A Low-Cost Smart Parking Solution for Smart Cities Based on Open Software and Hardware

Carlos Serrão[1(✉)] and Nuno Garrido[2]

[1] Information Sciences, Technologies and Architecture Research Center (ISTAR-IUL), ISCTE – Instituto Universitário de Lisboa, Ed. ISCTE, Av. das Forças Armadas, 1649-026 Lisbon, Portugal
carlos.serrao@iscte-iul.pt

[2] Instituto de Telecomunicações (IT-IUL), ISCTE – Instituto Universitário de Lisboa, Ed. ISCTE, Av. das Forças Armadas, 1649-026 Lisbon, Portugal
nuno.garrido@iscte-iul.pt

Abstract. Traffic management and car parking on modern cities continues to be a problem both for citizens and for city officials. The increasing number of vehicles flowing into the city drain the existing scarce parking resources, and the increase in time spent looking for a parking spot leads to more congestions, parasitic traffic, whilst augmenting fuel consumption and air pollution. In this paper we present an integrated flexible solution developed to help address this issue, using open hardware and software components to develop a low-cost smart parking system suitable for contemporary metropolitan cities. The smart parking solution is based on Arduino boards for the sensors network and on Raspberry Pi single-board computers for the gateway devices, integrated through specific developed software components and a mobile application for the end-users.

Keywords: Smart parking · Smart cities · Prototype · Parking · Arduino · Raspberry Pi · Android · iOS

1 Introduction

Modern cities must deal with different problems and challenges. One of the most important challenges of modern cities is the number of vehicles that cross the city borders every day creating major problems that citizens and city authorities have to face on a daily basis. It is important to find appropriate solutions capable of improving the quality of the city's life and therefore IT has helped the creation of a new type of smart cities. This has been regarded as the answer to some of the major city problems and is slowly changing the citizens way of interacting with their cities. "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" can be described as a smart city (Chourabi et al. 2012).

J. C. Ferreira et al. (Eds.): INTSYS 2018, LNICST 267, pp. 15–25, 2019.
https://doi.org/10.1007/978-3-030-14757-0_2

Parking and parking management systems have always been amongst some of the major cities concerns. City officials have conducted long term studies on the smart parking concept and of which might be the social, economic, political and environmental impact such systems may cause. Nevertheless, the large investment that is required on this type of systems has created political problems in the cities for their widespread adoption.

The purpose of this paper is to present a prototype of an integrated low-cost system based on open hardware and software components and designed to address the needs of the cities that require monitoring and measurement, not only of the parking areas, but also of other environment data such as mobility, pollution, temperature and humidity.

As part of the research conducted to address the problem of creating a hardware and software solution for this problem a prototype was developed and will be succinctly described on this paper. This prototype integrates hardware components to operate as parking sensors, and gateways that integrate the different parking sensors in a parking area, it also includes all the necessary software for the components to operate and communicate with the backend, and finally a mobile application that is used by end-users to find and give driving directions to existing free parking spots.

The first part of this paper introduces the smart parking topics and determines its importance in the context of smart cities and how they can contribute to solve some of the parking problems cities and city officials have to face. The second part makes a small overview over some already existing intelligent parking solutions that were developed and are being used on the context of smart cities. In the third 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 in the final section of the paper we present some conclusions and point out some future work that needs to be accomplished.

2 Smart Cities Intelligent Parking Solutions

As previously referred, many cities around the world are already implementing or considering the future implementation of smart parking solutions to solve some of the existing problems with parking pressure over their existing parking equipment.

Many different studies have been conducted around the world about this theme that refer the importance of improving the existing parking systems and the way they are used and managed in order to provide benefits to smart cities (Pham et al. 2015; Giuffrè et al. 2012; Mohd et al. 2009). The pressure of the number of vehicles either existing on the cities or crossing its borders everyday needs to be tackled. This has contributed to the development of innovative parking technologies (Fraifer and Fernström 2016).

There are already some examples of smart cities using intelligent parking solutions that try to address the parking problems cities have to deal with. In (Pham et al. 2015) the authors make a proposal to create a system based on an IoT network that can help drivers to find free parking spaces at the lowest possible cost based on different metrics, considering the geolocation of the vehicle, the distance between the parking areas and the total number of free slots in the parking area. 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 and Hancke Jr 2012) technology which monitors the parking lots through RFID (Akyildiz and 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.

