. Some of what is known about SIGFOX is that it uses Ultra Narrow Band (UNB) wireless modulation on the physical layer but most of technical details are closed from public. SIGFOX as a company, has positioned itself as a network operator (Centenaro el at., 2015) for IoT services and it has already SIGFOX network coverage in countries including UK, France and Spain (Vangelista et al., 2015). The technology supports bidirectional communication between the end device and the aggregator. The range that SIGFOX technology covers is between 30-50 km in less congested areas and 3-10 km in built up areas (Centenaro el at., 2015; Vangelista et al., 2015).

Ingenu

Ingenu is another upcoming long range communication technology on the block, available for IoT solutions and M2M communication. It was developed by On-Ramp Wireless based in San Diego (USA). The physical layer is based on the proprietary Random Phase Multiple Access (RPMA) technology and the technology can be deployed on other networks. The robust design of the technology on the physical layer allows the use of the 2.4 GHz band to operate over long range wireless links. (Centenaro et al., 2015; Vangelista et al., 2015)

LoRa

LoRa is a modulation scheme or a physical layer designed and patented by the Semetch Corporation - an analog and mixed-signal IC maker. It is derived from Chirp Spread Spectrum (CSS) with some phase tuning on symbol properties of the preamble to achieve better timing and frequency synchronization, which in turn allows a stable local clock on the LoRa tag without strict requirements on quality of components used. (Centenaro et al., 2015; Vangelista et al., 2015). In other words, data is encoded from the extent of frequency changes over time (Ian Poole ,2014). Ian Poole (2014), describes LoRa as a spread spectrum modulation technique that utilizes wideband linear frequency modulated pulses.

The technology achieves noise and interference reduction at the receiver by use of a spreading technique that sees the symbols encoded into a longer sequence of bits. The flexibility of the spreading code lengths allows for tradeoffs between coverage range, or link robustness, or energy demand against performance resulting from variable date rates. (Centenaro et al., 2015)

LoRa takes advantage of the unlicensed frequency bands that are usable all over the world, these includes 868 MHz for Europe, 915 MHz for North America and 433 MHz band for Asia. The choice of lower frequencies is to enable longer range and resistance to interference. It is worth mentioning that LoRa is not frequency specific, it can be used on many other frequencies with no serious changes. (Ian Poole ,2014) In fact, Semtech is already developing a chip that will use LoRa in 2.4 GHZ frequency band.

The LoRa employs an adaptive link in which different bandwidths are selected depending on the data rate and the link conditions. The available bandwidths include: 7.8 kHz; 10.4 kHz; 15.6 kHz; 20.8 kHz; 31.2 kHz; 41.7 kHz; 62.5 kHz; 125 kHz; 250 kHz; 500 kHz. It turns out that the choice of the data rate and the condition of the link plus some other factors also affects the power level adjustments. The power level adjustments are done in such way that fast communication is supported and battery life maximized without affecting network capacity. The choice of the data rate is a compromise between range and message time-on-air. (Ian Poole ,2014)

2.3.3 Reverse SSH Tunneling based remote control

The Secure Shell (SSH) Protocol is a protocol for secure remote login and other secure network services over an insecure network. The SSH protocol is made up of three main components each of which holding a particular role. The first one is the *Transport Layer Protocol*, responsible for server authentication, confidentiality, and integrity with perfect forward secrecy. The authentication of clients to the server is the responsibility of the *User Authentication Protocol* - the second component. And the third is the *Connection Protocol* that multiplexes the encrypted tunnel into multiple logical channels. (Ylonen & Lonvick, 2006; Egners et al., 2012)

The port-forwarding functionality of the SSH protocol support both local, dynamic and remote forwarding that allows it to redirect TCP/IP session over encrypted tunnels (Egners et al., 2012;). With the local port forwarding, all the traffic directed to local port is redirected to a server and a port through an SSH tunnel to a third node (Egners et al., 2012). The reverse of the local port-forwarding is the reverse SSH tunneling (Egners et al., 2012). In this case, the SSH client reverses the direction of traffic flow of the port on the SSH server where by packets from the third node to the port on the SSH server are redirected to a port to the SSH client.

Some other technologies that could be used to build a secure communication channel includes the Virtual Private Network(VPN) and the Secure Socket Layer (SSL) or Transport Layer Security(TLS) (Atighetchi et al., 2013) - the latest version of SSL. Unlike the SSH protocol that requires mainly only updating the server's port address to point to the tunneled port, the VPN may require special privileges for the operating system and utilizing it within a user application may not be possible still (Plesowicz & Laszczyk, 2013). The usage of SSL protocol may require a redesign if not a recompilation of all the applications (Plesowicz & Laszczyk, 2013) and also requires certificate which can self-made or obtained from the certificate authority.

The idea of reverse SSH tunneling has found use in different applications that requires remote access to clients behind a fire wall. For instance, Le Blond et al. (2012) and Holt (2014) used reverse SSH tunneling to establish a remote connection between a ZigBee Access Point and a remote collector server for a smart metering system. The beauty of using the SSH protocol is that, it uses public-key cryptography in which a key-pair is used to authenticate clients to server, where by a server can verify that a client is who it claims to be.

Creating a Reverse SSH Tunnel

There are situations when a remote secure connection to a device/server behind a private network is needed. Usually, the IP of the remote device may not be known to connect to it via SSH directly or if it is known, it is behind a firewall that renders it unreachable, presumably. This is when the Reverse SSH Tunneling comes in to facilitate a reachable host to forward a connection to a client that otherwise could not be reached directly.

Based on Powers (2014), the following explains how a Reverse SSH Tunneling could be configured: With a known IP address of a server, a remote device could be configured to open a socket to this server and instructing the server to forward all packets directed to this specified port to the remote device. For example, (Powers, 2014) assuming that the remote device and the server are running some Linux version of OS, a command:

```
ssh -N -R 2121:localhost:22 serverUserName@ServerIP,
```

instructs the server to open port 2121. Where; *serverUserName* is the user name of the device user to the server, *ServerIP* is the IP of the server, the N-Tag rules out the interactive shell when a connection is established and R is reverse connection. Now to connect to the remote device, a command like this:

```
sh -l DeviceUser -p 2121 localhost
```

would do the trick, but it would require a script that would keep this connection up by reconnecting when it goes down. Powers (2014) further explains how to configure *autossh* utility to create a secure and persistent reverse SSH tunnel to the remote device.

2.4 Web technologies and Frameworks

The overview of the web technologies that are relevant to the development of a Smart Parking system are given under this section. These sections present the background information on technologies, protocols and concepts covering areas such as the technologies that support sharing of data between different systems, technologies for responsive web applications and tools/Frameworks used for web software development.

2.4.1 Restful Application interface

Restful Applications are supported by the Representational State Transfer (REST) technique, which is a form of software architecture for distributed hypermedia systems, like World Wide Web (Herrero et al.,2015). REST approach is built on technologies including the Hypertext Transfer Protocol (HTTP), markup languages such as HTML and XML, web-friendly formats such as JavaScript Object Notation (JSON), and Uniform Resource Identifier (URI) (Kim et al.,2016). It is based on client-server architecture taking advantage of HTTP communication protocol (Kim et al.,2016). The REST has become a favorite across the Web as it is simpler compared to SOAP- and Web Services Description Language (WSDL)-based Web services (Rodriguez, 2008; Alarcon and Wilde, 2010), it is light weight and with high performance (Kim et al.,2016, Jeliazkova and Jeliazkov, 2011). The REST architectural style permits data to be accessed in ASCII format which allows data exchange by use of URIs (Stirbu, 2010; Wilke et al.,2015).

The client applications can interact with the remote system using standard commands such as the GET, PUT, POST, and DELETE from HTTP protocol (Stirbu, 2010; Herrero et al.,2015; Kim et al.,2016). The REST approach offers a number of important benefits for system scalability, it facilitates the visibility of the connotations of requests at the HTTP protocol layer. This makes a system easier to scale, optimize, and strengthen through the use of HTTP level appliances providing security, caching, and proxy capabilities. Furthermore, the absence of endpoints but the just a set of resource URIs and standard operations (Alarcon and Wilde, 2010) makes RESTful services easy to integrate as it requires just a handful of prerequisites to get a system up and running. It is platform and programming language independent (Jeliazkova and Jeliazkov, 2011), any language that supports HTTP and JSON or command line utilities, such as "curl" can easily utilize the REST APIs. (Wilke et al.,2015)

Despite the fact that most of the RESTful services use HTTP protocol, the REST setup itself is protocol independent, for that reason, REST systems that do not depend on HTTP protocol could be built (Jeliazkova and Jeliazkov, 2011). The REST APIs usage is simple, it just requires sending standard messages to the remote system indicating the URI of the requested resource, the method, the payload of the message and metadata (Alarcon and Wilde, 2010). For the RESTful services that use HTTP protocol, the system design involves identifying the domain object, linking the object to system resources and specifying the identifiers (URI designs) and operations (HTTP methods) on every resource (Jeliazkova and Jeliazkov, 2011). However, the downside of the REST procedure is the one-way interaction initiation, that is, the communication follows the request-response pattern, where only the client can start the conversation (Stirbu, 2010).

2.4.2 HTML 5 and CSS3

HTML5 standard is the latest of the HyperText Markup Language(HTML) versions. It was released in October 2014 by W3C. The HTML is a markup language for describing documents in a way to make it possible for the Web browsers to deal with them intelligently (Anthes, 2012). HTML5 is not only a standard but it is also a collection of technologies, it changed the way the web develops, works, and how it is used (Anthes, 2012). The HTML5 is platform independent, it supports the creation of websites and applications for platforms including mobile devices and desktop computers (Park et al., 2014).

Many of the features that developer imported from additional add-ons or plug-ins are available natively to the HTML5, features such as parallel processing, data storage and real-time communication (RTC) (Nurminen et al., 2013). The HTML5 and its RTC have been used in an experiment where Nurminen et al. (2013) proved the feasibility of a peer-to-peer video streaming between browsers with no need for external plug-ins. The support for client-server interacting application implementations is possible with HTML5, something that was a nightmare with other HTML versions (Park et al., 2014).

Among the most useful features availed by the HTML5 includes the WebSocket, which permits the exchange of date in real time using an always-on TCP connection. The mechanism is that the WebSocket protocol uses HTTP to establish a connection but the rest of the communication is handled by the WebSocket reader protocol. The benefits of this WebSocket is a smaller feature header which in turn reduces the communication overhead. (Park et al., 2014) The presence of CSS3 empowers the developers with the ability to develop User interfaces natively in the browser (Anthes, 2012) using HTML5. The support for the responsive web design means that developers can use HTML5 and CSS3 to produce applications that runs on any platform without worrying about the end device's screen size (Yeh et al., 2015). LaGrone (2013) teaches how to develop responsive websites that meets current mobile internet devices. He demonstrates how intuitive UI could be implemented using HTML5, CSS3 and JQuery.

2.4.3 Spring Framework

The Spring Framework is an open source application framework and inversion of control container for the java platform. The framework avails a complete programing and configuration model for present-day java-based enterprise applications and supports RESTful web services development (The Spring Team, 2016; Armando et al., 2014). The spring framework not only supports the development of Java applications but, with some extensions, web applications can also be built on top of Java EE platform. It is a single place for Java based application on all layers including one tier- stand-alone java application, web tier- in web application and enterprise tier tier- Enterprise Java Beans (Munsi et al., 2014). Its modularity allows choosing only those modules required for an application at hand.

Some of the features that Spring framework offers includes Dependency Injection (DI), Aspect-Oriented Programming (AOP) including Spring's declarative transaction management, Spring model-view-controller (MVC) web application and RESTful web service framework, Foundational support for Java Database Connectivity (JDBC) – for interacting with relational databases, Java Persistence API (JPA) – an Object-relational mapping (ORM) layer on top of databases, Java Messaging Service (JMS)- for queue/messaging, to mention but a few (The Spring Team, 2016; Soni, 2015). Figure 11 shows the components that permits the Spring framework to put available a range of services for developers to utilize. The components are grouped into seven categories, Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test just as shown in Figure 11.

