ISBN: 978-989-8533-80-7 © 2018

DEVELOPMENT OF A LOW-COST SMART PARKING SOLUTION FOR SMART CITIES

João Fernandes, Carlos Serrão[1] and Nuno Garrido[2]
ISCTE – Instituto Universitário de Lisboa
[1]*Information Sciences, Technologies and Architecture Research Center (ISTAR-IUL)*
[2]*Instituto de Telecomunicações (IT-IUL)*
Ed. ISCTE, Av. das Forças Armadas, 1649-026, Lisbon, Portugal

ABSTRACT

The pressure of traffic on modern cities keeps growing. More and more vehicles flow into the city draining the existing parking resources and increasing traffic congestions and fueling the pollution increase. In this paper we present a solution for a low-cost smart parking system and all the software and hardware components that were developed and integrated in a smart parking prototype. The developed prototype is based on Arduino for the sensor network and on the Raspberry Pi for the gateway device. This paper also describes some of the technology aspects used in the communication between the sensors and the gateway – based on the ZigBee protocol. In addition to the hardware part of the prototype, the used backend architecture and the mobile application, that enables the user to find quickly and efficiently a parking spot, are also analyzed in this paper.

KEYWORDS

Smart Parking, Smart Cities, Prototype, Parking, Arduino, Raspberry Pi, Android, iOS

1. INTRODUCTION

Cities' traffic congestion and the increase in the number of the city inflow vehicles are some of the major problems that citizens and city authorities all over the world must face daily. The importance of finding solutions to improve the quality of the city's life has grown in the past years and the use of IT has created a new breed of somewhat intelligent cities – often referred to as smart cities. The concept of "smart" applied to the cities has been regarded as the answer to some of the major city problems and is slowly changing the way we live and interact with our cities. A smart city can be described has "a city that monitors and integrates conditions of all of its critical infrastructures, including roads, bridges, tunnels, railways, subways, airports, seaports, communications, water, power, even major buildings, and that can better optimize its resources, plan its preventive maintenance activities, and monitor security aspects while maximizing services to its citizens" (Chourabi et al., 2012).

For years, parking management systems have been amongst some of the major cities concerns, and some cities have carried out studies on the smart parking concept and what impacts these systems can bring to social, economic, political and environmental issues. However, there have been some problems related to smart parking systems implementation which raised political issues, like the existence of more than one stakeholder which can be a problem in terms of investment management and decision making during the project.

This paper presents an integrated low-cost system based on open hardware and software components tailored for the city needs to monitor and measure, not only the management of the parking areas, but also other environment data such as mobility, temperature and humidity. The developed prototype presented in this paper is considered as proof of concept and a reference for the future implementation of similar system approaches in any city in the world, in a timely and cost-efficient manner. The prototype was designed in a way that it can be used with different architecture or software solutions tailored for parking management systems.

This paper starts by introducing the smart parking topic and its importance in the context of smart cities. Following this section, we present a short overview of current existing smart parking solutions. On the next section of the paper the proposed system is presented as well as the different components and how they are integrated. The tests and results from the developed solution are discussed on the following section and on the final section of the paper we present some conclusions and point out some future work that needs to be accomplished.

2. OVERVIEW OF SMART PARKING SOLUTIONS

"Smart cities deploy various information and communication technology components like radio-frequency identification (RFID), embedded systems, Wi-Fi, WiMax, LTE, ZigBee, machine-to-machine communication (M2M), asset management, energy management, unified communications, and collaboration" (Washburn et al., 2010). Hwaseong Dongtan, in South Korea, is one of the most relevant examples of a smart city that first introduced a global communications network where citizens can interact to get real-time information and, therefore, to prevent crime, control traffic systems and build a smart parking infrastructure.