SmartParking Systems has presented another solution that consists of an advanced navigation system that signals the availability of a parking spot and directs the user towards it (Smart Parking Systems 2017). This system is based on the LoRaWAN technology (Adelantado et al. 2017) making it 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. This system provides also a smartphone and a tablet application that permits 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.

There are other systems that can be used to implement intelligent parking solutions and help the management of the existing parking areas in the city, however the referred ones are amongst some of the most relevant ones. In the following section, the developed low-cost intelligent parking solution based on open hardware and software is presented.

3 Low-Cost Intelligent Parking System

The objective of this paper is to describe the implementation of a low-cost intelligent parking solution based on open hardware and software. The developed prototype also regarded scalability and upgradability issues that can allow some of the components currently on the system to be replaced by others in the future, to better adequate to the specific city parking solution requirements.

The architecture of the system depicted on Fig. 1 is composed of a set of components that work in an integrated manner to provide the necessary functionalities required by the intelligent smart parking solution.

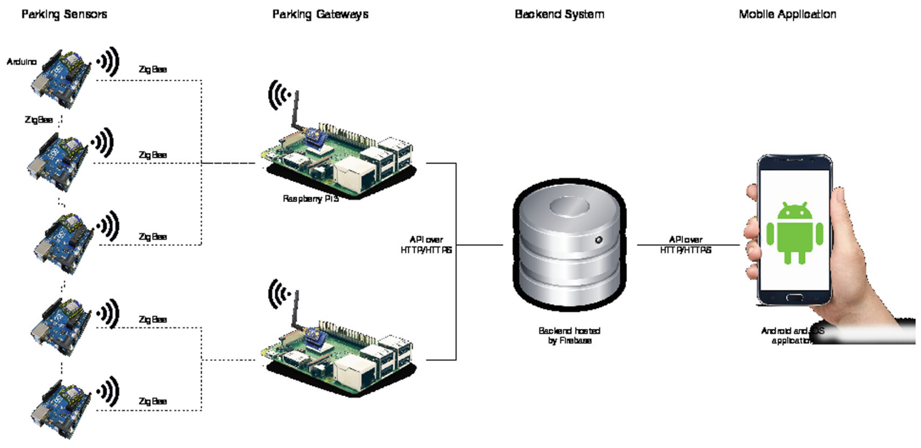


Fig. 1. Integrated smart parking solution architecture

The major components of the system are:

- Parking sensors: the hardware and software cells required to implement the detection of vehicles on the different parking spots;
- Parking gateways: the hardware and software hubs that manage and integrate a set of parking sensors on a parking area and connect the collected data with the system backend, through an Internet connection;
- Backend system: web-based system that is responsible for collecting information from the different gateways, process that information and provide the required information to the mobile application;
- Mobile application: the end-user mobile application that supplies information to the end-user about the availability and location of parking spaces.

In the following sections the different components of the system will be presented and described in detail, specifying their major functionalities.

3.1 Parking Sensors

One of the major components on any smart parking solution is the component capable of detecting the presence of a vehicle on the parking spot, thus allowing the signaling of the availability of the parking place to the system. For this specific prototype, the implementation of the parking sensor is based on an Arduino board integrated with an infrared proximity sensor used to detect the presence of a car over the parking spot. This sensor can detect the presence of obstacles based on the reflection of infrared radiation emitted by a transparent LED and collected by a photoelectric device. This 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. The basic different hardware components that integrate the parking sensor are the Arduino Uno R3 board, a Xbee 2mW Wire Antenna, an infrared sensor, a Shield (connector between the Xbee Antenna and the

Arduino board) and a power supply. The developed sensor is only used to simulate one of the many parking sensors that can be applied for this smart parking solution, it is not a real issue for the smart parking system because one of the objectives is 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 three 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 the XBee transceiver. The sensor software starts by initializing the different elements that are part of parking sensor. Firstly, the proximity sensor is initialized by receiving 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 also defined. After this initial process the sensor enters in a loop mode that enables its execution until some shutdown will need to be conducted. The diagram Fig. 2 displays the different states that might occur during the looping on the Arduino software.

