Comparison of Communication Protocols for Low Cost Internet of Things Devices

André Glória[1,2], Francisco Cercas[1,2], Nuno Souto[1,2]
[1] ISCTE-IUL, Av. das Forças Armadas, Lisbon, Portugal
[2] Instituto de Telecomunicações, Av. Rovisco Pais, 1, Lisbon, Portugal

***Abstract*—Internet of Things (IoT) uses the connection between devices to improve their efficiency and user experience, being the communication one of the main elements for a proper IoT network. This paper presents a review of the most common wired and wireless communication protocols, discusses their characteristics, advantages and disadvantages as well as a comparison study to choose the best bidirectional sensor network composed by low power devices such as Arduino, ESP-12 and Raspberry Pi.**

***Index Terms*—Internet of Things, Sensor Networks, I²C, RS232, ZigBee, LoRa, Arduino, ESP-12, Raspberry Pi**

I. Introduction

In order to provide full control of physical devices over the web the urge to connect everything to the Internet is growing, with an expected 200 billion smart devices being connected to the Internet by 2020 [1]. With the capability of gathering, analyzing and distributing the data, Internet of Things consists in the connection between the Internet and a range of devices and sensors and was designed to play a great role improving our quality of live.

IoT relies on group of devices connected over a network controlled by an application using a communication protocol. These devices can be anything capable of retrieving and storing data, but the most usual are microcontrolers or System on Chip (SoC) devices. SoC [2] is a replacement for a computer, based on an electronic chip or integrated circuit containing the whole computing system. With this approach the cost of producing a computing system has drop, making these devices much cheaper for the end consumer. The most common examples of this platforms are the Arduino [3], an open-source electronics single board based on a simple input/output capable of interfacing different peripherals, sensor and wireless communication devices, the Raspberry Pi [4], a low-cost credit card sized computer with enough processing power and memory, that support programmable I/O ports and the use of standard peripherals, and most recently the ESP-12 [5], a WiFi module based on the ESP8266 core processor, providing the ability to interface with sensors and actuators through I/O pins and offering a complete and self-contained WiFi (IEEE802.11 b/g/n) networking solution as a low-power, low-cost and minimal space device.

For an IoT system capable of creating smart environments [6], the need to adapt to any specification requires that multiple options for devices and communication protocols be available. Fig. 1 shows the system architecture and it is possible to see that two types of devices are essential, an aggregation node that provide a gateway between the user and the Sensor Network and also perform some data analyses, and a sensor node responsible for gather information from sensors and perform user actions. The previously presented devices are capable of performing the task needed for this nodes, with the Raspberry Pi and the ESP-12 being more suitable for the aggregation node, as they have more computational power, memory and WiFi connectivity, whereas the Arduino is the best choice for the sensor node, as it has more I/O pin, with analog inputs available, and lower power consumption. Regardless of the platforms chosen, these two nodes require a bidirectional communication mechanisms in order to exchange the data retrieved from sensors to the aggregation node and also get the user actions from the aggregation node to the sensor node to be applied to the actuators. Other thing to have in mind when choosing the right communication protocol is the need to have a multi-node capability, due to the fact that multiple sensor nodes can be included in the system, all communicating with the same aggregation node. Also, and bearing in mind the low-cost and power saving capabilities of the system, is important that the protocol have these features as well.

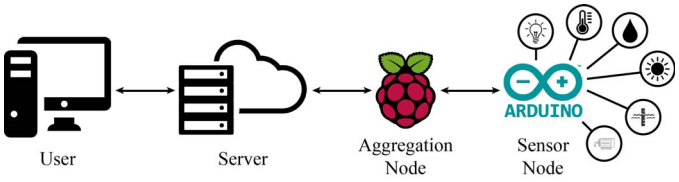

Fig. 1. System Architecture [6]

In this paper the authors present a detailed review of the main communication protocols available, as well as a multiple scenario analysis to comprehend if distance and different board combination affects delay, data rate and efficiency for the tested protocols in order to choose the best communication protocol for each scenario.