Different studies on smart parking systems (Thanh Nam Pham et al., 2015) (Giuffrè, Siniscalchi & Tesoriere, 2012) (Idris et al., 2009) mention the need to improve the current parking systems and how these systems can bring benefits to the development of smart cities. The growth of the cities and the increase in the number of vehicles have enabled the development of parking technologies, increasing the need to implement more advanced and intelligent parking systems in metropolitan areas of the cities (Fraifer & Fernström, 2016).

In (Thanh Nam Pham et al., 2015) a proposal is made to create a system like an IoT network that assists drivers on finding a free parking space at the lowest possible cost based on new performance metrics for the calculation of the parking cost, considering the geolocation of the vehicle, the distance between the parking areas and the total number of free slots in the parking lot. If the car park is full, the driver is redirected to another location until he can park the vehicle. Each car park uses WSN (Wireless Sensor Network) (Hancke & Hancke Jr, 2012) technology which monitors the parking lots through RFID (Akyildiz & Kasimoglu, 2004). The system works in real-time and gives the user the choice of the most suitable parking place, sending directions to the destination. Whenever a vehicle enters or exits the park, the data is updated by communicating with the parking lot WSN.

Another solution, presented by SmartParking Systems, consists of an advanced navigation system that signals the availability and directs towards the parking spot closest to the destination chosen by the user (Smart Parking Systems, 2017). The system uses LoRaWAN technology (Adelantado et al., 2017), capable of connecting sensors over long distances, requiring minimal structure while delivering optimal battery life. This offers advantages such as mobility, security and optimized location/positioning, as well as cost savings. A smartphone and tablet application allows the user to see real-time parking spaces and helps drivers to choose the best location without having to move around to check available parking spots. Time wasted in finding available spots is eliminated; the user saves time and the traffic in residential areas is relieved. The user can pay for parking easily using the application that is associated with a credit card.

3. SMART PARKING SYSTEM ARCHITECTURE

The main purpose of this work was to study, design and implement a low-cost smart parking system, using open hardware and open software resources. Figure 1 shows an architectural overview of the system and represents the implementation of the prototype. The architecture of the prototype is divided into four essential components: sensors, gateways, backend and mobile application. These components will be individually analyzed to understand their relevance on the overall operation of the system proposed in this article.

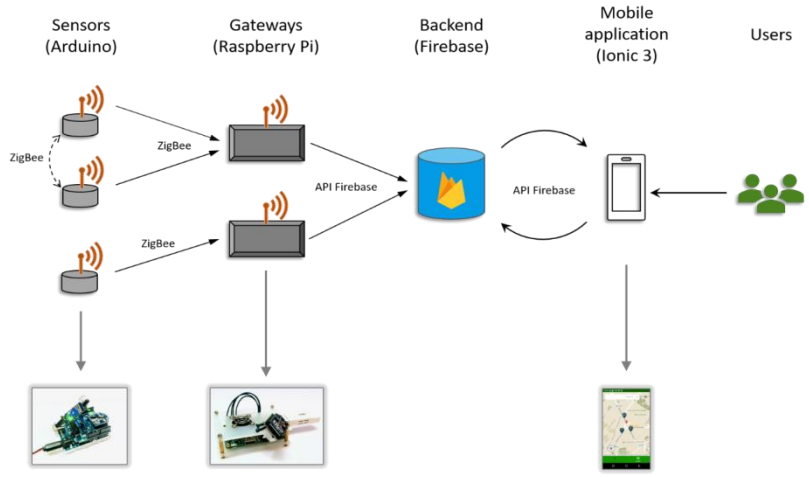

Figure 1. Smart parking system global architecture

The overall smart parking solution integrates a set of hardware components and the necessary software to make them operate in an integrated manner. The following sections of the article will present the different components.