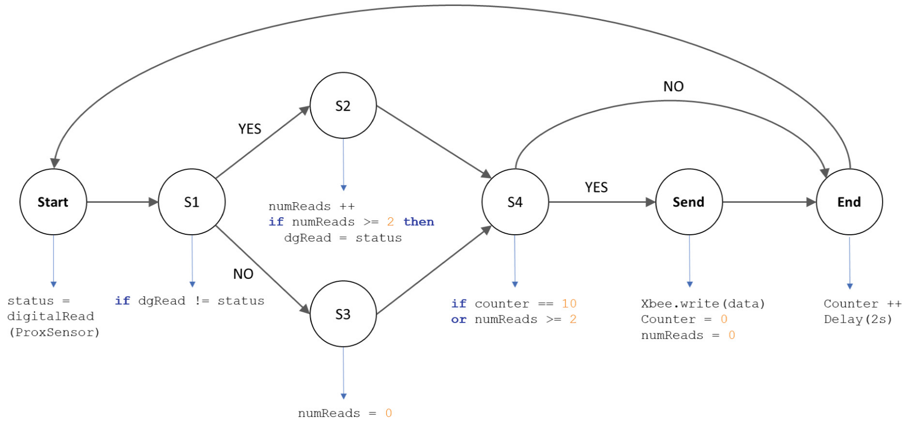


Fig. 2. State machine and pseudocode diagram of the Arduino loop function.

The looping part of the Arduino software contains the logic that handles the collection, processing and communication of data. The first state of this loop 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 is responsible for sending the data through the XBee to the parking gateway. The 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. The variable '**numReads**' must be greater than or equal to 2 (corresponding to the '**S4**' state), 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 important 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 displayed in Fig. 3.

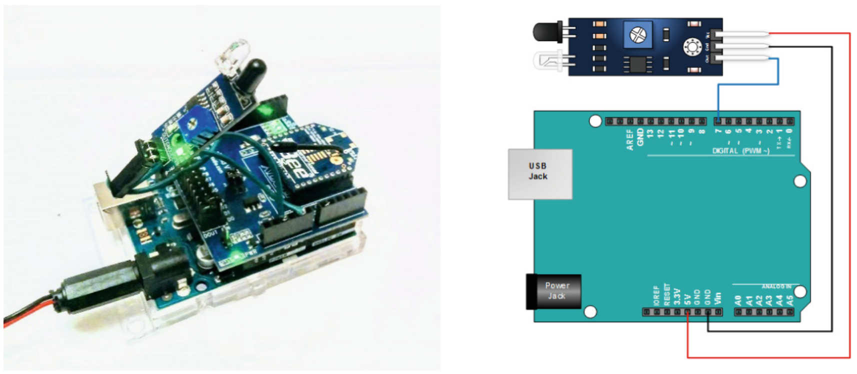


Fig. 3. Prototype proximity sensor and IR sensor connection to the Arduino (Vxlabs 2018).

3.2 Parking Gateway

The parking gateway is one of the most important components 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 (using a REST API, through HTTP/HTTPS). The gateway component is also composed by hardware and specifically designed software to implement its functionality. The hardware component that was selected to implement this gateway prototype was the Raspberry Pi 3, running on the Raspbian Linux operating system. The basic hardware components of the parking gateway are a Raspberry Pi 3, a USB Wi-Fi or 4G adapter, a Xbee 2mW Wire Antenna, a Socket (connector between the Xbee Antenna and the Raspberry Pi) and a power supply. The following image (Fig. 4) 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. The software developed for this gateway also implements a loop that enables the gateway to run forever until it is shutdown. Within this loop the information from the sensors will be collected from the XBee device. This data will then be transferred to a vector which separates a string from a defined character. After this, the correct reception of the data is verified and then the information regarding each specific sensor is sent to the backend database.

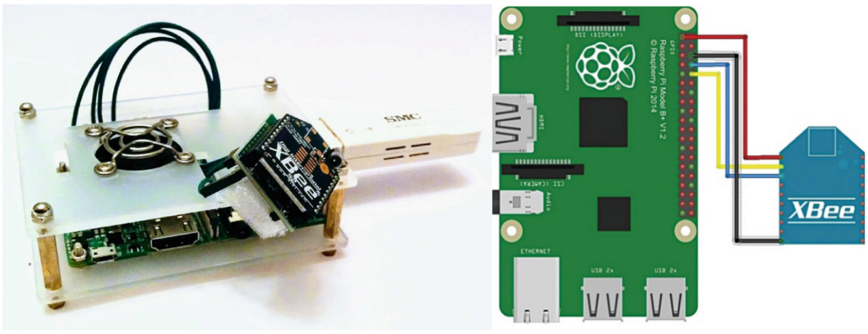


Fig. 4. Gateway prototype and connecting of the XBee transceiver to the Raspberry Pi (Electronics For You 2016).

3.3 Backend System

For the system to work, the backend 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.

