







IoT Android Gateway for Monitoring and Control a WSN

Dixys Hernandez-Rojas^{1,2} , Bertha Mazon-Olivo^{1,2} , Johnny Novillo-Vicuña^{1,2} ,
Carlos Escudero-Cascon² , Alberto Pan-Bermudez² ,
and Gustavo Belduma-Vacacela¹ 

¹ Universidad Técnica de Machala, 5.5 km Pan-American Av, Machala, El Oro, Ecuador
dhernandez@utmachala.edu.ec

² Department of Computer Science, Universidade da Coruña,
15071 A Coruña (03082), Spain

Abstract. Precision agriculture and the automation of agricultural processes such as irrigation with the use of current technologies, create the need to develop IoT gateways for WSN with Bluetooth Low Energy motes that demonstrate their functionality through a testbed. The challenge of developing both the testbed and an application for Android smart phones with gateway functionalities to package the sensor data into MQTT messages and send it to remote server or cloud computing and, in turn, be a tool for monitoring on-site of sensors and controlling the actuators present in the mote motivated the development of this work. We carried out tests on the proposed testbed, which denotes its functionality, and shows there is a fast and stable IoT gateway with low CPU and RAM usage, ready for IoT applications such as Precision Agriculture. The application design is extensible to other OS for smart phones and tablets.

Keywords: Android · Bluetooth Low Energy · Gateway · Internet of Things
Wireless sensor network

1 Introduction

Nowadays, not only people connect to the Internet; but also, millions of objects or things (smart objects) and this number is growing exponentially. The Internet of Things (IoT) is the Internet of physical objects with an intelligence that enables them to communicate with other objects and human beings, for controlling critical or relevant situations to a specific domain or application.

The devices (motes) integrate sensors and actuators; and, these are located geographically forming wireless sensor networks (WSN) [1]. For real-time monitoring and control of smart objects, software applications have increased focus on different IoT domains, some of them are Smart Cities, Smart Homes, Smart Buildings, Smart Healthcare, Precision Farming or Smart Agriculture and others [2, 3]. The desired characteristics of these types of applications according to [4] are automation of data management, data capture, communication, data processing and collaboration with other objects [5].

The domain IoT approached in this paper is Precision Agriculture (PA), known by some authors as Intelligent Agriculture and is defined as the use of Information and Communication Technologies (ICT) in the localized management of crops or

agricultural parcels [6, 7]. An example is drones that come equipped with global positioning systems (GPS), cameras and sensors, that are used to obtain images and geo-referenced data of a parcel or crop, with the purpose of generating maps of performance in Geographic Information Systems (GIS). It helps the farmer to make decisions that benefit the production process by integrating monitoring and controlling systems in real time based on IoT [8]. However, some of these technologies are still being developed and there are several challenges to be resolved. Some of these challenges include the optimization of data capture and communication between Smart objects and the Data Processing Center (DPC) or cloud computing, handling and integrating large volumes of data to generate information for a more accurate evaluation of the optimum sowing density, estimating fertilizers, more accurately predicting crop production, etc. [7, 9].

The purpose of this research work is communication between the wireless devices (motes) and the DPC, through the implementation of an IoT Gateway based on the Android operating system that allows automation of the following processes: (1) Discovering elements of WSN called motes (devices that integrate sensors and actuators); (2) connecting with the motes using Bluetooth Low Energy (BLE) technology; (3) gathering and monitoring of sensor data; (4) controlling of actuators; and (5) Internet connecting and communicating with the DPC or cloud computing that remotely stores, processes, controls and monitors devices installed in an agricultural parcel [10].

2 Background and Related Work

In this section, we make a small revision of the technological effects of the technological linkage and of the gateway system, as well as, similar researches in different IoT gateways.

2.1 Wireless Sensor Network (WSN)

The advances and advantages of modern wireless technologies, such as Wifi, Zigbee, Bluetooth Low Energy, Z-Wave, Lora, Sigfox and LTE, among others, have found their main niche on the Internet. That is why smart sensors [11], also known as motes, incorporate these wireless communication modules that allow them to group and intercommunicate in WSN networks. Currently, we can cite the most popular commercial open-mote hardware: Arduino, Raspberry Pi [12, 13] and RedbearLab [14]. In this paper, we use the hardware RedbearLab nRF51822.

One of the variables most used in IoT applications is temperature [15–19], either to measure the ambient temperature, soil, water or other material or process. The Resistance Temperature Devices (RTD) are devices that vary the metal resistance (mainly Platinum) as a function of the temperature variation, typically characterized by the Callendar-Van Dusen equation [20]. In this paper, we use Platinum RTD PT1000.

2.2 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) is a short-range wireless technology, also known as Bluetooth Smart. Unlike the classic Bluetooth present in computers, keyboards and mice; this technology is low-power, small size, low cost, robust, efficient and allows interoperability between different manufacturers. Nokia created this technology and they called Wibree [21], to be later adopted by Bluetooth SIG [22]. The dizzying development of BLE is due to the fact that the main smartphone manufacturers have incorporated it into their computers with iOS, Android, Apple OS or Windows operating systems.