3.1 Parking Sensors

For this prototype, a simple infrared proximity sensor is used for the implementation of the vehicle proximity sensor. This sensor detects the presence of obstacles based on the reflection of infrared radiation emitted by a transparent led and collected by a photoelectric device. It is a basic and low-cost solution used only for prototyping purposes and may not be adequate for use in a final smart parking system, since different things may incorrectly trigger the device and emulate a parked car. This is not a real issue for the smart parking system because one of the objectives was to provide the ability to support any kind of sensor (or groups of sensors) capable of accurately detecting a parked car. The proximity sensor is integrated within an Arduino board to implement the necessary logic switch for the parking sensor.

To implement a wireless communication meshed network between the different parking sensors, XBee was the selected data transceiver used to enable the communication between the different sensors and the gateways using the ZigBee protocol. For the prototype sake all devices were configured as Router (Silicon Laboratories, 2018), due to the existence of only 3 sensors in the sensor network and thus increasing the radius of communication between the sensors and the gateway. Although this configuration is not the most efficient in terms of energy consumption, the sensors energy consumption is extremely low and increasing the range of the sensor relative to the gateway is an important implementation aspect of the system that justifies the followed approach.

The Arduino board integrates the different components of the sensor and enables the communication obtaining the sensor status and sending the data to the gateway. To achieve this, specific software was developed for the Arduino UART controller that allows the initialization of the variables inherent to the proximity sensor and XBee. The remaining global variables are in line with the operation of the software, including the collection and processing of data as depicted in Figure 2.

```
void setup() {
  pinMode(ProxSensor, INPUT);
  Serial.begin(115200);
  xbee.begin(115200);
}
```

Figure 2. Setup function of the Arduino program

In this software, the '`setup()`' function is responsible for initialization of the different elements that are part of parking sensor. First, the initialization of the proximity sensor is performed through the '`pinMode`' function, which receives as arguments the identifier of where the sensor is connected, and the type of the connected element (in this case, '`INPUT`', because the intention here is to obtain the state of the sensor as data input for the Arduino board). After this process, the baud rate (bit per second) of the data transmission is defined.