II. Communications protocols

Communication is one of the main elements of IoT, as the need to trade data between devices is crucial to the efficiency of an IoT project. Besides the word Internet being part of IoT, not all the situations rely on Internet related protocols. With

the increased use of Sensor Networks and applications in the most diverse environments, the need of different protocols, both wired and wireless, is growing. Wired protocols are still used to connect devices, since they are more reliable, secure and can transfer data at higher rates. The most common wired technologies are Serial Communication or Universal Asynchronous Receiver and Transmitter (UART), mainly in the form of USB or RS232, RS485, SPI, I^2C and PLC. Also CAN bus can be used for networks that need a bigger reliability. Wireless communication protocols, offer all what is needed to work with the advantage of a wireless low space environment. The most common wireless technologies are WiFi, ZigBee, Bluetooth, with new technologies like 6LoWPAN, LoRaWAN and even LTE-A being increasingly used.

Present in Table I are some key characteristics of the main communications protocols.

Without the possibility to test every protocol and bearing in mind that the goal is to use a low-cost, low-power, high range, multi-node communication protocol that also has a low complexity hardware configuration, the chosen protocols to do further tests are I^2C, RS232, ZigBee and LoRa. A more precise description, as well as advantages and disadvantages, of each one can be found in the following subsections.

A. Inter-Integrated Circuit

The Inter-Integrated Circuit [7] was developed in 1982 with the purpose of reducing the number of wiring printed in the PCB and avoid additional logic, using just two wires for connecting the peripherals to the microcontroller, called Serial Data (SDA) and Serial Clock (SCL), making this protocol ideal for communication between integrated circuits and slow communication with on-board peripherals. With this two lines it is possible to connect multiple masters and slaves, making I^2C a multi-master protocol. This two signal lines make communications between devices possible using a protocol that defines a unique 7-bit slave address, used to connect to the bus, 8 bytes containing data and a few bytes for communication control. Besides this two lines, physically, the bus also needs a ground connection. Both active lines are bi-directional making the device that starts the data transfer on the bus a master, being the other slaves.

I^2C has a range of data rates [7] in which we can choose to transmit. *Standart mode*, at 100 kb/s, *fast mode*, at 400 kb/s, and *high speed mode*, at 3.4 Mb/s, are the main data rates for I^2C. But some variants of the protocol add *low speed mode*, at 10 kb/s, and *fast mode +*, at 1 Mb/s, as valid speeds.

Advantages of I^2C [8]:

- Easy to connect;
- Widely supported;
- Automatically configured;
- Low power consumption;

Disadvantages of I^2C [8]:

- Does not support long distance communication;
- Does not support High Speed connections;
- Number of nodes is limited by the address space on the bus;

B. Universal Asynchronous Receiver and Transmitter

The UART is a programmable integrated circuit capable of interfacing serial devices [9]. This interface provide operations such as converting the bytes to a single serial bit stream for outbound transmission, and vice versa for inbound transmission, adding a start and stop bits to signal the beginning and end of a data word, and a parity bit for error detection [9], [10]. To a correct stream of data there is no need for a clock signal since UART is asynchronous but both ends of the line must operate with the same baud rate [11].

There are four different standarts for UARTs: RS232, RS423, RS422 and RS485. The RS232 has a signe-ended line configuration with a Simplex or Full Duplex operation, being able to achieve up to 15 meters at 20 kbits/s.

