Evaluating ThingSpeak as an IoT Event Platform on building a Smart Parking Application

Aluizio Rocha, Arthur Souza, Dannylo Johnathan, Gibeon Aquino, Rafael Queiroz and Mário Melo

[1]Department of Informatics e Applied Mathematic
Federal University of Rio Grande do Norte (UFRN)
Mailbox 1524 – 57.078-970 – Natal – RN – Brazil

{aluiziorocha,arthurecassio,dannylojohnathan}@gmail.com

{gibeon,rfqueiroz.91,mariovmelo}@gmail.com

Abstract. *Many technologies are needed to build computational systems in the context of Smart Cities. At present, the construction of scalable, intelligent and ubiquitous systems is a constant challenge for developers. The evolution of the Internet of Things is one of the several paths to be pursued with the aid of this development. In this way, new models and platforms have been proposed to support the development of its applications. Allied to this challenge this work presents a generic and a conceptual model of an IoT application for Smart Parking. This model is implemented as RFID and RaspberryPi. In addition, it evaluates the ThingSpeak cloud service for its use as an IoT platform.*

1. Introduction

The Internet of Things (IoT) has gained more and more notoriety in recent years. This paradigm encompasses an infrastructure of hardware, software, and services that connect physical objects to a computer network. It consists of connecting various real-world devices and/or objects, called things so that they interact with other objects and applications with the purpose of collaborating with each other to perform various tasks or to provide useful services and information to people.

According to [Aazam et al. 2014] The IoT architecture usually comprises three layers: a) Sensing layer includes the various objects of IoT responsible for gauging and providing information about the environment (sensors) and for performing tasks (actuators) that in response to the sensed data, will change the environment in which they are inserted; b) Network layer is responsible for managing communication between devices and applications; and c) Application layer, where the applications make use of the data and/or functionalities provided by the devices of the sensing layer. Although the application of the IoT paradigm is interesting in several scenarios, connecting such devices involves overcoming a series of challenges in different layers.

For [Cavalcante et al. 2016] some these challenges are: a) standardization of data, currently the IoT devices do not follow any standardization of the data supply, they provide their data in different formats or units, which makes it difficult to use their information; b) device virtualization models, to facilitate the consumption and use of the functionality provided by the IoT devices, it is necessary to establish interfaces that sponge the data and functionalities of the devices through the Internet, preferably using open Web

standards; c) supporting application development, developing applications that are useful to people and using its devices is not a trivial task; this involves dealing with network issues, heterogeneity of data and devices, device discovery, etc. Some of the promising solutions to these challenges are the use of middleware and/or platforms that facilitate the sharing and consumption of data from it devices, offering different levels of abstractions for information providers and application developers, so that these do not have to deal with the challenges inherent in the different layers to carry out their tasks.

Considering these challenges, this work evaluates the use of ThingSpeak[1] as a platform to support the development of IoT applications. The choice of ThingSpeak was made due to its ease of use and practicality in the development of applications that generate large volumes of data. Other platforms were considered, such as FIWARE and Kaa, but ThingSpeak was much more friendly in the rapid development of these applications. In this way, two guiding questions are formulated: a) ThingSpeak facilitates the development of IoT applications? b) How can IoT applications do efficient use of ThingSpeak? In the specific purpose of answering these questions, the present exploratory study is organized into 4 sections. Section 2 presents the design of a smart parking system, and section 3 approaches how this system was implemented using the chosen technologies. Finally, section 4 shows the conclusion.

2. Related works and Architectures

Smart Parking is a concept that has been researched more extensively after the emergence of the Internet of Things and Smart Cities. This theme has been explored in several works, [He et al. 2014], [Rao 2017], [Lee et al. 2008] and [Ji et al. 2014]. For [Zanella et al. 2014] the development of Smart Parking is a prime service in building an Intelligent City since it brings a benefit easily perceived by the population. IoT technologies form the technological framework that allows the construction of this Smart Parking. This work further explores this theme by proposing an Intelligent Parking prototype easily portable for any IoT platform. Initially, the ThingSpeak platform was used in order to explore its characteristics and identify its strengths and weaknesses.