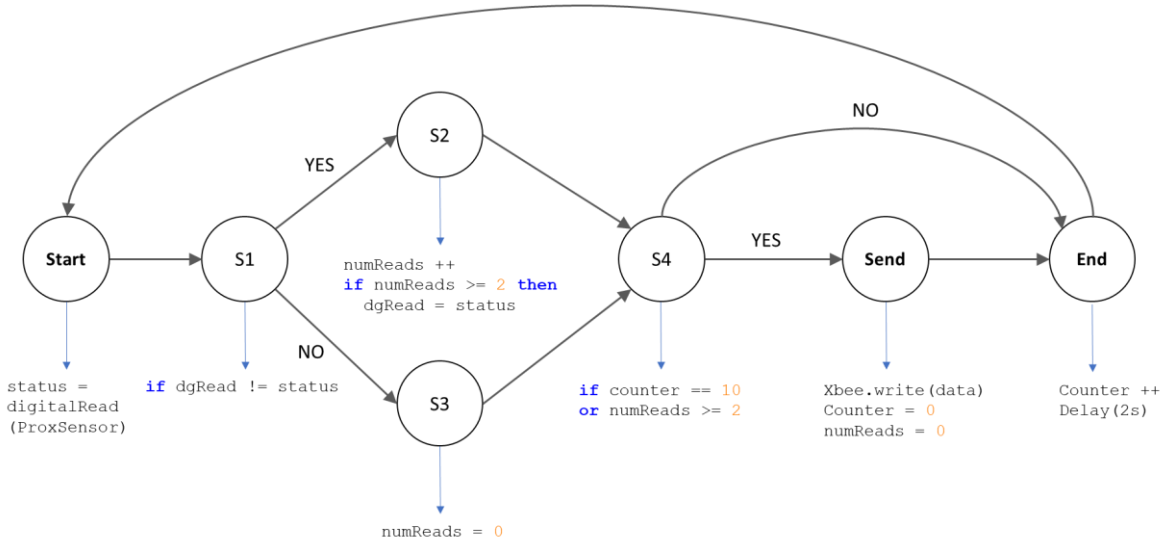

Figure 3. State machine and pseudocode diagram of the Arduino loop function

It is in the looping part of the Arduino software that the collection, processing and sending of data will take place. Figure 3 shows the state machine diagram of the Arduino loop function. The first state is called 'Start' and is responsible for gathering the status of the proximity sensor by calling function '`digitalRead()`' and saving it to a '`status`' variable. State 'Send' contains the function '`Xbee.write()`' which sends the data through the XBee to the gateway. Variables '`counter`' and '`numReads`' respectively specify a counter for each iteration of the cycle, and the number of readings in which the previous state of the sensor represented by '`dgRead`' is different from the current state indicated by the variable '`status`'. The goal is to ensure that there are no false detections. '`numReads`' must be greater than or equal to 2 (state 'S4'), assuring the first change of status was confirmed by a second iteration of the cycle, thus guaranteeing a true detection. This allows the sensor to respond immediately when sensing a consistent status change while avoiding constantly sending redundant information and thus save energy. The counter can be set to send data every 10 iterations even if there are no state changes, informing the gateway that the sensor is active on the network thus providing fast battery failure detection. The last function of the software implements a delay of two seconds before returning to the first instruction of the cycle, therefore setting the sampling rate of the sensor cell cycle. After connecting all elements, the result of the parking sensor is shown in Figure 4.

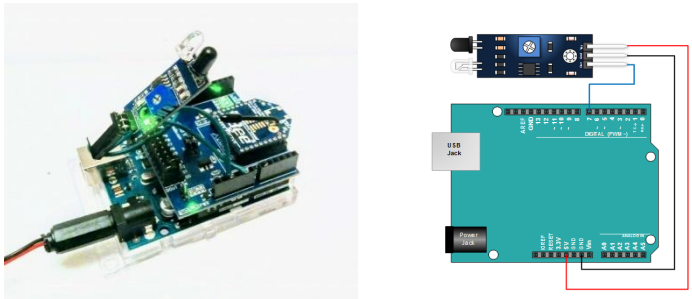

Figure 4. Prototype proximity sensor and IR sensor connection to the Arduino (Vxlabs, 2018)

3.2 Gateway

The gateway is an important component on the system and it will be responsible for allowing the communication of the parking sensors network with the management backend system, through the Internet. The gateway component is composed by hardware and specifically designed software to implement its functionality. Raspberry Pi 3, running on the Raspbian Linux operating system, was the hardware platform selected to implement the gateway prototype. Figure 5 represents the gateway prototype and depicts the connection of the XBee transceiver to the GPIO ports of the Raspberry Pi (Electronics For You, 2016). This will enable the gateway to communication with the neighboring sensor network.

The GPIO ports of the Raspberry Pi are set to use a baud rate of 115200 to connected to the XBee, coherently with the settings of the sensors XBee devices. Since for the backend part of the system, Firebase will be used, on the gateway software the Firebase API needs to be initialized and will be used to send the collected data. It is within the infinite loop shown in Figure 6, that the information from the sensors will be collected from the XBee device. This data will then be transferred to a vector using the '`split()`' method which separates a string from a defined character. The 'if' statement verifies the correct reception of the data and then the information regarding each specific sensor is sent to the database.