We establish the communication between BLE devices through a star network topology, where these devices adopt peripherals and central roles. The peripherals are small devices with low power (motors) and the central roles are mainly occupied by smartphones, which acts as a master device for communicating and scanning, while the peripherals act as slaves doing the advertisement. When a central device is connected to peripheral, they make use of the GATT (Generic Attribute Profile) services and characteristics to communicate. For this, we use the popular UART service with two characteristics: one to transmit (Tx) and one to receive data (Rx).

The physical layer of BLE uses the free 2.4 GHz frequency band, shared with other technologies like Zigbee and Wi-Fi. This is the reason, we use the technique of frequency to avoid or reduce interference.

2.3 Android Evolution

Android is a Linux-based operating system, developed by Google. It leads the market with approximately an 82.2% share compared to its immediate follower iOS that has 14.6% [23]. Android controls more than one billion mobile touch devices of different makes and models such as Smartphones, tablets, televisions, car radios, hand watches, etc. [24]. To develop Android applications, we need a software development kit (SDK) known as the Application programming interface (API).

2.4 Related Works

Given the heterogeneity of the sensors used in different IoT application domains, we can find three types of gateways in the scientific literature, which allows us to discover the motes present in the WSN, to gather the data of its sensors and to transmit it in another technology to a server or the cloud. These three gateways are: gateways based on raspberry pi, gateways based on smart phones (using Android mainly) and custom gateways for ZigBee or BLE networks and/or Ethernet or cellular data communication ports (4G or LTE) that access a server or the cloud. This is the case of [25] which develops an efficient IoT Gateway over 5G Wireless. While in [26], an Android App is used as a proxy to connect a BLE sensor from a medical device that does not meet the standard through an MQTT broker within the gateway or middleware developed with the standard IEEE 11073 software stack so that, other applications can access the patient's medical information. In [27], they developed a custom gateway using an IT ZigBee module as a coordinating node and a GSM/GPRS module to send SMS with the data acquired by

the sensors. In [28], they use the raspberry pi as (IoMTaaS) Platform of a WSN based on ZigBee motes. In [29], they also use a raspberry pi as a gateway, in which they implement a docker for ARM to detect BLE motes in their environment.

3 Methodology

This section describes the hardware used in the implementation of the WSN, the programming of the mote, the development process of the mobile application and the communication protocol used between the mote and the mobile application.

3.1 BLE Mote

For the demonstration of our testbed of a WSN, we chose RedbearLab because it is a commercial kit that allows a quick introduction to IoT, since it has connectors with a compatible Arduino pinout (Fig. 1). It allows two methods to burn your firmware either through the J-Link SWD interface or the mbed platform. This kit is based on the Nordic chip nRF51822, in which Nordic is the leading manufacturer of chips for Bluetooth Low Energy.

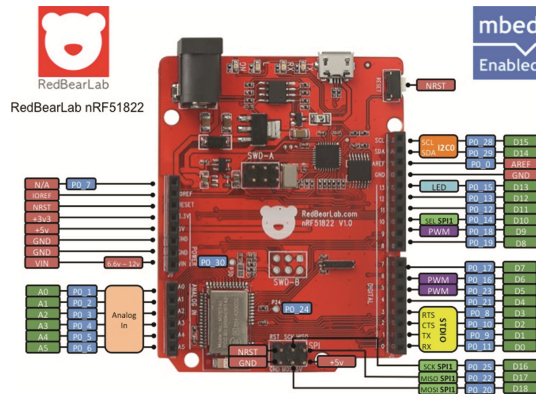


Fig. 1. Mote RedBearLab nRF51822

An important feature of this card is the mbed programming mode is enabled [30]. Enabled mbed mode means that the board has been validated by the enabled mbed program, which guarantees compliance with the criteria and technical requirements for interoperability with other enabled mbed products (Arm community). Some of the benefits of belonging to this program are: the developed code can be downloaded to any hardware with enabled mbed without making any changes; and the availability of an online development tool that allows us to edit, compile and download your application to our PC desktop and drag it onto a development board. Mbed has a large community of programmers, so it has a lot of reusable code. It also includes an OS that allows the programmer to control the hardware and connection with the cloud in a transparent way.

3.2 Testbed

With the chosen hardware, it is possible to evaluate the operation of a gateway based on Android. For this reason, we implement a testbed that allows us to monitor and to control the WSN nodes. To test analogue sensor monitoring, an RTD (PT1000) temperature sensor (DIN EN 60751) may be used within a resistance divider as shown in Fig. 2. We measure the proportional voltage by the AC/DC of the kit and analyze the interpretation of the measurements in the next section. Digital sensors are tested with the use of a simple pushbutton. To check the control on actuators, the use of LEDs is sufficient, as these can be easily replaced by relay to control motors and valves for irrigation control. We added a PWM output to the testbed to demonstrate its functionality in the case of using controllable motors or actuators with PWM signals. In this case, the light intensity of the LED varies depending on the PWM signal sent.