The solution presented in [Rao 2017] is based on WIFI and microcontroller. The microcontrollers are connected to the WIFI network and have control status of the web via infrared sensors. The driver, when connecting to the same WIFI of the parking, receives the information of which lots are available. Although the solution is simple and apparently low cost, the author does not make clear how to scale this solution. In [Lee et al. 2008] the authors treat intelligent parking with the focus only on sensing. They evaluate two types of sensors: Ultrasonic and Magnetometer for better detection of cars in a smart parking environment. Differently, the present work does not evaluate types of sensing, seeking to present a complete solution for intelligent parking.

The work in [He et al. 2014] presents a conceptual model for the management of parking lot in an intelligent city. In this model cars act as IoT agents that travel on the roads and are alerted about parking lot available. For that to be possible a smart parking service is proposed in a cloud. The cloud receives all vehicle data as well as parking space data. The agents and actors presented in the model were: the lot, the car, the services cloud and the roads. The parking lot and the car must be equipped with devices

[1] https://thingspeak.com

that allow the identification of the car, the lot, and the status of this parking space (free or occupied). The idealized infrastructure considers that the roads have sensory antennas that identify automobiles. Once a car is identified, the data is sent to the cloud service which previously selects available parking lot to properly target the vehicle.

Our work uses the same concepts of layers: sensing, communication and application; and agents: the parking lot and the cars. Furthermore, we experimented the model with a functional prototype of these concepts.

2.1. Conceptual Model and Architecture

Building solutions within the Internet context of Things as well as Intelligent Cities converge on the use and integration of various models and data protocols. This commonly called multilayer approach is used in a number of works [Hachani et al. 2013], [Yang et al. 2012], [Mietzner et al. 2011] and has been initialized and supported by service-oriented architecture. Similarly, this work seeks to delimit, define and implement software and hardware components that provide a layered architecture.

The first step in this journey is to define the conceptual model of the business object and study, in our case, a smart parking system. The papers presented [He et al. 2014], [Ji et al. 2014] corroborate the choice of the layered approach, as well as define theoretical and conceptual models of intelligent parking. The model adopted in this work is based on the one presented in [Ji et al. 2014] and specifies the operation of intelligent parking. Figure 1 shows the elements model mapped to the smart parking system. The elements named A1, A2, A3... A10 represent parking spaces (lots). Each parking lot will be equipped with a sensor (element 4 of Fig. 1) which shall identify the presence of a car parked. Unlike the model proposed in [He et al. 2014], the proposed model in this article do not equip the car with sensors, but sticking an RFID tag to identify the car and the driver. The reading of RFID tag must be performed by RFID sensors (element 3) which are located at the entrances and exits of the parking lot. The parking sensors present in the lots are managed through a microcontroller (element 1). The use of microcontroller as gateway for the lot sensors provides a scalable solution since each microcontroller can handle a suitable number of sensors. Since there is a need to increase the number of parking lot, it will be enough to include more microcontrollers. The effective control of lot reservation and occupation is made by services and applications deployed in the cloud or IoT platform (element 5). The IoT platform should also be responsible for managing communication between the various components of the model. Reservation of a lot and any other communication with the driver will be made via the mobile application running on his Smart Phone. The mobile application (element 6) is the channel of communication with the driver. Finally, elements 2 and 7 respectively represent an LCD screen for messages of common interest to drivers and a dashboard application for administrative use by the parking system managers.

From this conceptual model, we chose the ThingSpeak as the IoT platform to implement a proof of concept in the construction of an smart parking system. The initial idea would be to use ThingSpeak as element 5 of the conceptual model, since it has already been used in previous work [Chandana et al. 2015] as IoT platform. In order to identify the scenarios in which the platform would be used and how the communication would occur among the elements of the model, we mapped three main flows of events

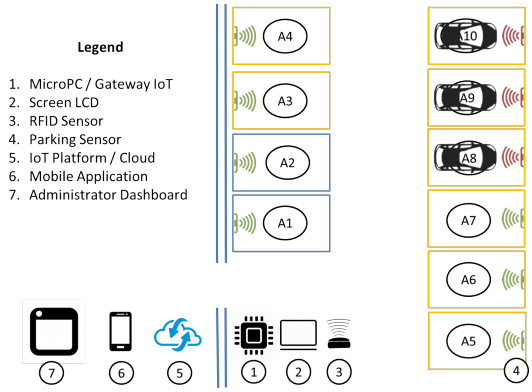

Figure 1. Smart Parking Conceptual Elements

for the application of smart parking. Three main events were mapped in the application, namely: arrival of the car, occupation of a parking lot and the exit of the car from the parking lot.