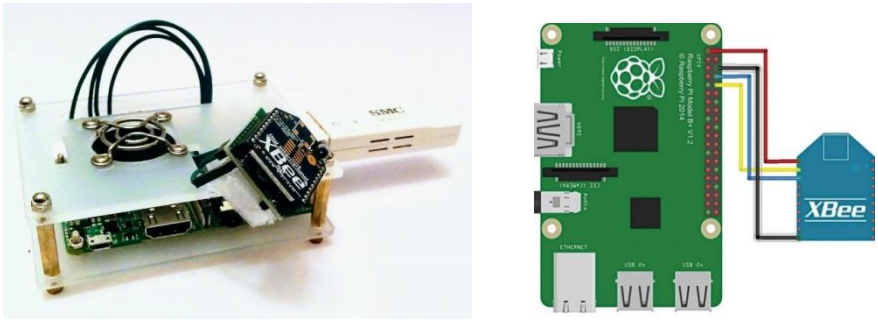

Figure 5. Gateway prototype and connecting of the XBee transceiver to the Raspeberry Pi (Electronics For You, 2016)

```
## Loop
while True:
    dataReceived = str(ser.readline().strip()).replace("b","")
    dataReceived = dataReceived.replace("'","").split(";")
    if dataReceived and len(dataReceived) == 4 and
       dataReceived[0]!= "":
        idS = "s" + dataReceived[0]
        addSensor(sensors, dataReceived)
 sensors[getSensorPosition(sensors,dataReceived[0])].cancelTimer()
        result = firebase.patch("/zonas/"+idZ+"/gateways/" + idG +
"/sensores", {idS: {"id": dataReceived[0],
                    "lat":dataReceived[1],
                    "lng":dataReceived[2],
                    "status":dataReceived[3]}})
 sensors[getSensorPosition(sensors,dataReceived[0])].startTimer()
```

Figure 6. Infinite loop of the gateway side program in Python

3.3 Backend System

For the system to work, the backend part of the prototype is based on the online Firebase platform. This platform was selected because it provides the essential services needed for a faster prototype development, however it may be replaced by any other backend platform if the basic necessary backend services are guaranteed. In the gateway component, authentication is initially performed on Firebase followed by the API initialization, which later allows the use of functions for collecting and sending data to the database. In the configuration of a Firebase project, chunks of code are provided for introduction into the Android, iOS, and Web application projects that allow Firebase API initialization. In the case of the Ionic Framework (used later for the application development), the code for Web applications is withdrawn, as it uses Web technologies for the development of hybrid mobile applications.

3.4 Mobile Application

The final objective of a smart parking solution integrated on a smart city is to provide the necessary tools for citizens to easily and quickly find ways to improve their lives through the usage of affordable technologies. In this specific case, the objective is to provide the end users with a simple tool to access the most convenient parking slot available at a given time. Due to the growing usage of mobile technologies, the best way to bring the solution to the end user is through the development of a mobile application. To produce a mobile application, in a way that it can be usable both on Android and iOS devices, we selected a hybrid mobile development framework - Ionic Framework (Ionic Framework, 2018). After finalizing the design and export of the created project through the Ionic Creator website (Ionic Creator, 2018), the Google Maps library was added to the project scope so that the user could navigate the map in search of parking zones represented by custom markers. Using the mobile application, it is possible to check the number of available spots in a given period represented by a number inserted in the markers of each zone, as shown in Figure 7. This number will be updated depending on the state changing of the sensors for a given zone. Whenever the user presses a marker, he will be taken to another screen that shows in more detail the information about the selected zone, i.e. the list of parking spots and their current state, and information about the parking zone. If the user selects a spot, the screen with the previous map will appear again, but, in this case, the map shows the route from the driver's location to the destination, and this operation can be confirmed or cancelled. There is another way to search for parking zones and it is in the second tab located at the bottom of the screen. The side menu offers completeness to the application, showing the most relevant sections of the application and each one was implemented separately.

All data generated and modified by the user is subsequently updated in the database, where each registered user has a reserved section. This is possible by calling methods implemented in the Firebase API library. During the application execution, information about the parking status in each zone can be updated in real-time, because of the listener implemented that can detect if the database has changed.