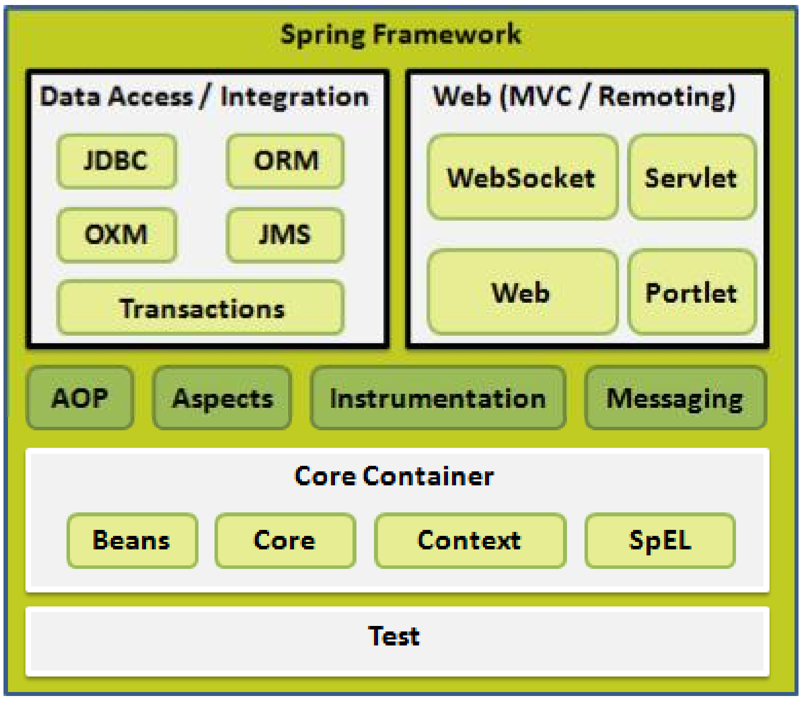

Figure 11: Spring Framework Architecture. [Reprint from (The Spring Team, 2016)]

Core Container
This category hosts modules that provide the essential parts of the framework including the IoC and Dependency Injection features. The support for integrating common third-party libraries into a Spring application context for caching, mailing, scheduling and template engines is made possible by this part of Spring framework.

Data Access/Integration
This is responsible for handling the common challenges developers face while working with databases in applications.

Web
This bring all the support needed to build robust and easy to maintain web application in a simplified manner. The web provides the MVC framework which provides model-view-controller architecture and ready components that can be used and loosely coupled web applications. The web-socket module for the WebSockect-based two-way communication between client and server in web applications also belongs to this category. The MVC provides a clean separation between binding request parameter, business objects and controller logic.

The MVC works in such a way that the HTTP request from the client in received by the core controller (DispatcherServlet) which looks up for the HandleMapping based on the URL to dispatch the request to the appropriate Controller. The Controller calls the necessary business logic to process the request, and the view name and ModelAndView object are returned to DispatcherServlet at the end of the process, and eventually by ViewResolver. The returned ModelAndView is rendered to the corresponding view. (Zhang et al., 2013)

AOP (Aspect Oriented Programming)
This module introduces the aspect-oriented programming implementation to the spring framework, permitting developers to write codes that have cross-cutting concerns handled cleanly by the framework. The AOP module make it possible to add new functionality to the existing code without changing its design (Munsi et al., 2014; Soni, 2015).

Aspects
This module is responsible for the integration with AspectJ which is another AOP framework.

Instrumentation
This module avails the class instrumentation support and class loader implementation that is useful to some application servers.
Test
The Unit and integration testing of the Spring components using Junit or TestNG are made possible by the Test module. The module also gives the developer the ability to test the code in isolation using the mock objects (Soni, 2015).

2.4.4 Django

Django is a free and open source full-stack framework with features covering all needed modules to develop a fully functional web application. It makes the building of the web applications easier and faster with fewer code (Taneja & Gupta, 2014), where by, in a matter of hours a developer can progress from concept to web application launch (Django, 2016). It is widely described as "the web framework for perfectionists with deadlines" (Taneja & Gupta, 2014; Plekhanova, 2009). Django is based on Python, one of the most popular web programming languages and one of the well documented, from which also Django derives its strength (Plekhanova, 2009). To demonstrate the spread of Django's popularity, Djangosites.org showcases websites powered by Django and, at the time of writing, there was 5080 websites listed.

With its inbuilt server, web applications can be developed and tested at just a click of a button. Some of the features that Django offers includes templates, support for rational database model, comprehensive security and many more (Taneja & Gupta, 2014). The Databases supported include MySQL, SQLite, Oracle and PostgreSQL (Taneja & Gupta, 2014). Django offers tight security and helps developers circumvent many known security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking (Django, 2016). Some of the security projections are achieved by its support for the management of user accounts and passwords through its user authentication system.

The possible reason for Django to be labeled as the web framework work for perfectionists with deadlines is probably due to the fact that Django does most of the heavy lifting, handling most of the common web development tasks including user authentication, content administration, site maps, RSS feeds, just to mention (Django, 2016). With all the heavy lifting done, the developer is left to focus on writing the application with no need for reinventing the wheel.

The Django framework uses the Model-Template-View(MTV), a variation of the MVC pattern. The Model works just like in the typical MVC. The Template produces the final HTML presented to the user by combining the HTML elements and data. The View gets the user inputs (HTTP request), they access the necessary models, and if needed some logic added, and then pass the data to the template. The template is rendered and returned to the user.

2.5 Machine Learning

Machine learning is about teaching the machine how to think. Alpaydin (2014), defines machine learning to be that act of programming computers to optimize a performance criterion using example data or past experience. A computer would be able to predict results of data not known before without being explicitly programmed. There are mainly three ways through which a machines can learn; through the supervised, unsupervised, or reinforcement learning. In reinforcement learning, a machine has to learn the sequence of correct actions to the desired outcome. In supervised learning, the machine learns the mapping between the input and the output using the correct "answers" given. But in the unsupervised learning, the machine derives the mapping from only the inputs given. (Alpaydin, 2014)

The problems to solve usually falls into two categories; one whose output(s) is discrete, called a classification problem or the one whose output is continuous, called a regression problem. A Classifier solves classification problems and the regressor handles regression problems. Clustering is an example of unsupervised learning that allows machines to learn from given data and then group together instances of data that look similar (Alpaydin, 2014).

CHAPTER 3

3.0 METHODS

The execution of the project had three phases, the first phase dealt with the validation of the hypotheses derived from the problems conceived. The second phase was recruiting the participants to the pilot and the third phase was the implementation and installation of the system. The second and third phase were executed in parallel since the two required different resources and were independent. A qualitative research approach was used for the first two phases of the project.

Qualitative approach is suitable for studying and understanding complex phenomenon by analyzing data gathered via interviews, surveys, and among other approaches. Studying complex phenomenon also requires the researcher to collect data from real world environments and situations (Seaman C., 1999). Saldana (2011) defines qualitative research as a broad term that encompasses a collection of approaches and methodologies including interviews, surveys, ground theory, and etcetera. Yet Strauss and Corbin, 1990) calls it "a type of research that produces findings not arrived at by statistical procedures or other means of quantification".

3.1 Qualitative approach

This research has one of its aims is of studying the problems drivers in Helsinki area face, through gathering information from real world environment. In this case, it was done by conducting interviews and qualitative research approach was used for the first two parts of research work. In the second part, survey was used to recruit participants to the smart parking system pilot project conducted at Katajonokka, an island located in Helsinki area.

3.1.1 Interviews

The interviews were conducted to find answers to the first two research questions. The first question seeks to know the challenges associated with parking in Helsinki Area, while the second question aims to assert that really a smart Parking in Helsinki Area will offer some advantages to drivers.

The recruitment of participant was based on the characteristic of the target group. Invitations were sent asking them to take part and interviews were scheduled based on their responses. Skype and phone interviews were used with those whose schedule changed abruptly. There were two face-to-face, one Skype and three phone interviews. The backgrounds of the participants are shown in the Table 1.

3.1.1.1 Procedure and Data Collection

One of the interviews was carried out from a place we chose. Here the interviewee was invited to the room where the sitting arrangements was as shown in figure 12. For the second interview, the participant made invitation to a place of his convenience and the same sitting arrangements as shown in figure 12 was used. The interview took 7 to 10 minutes. At the start of the interview, the aim of the interview was explained and participants were informed about the intentions. The intentions were not to examine them but to understand the challenges associated with parking in Helsinki area. They were also informed that there was no correct or wrong answer. The first column of the table shows the code name we gave to the interviewee in order from the first to the last interviewed. The second column shows the relevant details about the interviewee.

***Table 1**: Details of the participants*

Interviewee	**Details**
Interviewee 1	Age group: 35+ Gender: Male Residence: Helsinki Driving skills: Experienced Drive Frequency to Helsinki Area: Once a week
Interviewee 2	Age group: 30-35 Gender: Male Residence: Espoo Driving skills: inexperienced Drive Frequency to Helsinki Area: Twice a week
Interviewee 3	Age group: 25-30 Gender: Female Residence: Espoo Driving skills: Experienced Drive Frequency to Helsinki Area: 2-3 times a week
Interviewee 4	Age group: 25-30 Gender: Male Residence: Espoo Driving skills: inexperienced Drive Frequency to Helsinki Area: Once a week
Interviewee 5	Age group: 20-25 Gender: Female Residence: Espoo Driving skills: Inexperienced Drive Frequency to Helsinki Area: Not regularly
Interviewee 6	Age group: 30-35 Gender: Male Residence: Espoo Driving skills: Experienced Drive Frequency to Helsinki Area: Once a week

Data was collected by conducting structured interviews. The target groups for this research work were people travelling by car in Helsinki area. That included native people and tourists. For studying challenges in parking in Helsinki area with tourists' perspective, international students who travel by car to Helsinki area were interviewed.

Two types of Interviews were used; the face-to-face and phone interviews. The face-to-face interview set-up was as shown in figure 12. A similar arrangement was used for a skype video calling interview. The setup for the phone interview was in such a way that the phone was in front but equal distance from each host.

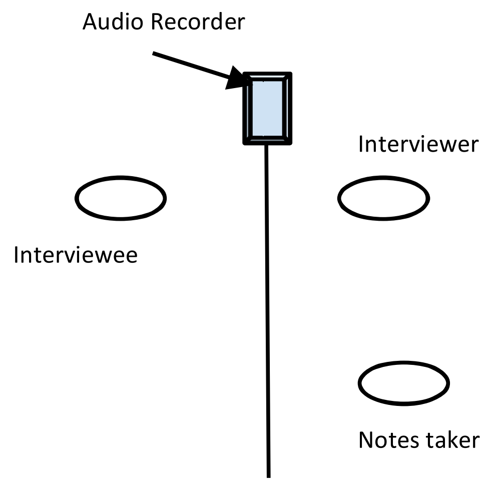

***Figure** 12: Face-to-face interview set-up*

All the interviews were conducted in a pair, where one person was interviewing while the other took notes, as illustrated in figure 12. An audio recorder was used too, to capture all the discussions during the interviewing process. At the end of each interview, the interviewer and the one taking notes discussed the event to interpret the collected data. This was done when they were still fresh and documented their conclusions.

The stratified sample that was used as the target group was divided into subgroups. The reason for this was to have the participants cover the different characteristics of the target group as widely as possible. Out of the total six interviewees, two of them were female and four were males. Three interviewees were aged between 18- to 29 and three were 30-45 years of age. Furthermore, three of the interviewees were new and inexperienced drivers whereas the other three were experienced drivers.

Interview questions were as follows:
1. How often do you travel in your car in Helsinki area?
2. How do you find a parking space?
3. How is the experience of finding a parking space?
4. Do you use any technology to get help in finding out nearest parking lot?
5. How do you pay for the parking time?
6. Are you comfortable with the process used to pay for the parking?
7. Have you heard about the smart parking systems?
"Interviewer Explains Smart parking"
8. What is your comment on this smart parking system?
9. Would you use the system if it was availed and why?