The arrival of the car, Figure 2, occurs when the driver drives to the parking entrance. When positioning the car near the parking gate, the RFID reader will read the RFID tag sticked in the vehicle. The information obtained will be forwarded to ThingSpeak (event 1 of Fig. 2), which in turn will execute the algorithm for the lot reservation (event 2 of Fig. 2). Finally, ThingSpeak will notify (event 3 of Fig. 2) the parking sensor present in the reserved parking lot and the mobile application to the driver with the location of this parking lot.

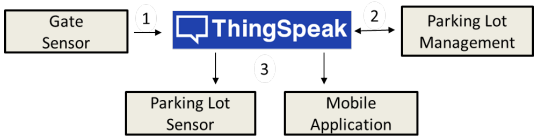

Figure 2. Car arrival events

Figure 3 shows the events related to the occupation of the parking lot changing the status of this space. Thus, when a vehicle occupies or leaves a parking lot, the parking sensor must notify the IoT platform about this (event 1 of Fig. 3). The IoT platform in our model ThingSpeak should update the job reservation service with the new status of the parking space (event 2 of Fig. 3).

Figure 3. Events from occupation/exit of a parking space

The last event flow (Figure 4) was identified when the vehicle leaves the parking lot. The exit notification of the vehicle is mainly important to control the incoming flow of other vehicles when the parking lot is filled to capacity close to the limit. A fact that will avoid congestion and queuing in the parking lot. This event is also important to check the time spent in the parking and assist the driver in fee payments. The time spent in parking is also important information for parking managers to provide an analysis for

future planning. The sequence of these events starts when the car approaches the parking exit gate. The RFID reader present at the exit gate will notify the IoT platform (event 1 of Fig. 4) that a car is leaving the parking lot. Thereafter, the IoT platform will notify the parking space reservation system that the number of total vehicles in the parking lot has decreased (event 2 of Fig. 4), as well as notify the driver's application with the data regarding his stay in the parking lot (Event 3 of Fig. 4).

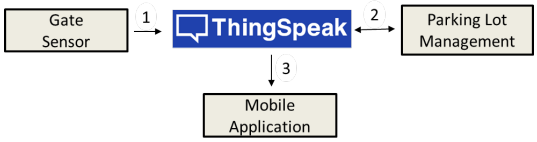

Figure 4. Car exit events

3. Smart Parking on ThinkSpeak

This section addresses the technical design aspects of the smart parking system. In this way, the technologies briefly presented earlier (RFID and ThingSpeak) will be better specified, including an explanation of how they are used in this work. RFID sensing is explained in the Entrance Control subsection. The Parking Lots Management subsection addresses the use of RaspberryPi as an IoT device for lots occupation. The use of ThingSpeak is reported in the ThingSpeak application. The last subsection brings the mobile application built for communication with the driver and their interaction with the platform ThingSpeak.

Building the intelligent parking system using only the previously presented conceptual modeling proved to be a challenge. Extensive use of ThingSpeak has been discouraged due to a number of limitations encountered. In other words, the lack of some communication elements, as well as the partial implementation of some protocols, led to the use of other tools to create a viable prototype. One clear example was the development of an isolated Web Service to control parking occupancy. Initially, this service was thought to be built on ThingSpeak. Figure 5 presents the final system architecture highlighting each element that interacts with ThingSpeak.

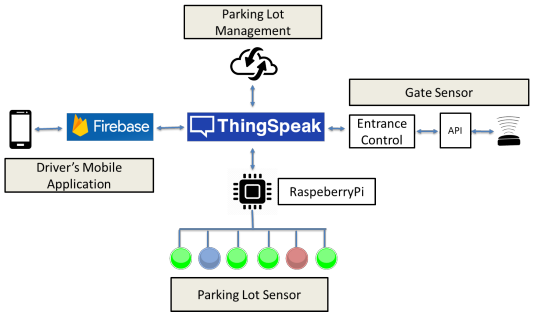

Figure 5. Final architecture of Smart Parking System

3.1. Entrance Control

Looking from an IoT perspective, the things intended to use the services available from Smart Parking are the vehicles. Cars and its drivers are the main actors of a Smart Parking

system. So cars need a unique identity to be used by the services offered by the parking system. In this way, all drivers are abstracted as vehicles and have an RFID ID known to the main application. This unique identification is stored in RFID tags (R-tags). R-tags have a specific frequency ranging from 900MHz to 915MHz that can be captured by the RFID reader.