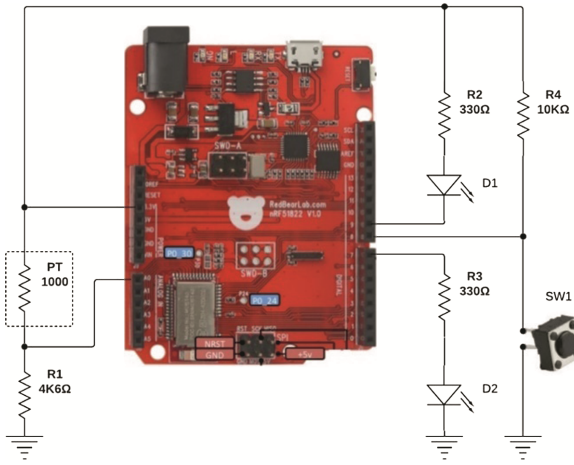


Fig. 2. Testbed for precision agriculture

3.3 Conversion of Measurements to Engineering Units (Pt1000)

In this section we analyze the logic followed for the interpretation of the digital measurements obtained from the Analog Digital Converter (ADC) to bring them to engineering units. In this case, the variable is temperature and the output is given in degrees Celsius ($^{\circ}\text{C}$).

The above-mentioned processor uses a 10-bit A/D converter and 3.3 VDC reference voltage (V_{ref}) for analog measurements, so a least significant bit value (LSB) is equal to:

$$LSB = \frac{V_{ref}}{2^{10 \text{ bits}}} = \frac{3.3 \text{ V}}{1024} = 3,22265625 \text{ mV} \quad (1)$$

$$V_{temp} = LSB * Pd \quad (2)$$

On the other hand, V_{temp} can be calculated by means of a voltage divider formed by the resistor of R4600 and the RTD (Pt1000), see (Fig. 3). So:

$$V_{temp} = \frac{V_{ref} * R4600}{R4600 + RTD} \quad (3)$$

$$RTD = \frac{15180}{V_{temp}} - 4600 \quad (4)$$

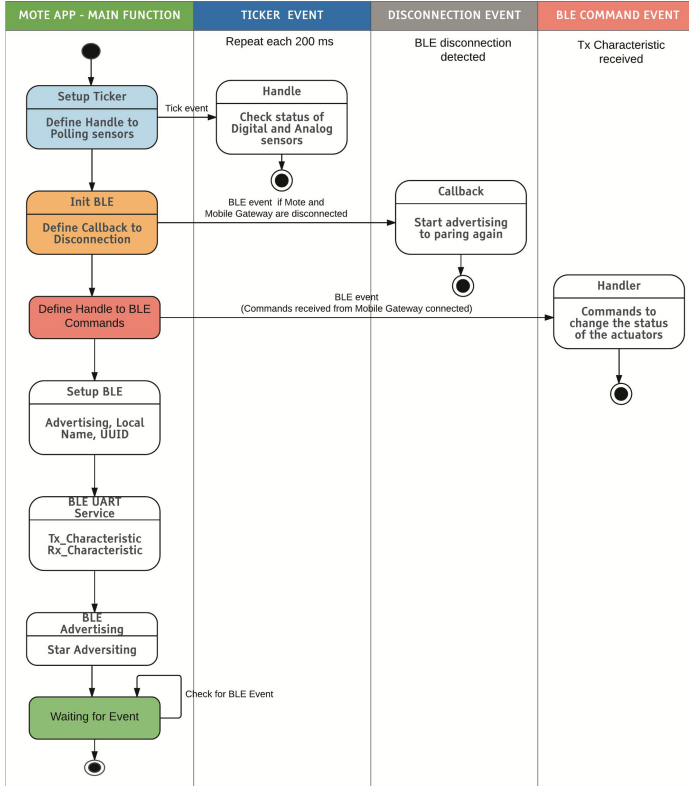


Fig. 3. Activity diagram of the mote firmware

Once the RTD value is known, we can then determine the temperature value using the following approximation:

$$RTD = R_0(1 + \alpha(T_f - T_i)) \quad (5)$$

where R_0 is the resistance at 0 °C, α is temperature coefficient of PT1000 at 0,00385055, T_f y T_i are final and initial temperature, respectively.

$$T_f = \frac{RTD - 1000}{3,85055} [^{\circ}\text{C}] \quad (6)$$

3.4 Mote Programming

The BLE uses UART Service and we program the mote application in “C” language using the BLE UART service. The main function has several parameters of BLE and the application is initialized and configured by itself. The rest of the functions of the mote are for attending to events such as sampling polling of the sensors and the sending of data, disconnecting of the mote with some low energy Bluetooth device and execution of the commands sent from the paired device BLE with the mote, as we can see in the Fig. 3.