3.1.1.3 Data Analysis

To ensure the quality of the analysis, immediately after each interview, the handwritten interview notes were transferred to a word file. The recorded audio file was replayed to ensure that the documentation was accurate and was not influenced by the researcher. The raw data such as interview notes and audio files were kept in one folder in google drive so that both the researchers can refer to it when needed. To answer the research question one, interviewees' comments about challenges associated in parking in Helsinki Area were extracted from interview notes and categorized under three main categories of problem area:
1) Finding parking space,
2) Paying parking fees, and
3) Costing of parking time.
The naming of the categories was adopted from the interviewees' responses. For instance, paying parking fees was chosen to be a category for research question one because several interviewees mentioned they had problems in paying parking fees.

To answer the research question two, interviewees' comments about the advantage of smart parking in Helsinki Area were extracted from the interview notes and categorized under three main categories:
1) Time Saving,
2) Convenience in finding a parking space and paying the parking fees, and
3) Favorable costing of the parking time.
The naming of the categories was adopted from interviewees' responses.

3.1.2 Survey

The recruitment of participants for the Smart Parking system pilot was conducted by the Helsinki city's project team in conjunction with Witrafi management team, using a survey. The survey was sent to all persons who reside or work at the Katajonokka Island. The maximum number of entrants desired were 300 people. The selection of the successful candidates depended on their response in the survey. For instance, to qualify;

- a respondent should own a car
- should have a Helsinki city issued parking permit that permit parking to Katajonokka
- no other family member should have qualified already

The distribution of the survey was done by uploading a link of the survey to the Helsinki city website and through news letters sent by Witrafi management team to all its stakeholders. More announcements about the survey were published in Helsinki Sanomat newspaper. When the survey was closed, respondents were contacted by emails to give them the verdict towards their participation to the pilot and availing to them information on how to proceed.

The pilot was estimated to last for three months and all the successful participants would be exempted from parking charges for the whole pilot period. They would receive a WinNode - an active RFID tag, to be left on the dashboard when cars are parked. It was also believed that some people would participate in the survey and later in the pilot because they wish to see parking getting better in the city.

3.2 Embedded Networking

The main reason behind Embedded Networking is to facilitate embedded device get connection to the internet and exchange data on a network (such as LAN or WAN) with a specific or standard protocols such as Ethernet or Internet. A concept cyber-physical system is used to describe a system that combines physical systems together with computing and communication core (Lee et al., 2015; Rajkuma et al., 2010).

The Smart Parking system has two main parts; the back-end and front-end. The back-end in this context refers to the part of the system that stores, processes data and provide information to users and decision makers. It consists of the presentation layer (the front end) and the data access layer (the backend) - when looked at from software engineering viewpoint. Similarly, the front-end refers to the hardware users interact with and that hardware that collects data and passes it to the data access layer. The front-end consists of two parts; the active RFID tag and the gateway. The active RFID tag was named WinNode and the gateway is the WinSpot.

The design of the system's front-end had the following constraints that I (the system engineer) had to respect:

i. Interoperability with other IoT systems
ii. WinNode should be as cheap as it is practically possible
iii. WinNode should be a Low power node with battery lasting for at least 2 years

The LoRa technology and LoRaWAN was chosen as it looks to be with the highest momentum in the area of IoT application. Furthermore, software and hardware supporting the technology are readily available. The LoRaWAN stack is free of charge and getting it requires no registration, no strings attached whatsoever. The next section explains how the system's architecture was developed from some of the already known architectures such as LoRaWAN and IoT Gateway Centric Architecture.

3.2.1 The Smart Parking System Architecture

The Smart Parking System architecture follows a modified LoRaWAN architecture presented in section 2.1.4. It also shares some similarities with the IoT Gateway Centric Architecture described in section 2.1.3, where by the Gateway corresponds to the M2M sensor Gateway and the M2M Gateway corresponds to the Back-end (Server) as shown in figure 13. One of the differences is that the Smart Parking System lacks the IP enabled devices that interacts directly with the server, instead the supported devices link through the gateway which transforms their packets to the format acceptable to the server.

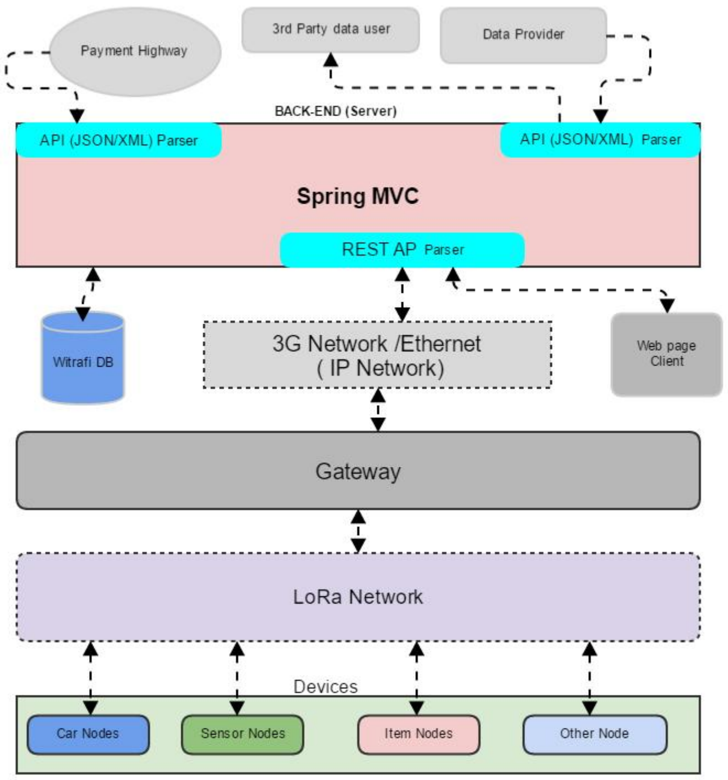

***Figure** 13: The Smart Parking System Architecture*

Unlike the LoRaWAN that includes a network server for routing packets to respective destination networks in case different users are being served by the same LoRa network, with the Smart Parking System, packets are routed by the gateway to the backend. The device layer supports LoRa based nodes that are able to interact with the server through the LoRa gateway using LoRa network. The device data could be from sensor node, item identification node, Car nodes (WinNode) or any other node based on LoRa MAC.

The gateway has two interfaces, the LoRa interface that communicates with the LoRa nodes and the IP network interface that interacts with the server. The IP network interface was implemented with 3G network. More on gateway is elaborated in section 4.2.3. A simplistic procedure description of how the Smart Parking System Architecture works, follows:

1. Data from the Car node is transmitted through the LoRa network to the Gateway
2. The Gateway adds on top of the packet the Gateway layer information, pack in supported format and add IP network information before passing it over to the server
3. The back-end parses that received packet using Spring functions for REST API parsing
4. After parsing the data, it is processed and the final logic might require data to be collected from the database after which the result is passed to users and/or stored in the database
5. A payment highway API is used to support payment functions of the Smart Parking System

CHAPTER 4

4.0 Implementation

This phase started with the design of the subsystems using block diagrams. The system block diagrams were presented to a second person with sufficient knowledge for both the system as whole and electronics in general. The aim was to get a second opinion, but ended up igniting some discussions which led to the discovery of some flaws in the system at this such an early stage. After a few revisions, we both agreed to move to the following stage of drawing the schematics. The seeking of a second opinion for the schematic files was also done and indeed some errors were discovered and corrected. The third stage was the PCB layout design, which is converting the schematic drawings to component footprints on which electronics components are mounted/installed.

The PCB layouts were also reviewed by a second person before sending them for manufacturing. To start with, 10 pieces of PCBs were ordered, assembled and tested to verify the working of a subsystem. During testing some errors were found and corrected. After three iterations, a working system was ready to be tested. Along the way the enclosures designs were also being iterated as the form factor is affected by the shape of the PCB and vice-vasa.

One of the roles I performed during the implementation of the smart parking system was to provide the hardware needed to build a platform on which the parking services would be delivered. There were two components I had to build; the WinSpot (LoRa gateway) and the WinNodes (LoRa Active RFID TAG). The following sections explains how the WinNode and WinSpot building was approached.

4.1 WinNode Design and Implementation

The WinNode (Active RFID TAG) is composed of a Microcontroller Unit (MCU), a sub-GHz radio core and a power supply as shown in figure 14. It is a 3.3 Volts system, so it requires a 3.3V power supply.

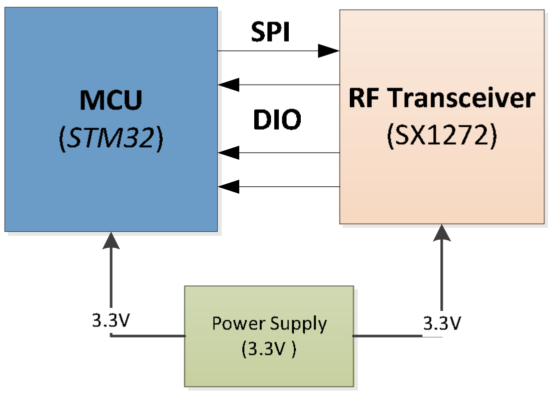

***Figure 14**: WinNode block diagram*

The communication between the MCU and the radio core happens via an SPI interface. The radio core interrupts are communicated to the MCU using the digital input/output pins (DIO). The MCU is the brain power of the node; it makes most of the decisions and interpretation and execution of the commands received from the gateway. The radio core provides an interface between the node and the rest of the LoRa network – the local network. The WinNode required LED light indicators and a push button for control.

4.1.1 The first working Version of WinNode

To lower to cost of the WinNode, more research and development on components was done and some options were chosen over others to achieve an optimum solution. The first decision was to use a printed circuit board (PCB) antenna for the LoRa system instead of other alternatives such as chip antenna, patch or helical antenna, etc. With a PCB antenna, once designed its costs are covered by the cost of the PCB manufacturing. A LoRa module was chosen for purposes of speeding up the development as it would take more time for board design iterations to have the WinNode conform to regulatory requirements. The module also removes the expert radio frequency(RF) knowledge required for RF designs.

The LoRa module hosts both the microcontroller and the radio chip with all the needed matching circuits. The design and implementation of the WinNode went through a number of iterations until a working version was realized. Figure 15 shows one of the working version of the WinNode. It uses a PCB antenna but also a provision for an external antenna was included using an SMA (SubMiniature version A) connector footprint. The LEDs (LED1 and LED2) and a button provide the user interface to the node user.

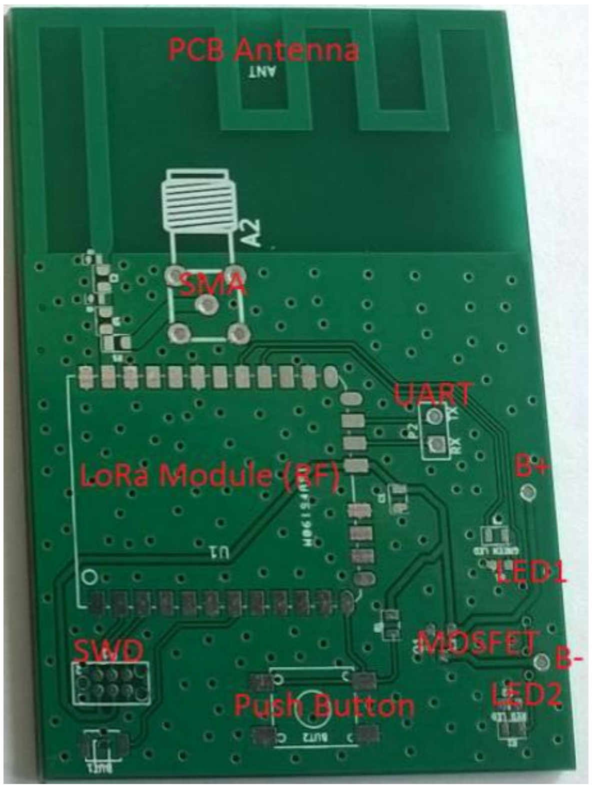

***Figure 15**: WinNode PCB and components.*

In comparison to figure 14 and figure 15, the LoRa module combines both the MCU and RF transceiver into one unit – the module. The module is soldered to an adapter board to allow the inclusion of other components including a button, LEDs, power supply and an antenna.