Advantages of UART:

- Widely supported;
- No clock signal needed;
- Robust to errors;

Disadvantages of UART:

- Limited size of 9 bits;

C. IEEE 802.15.4 (ZigBee)

Introduced in 2002, ZigBee uses the IEEE 802.15.4 protocol as a base. Created for low-rate wireless private areas networks (LR-WPAN) it is one of the most used communication protocols for IoT due to is low consumption, low data rate, low cost and high message throughput. It can also provide high reliability, security, with both encryption and authentication

TABLE I
MAJOR COMMUNICATION PROTOCOLS CHARACTERISTICS

Feature	**SPI**	**I^2C**	**RS232**	**RS485**	**WiFi**	**Bluetooth**	**ZigBee**	**LoRa**
Based Data Rate [Mbps]	20	0.1	0.02	10	11	1	0.25	0.11
Frequency [GHz]	-	-	-	-	2,4	2,4	2,4	0.433
Version	-	Standard	-	-	802.11b	4.0	-	-
Range [m]	100	10	15	60	1-100	10-100	10-100	2000
Nodes/Masters	3	1024	256	256	32	7	65540	-
Power Consumption [mA]	-	-	-	-	100-350	1-35	1-10	1-10
Complexity	Medium	Low	Low	Low	High	Medium	Medium	Low
Security	-	-	-	-	WPA/WPA2	128 bit	128 bit	128 bit

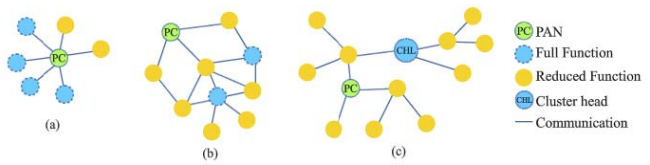

Fig. 2. IEEE 802.15.4 topologies. (a) Star, (b) Peer-to-Peer, (c) Cluster-tree

services, works with different platforms and can handle up to 65000 nodes.The IEEE 802.15.4 works in three frequencies, with a DSSS method, capable of transferring data at 250 kbps at 2.4 GHz, the most usual [8], [12]. To ensure that collisions are avoid, the IEEE 802.15.4 MAC sublayer uses the CSMA/CA protocol. The MAC sublayer is also responsible for flow control via acknowledged frame delivery, frame validations as well as maintaining network synchronization, controlling the association, administering device security and scheming the guaranteed time slot mechanism [13].

This standard can support two types of network nodes, Full Function Device (FFD), responsible for creating, controlling and maintain the network, and Reduce Function Device (RFD), simple nodes with low resources being just able to communicate with the coordinator in a star topology [12], [14]. Fig. 2 shows the topologies for the IEEE 802.15.4 protocol.

Advantages of ZigBee [8]:

- Low power consumption, allows devices to operate from batteries;
- One coordinator can control a numerous amount of slaves;
- Self-organizing network capabilities;
- Highly secure, with 128-bit AES encryption;

Disadvantages of ZigBee [8]:

- Setting up the network requires additional devices, which can increase costs;
- Low data transmission, only 127 bytes per message;
- Is incompatible with other network protocols and lacks Internet Protocol support;

1) LoRa: A bidirectional communication protocol that uses the LoRa physical layer in order to provide low power long-range communications. To achieve this, LoRa is based on Chirp Spread Spectrum (CSS) modulation that have the same low power characteristics as FSK modulation (present in great part of the other wireless communication protocols) but with a significant increase in the communication range. With a single base station it is possible to cover up to hundreds of square kilometers. To guarantee that all communications are completed, the LoRa MAC layer is responsible for join and accept the end-point and the gateway, schedule the receiving slots for end-points and confirm the reception of the received packets [15]. LoRa uses a star topology in order to maintain the low power long communication viable, reducing complexity and increasing network capacity and lifetime as opposed by a mesh topology. This is possible by using a higher link budget, lower license-free frequency such as 433 or 900 MHz and software adaptive power output with transceiver modules [16].

Advantages of LoRa [16]:

- Low power consumption, allows devices to operate from batteries;
- Long communication range;
- Highly secure, with 128-bit AES encryption;

Disadvantages of LoRa:

- Low data transmission, only 55 bytes per message;

III. Tests Scenarios

To ensure that the best communication protocols were used in the system, a set of tests was applied to the network nodes in order to choose the best one.