Main Function. In this function, we configure the necessary parameters for the correct operation of the testbed, such as sample time for measurements, the creation of handlers and callbacks for each event of the application. After this, the rest of the mote operations are by events.

Disconnection Function. This function attends the disconnect event, when the device that was paired with the mote is disconnected for any reason and restarts the “Advertising” process. Figure 4(a) shows the diagram of the programming logic of the class.

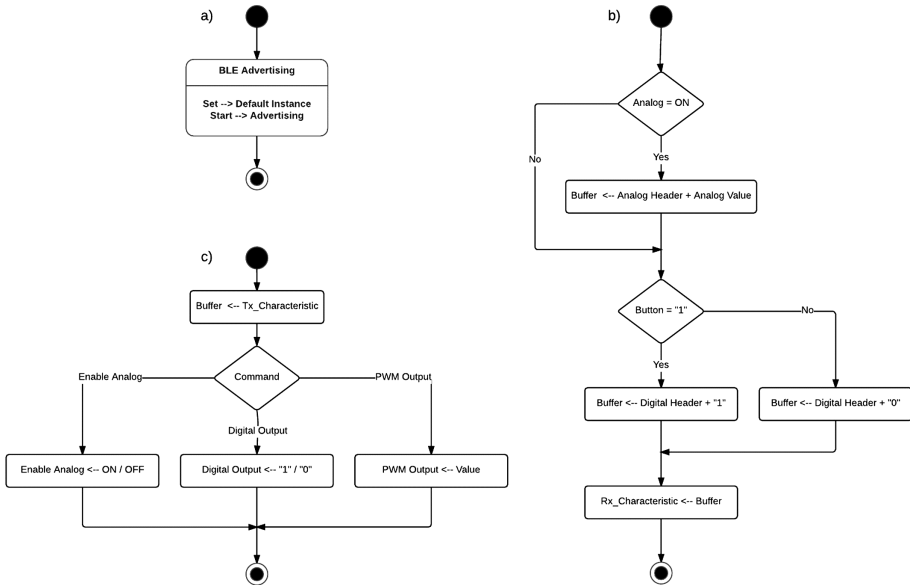


Fig. 4. Events: (a) Disconnection function, (b) Polling function, (c) Command functions

Polling Function. This function records the Tick event of the operating system, so every time a tick occurs, this function is executed. It uses this execution to read the corresponding analog and digital inputs, as shown in Fig. 4(b).

Command Functions. These functions handle the event when the paired device sends data or commands via BLE. First, it extracts commands and data from the “tx_characteristic” according to the established communication protocol.

The operation is then performed by modifying the output variable of the corresponding mote, either as a digital output by turning on or off an LED, or a PWM (Pulse Width Modulation) output with the value defined in the user interface of the smart phone. See Fig. 4(c).

3.5 Communication Protocol Used

As mentioned the mote firmware should be as light as possible, so the communications protocol is light too. In Fig. 5, we can see the protocol used, where the first byte corresponds to the command or action to be executed on one of the mote outputs, if it is digital or analog. Then, there is another byte that indicates which output variable is modified and finally, two bytes indicate the desired value to obtain at the output variable. For the testbed, we use three types of commands, however different types of commands in a byte, are sent up to 256, enough for almost any IoT application.

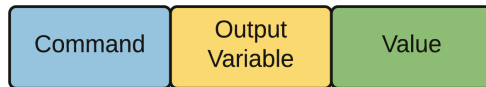


Fig. 5. Communication protocol

3.6 Implementation of the Android Application

In Fig. 6, we show the workflow of the developed telemetry system, which has three main components: mote, gateway and server. The mote or “thing” is in charge of interacting with the specific process. It acts as peripheral advertising every 100 ms in the application, and it waits for a central device to pair. Once a central device discovers it, the central device connects to it, so, both the central and peripheral switch to a state called paired. At that moment, the communication is done through the generic profile UART, with a characteristic (Tx) to send data from the mote to the gateway, and another characteristic (Rx) to receive the commands.