The Green(LED1) and Red (LED2) LEDs are used to signal to the user the status changes as communicated from the data access layer via the gateway. The user communicates to system using the button and web interface. The SWD (Serial Wire Debug) interface is used for reprogramming the MCU and the UART (universal asynchronous receiver/transmitter) was intended for logging some events at runtime for purposes of debugging the WinNode embedded software. The WinNode was designed to use 2xAA batteries in series providing a 3.0 Volts power supply. To protect the system from the dangers of reverse polarity, a MOSFET (metal–oxide–semiconductor field-effect transistor) was installed in series with the battery negative (B-) terminal. The MOSFET was chosen because I would cause lower voltage drop than if a diode was used. The next step is to have the PCB assembled in to a housing.

4.1.2 The WinNode Assembly and Enclosure

The posture of the enclosure was inspired the later "L" shaped figure. The reason behind the L-shaped came from the desire to mount the antenna in such a way that it would never be in close proximity with metal surfaces. The presence of metal surfaces causes the antenna to be detuned by changing its properties that could affect its efficiency, resonance, band width etc. By positioning the PCB with the antenna facing up, that risk could be reduced. Components on the PCB shown in figure 15 were assembled and the PCB installed in the casing shown in figure 16. The front plate was designed to be the button at the same time. The words are saying "*Press Here*".

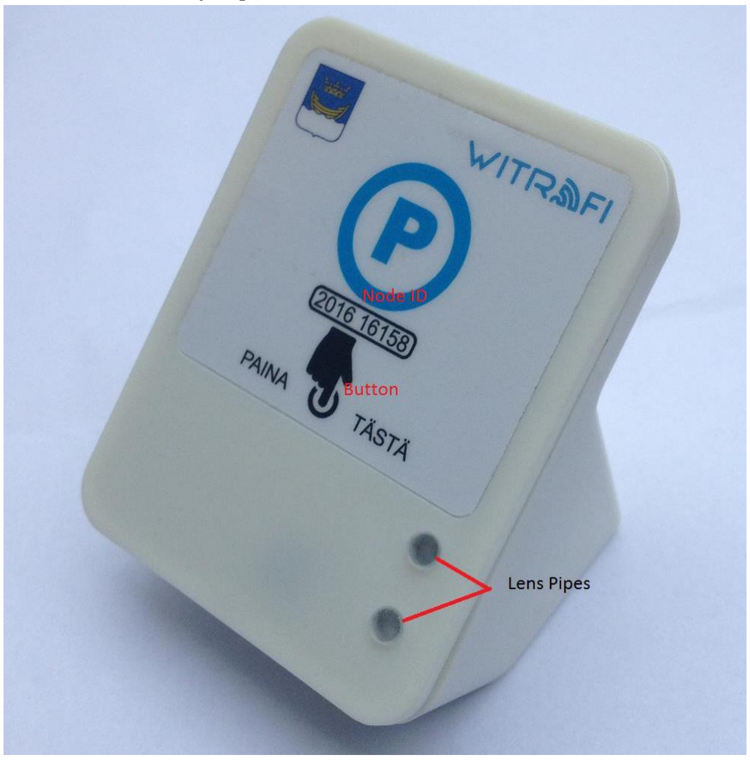

***Figure** 16: The WinNode.*

To increase the visibility of the indictor LED lights, lens pipes were added on to the enclosure. The usage and placement of the WinNode should be that it is turned on by placing the button, and then have it placed on the dashboard with the front plate facing the car windscreen. The reason behind having the WinNode face the car's windscreen is to make it possible for the parking wardens (parking enforcement officers) to see the identity(ID) of the tag – the Node ID. The ID of the tag is attached to the user and the car's registration number at the backend(server) and with it, wardens are able to tell which car has paid the parking fees. The car's registration number could be used too though.

The design of the enclosure was done with system's total cost reduction in mind. For instance, the parts making up the enclosure were designed in such a way that they could be manufactured by one mould that opens in one direction – no sliding. Furthermore, the parts themselves were designed in a manner that they would require no tool to assemble – the assembly functionality was embedded into the parts. This in turn reduced on the cost of WinNode production due to reduced cost of assembly.

The time for testing had arrived. This version was tested by loading it to maximum power the node could consumer while recording the time it is active. A LoRa nodes consumes most while in transmitting mode, it takes up to 125mA, so by programing it to send a continuous wave with all other peripheral on, it was possible to estimate its maximum energy demand. The main interest was to determine the WinNode's cut off voltage. To test this, a linear DC power supply with voltmeter and ammeter was used. A WinNode was powered to 3.3V from the power supply and then the voltage lowered until the stopped working. At that point the voltage reading from the voltmeter was the low voltage cut off voltage.

The cut off voltage was 1.8V but the WinNode stopped working properly when the power supply voltage went below 2.0V. It was clear that this limitation required a stabilizer to maintain a constant voltage despite the fluctuations in the input voltage.

4.1.3 Voltage Stabilizer for WinNode

The devices for IoT systems are required to live for year on a small battery without being recharged or replaced. The WinNode is no exception, it is expected to live on two AA batteries for at least two years. To achieve this target, the device has to squeeze out all the "juice" from the battery. The design shown in Figure 15 had a higher low voltage cut off that could not exhaust all the charge from the battery.

A 3.3V DC-DC converter shown in figure 17 was used to stabilize the output voltage of the system to 3.3 Volts, for the range of input volts from 0.8 Volts to 5.5 volts. This allows "sipping dry" the battery while maintaining the system voltage stable at 3.3V and the node can live for a longer time before the need to replace the batteries.

The same test for the lower cut off voltage as described in section 4.1.2, was performed and this time the cut off voltage had gone down as low as 0.8 Volts. This was good news as it meant that, almost all the battery charge would be drained during the operation of the WinNode in the field.

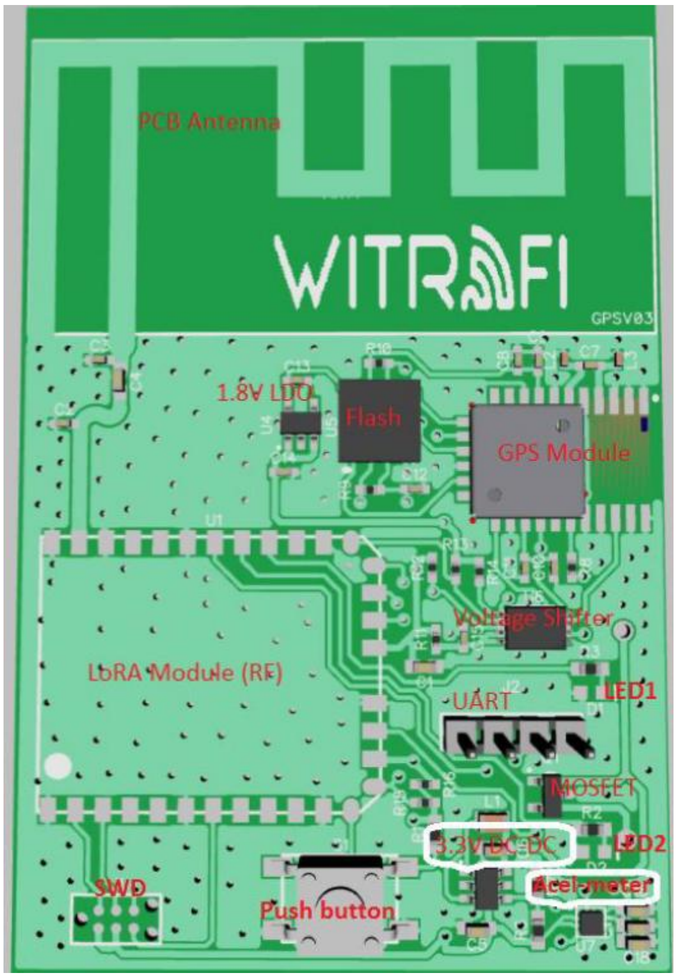

***Figure** 17: WinNode Board components – final version.*

After fixing the WinNode's energy draining limitations, the next step was to install on the WinNode the field software to facilitate field simulated tests.

4.1.4 The WinNode Embedded software

The embedded software development team was responsible for developing all the needed embedded software. The software running on the WinNode is based on the LoRa MAC Class A specification (V3.0), described in section 2.2.4 (the LoRaWAN protocol). The WinNode mainly just transmits its encrypted unique ID and the RSSI values are computed by the gateway. The WinNodes have some intelligence that allows them to figure out if they are in range or out of reach from the gateway, and automatically adjust their data rate to much their range. Usually, lower data rates are used for long ranges and higher data rates for short distances from the gateway.

The WinNodes were configured to use an adaptive data rate scheme (ADRS). Under this scheme, nodes automatically determine their connection status and adjust their data rates (DR) accordingly. The details of the data rates the WinNodes uses to switch between as defined by the scheme are as shown in table 2. The node starts with the highest data rate – DR5 and lower the data rate towards DR0 when it does not receive a response for its transmission.

***Table 2**: The data rates used by the WinNode*

Data rate #	DR0	DR1	DR2	DR3	DR4	DR5
Bandwidth	125 kHz	125 kHz	125 kHz	125 kHz	125 kHz	125 kHz
Data rate	293 bps	537 bps	977 bps	1758 bps	3125 bps	5469 bps

The WinNode software was developed in such way that it is possible to do some configuration updates remotely. The means by which a WinNode remote access in accomplished is explained in coming sections. Some of the functionalities such as ping frequency and LED behaviors of a WinNode can be altered using commands through a given downstream port. The WinNode is programmed with a unique identifier and it advertises it at a programmable time interval when it is first switched on. When the car is parked, the green LEDs of the WinNode blink very slowly. The reason for slow blinking is to conserve battery life. If the car is detected by the backend to be located in a wrong parking lot, the backend sends a warning packet through the gateway to the WinNode.

When a warning packet is received, the red LED of the WinNode blinks very slowly. The button of the WinNode can be configured to have several functions such as turning ON/OFF of the WinNode, initiating a parking event, opening of a barrier or a garage door, approval of payment for the parking, to mention but a few. The results from position estimation as explained in section 4.5.2 - WinNode Position Estimation Methods, indicated that there was room for some improvements. The next section explains how these improvements were approached.

4.1.5 Enhancing the positioning capability of the WinNode

Introducing a Global Positioning System (GPS) was to enhance the positioning capability of the WinNode. To address some of the challenges associated with Received Signal Strength Indicator (RSSI) based positioning, as explained in section 4.5.2, to the second version of the WinNode a GPS module was added. The presence of the GPS should reduce on the number of gateways installed as under this arrangement, a single gateway could cover at least a square kilometer. However, in places where the GPS is also quite inaccurate, the combination of the two – the GPS coordinates and the RSSI values could be used to estimate the position of the WinNode.

The added GPS module is shown in figure 17. The PCB antenna is designed for a fixed size of the board and hence limiting the flexibility to just expand the board size to accommodate more components when needed. To meet the limitation of the space, a small GPS module with on board antenna was used. Of course when a GPS in added, the energy demands of the system increases. However, if the GPS could acquire a fix fast enough, it could be switched off to save the energy. To achieve that, some Client Generated Extended Ephemeris (CGEE) files - calculated on the device and are required to be stored locally for the GPS receiver to use in estimating its position on Earth. The Server Generated Extended Ephemeris (SGEE) files were not used since the WinNode does not have internet connection for the files to be downloaded and stored.

The flash memory shown in figure 17 is used for storing those Ephemeris files. The challenge then, was how to integrate a 1.8V GPS module and flash to a 3.3V node. The solution to that was the 1.8V low-dropout (LDO) regulator and a voltage shifter shown in figure 17. The 1.8V LDO regulator takes in 3.3V system voltage and output 1.8 Volts for the flash and GPS module. To facilitate the intercommunication between 3.3V LoRa module and a 1.8V GPS module, a voltage shifter was used. Its operation is in such a way that one side is configured to 3.3V and the other to 1.8V. Communications from one side is converted to the voltage levels of the other side and vice versa.
The next task was to find ways that could be used to reduce on the energy demands of the WinNode to have the life span of the battery extended.