A basic requirement of the application is to capture correctly the unique identification stored in R-tags. For this, it was designed a module for the entrance control of smart parking based on R-tags reading. The reader used in this project was a long distance NESSLAB NL-RF 1000 Reader, with 900MHz of frequency. Each reader can manage up to four antennas and it has two kinds of communication: Serial and Ethernet. After some tests to find out the range of antennas used, it was possible to discover that the antenna has a directional spectrum, with a reach gain of four meters approximately. Exceeding this limit there is a loss of accuracy and the reader may not read the tags.

The entrance control module is responsible for reading the cars' tags, and it uses the API RFID to communicate with the RFID reader. This API is public and available on Java language in *GitHub* (*https://github.com/dannylo/rfid-readernesslab-api-java*). Through the communication services offered by API RFID running on the RaspberryPi, it was possible the reading of the tags and to transmit its data for the *ThingSpeak* platform, as illustrated in Figure 6. The data sent by API RFID are in JSON format and the RaspberryPi transmits this data through HTTP RESTful to the ThingSpeak channel *entrance_car_channel*. This channel logs all the tags read.

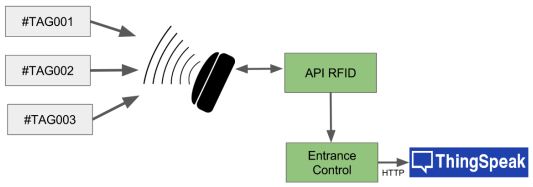

Figure 6. RFID Reader communication schema with ThingSpeak platform.

In order to avoid R-tag reading duplicity, the API RFID have a perception *timer*. This timer is configured by the entrance control module to 10 minutes, meaning that in this period any R-tag can be read only one time. An example of a R-tag entrance control application is available in GitHub (*https://github.com/dannylo/entrance-rfid-app*).

3.2. Parking Lots Management

Whenever a car comes to the parking gate, some free parking lot must be reserved for it. At this moment, a free parking lot is chosen by the system and the driver receives a notification in his Mobile Application about the parking lot reserved to his car.

The Smart Parking application has a web server in the cloud (*https://smartparkingiot.herokuapp.com/app/parkinglot/*) as a system manager with database to store informations about the lots, vehicles and users. The others Smart Parking application components send/request informations to/from the web service and act according to its response. This web service was developed in Java and it is available in GitHub (*https://github.com/rafaelfqueiroz/smartparking*). The web service is RESTful, i.e., the message exchange between web service and others applications occurs over the HTTP protocol and the data are in JSON format.

The web service just gets the first parking lot available and returns it as response to the request. This request occurs when a car comes to the parking entrance and is performed by the entrance control module. At this time, the web service also notifies the driver, the user associated with the car, about the reserved lot through the mobile application.

To produce a simulation in laboratory of the parking lot occupation, we made a prototype (Figure 7) with the components used in the system. This prototype has four parking lots. Each parking lot has a LED showing its status and a push button to simulate status changes. When a lot is free its LED is green, and red when it is occupied. Whenever a car arrives at the gate and the system choses a lot, its LED is going to be blue.

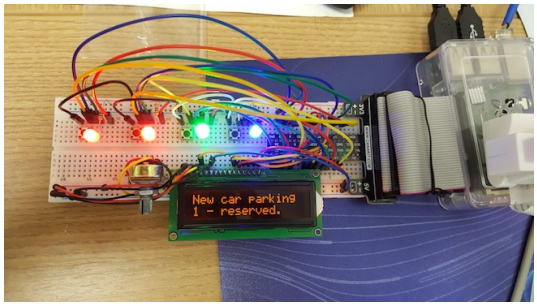

Figure 7. Parking lot prototype.