The application on Android (App) has dual functionality, monitoring and controlling the WSN as a gateway. To do this, it verifies that the BLE service on your smartphone is active; otherwise the application is terminated. Then, there are several screens or interfaces for authentication on the server, for interacting in real time with the motes of a WSN and for plotting the sensor data in real time. All these functions are accessible through functional buttons. It then lists all the discovered tags for the user to select who to connect to. Once the smartphone enters a paired state, the application allows it to

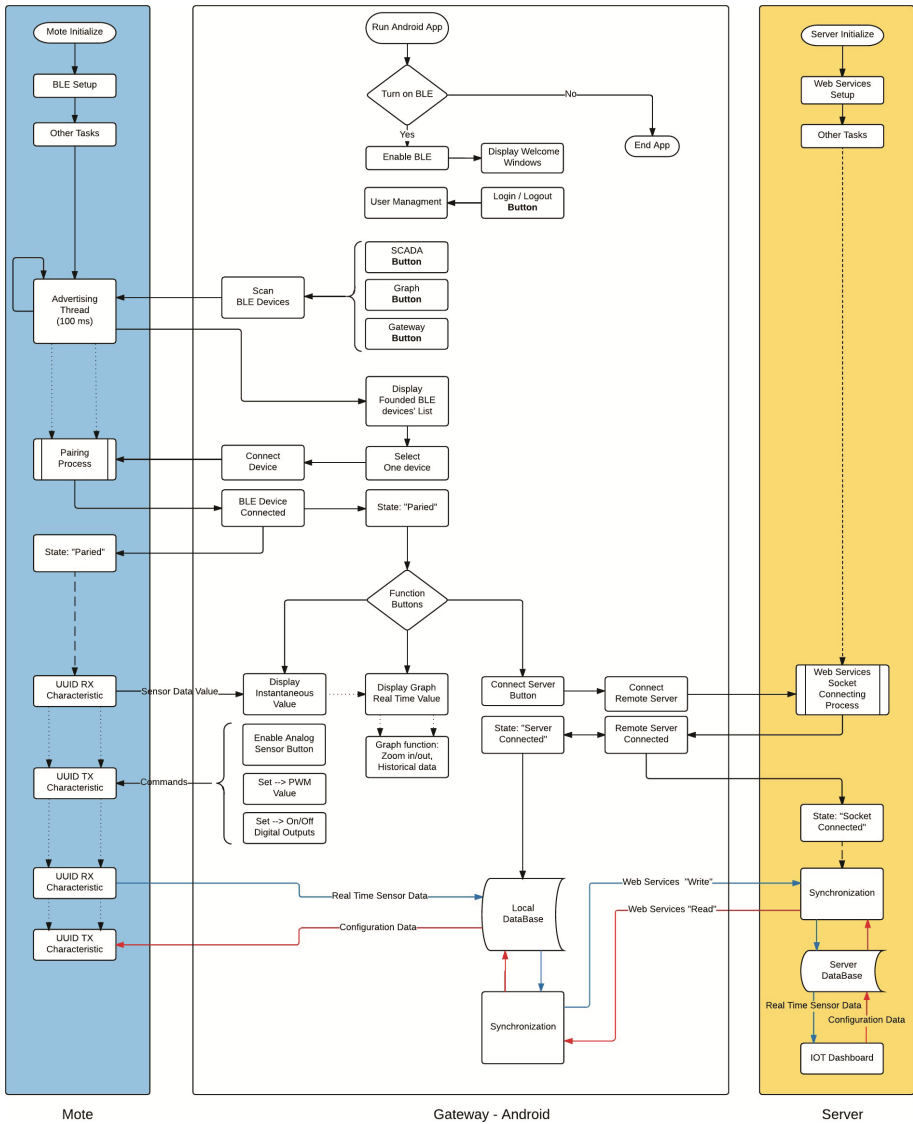


Fig. 6. Telemetry workflow

interact directly with the mote; and as a gateway, to establish a data gateway from the mote to the server and vice versa through a mechanism of synchronization of both ends with data persistence and commands in the local database. Finally, we have the server, where an end user through a dashboard from a traditional browser can remotely monitor the variables of a process in real time, as well as remotely send commands to open or close a solenoid valve, turn on an LED, etc. This is possible in the system thanks to the synchronization mechanisms implemented in the server.

4 Results and Discussion

Following the flow diagram described in the previous section in Fig. 6, we implement a telemetry system by developing firmware for a Mote BLE, IoT gateway and cloud computing. The implementation results are shown in this section.

4.1 Android Application Functionality

We implement the Android application with two requirements: monitoring and controlling in situ a WSN and acting as a gateway between the BLE motes with cloud computing, that allows users to access the data acquired of the IoT application domain (Agriculture) from the web and to send a remote command, for example to turn on or off a water pump or solenoid valve. Figure 7 shows several screenshots of the Android application that demonstrate the fulfillment of the stated objectives. In (a) the initial screen of the application, with a menu of options based on buttons. In (b) an interface that allows the configuration of the necessary parameters to make a connection with a remote server in the cloud. In (c) the first screen that appears when the SCADA option is pressed, where all discovered motes appear. Clicking on one of them, establishes a connection between the mote and the smartphone that permits visualization of instantaneous values of variables that are measuring the mote, as well as the controls necessary to send commands to the actuators, as seen in (d).