4.1.6 More WinNode Power saving using accelerometer

The cardinal use of the WinNode is for it to tell the backend system its location every time it changes. This allows the backend to start or to end a parking event and calculate the due parking fee, using the WinNode location in combination with the information received from other sources/systems. Keeping track of position of the WinNode would require the node to send its location at a regular interval, say every minute. Well, If the car moves within that minute, the back-end would not know until it receives a ping at the end of the minute. What if there is a mechanism to detect a start and stop of movement in which case the backend is updated only when this motion comes to a halt. This would mean that if a car is parked once a day the WinNode would only transmit once a day and the power saving would be enormous in this case. Using an accelerometer, shown in figure 17, tells if the car is stationary or in motion.

The accelerometer has an interrupt output pin that wakes up the main MCU, which is normally in very low power mode together with other components. The accelerometer could be configured to interrupt the MCU when the motion stops. When the MCU wakes up due to the accelerometer's interruption, it turns on the GPS to get the WinNode's current position and transmit it to the backend. At the end of the transmission, the MCU turns off all the peripherals and goes back to sleep and waits for another interrupt.

4.2 WinSpot (Gateway) Design and Implementation

The Version of the gateway that uses a Raspberry PI for a central processing unit was preferred for the WinSpot. It consists of four main parts; the 3G modem, the Raspberry Pi 2, the Concentrator and the power supply as they are shown in figure 18. The other option uses a microcontroller for a central processing unit to control other components making up the gateway.

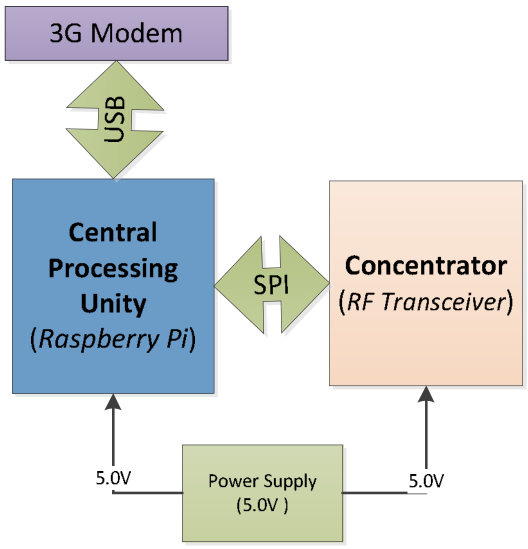
3G Modem
USB
Central Processing Unity (Raspberry Pi)
SPI
Concentrator (RF Transceiver)
5.0V
Power Supply (5.0V)
5.0V

***Figure** 18: WinSpot block diagram for Raspberry Pi Version*

The 3G modem is a USB dongle that provides networking capabilities to the LoRa network – implementing the backhaul IP stack shown in figure 10 in section 2.2.4. The Raspberry Pi runs a Linux distribution on which the gateway functionality was implemented using the open source LoRa gateway code base and libraries located at (https://github.com/Lora-net). The concentrator unit is based on Semtech SX1301 multi-channel modem and SX1257/SX1255 RF transceivers. The interaction between the central processing unit and the concentrator is through the SPI interface. WinSpot acts as gateway towards the IP-world.

To achieve easier interconnection between the WinSpot components, a raspberry Pi 2 connector, shown in figure 19, was designed. To be able to monitor the temperature of the gateway unit, a temperature sensor was added to the Pi connector board. The Inter-Integrated Circuit (I2C) interface facilitates the intercommunication between the Pi and the temperature sensor. The purpose of monitoring the temperature was to learn how the temperature fluctuations would affect the functioning of the gateway, especially during cold times like in winter seasons. The results of the temperature measurements were to help decide the necessity of installation inside the gateway enclosure a heater for winter and cooler for hot summer.

The hardware watchdog is responsible for resetting the system in case of a total system crash or when it hangs in some software subroutine loop. The external hardware watchdog used has watchdog service time from 1ms to 3 minutes that are configurable by hardware. The three minutes service time was chosen. The main system has to service the watchdog by toggling an input/output pin before the set reset time elapses. The Pi connector header has other interfaces made accessible that makes attaching other subsystems smooth.

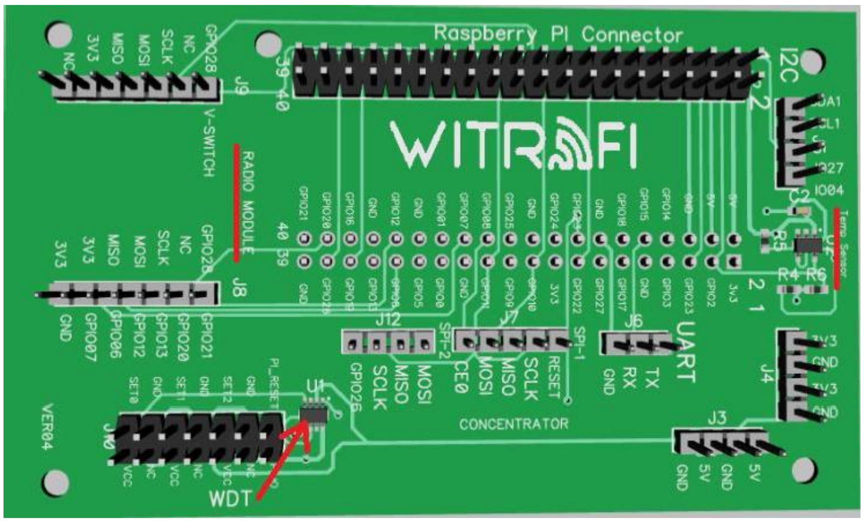

Figure 19: Raspberry Pi 2 connector

The interfaces include Serial Peripheral Interface (SPI), and Universal Asynchronous Receiver/Transmitter (UART). An expansion header for another radio module was also added whose use is still an ongoing research on how to use more than one LoRa transmitter on a single gateway.

4.3 System components inter-communication

The data flows through the different components of the system, from the data source to the destination. Ideally, there is the source of the data, the channel with/without repeaters/relay and the destination. The smart parking system has the WinNode and server interchanging roles for a source and a destination and the WinSpot is a relaying the data between the two. The following sections explains how data moves from source to destination with in the smart parking system.

4.3.1 WinNode-to-WinSpot communication

The WinNode and the WinSpot both use the LoRa network to communicate to each other. The concentrator has the capability to scan and receive thousands of communications from the WinNode within a matter of seconds. The WinNode uses up-stream communication to send packets to the WinSpot and the WinSpot uses the down-stream communication to respond or contact the WinNode.

When the WinNode sends a packet to the WinSpot, it goes to sleep mode and wakes up later, after a programmed waiting time, to listen and receive a packets. If no packet is received, it retries 3 times and if nothing is received it sleeps and retry after typically either 15 minutes or two minutes.

4.3.2 WinSpot to server communication

The parking facility, region or zone profile is directly linked to WinSpot profile. Depending on the size of the facility, more than one WinSpot profiles might contribute to a single facility profile. The profile is composed of the Nodes IDs the WinSpot (s) has read. Every node the WinSpot reads is added to its profile and if the node has not advertised itself for about ten minutes or so, it is cleared from the profile – it is assumed to have left the parking facility. This applies to WinNode version without an accelerometer. As with the accelerometer, the WinNode has to report changes in motion – stationary state or moving state.

Nonetheless, some pings to the backend are required to keep in touch just in case the backend has some queries to the WinNode.

The WinSpot is registered as a user to the database, this way it is easy to create and manage profiles. The authentication and authorization of the WinSpot is based on the fact that it is a user. When the WinSpot sends packets to the database, the backend first authenticates it before registering any transaction to the database. Therefore, for that matter, the WinSpot packet must always contain an authentication/session token depending on scheme chosen by the systems administrator. The public private key pair scheme is also used for authentication.

When the WinSpot receives a warning packet response from the server, the target WinNode ID is extracted and via unicast, the specified WinNode is contacted. The WinSpot waits until the WinNode contacts it through the regular pings to forwards the packet from the backend.

4.3.3 Server-to- WinNode communication

The architecture being a request/response, the server has to wait until the client – the WinNode, contacts it for the server to present its queries. The server knows that every contact the WinNode makes to the backend, there are receive windows in which the node waits for downstream communication. The server sends its queries to the WinSpot which in turn forwards them to the WinNode during one of its receive windows.

4.4 WinSpot Installation and Deployment

Although the main requirement of the system was just to be able to tell if a car is parked within Katajonokka area, it was of great interest to be able to specify the part of Katajonokka the car is. The area is just about one squire kilometer and just one LoRa gateway would be enough to have a total area coverage. The first approach was to have RSSI base positioning, and having planned a cell-based design, more gateways(WinSpots) were needed and their locations are as shown in figure 20.

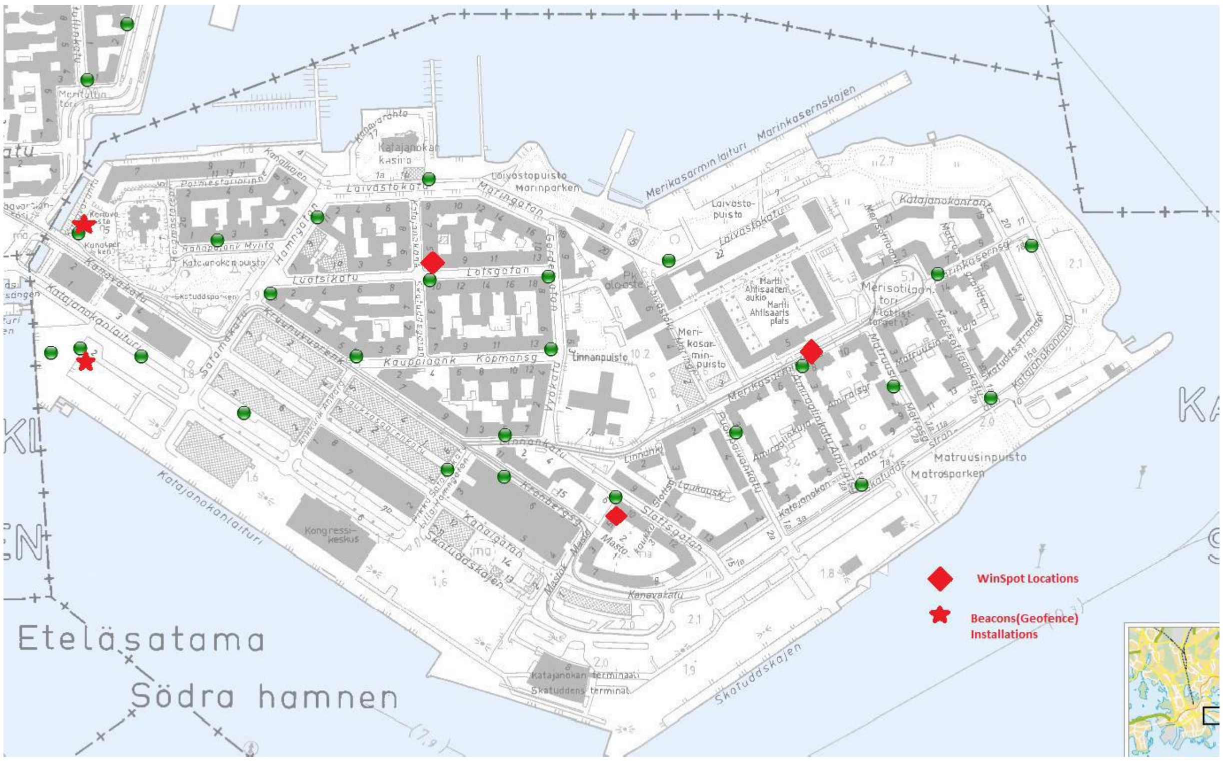

***Figure 20**: WinSpots Installation points at Katajonokka*

The beacons/Geofence were installed at the boundary of the area of interest to be able to know when a car leaves or enters the region. This was based on the fact that it is possible for a WinNode across the other side of the area of interest to have the same or even higher RSSI than the one inside the area. The beacons would be transmitting packets at a very low power, limiting their coverage within about ten to fifteen meters. The green spots shown in figure 20 are the ticket machine locations. Their location gives an idea where the parking slots are positioned. The distance between the closest parking spots from the two areas is about fifty (50) meters and by limiting the beacon coverage to at most fifteen (15) meters, cars from the other side should not pick up beacon transitions.

The WinSpots were installed on poles, about three and half meters tall. The poles were fastened to the ticket machine and the power for the WinSpots was tapped from socket provisions located in the ticket machines. At the end of each installation, the WinSpot was turned on and tested to make sure that it connects to the backend.