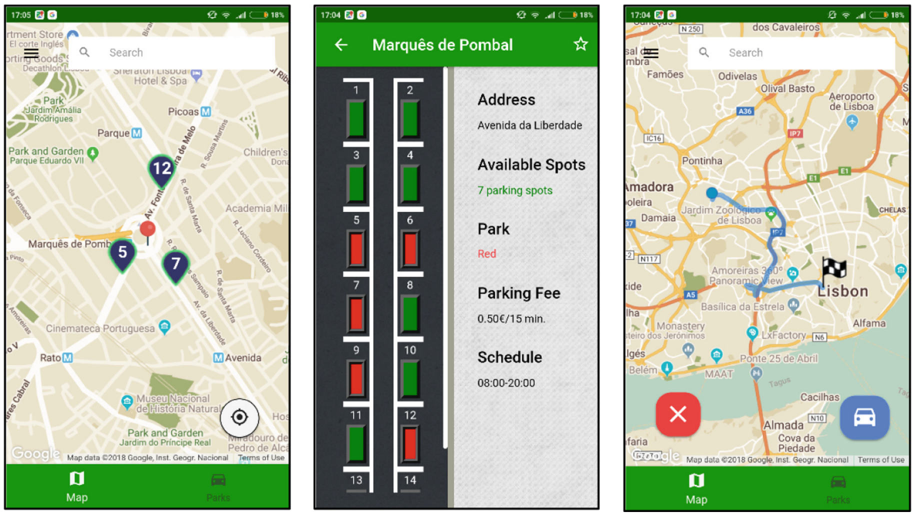
Access to the Firebase backend database is performed through a REST API served through the HTTP/HTTPS protocols.

3.4 Mobile Application

The final objective of a smart parking solution integrated on a smart city is to provide the necessary intelligent and intuitive 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 and give the means to find their way to the available parking spot. Due to the growing usage of mobile technologies, the best solution was the development of a mobile smart parking application. To produce a multiplatform mobile application (available on Android and iOS), an hybrid mobile development framework - Ionic Framework (Ionic Framework 2018) was selected. 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 Fig. 5. 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 displays 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.


Marquês de Pombal

Address
Avenida da Liberdade

Available Spots
7 parking spots

Park
Red

Parking Fee
0.50€/15 min.

Schedule
08:00-20:00

Fig. 5. Application prototype layout.

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.

4 Functional Prototype

As proof of concept for this work, a fully functional prototype was implemented and tested. The proposed prototype consists of two gateways as depicted in Fig. 6, one on the right side and another on the left side, simulating two separate hubs managing two different parking zones, with a total of three parking sensors (two on the right and one the left), and a mobile application for the end-user running on an Android phone (on the bottom).

For testing purpose, one of the gateways is powered by a photovoltaic cell driving a lithium polymer rechargeable power bank, the other gateway runs on a common 5 V USB charger and the parking sensor cells are supplied by 9 V batteries.

For the parking sensor cells, we used infrared sensors as a simple solution for the prototype, that can be easily replaced by a magnetic sensor, specific for the detection of vehicles.

The system was first tested on the software console of the Arduino, and also checked in real-time through the Firebase console. The system is coherent and robust, and the components interconnect correctly as expected demonstrating full functionality and showing status changes on the end-user application in real-time with the sensor stimulation.