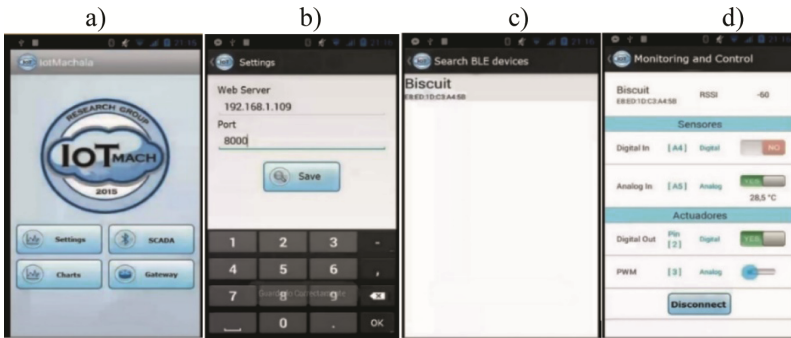


Fig. 7. Mobile gateway screenshots for IoT. (a) Starting screen, (b) Server connection parameters, (c) BLE devices found, (d) Monitoring and control.

4.2 Evaluations

Similar to the functionality demonstration of the application, these are according to the design requirements. Tests are also performed to measure the response rate of the proposed testbed for both local solutions and a complete IoT solution, i.e. the response time from the sensor to the cloud. The performance of the gateway was also measured in terms of the use of CPU and memory of the smart phone with different messenger loads.

The proposed test scenario consisted of three components: Mote, Mobile Application and IOTMACH Server. We used the following tools for gathering the measurements: Software Tera Term to capture the time when the mote packs a new sensor reading and to send it to the gateway. The Tera Term runs on the PC where the mote (RedbearLab) is connected via a USB serial port. Android Studio 2.3.3 was also used on a PC to take the logs sent by the Smart phone (Samsung J500 M, Version of android: 6.0.1) connected via USB to the PC. To capture the arrival time of messages from the Gateway on the server, we used the MQTT SPY, connected to the same MQTT Broker where the Gateway connects to send its messages. The connection between the Gateway and the mote is via Bluetooth Low Energy and via Wi-Fi for connection with the cloud (IoT MACH sever). An overview of this scenario is shown in Fig. 8(a).

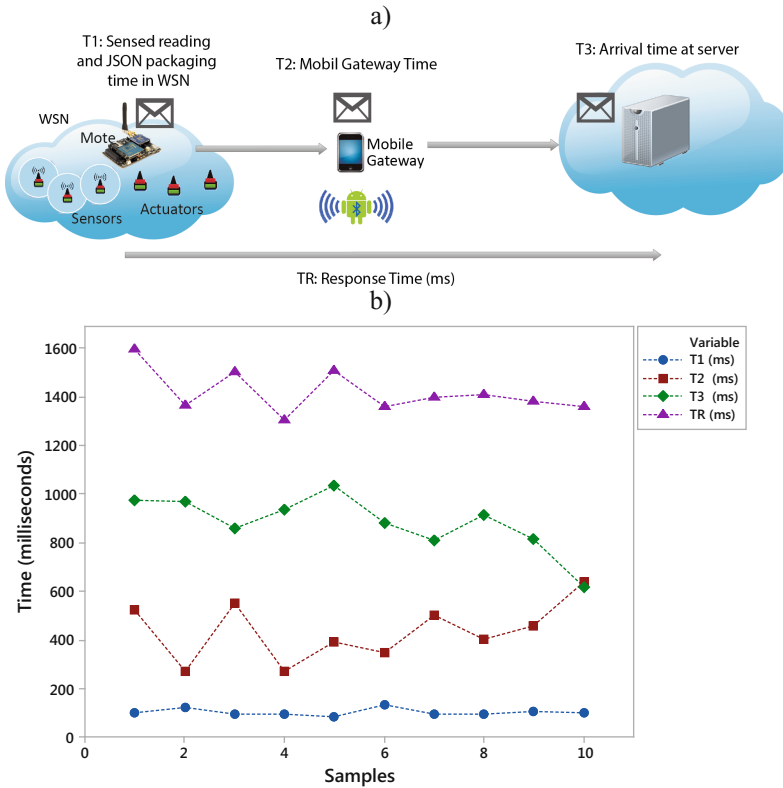


Fig. 8. Response time evaluation. (a) Test scenario, (b) Response time per connections and overall.

The experiment results, sending ten messages from the mote to the cloud can be observed in Fig. 8(b), where T1 represents the delay introduced by the mote to pack the data of the sensor. T2 represents the media delay (BLE) for sending the data between the mote and the gateway. T3 is the delay between the Gateway and the server. These times can vary depending on the medium used, whether Wi-Fi or GPS, 4G or LTE and

depending on the data plan and smart phone used. TR represents the overall time of a sensor reading until it reaches the cloud. In Table 1 you can find more statistical details of the experiment, where we can highlight the speed of our testbed, with an average response time of only 1.4 s. This is excellent for precision agriculture applications where variables like temperature, humidity, etc. vary slowly over time.

Table 1. Response time statistics.