4.5 Backend Implementations

The implementation of the backend system had two parts; the pilot version development and the proof of concept. These two parts were handled by two different teams. The proof of concept team used Django whereas the spring framework was used to develop the pilot version. The proof of concept version was designed to learn how the WinSpot sends and receives data from the backend.

4.5.1 Using Django for a proof of concept

The main objective of the proof of concept was to test how the intercommunication between the gateway and the server could be accomplished. A database was designed and implemented in a python based framework – Django. The design of the database shown in figure 21 was based on the idea that the position of a car could be determined by the location of the CarTag (WinNode) using trilateration. Under this approach, a WinNode should be read by at least three AccessPoints (WinSpots) for the location of the node to be estimated.

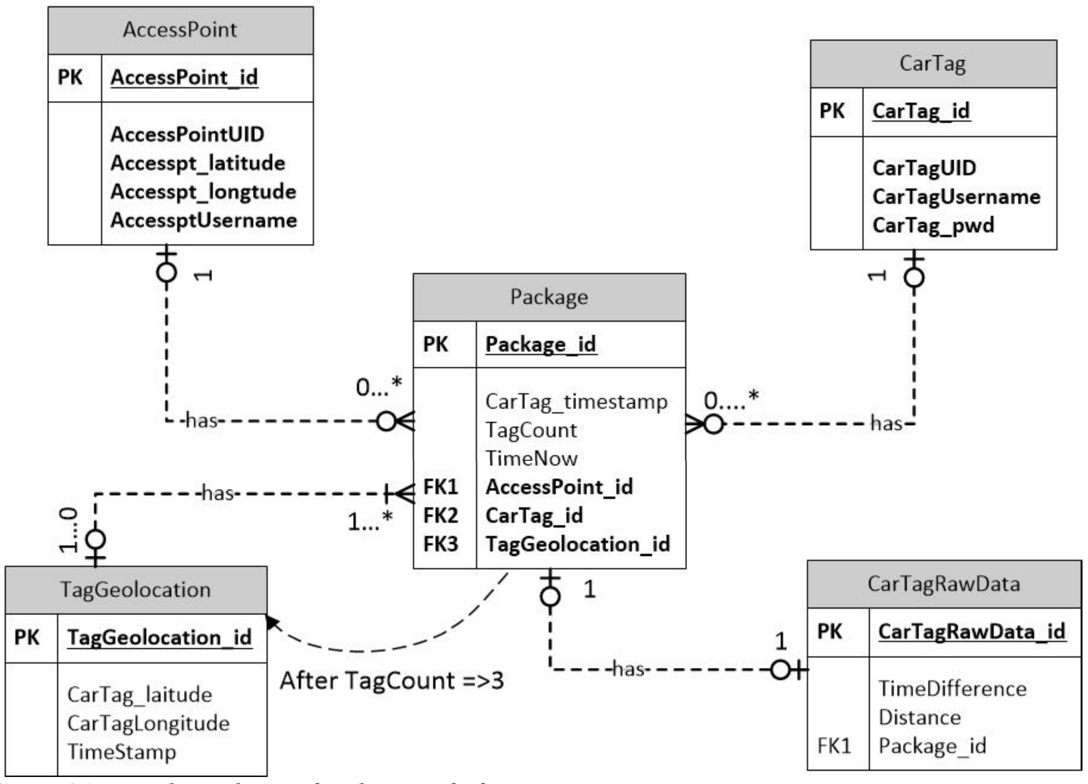

***Figure 21**: Database design for the proof of concept*

The database has five tables with the *Package* table linking all the other tables. The AccessPoint table stores the Access point unique Id, its user name and locations as latitude and longitude. The CarTag table stores the unique id of the tag, the user name and the password hash. The CarTagRawData table stores all the raw data used to estimate the tag's location which includes time of flight, distance and the package Id. The last table, the TagGeolocation table keeps the coordinates for the location of the tag and the time stamp at which the location was determined. The pilot version has some modification from this, as RSSI values and GPS coordinates were used.

4.5.2 WinNode Position Estimation Methods

There are two approaches for estimating the position of the WinNode that were tested during this project, the first one was the cell-based positioning where RSSI values were used to estimate the cell in which the WinNode was located. There were two ways of estimating these cells; the first one was similar to finger printing but not exactly. A person with a WinNode moved around to the furthest points of the cells while recording the RSSI values at those points. This task was designed in such a way that a known WinNode was used, when the button was pressed, a packet was sent to the backend which recorded the WinNode ID, Time stamp and RSSI value. At the same time, the person carrying out the task recorded the location and the time of button press. From the results, the lowest RSSI value was used as the cut off value for that cell. This means that when a WinNode pings to the backend, it would only be placed to a particular cell if its RSSI value passes the cut of limit of that cell.

The first method suffers some drawbacks, especially when the environmental conditions changed the threshold RSSI values could not be changed. The second method applied machine learning algorithms to achieve an adaptive system that can respond to changes in the environment. The Radial Basis Function Network (RBFN) in particular, was used for its ability to carry out classification by measuring the homogeneity between the data. The algorithm takes as inputs the WinNode RSSI values to perform clustering and returns weights assigned to different WinSpots. The weights tell how likely the WinNode belonging to the cell represented by the WinSpot ID. The WinNode is placed to the cell with the highest weight. Figure 22 shows the improvements a machine learning algorithms introduces.

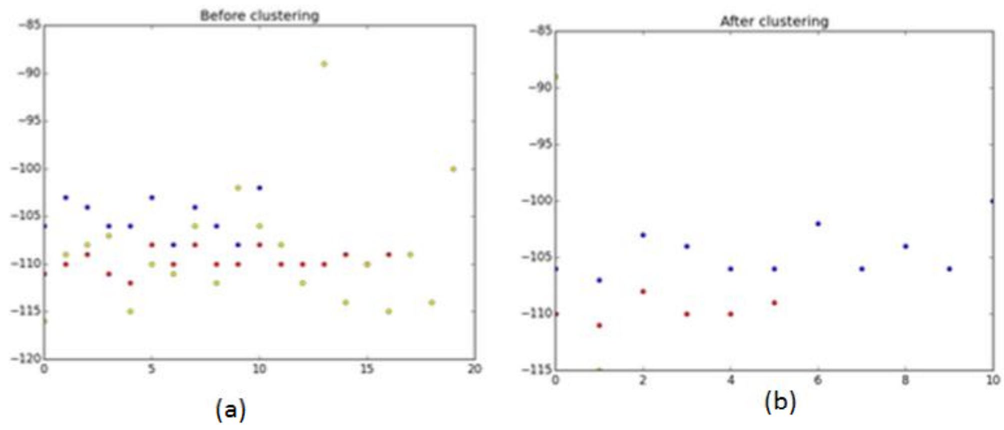

***Figure 22**: Cell prediction using machine learning [Source: Witrafi Oy]*

In figure 22 (a), there are three cells to which a node is predicted to belong, with each color representing a cell in a given gateway resides. Figure 22 (b) shows the result of the clustering algorithm with distinct boundaries. After clustering it is clear that the node belongs to cell coded with blue color code.

The second approach used for estimating the position of the WinNode was the use of GPS coordinates. When a car is parked, the GSP coordinated are read from the GPS module and together with the node ID they are transmitted to the gateway. The gateway reads the RSSI value and add it the packet together with its ID. At the backend, the WinNode positon is estimated from the GPS coordinates. But in case the WinNode is located in a place where the GPS coordinates cannot be reliably obtained, the RSSI values are used to estimate its location.

4.6 Smart Parking System area coverage test

The tests for the area coverage were aimed at validating the hypothesis made when choosing the technology. The location of the pilot is a densely built up are with many tall building that are expected to abstract and or absorb radio signals. But the LoRa technology promises higher interference immunity and better concrete penetration. Worse still, the technology is now being used an application that requires the node to be inside a car – something made out of metal, so expected to affect the range/coverage.

The test setup to have the coverage test made was designed in such a way that when a button on the WinNode is pressed, it sends a packet request for a response, the WinSpot in range responds. And when the WinNode receives the response packet, it blinks the green LED. The test was carried out by having a WinNode in a car and driving around the pilot area. One person was driving the car around parking places considered obstructed while the other pressing the WinNode button and recording the coverage on the map. During the button pressing, the WinNode was placed on the dashboard to mimic the operating position of the node. After covering all the planed points from the pilot area, the testing also continued out of the pilot area just see how far the coverage could go.

CHAPTER 5

5.0 Results

This section presents the research and measurement results for the different phases of the project. The interviews produced results answering research questions 1 and 2. The survey presents the outcome of the recruitment process for the participants to the project pilot. Some measurement results aimed at checking the performance of the chosen technology, such as coverage is also presented.

5.1 Results from the interviews

The results from the interview confirmed the existence of a number of challenges associated with parking in Helsinki area. There are three key problem areas that were identified; finding parking, paying parking fee and costing of the parking time. The key problem areas and their associated problems are listed in Table 2. The same results also identified some of the benefits interviewees think they would gain from using a smart parking system, if it became operational. These benefits included time saving, convenience in finding a parking space and paying the parking fees, and the favorable costing of the parking time. These advantages are shown in Table 2.

All interviewees pointed out that, finding free parking space in Helsinki area is real challenge. Four interviewees described it as a frustrating and a time consuming task. "It's frustrating, you need to drive a lot of back and forth", interviewee 1 recounted. On the challenges related to paying the parking fees, not having coins turned out to be a big obstacle as some vending machine requires coins for parking tickets. One interviewee narrated the frustration, "It's difficult, I have to look around to find coins and sometimes I go to shop, buy something, and get some coins from the shop". It was noted that 100% of the interviewee had a complaint about having to pay for the whole period for which the ticket was bought even though they, very often, leave before the paid time expires. Table 2 shows some of the key problem areas associated with parking in the Helsinki Area.

***Table 2**: Key challenges associated with parking in Helsinki Area*

Problem Area	**Problems**	**Interviewees' Comments**
Finding parking space	- Time wastage - Fuel wastage - Drivers' frustrations	- Time consuming - Frustrating - Not convenient - Feels like trial and error - Waste of fuel
Paying parking fees	- Inconvenient payment method	- Not convenient because you do not have coins always
Costing of parking time	- Expensive	- Feels like cheating because most of the time you pay more than you use - Minimum charging is too much

The data collected from the interview suggested that a smart parking system could solve most of the problems associated with parking. After explaining the concept of smart parking, all the participants were like, when will the system hit the shelves? A belief that the smart parking system would reduce, if it does not totally eliminate, the back and forth driving looking for the free slot and hence saving time and fuel, was evident from all the people that were interviewed. "I don't have to remember when my time will expire", interviewee 3 excitedly exclaimed. And interviewee 6 was happy that there would be no more need for having coins. It was vivid, from the interview that smart parking system could solve the existing problems and that there is a sheer need of such system. However, cost of using smart parking system will be a deciding factor for some users as interviewee 2 clearly mentioned "Yes, but I have a question about how much it costs. If it costs way more than manually trying to find a free parking space, I would not use". Table 3 highlights some of the key advantages of smart parking system pointed out by interviewees.

***Table 3**: The advantage of smart parking in Helsinki Area*

Advantage of Smart Parking	**Interviewees' Comments**
Time Saving	- Saves time and fuel
Convenience in finding a parking space and paying the parking fees	- It would be very handy - Sounds very convenient - I don't have to remember when my time will expire - No need of having coins
Favorable costing of the parking time	- Can pay only for time you park - Don't have to pay extra

As can be seen from the comments in Table 3**,** participants believe that smart parking system can solve the key problems they are facing today. Apart from these comments, participants showed their excitement and willingness to use the system if such system would be made available.

5.2 Results from the Survey

The survey was used to recruit participants to the pilot and the target maximum number that was needed was 300 participants. The response to the survey was below our expectations but enough to support the research the pilot was intended to perform. Out of more than 300 contacts who were sent the survey, 103 of them responded. The number of respondents who qualified were 80 in number, and they were chosen to participate in the pilot

5.3 Smart Parking System area coverage and performance

The area of interest was the Katajonokka island and the aim was to have coverage to the whole area. The coverage test was carried out verify that a connection could be established at all points in the pilot area.