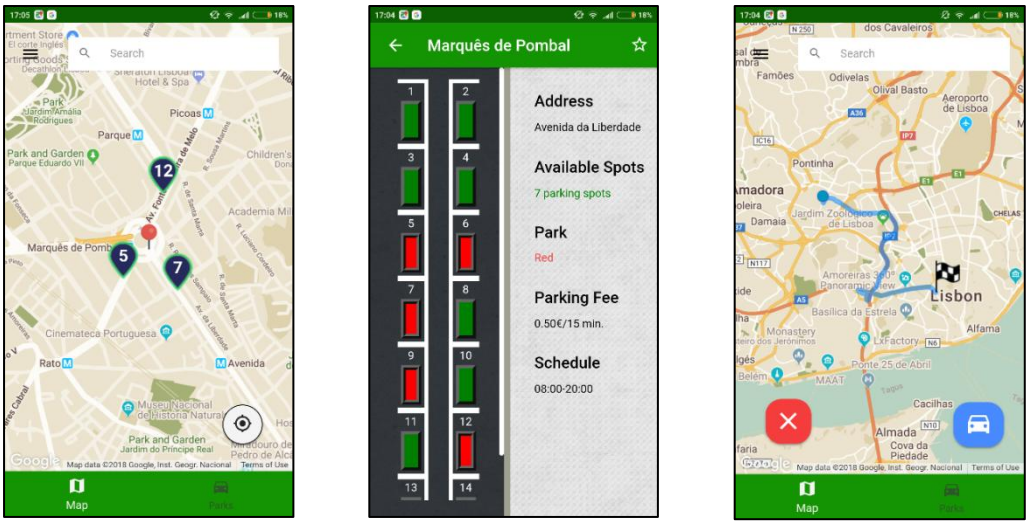

Figure 7. Application prototype layout

4. TESTS AND RESULTS

Through the software console provided for the Arduino's programming and configuration, it is possible to verify that the printed data is consistent with what is intended. The developed Python program works correctly as expected and its operation can be checked in real-time through the Firebase console. During the gateway operation, it is possible to check real-time sensor status changes. All information contained in the database is collected and handled by the application and shown to the user in an organized manner.

The fully functional prototype tested consists of two gateways, three parking sensors and the mobile application installed on an Android device. The gateway on the right side of figure 8 receives information from both sensors, also on the right side of the photo, and the gateway to the left receives information only from the sensor to the left. The intention is to simulate the existence of two active parking zones. One of the gateways is powered by a power bank with photovoltaic cells, which draw power to the lithium polymer rechargeable battery thus presenting a renewable and environmental friendly energy solution. Considering the existence of this method for supplying energy to traffic lights, for example, these devices require a much lower amount of energy, which makes possible the use of renewable energies to power the gateways. The sensors are powered by 9V batteries which are sufficient for the operation of the devices in this prototype environment.

The prototype works as expected and faithfully demonstrates the functionality of the smart parking system. All components interconnect in a consistent way and the data transmitted and subsequently received in the application are coherent and robust. The responsiveness to changes in sensor state is within the minimum standards and ZigBee technology ensures full-time communication between physical components. The range of the sensor network built for the prototype is acceptable although obstacles can negatively influence communication. It was impossible to test the prototype in a real environment; however, the obtained results exceeded the initial expectations for a real-time smart parking management system.

Figure 8. Complete smart parking prototype test setup

5. CONCLUSIONS AND FUTURE WORK

This paper presents a low cost open hardware and software smart parking solution to mitigate the problem of parking in cities and the traffic associated to the search of parking spots. The study of existing systems and the designed prototype built have led to the conclusion that there are advantages in the implementation of a smart parking system in any city, and this concept will be a reality soon that will surely benefit citizens and smart cities.

We present an open solution that allows integration with other components alternative to those chosen for this prototype, which makes it fully scalable. The use of Arduino and Raspberry Pi for parking sensors and gateways, respectively, enables the option to use other types of sensors and data communication methods alternative to those applied during the implementation of the prototype. The use of infrared sensors served as a simple solution for the prototype, but can be easily replaced by a magnetic sensor, specific for the detection of vehicles. This project demonstrates the use of ZigBee technology for the communication between the physical elements of the system. This communication protocol is commonly used in other existing smart parking systems and in other smart cities applications as an appropriate method of communication in terms of energy savings and cost efficiency.