These tests focus on the exchange of strings of data between the aggregation node, the master, and the sensor node, the slave. The main goal was to see if the communication protocol can work with these platforms, how well they perform when increasing the distance between nodes and also to compare the results with theoretical ones.

For these, three different tests were made in order to see the following values:

- Throughput;
- Message Delay;
- Efficiency;

Also the complexity of setting the system to work and master slave combination were evaluated.

For the first test, the master sends a continuous amount of data packets, with no interval between them, and the slave just receives them, registers the timestamp and increases the number of packets received. Then we see how many packets the slave can process in a second, in order to obtain the true system throughput.

In the second one, a similar method is used, but this time only 100 packets are sent with an interval of one second. The slave receives the packets and registers the time between packets. Then the average time is calculated, in order to achieve the delay, over the one second, that the system needs to receive the packets.

In the last one, 1000 packets were sent to the slave and the slave only increase the number of received packets if the data is equal to the one sent. This way it is possible to see how many packets were lost or have errors.

All the tests have been done with a distance of 1, 2, 5, 10, 20 and 50 meters between nodes and with the following board combination, as both master and slave: Raspberry Pi - Arduino, Arduino - Arduino, ESP12 - Arduino, in order to evaluate every scenario possible.

A. Complexity

For I^2C and RS232, Arduino, ESP12 and Raspberry Pi platforms have native pins to use this protocols, so no additional hardware is needed. ZigBee and LoRa need the external radio interfaces in order to communicate. For ZigBee a pair of XBee S2, an Arduino Wireless Proto Shield and XBee Explorer were

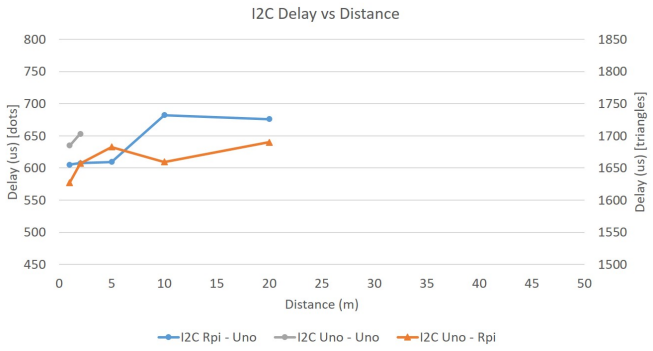

Fig. 3. I^{2}C Delay vs Distance results

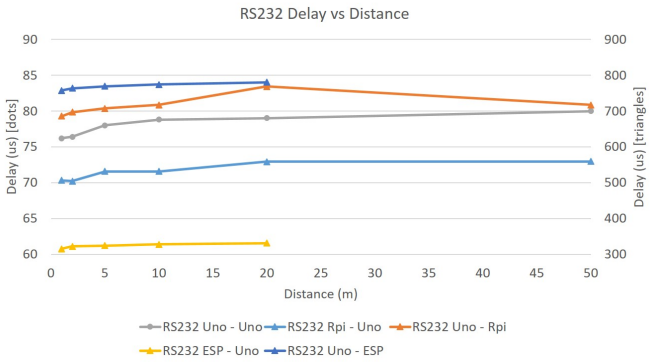

Fig. 4. RS232 Delay vs Distance results

used to connect everything. For LoRa only a pair of Adafruit RFM69HCW is needed.

In terms of software, for I^{2}C in the Raspberry Pi the smbus library was used and in the Arduino and ESP12 the Wired library. For a correct communication only the slave address needs to be set on both platforms as well as the callbacks on both ends. I^{2}C does not allow the Raspberry Pi to be a Slave, so some modification to the code was needed in order to set the Arduino as a Master. Also a minimum of 1 ms was needed between packets in order to avoid overflow of packets when sending to the Arduino.