The results from the coverage tests were good, the whole pilot area was well covered. Despite the heavy concentration of concrete building in the pilot area, LoRa technology performed quite good considering the height at which the gateways were installed. Figure 23 show the coverage area map tested during the pilot.

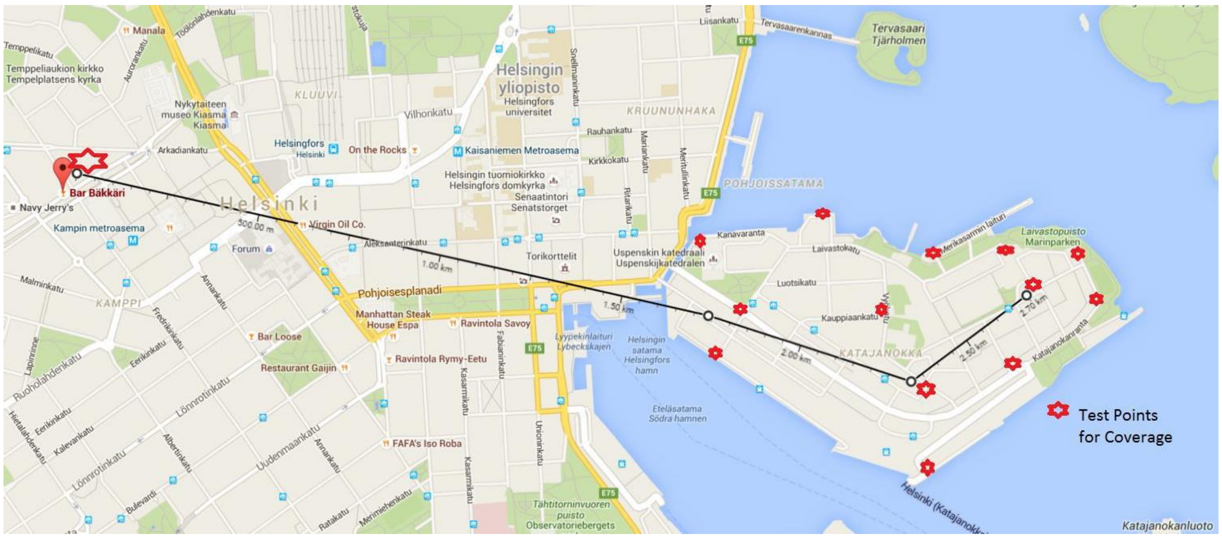

Figure *23: Smart parking coverage map for Katajonokka*

The furthest point that was measured was about two (2) kilometers away from the pilot area, near the Bar Bakkari. We believe that if the elevation at which the WinSpots where installed was much higher, the coverage to other areas would have been even better.

5.4 Parking facility Occupancy Prediction

One of the purpose of the system is to be able to predict the probability of finding a vacant parking space in a given parking facility. This is expected to reduce on drivers' cycling around the busy city looking for where to park, since the driver will be able to know in advance where he is likely to find parking. History data about the parking situation in Katajonokka was used to test the machine learning algorithms that were developed to predict occupancy. The results indicated that it is possible to predict the occupancy up to an hour with the certainty ranging from 65% to 95%. Figure 24 shows an example of a parking occupancy prediction for a day's hourly parking situation. The x-axis shows the time of the day and the y-axis percentage occupancy.

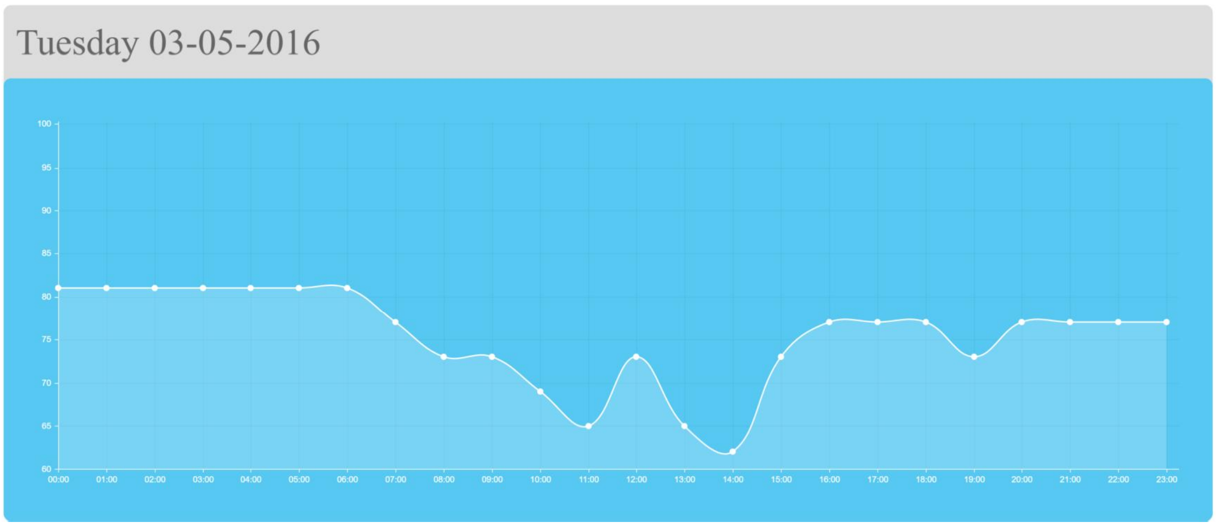

***Figure 24**: Parking Occupancy prediction for a day at Katajonokka*

The 100% occupancy was modeled to be the total number of participants who received the parking nodes – WinNodes. The algorithm uses the history parking data to learn the parking behaviors of these drivers to be able to predict the parking situation for a future date. Since Katajonokka is at the same time a residential area, the occupancy is seen dropping starting at 06:00 when residents start leaving to work, but start building back at 15:00 as they return from work.

CHAPTER 6

6.0 Discussion and Conclusion

The objectives of this work were to study the short comings drivers in Helsinki area faces and also to validate whether a smart parking system could address these short comings. It was intended that a smart parking system would be designed, implemented and deployed at Katajanokka – Helsinki area, with the aim of identifying the challenges such a system would face. The objectives were formulated into three research questions:

1. What are the challenges associated with parking in Helsinki Area?
2. What are the advantages of smart Parking in Helsinki Area?
3. What are the challenges a Smart Parking system for Helsinki Area would face?

The research questions were answered and a smart parking system was designed, implemented and deployed at Katajanokka.

Several challenges associated with parking in Helsinki area were highlighted by participants of the interview. All the participants believed that the smart parking system could solve their city parking difficulties. However, the cost of using a smart parking system also appeared to be a deciding factor to whether or not use the system. One of the participant stressed that the decision of whether or not to use the smart parking system depends of the cost.

The preliminary research through interviews identified three main problem areas associated with parking in Helsinki area:
1) Finding parking space,
2) Paying parking fees, and
3) Costing of parking time.
Smart parking being a good solution, there was willingness already among the people interviewed to use such system in Helsinki area. However, cost of using smart parking system should be kept moderate and comparable to the existing solutions.

A smart parking system was built based on LoRa technology – one of the long range communication technologies developed to support the IoT systems. Some protocols developed to support IoT are locked where by the only way to use them one has to deal their vendors. An example of such locked technologies include SIGFOX, one is not able to build a private network based on SIGFOX like it is possible with LoRa. The absence of central IoT standards is hindering the efforts to build IoT systems that can interoperate. Some good application level standards such as Open Data Format standard, and MQTT-SN already exist, the adaptation of a single one of them by the IoT community has not happened yet. What is being seen happening is the creation of ecosystems such as LoRa Alliance, Intel IoT solution Alliance, and many others but many of which are based on a given platform they are promoting. These ecosystems are closing out other players and hence creating a barrier towards the interoperability of IoT system.

It is quite unlikely that one is able to build a single IoT system that would meet all the requirements, but instead, a system of systems could be the most appropriate way to go for a cost effective solution. Many

IoT systems have mixed requirements to fulfil which require a combination of different technologies. For instance, the requirement for low power consumption and long range that must connect to the internet.

Some of the challenges for smart parking system

The challenges identified were based on the perspective of a startup company with limited resources, as it is usually the case. Some of the challenges a smart parking system faces in their corresponding system development phases include:

1. **Design challenges**
    a. There is no one technology that addresses all the needs of the system based on IoT paradigm, different technologies have to be combined to meet the requirements of the project. For instance, LoRa technology was used to meet the demands on low power nodes and 3G connected the nodes to the internet.
    b. The interoperability with other IoT systems without a common standard agreed upon, is not likely to happen soon

2. **Implementation challenges**
    a. There is a limitation on using the industrial, scientific and medical (ISM) radio band, it poses a restriction on the quantity of data to be shared between the Node and the gateway due shorter time on air imposed. The transmissions cannot happen at any time, the duty cycle must be respected and hence limiting the real time communication capabilities.
    b. The implementation of the system requires knowledge from a number of fields such as radio electronics and systems, power electronics, embedded systems, web technology and databases, to mention a few.
    c. Reducing the Cost of system: it takes enormous amount of time to look for the best deal from components vendors. For a small company with low volumes, the components cost is usually high, mainly due to lack of volume bargain.

3. **Deployment/Installation challenges**
    a. The need for a raised point from the ground, possibly on top of buildings, to mount the gateway as higher heights improves communication range and reduces the number of gateways needed. This would require renting a small part of the building's roof top or use telecommunication network masts, a move that would blow up the budget.

4. **Operations challenges**
    a. People fear to be tracked yet they love to be served
    b. Visibility of the WinNode under heavy snow fall
    c. Charging of the WinNode to full charge when less time is spent in that car – battery life under quick charging is threatened
    d. The accurate prediction of parking occupancy in the Helsinki area would require that all drivers have a WinNode, at least the majority of them.

Future improvements and developments

Even though the objectives of the project were achieved, there is still room from more improvements, to have the system fine-tuned for even better results. Among the improvements and future research include:

- Automatic determination/reading of WinNode ID/Car registration number in situations when it is not visible. In heavy snow fall of winter, reading registration numbers and node IDs could be improved if done automatically. So more research on ways to have such a feature added is needed.
- Charging of the WinNode to full charge when less time is spent in that car – battery life under quick charging. As more features are added, more energy is needed and using non-rechargeable batteries might not might not be feasible as the size of the node would grow bigger to be able to store enough energy for longer node life. A good option could be to use reachable but with a charger that supports quick charge as les time is spent in the car might not be enough to load enough energy for a day's usage.
- Automating parking facilities' barrier opening and parking fee payments, where driver drives next to the barrier and the barrier detects the node and opens the barrier, he/she simply parks the car and drives a way at the end – payments taken care of automatically.
- The use of the button to switch the modes of the WinNode from, for instance, a parking permit to the parking disk when in places where no payment is required. The node could have a display and a timer to show the time at which the car was parked.
- Cellular based Nodes that talks directly to the backend could eliminate the use of gateways and the challenges associated with their installation. More research on node-to-server architecture is needed especially in the area of power consumption reduction.