The development of an application that previously reveals the availability of parking spaces in the areas with the highest occupancy and daily demand, could please the citizens, but the issue of reservation of parking spots was not addressed. The integration of smart parking in a city can change the paradigm of traffic and parking as the biggest problems of the cities and improve the solutions and proposals that exist over time.

ACKNOWLEDGEMENTS

This project was partially funded by FCT | UID/MULTI/4466/2016.

REFERENCES

Adelantado, Ferran, Xavier Vilajosana, Pere Tuset, Borja Martínez and Joan Melià. "Understanding the Limits of LoRaWAN." *IEEE Communications Magazine* 55 (2017): 34-40.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., ... & Scholl, H. J. (2012, January). Understanding *Smart Cities*: An integrative framework. In System Science (HICSS), 2012 45th Hawaii International Conference on (pp. 2289-2297). IEEE.

Electronics For You. (2016) *XBee interfacing Raspberry Pi Model 2*. Available from: https://electronicsforu.com/electronics-projects/XBee-interfacing-raspberry-pi-model-2/2 [Accessed 15th November 2017].

Fraifer, Muftah & Fernström, Mikael. (2016). Investigation of Smart Parking Systems and their technologies

Giuffrè, T., Siniscalchi, S. M., & Tesoriere, G. (2012). A novel architecture of parking management for *Smart Cities*. Procedia-Social and Behavioral Sciences, 53, 16-28.

Hancke, G. P., & Hancke Jr, G. P. (2012). The role of advanced sensing in Smart Cities. Sensors, 13(1), 393-425

I. F. Akyildiz, I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges", Ad Hoc Netw., vol. 2, no. 4, pp. 351-367, Oct. 2004.

Idris, Mohd & Y.Y, Leng & E.M, Tamil & N.M, Noor & Razak, Zaidi. (2009). Car Park System: A Review of Smart Parking System and its Technology. Information Technology Journal. 8. 10.3923/itj.2009.101.113

Ionic Creator. (2018) Available from: https://creator.ionic.io/

Ionic Framework. (2018) *Build amazing apps in one codebase, for any platform, with the web.* Available from: https://ionicframework.com/

Silicon Laboratories. (2018) *What is the difference between an end device, a router, and a coordinator?* Available from: https://www.silabs.com/community/wireless/zigbee-and-thread/knowledge-base.entry.html/2012/07/02/what_is_the_differen-IYze [Accessed 4th November 2017].

Smart Parking Systems. (2017) *Shape the future of tomorrow's cities.* Available from: http://www.smartparkingsystems.com/, [Accessed 19th December 2016].

Thanh Nam Pham, Ming-Fong Tsai, Duc Binh Nguyen, Chyi-Ren Dow, and Der-Jiunn Deng. "A *Cloud*-based Smart-parking System based on Internet-of-Things technologies." digital object identifier 10.1109/access.2015.2477299 (2015): 1581-1591.

Vxlabs. (2018) *Which jumper to set on the ITEAD XBee shield v1.1 for use with a 3.3V Arduino.* Available from: https://vxlabs.com/2018/03/23/which-jumper-to-set-on-the-itead-XBee-shield-v1-1-for-use-with-a-3-3v-arduino/ [Accessed 25th October 2017].

Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N. M., & Nelson, L. E. (2010). Helping CIOs Understand "*Smart City*" Initiatives: Defining the *Smart City*, Its Drivers, and the Role of the CIO. Cambridge, MA: Forrester Research, Inc. Available from http://public.dhe.ibm.com/partnerworld/pub/smb/smarterplanet/forr_help_cios_und_smart_city_initiatives.pdf.