For RS232 the communication works as a simple Serial Communication with the RX and TX ports on both ends, with a simple modification on settings of the Raspberry Pi in order to tell the OS not to use that port for console communication. Also the pySerial library was needed.

For ZigBee, the most complex configuration, it was necessary to pair the radios using XCTU software. After that each radio was assigned with an address that the other end needed to know in order to send the packet. On the Raspberry Pi the XBee library was used and on the Arduino and ESP12 a 10ms between packets was needed to ensure a correct transmission of packets.

For LoRa, the RadioHead library was used on all platforms. To configure it only an address for each node had to the chosen, guaranteeing it was different for each node.

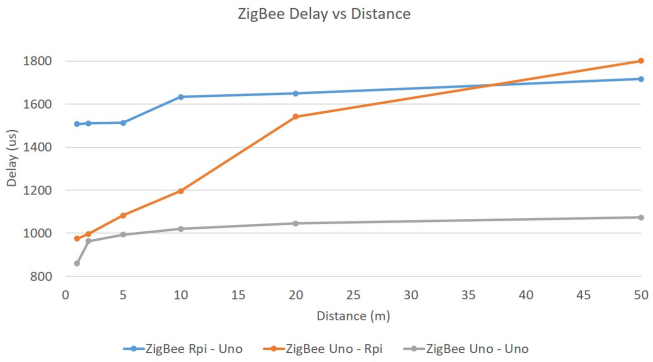

Fig. 5. ZigBee Delay vs Distance results

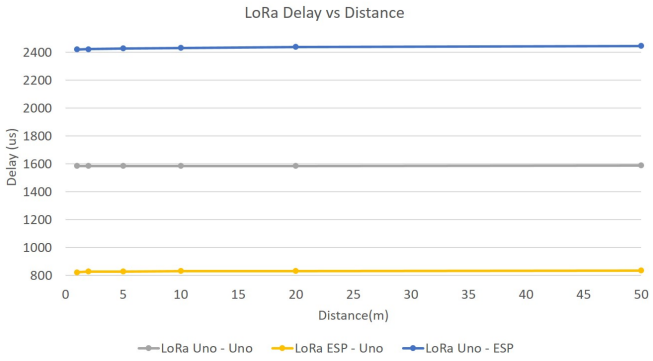

Fig. 6. LoRa Delay vs Distance results

IV. Results

The first thing to notice is that some protocols were not able to connect in some platforms, I^{2}C with ESP-12 is one of the cases and also LoRa with the Raspberry Pi. The ESP-12 and ZigBee connection can be done, but due to ZigBee interference with WiFi, the results were not viable thus it was discarded as a possibility. One disadvantage detected was the fact that the Raspberry Pi can not be a I^{2}C slave, having the Arduino to wait a second between packets and therefore affecting the results.

With these delay results, Fig. 3, 4, 5 and 6, it is possible to understand that the increase of distance does not affect much the delay value, with only a slight increase in all protocols. RS232 has the best results in all scenarios, with a delay ranging from around 75 to 800 us. LoRa has the higher delay with values ranging from 800 to 2500 us and also the biggest variation between scenarios. As of distance, the only one really affected is ZigBee with a bigger increase in delay over 10 meters in all scenarios. Regrading the master slave combination, is when both platforms are the Arduino Uno that the best results are achieved and, with only one Arduino Uno present, the delay decreases when the Uno is the slave.