In real scenario, there will be sensors installed at each parking lot to realize when a car enters or leaves the lot. Movement or distance sensors can be used for this perception in the lot. For the simulation with this prototype, the usage of such sensors, very close to each other, produced some unexpected events. Thus, we decided to replace real sensors with push buttons that change the lot status. For example, pressing the button when the lot is occupied indicates that the car has left and now this lot is free. The status loop is: free - reserved - occupied - free. A 16x2 characters LCD display was used to show informations about the events. In Figure 7, this display is showing that a new car parking event has happened, and lot number 1 (right to left counting) is reserved for this car (blue LED). Lot 2 is free (green LED), and lots 3 and 4 are occupied (red LEDs).

3.3. ThingSpeak Applications

Thing Speak is an IoT platform focused on services that store, visualize and analyse data from devices. The plataform also supports MATLAB code, enabling more complex analyses of the stored data. ThingSpeak platform has some event triggers that are called ThingSpeak Applications (app). These apps are used to transform and visualize data or trigger an action whenever a data insert condition happens. Some of these apps - React, ThingHTTP and TalkBack - were used in this project.

3.3.1. React

The React application works with the ThingHTTP application to perform actions when channel data meets a certain condition. In this project, we used React to program data insertion triggers that notifies the Web Service and the IoT device (Raspberry) about the parking lot events. For the Smart Parking system we developed four React apps:

- *car_entrance_event*: Raspberry notifies the Web Service that a new car has arrived. This React is triggered when inserting any data on channel *car_entrance_channel*. This insertion is performed by the Raspberry when it reads the car RFID tag. This channel has one field only (*tag*) and it logs the cars ID that have used the Smart Parking.
- *reserve_lot_event*: Web Service notifies the Raspberry that a parking lot was reserved.
- *occupy_lot_event*: Raspberry notifies the Web Service that a parking lot was occupied.
- *free_lot_event*: Raspberry notifies the Web Service that a car left its parking lot and now this lot is free.

These last tree React apps are triggered when inserting any data on channel *parking_channel*. This channel has two fields: *lot_number* and *status* that stores these numbers that represent a possible status for each lot: 0 - free, 1 - reserved and 2 - occupied. Raspberry is responsible to inform that a parking lot is free or occupied, and the Web Service is responsible to reserve a parking lot and to notify Raspberry about this.

3.3.2. ThingHTTP

The ThingHTTP app enables the communication with other applications using HTTP protocol. ThingHTTP basically works sending a HTTP request message (GET, POST, PUT or DELETE) to an URL informed in its configuration to inform other application about some data insertion event in the ThingSpeak channel. Three ThingHTTP were created:

- *notify_car_entrance*: this ThingHTTP app is called by the React *car_entrance_event* to send a notification to the Web Service that a car arrived at the gate.
- *notify_rasp_reserved_lot*: forwards the data corresponding to the parking lot chosen for the entrancing car. This event is called by the React *reserve_lot_event* to send the chosen lot number to the TalkBack app *reserve_lot_command*.
- *notify_WS_status_changed*: notifies the Web Service that a car has occupied or left a parking lot. This is triggered by the React apps *occupy_lot_event* and *free_lot_event*.

3.3.3. TalkBack

The TalkBack app enables the ThingSpeak platform communication with any IoT device. It uses a queue structure to send data or commands to the IoT device that periodically queries this queue, i.e. a pooling method notification. The free ThingSpeak account limits successive queries in minimum intervals of 15 seconds.

The TalkBack app *reserve_lot_command* pushes the reserved parking lot number to a FIFO (First In, First Out) queue. Every 15 seconds Raspberry checks if there is any data in this queue, and when it has it updates the status of the parking lot. For each Raspberry query ThingSpeak removes the first entry in this queue.

3.4. Mobile Application

The mobile application was designed for the Android platform and it serves as a mobile user interface for the smart parking system, showing all the lot status for the final user. Furthermore, the mobile application receives a message indicating which lot was reserved to the car associated with the user.

The Android notification service used was the Firebase Cloud Message (FCM), provided by Google as a publish/subscribe service. According to the FCM rules, the Web Service is the publisher and it must have a key representing the credential for using the FCM service (the broker). The mobile app is the subscriber and it must use a temporary token, generated by the broker, to be notified about the publishing of the Web Service. The token represents a user's app session and it is used by the Web Service to notify the mobile app which lot was reserved for the car associated with that user. This association is made through the user and his tag sticked in the car. All tokens are informed to the publisher, the Web Service.