Variable	Average	Standard deviation	Min	Q1	Q3	Max
T1 (ms)	101,5	14,93	80	93,5	111	130
T2 (ms)	435,1	121,3	269	329,8	529	640
T3 (ms)	880,9	118,3	616	812,3	972	1036
TR = T1 + T2 + T3 (ms)	1417,5	88,8	1304	1357,8	1503	1594

In the second experiment, we measured the performance of the Gateway. The Android application was modified to increase the messaging load between the Gateway and the server, which would allow a total of 30 separate observations in three series with 10 iterations for each one. In the first series, 10 messages were sent, in the second series, 100 messages and in the third series, 1000 messages. Using the internal functions of Android Studio, it is possible to obtain the use of CPU and memory for each one of the mentioned series. The results are shown in Fig. 9 and statistical details in Table 2, where we can highlight that the CPU usage is practically constant regardless of the number of messages handled, with a small average variation between 11.79% and 13.30%. The use of the RAM shows a logical behavior proportional to the number of messages transmitted, which has to be processed in the Gateway.

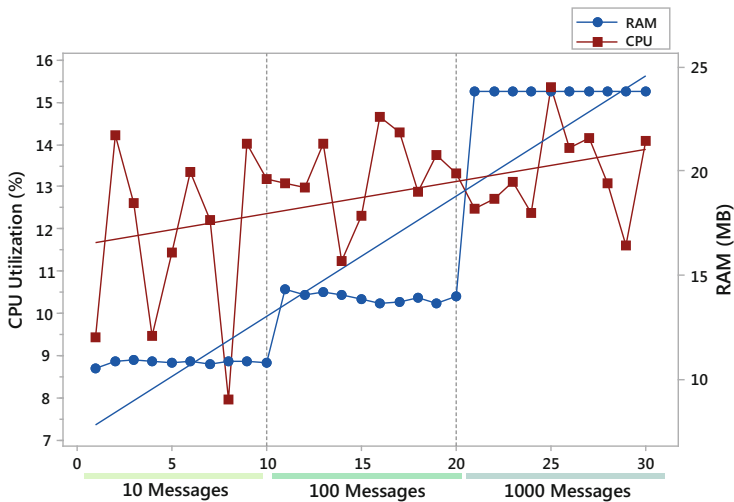


Fig. 9. Performance of the gateway, CPU and RAM utilization

Table 2. Smart phone performance statistics.

#Messages	CPU utilization AVG %	CPU utilization stan. dev.	RAM (MB) AVG	RAM stan. dev.
10	11,79	2,161	10,80	0,102
100	13,27	1,000	13,94	1,104
1000	13,30	1,097	23,87	1,414

5 Conclusions

We develop an Android application in this paper, which discovers BLE tags in the environment and when establishes connections with motes. The application allows the monitoring and controlling in situ of the sensors and actuators connected to the mote. Also, it works as an IoT gateway by receiving the messages coming from the WSN and by encapsulating them to be transmitted via MQTT to the cloud. This allows users of different application domains, as is the case of Precision Agriculture, to remotely, via the web, manage their processes by monitoring variables such as temperature and humidity; or by manipulating electro valves and water pumps to control irrigation. The results of our experiments demonstrate that a fast gateway is available, with low latency and excellent performance in commercial smart phones. The low CPU usage allows the smart phone to perform other functions and operations without affecting the gateway operation. The proportional cost of RAM offers the possibility of handling high rates of messaging or several motes. Following the workflow of the telemetry system shown in this work, this testbed can be implemented on other smartphone operating systems.

References

1. Atzori, L., Lera, A., Morabito, G.: The internet of things: a survey. *Comput. Netw.* **54**(15), 2787–2805 (2010). <https://doi.org/10.1016/j.comnet.2010.05.010>
2. Botta, A., et al.: Integration of cloud computing and internet of things: a survey. *Future Gener. Comput. Syst.* **56**, 684–700 (2015). <https://doi.org/10.1016/j.future.2015.09.021>
3. Chen, S., et al.: A vision of IoT: applications, challenges, and opportunities with China perspective. *IEEE Internet Things J.* **1**(4), 349–359 (2014). <https://doi.org/10.1109/JIOT.2014.2337336>
4. Sarkar, C., et al.: A scalable distributed architecture towards unifying IoT applications. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 508–513. IEEE, Seoul (2014). <https://doi.org/10.1109/WF-IoT.2014.6803220>
5. Campoverde, A., Hernández, D., Mazón, B.: Cloud computing con herramientas open-source para Internet de las cosas. *Maskana* **6**(No. Especial), 173–182 (2015)
6. Mat, I., Kassim, M., Harun, A.: Precision irrigation performance measurement using wireless sensor network. In: 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 154–157. IEEE, Shanghai (2014). <https://doi.org/10.1109/ICUFN.2014.6876771>