Regrading throughput results, Fig. 7, 8, 9 and 10, it is possible to understand that distance affects the results, with a different pattern in each scenario, and that in every protocol the maximum value reached is widely different from the

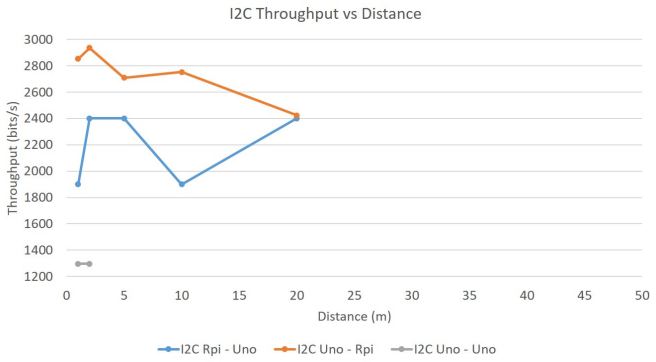

Fig. 7. I^{2}C Throughput vs Distance results

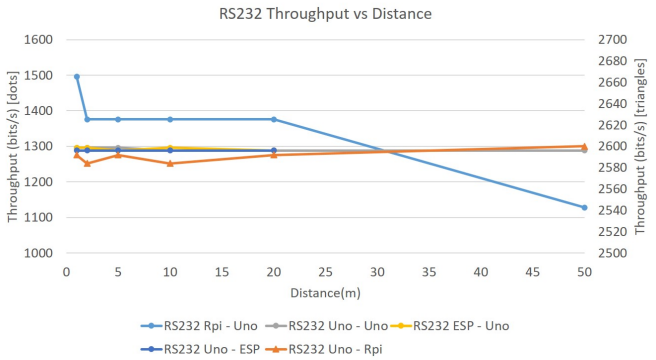

Fig. 8. RS232 Throughput vs Distance results

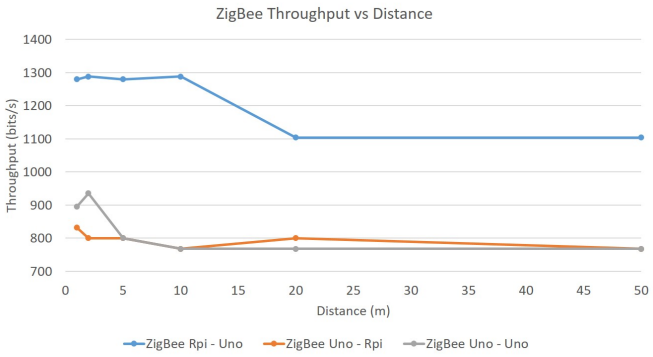

Fig. 9. ZigBee Throughput vs Distance results

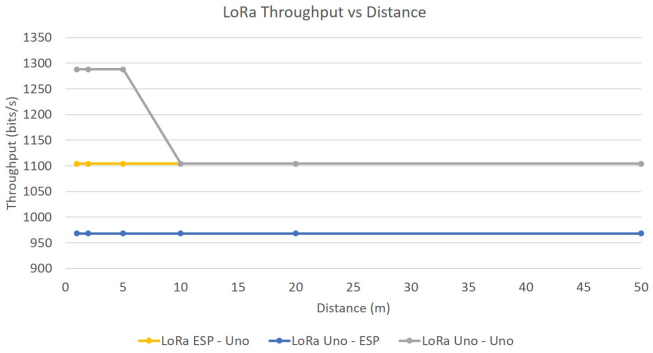

Fig. 10. LoRa Throughput vs Distance results

theoretical values. In some cases there is up and downs with the increase of distance but always with slight variations, so that was not taken in consideration. I^{2}C has the most inconsistent results with big variations in both distance and master slave combination, with a data rate raging from 1.3 to 2.9 kbits/s. On the other hand, RS232 has similiar data rate results, raging from 1.1 to 2.6 kbits/s, but with a more consistent and linear results. LoRa and ZigBee got very similar results, ranging from 0.75 to 1.3 kbits/s, with LoRa having the most linear results, almost none variation with the increased distance. Regrading the master slave combination, the best results are achieved when only one Arduino Uno is present, with some variations as it is the master or the slave.