When a lot is reserved by the Web Service for the entrancing car, the Web Service sends to FCM a message in JSON format containing this data: the token that represents the mobile user app that will be notified and the reserved lot. When this message arrives at the broker, it forwards this message to the mobile app associated with the token into the message.

4. Final Remarks

The Internet of Things technologies are being increasingly used mainly with the advent of Smart Cities. Generally the IoT applications are organized in layers of software and hardware elements that integrate to act in the diverse scenarios presented in the construction of these systems. Among the many challenges presented in this way, this work investigates the use of the ThingSpeak platform as an IoT platform. For this, a Smart Parking system was modeled and a proof of concept was implemented using ThingSpeak. In order to evaluate ThingSpeak in a concise way, two main questions were proposed: a) ThingSpeak facilitates the development of IoT applications? b) How can IoT applications do efficient use of ThingSpeak? To answer these questions, some aspects of ThingSpeak have been raised with the Smart Parking system experimentation:

- ThingSpeak is unable to run an entire application developed in Java, Python, C# or C++. Only the available programming language is JavaScript. So, probably it is necessary another platform to run a system with some extra software components.
- MQTT protocol support is partial. ThingSpeak supports only the MQTT publish method, used when a device posts its data to a channel. Platform communication with the IoT device is done through TalkBack application that uses pooling queries with 15 seconds[2] minimum interval.
- Excessive delay in data exchange through the platform. All data exchange uses HTTP(S) protocol over TCP connections that is based on send-reply messages. Thus, if the amount of data is low, all these messages cause unnecessary delays.

Therefore, the ThingSpeak platform is good when analyzing the collected data is the main purpose of the application. Its integration with MATLAB tools greatly facilitates

[2]Limited by the free account.

the development of Big Data analytics and Deep Learning applications. IoT applications, in which the algorithm needs few data, are not so suitable for this platform.

The main purpose of any IoT platform is to facilitate the data exchange between devices and software components. The literature has shown that the MQTT protocol has been used as a very promising and ease type of communication. In this work, probably a simple Web Service with an MQTT publish/subscribe service would not have presented the communication problems aforementioned. As future works we intend to perform more exhaustive tests on the prototype, so as to reap more concrete results for more critical analysis and evaluations by other perspectives.

References

Aazam, M., Hung, P. P., and Huh, E.-N. (2014). Smart gateway based communication for cloud of things. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6. IEEE.

Cavalcante, E., Pereira, J., Alves, M. P., Maia, P., Moura, R., Batista, T., Delicato, F. C., and Pires, P. F. (2016). On the interplay of internet of things and cloud computing: A systematic mapping study. *Computer Communications*, 89:17–33.

Chandana, R., Jilani, S., and Javeed Hussain, S. (2015). Smart surveillance system using think speak and raspberry pi. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(7):214–218.

Hachani, S., Gzara, L., and Verjus, H. (2013). A service-oriented approach for flexible process support within enterprises: application on plm systems. *Enterprise Information Systems*, 7(1):79–99.

He, W., Yan, G., and Da Xu, L. (2014). Developing vehicular data cloud services in the iot environment. *IEEE Transactions on Industrial Informatics*, 10(2):1587–1595.

Ji, Z., Ganchev, I., O'Droma, M., and Zhang, X. (2014). A cloud-based intelligent car parking services for smart cities. In *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, pages 1–4. IEEE.

Lee, S., Yoon, D., and Ghosh, A. (2008). Intelligent parking lot application using wireless sensor networks. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pages 48–57. IEEE.

Mietzner, R., Leymann, F., and Unger, T. (2011). Horizontal and vertical combination of multi-tenancy patterns in service-oriented applications. *Enterprise Information Systems*, 5(1):59–77.

Rao, Y. R. (2017). Automatic smart parking system using internet of things (iot). *International Journal of Engineering Technology Science and Research (IJETSR)*, 4(5):225–228.

Yang, X., Li, Z., Geng, Z., and Zhang, H. (2012). A multi-layer security model for internet of things. *Internet of things*, pages 388–393.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32.