7. Bendre, M.R., Thool, R.C., Thool, V.R.: Big data in precision agriculture: weather forecasting for future farming. In: 2015 1st International Conference on Next Generation Computing Technologies (NGCT), pp. 4–5. IEEE, Dehradun (2015). <https://doi.org/10.1109/NGCT.2015.7375220>
8. Zhang, Q.: Precision Agriculture Technology for Crop Farming, 1st edn. CRC Press Taylor & Francis Group, Washington (2016)
9. Ivanov, S., Bhargava, K., Donnelly, W.: Precision farming: sensor analytics. *IEEE Intell. Syst.* **30**(4), 76–80 (2015)
10. Gartner Newsroom: Gartner Says Worldwide Smartphone Sales Recorded Slowest Growth Rate Since 2013. <http://www.gartner.com/newsroom/id/3115517>. Accessed 27 July 2017
11. Islam, T., Mukhopadhyay, S., Suryadevara, N.: Smart sensors and internet of things: a postgraduate paper. *IEEE Sens. J.* **17**(3), 577–584 (2017). <https://doi.org/10.1109/JSEN.2016.2630124>
12. Raspberry Pi 3 Model B Homepage. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Accessed 27 July 2017
13. Novillo, J., Redrován, F., Espinoza, F., Molina, J.: Raspberry analysis in the teaching of computer sciences. *Int. J. Appl. Eng. Res.* **12**(7), 1182–1189 (2017)
14. RedBearLab nRF51822 Homepage. <http://redbearlab.com/redbearlab-nrf51822/>. Accessed 27 July 2017
15. Sreekantha, D., Kavya, A.: Agricultural crop monitoring using IOT - a study. In: 2017 11th International Conference on Intelligent Systems and Control (ISCO), pp. 134–139. IEEE, Coimbatore (2017). <https://doi.org/10.1109/ISCO.2017.7855968>
16. Alahi, M., et al.: A temperature compensated smart nitrate-sensor for agricultural industry. *IEEE Trans. Ind. Electron.* **64**, 7333–7341 (2016). <https://doi.org/10.1109/TIE.2017.2696508>
17. Shailaja, M., Nikkam, G., Pawar, V.: Water parameter analysis for industrial application using IoT. In: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 703–707. IEEE, Bangalore (2016). <https://doi.org/10.1109/ICATCCT.2016.7912090>
18. Maddikatla, S., Jandhyala, S.: An accurate all CMOS temperature sensor for IoT applications. In: 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 349–354. IEEE, Pittsburgh (2016). <https://doi.org/10.1109/ISVLSI.2016.113>
19. Russell, L., et al.: Sensor modality shifting in IoT deployment: measuring non-temperature data using temperature sensors. In: 2017 IEEE Sensors Applications Symposium (SAS), pp. 1–6. IEEE, Glassboro (2017). <https://doi.org/10.1109/SAS.2017.7894057>
20. Sinclair, I.: Sensors and Transducers, 3rd edn. Newnes, Great Britain (2001)
21. Agarwal, K., Sharma, D.: Wireless communication wibree (bluetooth low energy technology). *EEC J.* **2**(2), 1–4 (2017). <https://doi.org/10.24001/eec.2.2.1>
22. Bluetooth Working Groups Homepage. <https://www.bluetooth.com/membership-working-groups/working-groups>. Accessed 27 July 2017
23. Android Homepage. <https://www.android.com>. Accessed 27 July 2017
24. Platform versions. <https://developer.android.com/about/dashboards/index.html#Platform>. Accessed 27 July 2017
25. Saxena, N., et al.: Efficient IoT gateway over 5G wireless: a new design with prototype and implementation results. *IEEE Commun. Mag.* **55**(2), 97–105 (2017)
26. Schmidt, M., Obermaisser, R.: Middleware for the integration of Bluetooth LE devices based on MQTT and ISO/IEEE 11073. In: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE (2017)
27. Dener, M.: A new gateway node for wireless sensor network applications. *Sci. Res. Essays* **11**(20), 213–220 (2016)

28. Maiti, P., et al.: Sensors data collection architecture in the internet of mobile things as a service (IoMTaaS) platform. In: IEEE International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC 2017), SCAD Institute of Technology, Coimbatore, India, 10–11 February 2017
29. Dupont, C., Raffaele, G., Luca, C.: Edge computing in IoT context: horizontal and vertical Linux container migration. In: Global Internet of Things Summit (GloTS). IEEE (2017)
30. mbed IoT Platform Homepage. <https://www.mbed.com/en/platform>. Accessed 27 July 2017

Technology Trends

Third International Conference, CITT 2017, Babahoyo,

Ecuador, November 8-10, 2017, Proceedings

Botto-Tobar, M.; Esparza-Cruz, N.; León-Acurio, J.;

Crespo-Torres, N.; Beltrán-Mora, M. (Eds.)

2018, XII, 231 p. 120 illus., Softcover

ISBN: 978-3-319-72726-4