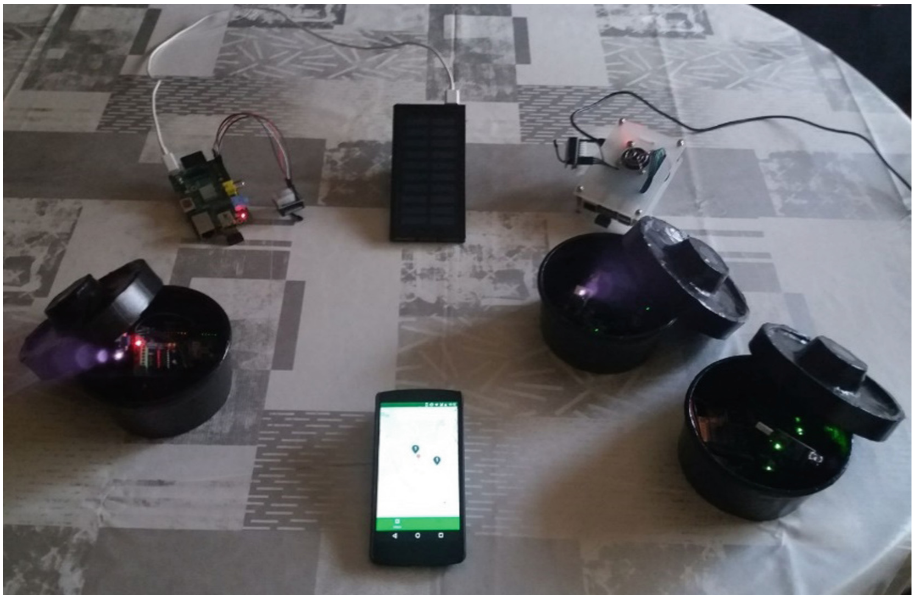


Fig. 6. Complete smart parking prototype test setup.

5 Conclusions

This paper presents a fully scalable low-cost open hardware and software smart parking system for parking management in smart cities alternative to existing studies with less flexible and more costly solutions.

The system consists of sensor cells, parking gateways, a web-based backend database, and an end-user mobile application. This is a complete and integrated system designed for flexibility and allowing diverse component implementation alternative to those selected for the functional prototype, such as other types of sensors, data communication methods, front-end software or backend solutions.

We propose the use of low-cost general-purpose Arduino and Raspberry Pi boards for the implementation of parking sensors and gateways, respectively. The prototype demonstrates the use of ZigBee technology for the communication between the physical elements of the system as a suitable scalable solution and efficient in terms of energy savings and cost. The cost of the implemented prototype is relatively low, when compared with other solutions existing on the market. The total cost of each parking sensors is estimated in around 40 euros, while the cost of the parking gateway is around 75 euros. These represent the costs of the individual components used to assemble both the parking sensors and parking gateway and does not consider any economy of scale effect (as the number of hardware components increases, the price of each individual components lowers). Moreover, it is also important to notice that since all the hardware and software used on the solution are open it does not require a specific company for its implementation allowing anyone to be able to build the parking sensors integrating different types of hardware components. The cost calculation for the solution do not include data communication costs nor the maintenance and operation costs, because they highly depend on the different solutions adopted.

The parking sensor network responds to changes in real-time using the communication between the physical components and allowing the mobile application to show the availability of parking spaces available for the user, even in areas with the highest occupancy and daily demand.

This smart parking system can improve the existing approaches or unveil new solutions that can satisfy the citizens and change the paradigm of traffic and parking as one of the biggest problems in the context of smart cities.

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

References

Adelantado, F., Vilajosana, X., Tuset, P., Martínez, B., Melià, J.: Understanding the limits of LoRaWAN. IEEE Commun. Mag. **55**, 34–40 (2017)

Chourabi, H., et al.: Understanding smart cities: an integrative framework. In: 2012 45th Hawaii International Conference on System Science (HICSS), pp. 2289–2297. IEEE, January 2012

Electronics For You: XBee interfacing Raspberry Pi Model 2 (2016). https://electronicsforu.com/electronics-projects/XBee-interfacing-raspberry-pi-model-2/2. Accessed 15 Nov 2017

Fraifer, M., Fernström, M.: Investigation of smart parking systems and their technologies (2016)

Giuffrè, T., Siniscalchi, S.M., Tesoriere, G.: A novel architecture of parking management for smart cities. Procedia-Soc. Behav. Sci. **53**, 16–28 (2012)

Hancke, G.P., Hancke Jr., G.P.: The role of advanced sensing in smart cities. Sensors **13**(1), 393–425 (2012)

Akyildiz, F., Kasimoglu, I.H.: Wireless sensor and actor networks: research challenges. Ad Hoc Netw. **2**(4), 351–367 (2004)

Mohd, I., Leng, Y.Y., Tamil, E.M., Noor, N.M., Zaidi, R.: Car park system: a review of smart parking system and its technology. Inf. Technol. J. **8**, 101–113 (2009). https://doi.org/10.3923/itj.2009.101.113

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

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

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

Smart Parking Systems: Shape the future of tomorrow's cities (2017). http://www.smartparkingsystems.com/. Accessed 19 Dec 2016

Pham, T.N., Tsai, M.F., Nguyen, D.B., Dow, C.R., Deng, D.J.: A cloud-based smart-parking system based on internet-of-things technologies. Digital Object Identifier, pp. 1581–1591 (2015). https://doi.org/10.1109/access.2015.2477299

Vxlabs: Which jumper to set on the ITEAD XBee shield v1.1 for use with a 3.3 V Arduino (2018). 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 25 Oct 2017