As for the efficiency tests, I^{2}C had the worst results as communications at 50 meters were not able to connect and when both platforms were Arduino Uno, nothing over 2 meters worked. This could be improved with additional hardware such as I^{2}C range extenders. RS232, when connecting ESP-12 and Arduino Uno was not able to connect over 20 meters. Every other scenario tested got a 100% efficiency rate, including LoRa and ZigBee for every distance.

V. Conclusions

In this paper the authors focused on comparing communication protocols, to choose the best for a set of scenarios using low-cost devices, based on delay, data rate and efficiency. Concluding the communication research, RS232 is the better choice when a wired protocol is needed and LoRa is a more reliable choice for a wireless protocol mainly because of the low complexity and cost needed and the fact that does not interferes with WiFi. Another focus was choosing the best devices for each node of the system, based on previous stated characteristics and the communication tests done. With all these results in mind, it is possible to say that the best suited combination includes an Arduino Uno as an Sensor Node and the ESP12 as the Aggregation Node. This last decision takes place not only because it is a cheaper solution, not putting in risk the reliability of the system, capable of sustaining the node specifications but also due to the Raspberry Pi 3 extra features, such as Internet browser, file systems and others, are not needed for the system, it has a bigger power consumption and the RS232 lines are also used in the system console, meaning that some interference can exist.

Acknowledgment

This work was partially supported by the FCT - Fundação para a Ciência e Tecnologia and Instituto de Telecomunicações under project UID/EEA/50008/2013.

References

[1] J. Gantz and D. Reinsel, "THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," *IDC iView: IDC Anal. Future*, vol. 2007, pp. 1–16, 2012.

[2] A. Ghosh, "Intelligent Appliances Controller Using Raspberry Pi Through Android Application and Browser," *IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016.

[3] M. Banzi, *Geting started with Arduino*, 2nd ed., O'Reilly, Ed. Make:Books, 2011.

[4] C. Bell, *Beginning Sensor Networks with Arduino and Raspberry Pi*, 1st ed., Apress, Ed. Apress, 2013.

[5] AI Thinker, "ESP-12E WiFi Module Datasheet," 2015. [Online]. Available: https://mintbox.in/media/esp-12e.pdf

[6] A. Glória, F. Cercas, and N. Souto, "Design and implementation of an IoT gateway to create smart environments," *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017)*, 2017.

[7] F. Leens, "An introduction to I2C and SPI protocols," *IEEE Instrumentation and Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.

[8] A. Hafeez, N. H. Kandil, B. Al-Omar, T. Landolsi, and A. R. Al-Ali, "Smart home area networks protocols within the smart grid context," *Journal of Communications*, vol. 9, no. 9, pp. 665–671, 2014.

[9] U. Nanda and S. K. Pattnaik, "Universal Asynchronous Receiver and Transmitter (UART)," *3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 22–23, 2016.

[10] L. Frenzel, "What's The Difference Between The RS-232 And RS-485 Serial Interfaces?" [Online]. Available: http://ocwitic.epsem.upc.edu/assignatures/se/recursos/rs232-vs-rs485

[11] M. Sharma, N. Agarwal, and S. Reddy, "Design and Development of Daughter Board for USB-UART Communication between Raspberry Pi and PC," *International Conference on Computing, Communication & Automation*, 2015.

[12] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, 2015.

[13] N. S. Bhat, "Design and Implementation of IEEE 802.15.4 Mac Protocol on FPGA," *Innovative Conference on Embedded Systems, Mobile Communication and Computing (ICEMC2)*, pp. 1–5, 2011.

[14] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007.

[15] LoRa™ Alliance, "LoRaWAN™ Specification," 2015. [Online]. Available: https://www.lora-alliance.org/portals/0/specs/LoRaWANSpecification1R0.pdf

[16] LoRa® Alliance Technical Marketing Workgroup, "LoRaWAN™ What is it? A technical overview of LoRa ® and LoRaWAN ™," *LoRa® Alliance, San Ramon, CA, White Paper*, 2015.