References

1. Ahmad, N.A.N. and Zamri, S.A.S., 2014, September. The cross platform application development adapted Spring framework to support front-end tendering services. In Computer, Communications, and Control Technology (I4CT), 2014 International Conference on (pp. 58-62). IEEE.
2. Alarcon, R. and Wilde, E., 2010, April. Linking data from restful services. In Third Workshop on Linked Data on the Web, Raleigh, North Carolina (April 2010).
3. Al-Fagih, A.E., Al-Turjman, F.M., Alsalih, W.M. and Hassanein, H.S., 2013. A priced public sensing framework for heterogeneous IoT architectures. Emerging Topics in Computing, IEEE Transactions on, 1(1), pp.133-147.
4. Alpaydin, E., 2014. Introduction to machine learning. MIT press.
5. Anthes, G., 2012. HTML5 leads a web revolution. Communications of the ACM, 55(7), pp.16-17.
6. Armando, A., Carbone, R., Chekole, E.G. and Ranise, S., 2014, June. Attribute based access control for APIs in spring security. In Proceedings of the 19th ACM symposium on Access control models and technologies (pp. 85-88). ACM.
7. Atighetchi, M., Soule, N., Pal, P., Loyall, J., Sinclair, A. and Grant, R., 2013, October. Safe configuration of TLS connections. In Communications and Network Security (CNS), 2013 IEEE Conference on (pp. 415-422). IEEE.
8. Bormann, C., Castellani, A.P. and Shelby, Z., 2012. Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Computing, 16(2), p.62.
9. Centenaro, M., Vangelista, L., Zanella, A. and Zorzi, M., 2015. Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. *arXiv preprint arXiv:1510.00620*.
10. Chinrungrueng, J., Sunantachaikul, U. and Triamlumlerd, S., 2007, January. Smart parking: An application of optical wireless sensor network. In Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on (pp. 66-66). IEEE.
11. Datta, S.K., Bonnet, C. and Nikaein, N., 2014, March. An IoT gateway centric architecture to provide novel M2M services. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 514-519). IEEE.
12. DiCicco-Bloom, B. and Crabtree, B.F., 2006. The qualitative research interview. Medical education, 40(4), pp.314-321.
13. Django, 2016. Django overview. Django Software Foundation. *Accessed 2 April 2016.* Available at: https://www.djangoproject.com/start/overview/
14. Egners, A., Gatzen, D., Panchenko, A. and Meyer, U., 2012, March. Introducing SOR: SSH-based onion routing. In Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on (pp. 280-286). IEEE.
15. Framling, K., Kubler, S. and Buda, A., 2014. Universal messaging standards for the IoT from a lifecycle management perspective. Internet of Things Journal, IEEE, 1(4), pp.319-327.
16. Fu, Q. and Retscher, G., 2009. Active RFID trilateration and location fingerprinting based on RSSI for pedestrian navigation. Journal of Navigation, 62(02), pp.323-340.
17. Geng, Y. and Cassandras, C.G., 2012. A new “smart parking” system infrastructure and implementation. Procedia-Social and Behavioral Sciences, 54, pp.1278-1287.
18. Govindan, K. and Azad, A.P., 2015, January. End-to-end service assurance in IoT MQTT-SN. In Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE (pp. 290-296). IEEE.
19. Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M., 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), pp.1645-1660.
20. Gupta, V., Goldman, R. and Udupi, P., 2011, June. A network architecture for the web of things. In Proceedings of the Second International Workshop on Web of Things (p. 3). ACM.

21. Herrero, J.L., Lozano, J. and Santos, J.P., 2015. A REstfull Approach for Classifying Pollutants in Water Using Neural Networks. In New Contributions in Information Systems and Technologies (pp. 371-380). Springer International Publishing.
22. Holt, G.K, Sep 16, 2014. Get Smart: SSH reverse tunnels for some access to smart meters. researchgate.net.
23. Hunkeler, U., Truong, H.L. and Stanford-Clark, A., 2008, January. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on (pp. 791-798). IEEE.
24. Ian Poole, 2014. LoRa Physical Layer & RF Interface. Radio-Electronics.com. *Accessed 10 March 2016*. Available:http://www.radio-electronics.com/info/wireless/lora/rf-interface-physical-layer.php
25. Jeliazkova, N. and Jeliazkov, V., 2011. AMBIT RESTful web services: an implementation of the OpenTox application programming interface. J. Cheminformatics, 3, p.18.
26. Kim, K.J., Kim, C.G., Whangbo, T.K. and Yoon, K., 2016. A continuous playing scheme on RESTful web service. Cluster Computing, pp.1-9.
27. King, N. and Horrocks, C., 2010. Interviews in qualitative research. Sage.
28. Krco, S., Pokric, B. and Carrez, F., 2014, March. Designing IoT architecture (s): A European perspective. In Internet of Things (WF-IoT), 2014 IEEE World Forum on (pp. 79-84). IEEE.
29. Ksairi, N., Tomasin, S. and Debbah, M., 2015. A Multi-Service Oriented Multiple-Access Scheme For Next-Generation Mobile Networks. *arXiv preprint arXiv:1512.00213*.
30. LaGrone, B., 2013. HTML5 and CSS3 Responsive Web Design Cookbook. Packt Publishing Ltd.
31. Le Blond, S.P., Holt, A. and White, P., 2012, October. 3eHouses: A smart metering pilot in UK living labs. In Innovative Smart Grid Technologies (ISGT Europe), 2012 3rd IEEE PES International Conference and Exhibition on (pp. 1-6). IEEE.
32. Lee, E.A. and Seshia, S.A., 2015. *Introduction to embedded systems: A cyber-physical systems approach*. Lee & Seshia.
33. Lee, S., Kim, H., Hong, D.K. and Ju, H., 2013, January. Correlation analysis of MQTT loss and delay according to QoS level. In Information Networking (ICOIN), 2013 International Conference on (pp. 714-717). IEEE.
34. Lomotey, R.K. and Deters, R., 2013, March. Reliable Consumption of Web Services in a Mobile-Cloud Ecosystem Using REST. In Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on (pp. 13-24). IEEE.
35. LoRa Alliance, 2016. LoRa Alliance Wide Area Networks for IoT, LoRa Allinace [Online]. Available: https://www.lora-alliance.org/ [Accessed: *10 March 2016*].
36. LoRaWAN Specification v1.0, LoRa Alliance, Inc. 2400 Camino Ramon, Suite 375 San Ramon, CA 94583, 2015. Available: https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf
37. Merriam, S.B. and Tisdell, E.J., 2015. Qualitative research: A guide to design and implementation. John Wiley & Sons.
38. Munsi, N., Sehrawat, N. and Jain, M., 2014. Integration of Struts, Spring and Hibernate for an E-Commerce System.
39. Nurminen, J.K., Meyn, A.J., Jalonen, E., Raivio, Y. and Garcıa Marrero, R., 2013, April. P2P media streaming with HTML5 and WebRTC. In Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on (pp. 63-64). IEEE.
40. Park, J.T., Hwang, H.S., Yun, J.S. and Moon, I.Y., 2014. Study of HTML5 WebSocket for a Multimedia Communication. development, 9(7).
41. Pautasso, C., Zimmermann, O. and Leymann, F., 2008, April. Restful web services vs. big'web services: making the right architectural decision. In Proceedings of the 17th international conference on World Wide Web (pp. 805-814). ACM.
42. Pcmag, 2014. Smart-parking [online]. [Accessed: *10 March 2016*] Available at: http://www.pcmag.com/encyclopedia/term/65086/smart-parking

43. Pimentel, J.R., 2014, May. An effective and easy to use IoT architecture. In 2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014).
44. Plekhanova, J., 2009. Evaluating web development frameworks: Django, Ruby on Rails and CakePHP. Institute for Business and Information Technology.
45. Plesowicz, P. and Laszczyk, P., 2013, August. Evaluation of secure signal transmission in automatic control using SSH tunneling. In Methods and Models in Automation and Robotics (MMAR), 2013 18th International Conference on (pp. 672-677). IEEE.
46. Polycarpou, E., Lambrinos, L. and Protopapadakis, E., 2013, June. Smart parking solutions for urban areas. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a (pp. 1-6). IEEE.
47. Powers, S., 2014. The open-source classroom: this is why we can't have nice things: SSH. Linux Journal, 2014(246), p.7.
48. Rajkumar, R.R., Lee, I., Sha, L. and Stankovic, J., 2010, June. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference* (pp. 731-736). ACM.
49. Retscher, G. and Fu, Q., 2008. Active RFID trilateration for indoor positioning. Coordinates, 4(5), pp.10-15. Available at: http://mycoordinates.org/active-rfid-trilateration-for-indoor-positioning/all/1/
50. Revathi, G., Dhulipala, V. and Sarma, R., 2012, February. Smart parking systems and sensors: A survey. In 2012 International Conference on Computing, Communication and Applications (ICCCA) (pp. 1-5).
51. Riliskis, L., Hong, J. and Levis, P., 2015, November. Ravel: Programming iot applications as distributed models, views, and controllers. In Proceedings of the 2015 International Workshop on Internet of Things towards Applications (pp. 1-6). ACM.
52. Rodriguez, A., 2008. Restful web services: The basics. IBM developerWorks.
53. Saldana, J., 2011. Fundamentals of qualitative research. Oxford university press.
54. Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. Software Engineering, IEEE Transactions on, 25(4), pp.557-572.
55. Shelby, Z., Hartke, K. and Bormann, C., 2014. The constrained application protocol (CoAP).
56. Soni, R.K., 2015. Learning Spring Application Development. Packt Publishing Ltd.
57. Srikanth, S.V., Pramod, P.J., Dileep, K.P., Tapas, S., Patil, M.U. and Sarat, C.B.N., 2009, May. Design and implementation of a prototype Smart PARKing (SPARK) system using wireless sensor networks. In Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on (pp. 401-406). IEEE.
58. Stanford-Clark, A.S. and Truong, H.L., 2013. MQTT for sensor networks (MQTT-S) protocol specification. International Business Machines Corporation version, 1.2.
59. Stirbu, V., 2010, April. A restful architecture for adaptive and multi-device application sharing. In Proceedings of the First International Workshop on RESTful Design (pp. 62-65). ACM.
60. Strauss, A. and Corbin, J., 1990. Basics of qualitative research (Vol. 15). Newbury Park, CA: Sage.
61. Taneja, S. and Gupta, P.R., 2014. Python as a Tool for Web Server Application Development.
62. Tesoriero, R., Gallud, J., Lozano, M. and Penichet, V., 2008. Using active and passive RFID technology to support indoor location-aware systems. Consumer Electronics, IEEE Transactions on, 54(2), pp.578-583.
63. The OpenGroup, 2014. Open Data Format Standard(O-DF), Available at: https://www2.opengroup.org/ogsys/catalog/C14B
64. The Spring Team, 2016. Spring Framework. Pivotal Software, Inc. *Accessed 28 March 2016.* Available at: https://projects.spring.io/spring-framework/#quick-start
65. Vangelista, L., Zanella, A. and Zorzi, M., 2015. Long-Range IoT Technologies: The Dawn of LoRa™. In *Future Access Enablers for Ubiquitous and Intelligent Infrastructures* (pp. 51-58). Springer International Publishing.
66. Vasco Lopes, N., Pinto, F., Furtado, P. and Silva, J., 2014, October. IoT architecture proposal for disabled people. In Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on (pp. 152-158). IEEE.

67. Villaverde, B.C., Pesch, D., De Paz Alberola, R., Fedor, S. and Boubekeur, M., 2012, July. Constrained application protocol for low power embedded networks: A survey. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on (pp. 702-707). IEEE.
68. Wang, C.S., Huang, C.H., Chen, Y.S. and Zheng, L.J., 2009, December. An implementation of positioning system in indoor environment based on active RFID. In Pervasive Computing (JCPC), 2009 Joint Conferences on (pp. 71-76). IEEE.
69. Weinstein, R., 2005. RFID: a technical overview and its application to the enterprise. IT professional, 7(3), pp.27-33.
70. Wilke, A., Bischof, J., Harrison, T., Brettin, T., D'Souza, M., Gerlach, W., Matthews, H., Paczian, T., Wilkening, J., Glass, E.M. and Desai, N., 2015. A RESTful API for accessing microbial community data for MG-RAST. PLoS Comput Biol, 11(1), p. e1004008.
71. Yeh, M.C., Yang, H.I. and Mei, H., 2015, March. A HTML5-Based Innovative Sleep Prescription Service. In Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on (pp. 221-222). IEEE.
72. Ylonen, T. and Lonvick, C., 2006. The secure shell (SSH) protocol architecture.
73. Yun, M. and Yuxin, B., 2010, June. Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid. In Advances in Energy Engineering (ICAEE), 2010 International Conference on (pp. 69-72). IEEE.
74. Zhang, D., Wei, Z. and Yang, Y., 2013, October. Research on Lightweight MVC Framework Based on Spring MVC and Mybatis. In Computational Intelligence and Design (ISCID), 2013 Sixth International Symposium on (Vol. 1, pp. 350-353). IEEE.
75. Zhang, E., Jiang, W., Kuang, Y. and Umer, M.A., 2011, October. Active RFID positioning of vehicles in road traffic. In Communications and Information Technologies (ISCIT), 2011 11th International Symposium on (pp. 222-227). IEEE.