Development of a medical monitoring system based on the Internet of Things

J. Bermejo Torres

Higher Polytechnic School, San Pablo CEU University, Madrid, Spain, j.bermejo1@usp.ceu.es

Abstract

The aim of the present project is the development of a wireless, little size, low cost and scalable platform, based in IoT. It will be capable of measure different physiological signal (temperature, pulse and movement) and internal device signal (temperature). Then, it will send them together in the same packet through Wi-Fi to a network, or the Internet, using the TCP/IP model to finally be received by a server and represented it in a graphical format. The final use of it is to monitor patients in medical environments remotely by specialists.

1. Introduction

The Internet of Things (IoT) term refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves [1]. In this context, Cisco states that IoT devices will account for 50 percent (14.7 billion) of all global networked devices by 2023 [2].

There is no doubt that this technology is growing unstoppably and every day more companies and sectors incorporate it. One of the industries that will have to face this change will be the medical instrumentation industry to connect all its medical devices to the Internet, which, whatever their function, are ultimately intended to obtain very useful physiological data for later analysis and according to them an adequate decision making.

Nowadays, there is an abundance of proprietary IoT healthcare platforms, for example, Apple Watch Series 3 combined with "*Health*" App, or Samsung Galaxy Watch 4 and "*Samsung Health*" App, can monitor the ECG, the oximetry, steps or sleep. But these systems and devices are very expensive, with watches prices above 400$ and phones between 600$ and 1100$ (smartwatches can't work properly without a smartphone). In many cases, these products are not scalable or modifiable because don't provide open-source software or hardware, and technical documentation too. Also, they need to be connected through bluetooth to the phone, so two connections (first bluetooth, then Wi-Fi or 5G) must be established and the probabilities of failures increase. This architecture could be minimized using only one connection to transfer data to the cloud.

Also, there are specialized healthcare monitoring IoT hardware platforms like *Mysignals HW*(*Libelium*). This device can collect a lot of physiological signals such as temperature, glucose in blood, electrocardiogram, electromyogram, etc. And all open source and scalable, due to it is based on Arduino. Another example of multisensory hardware platform for measure physiological signals is *Bitalino*, but it only works with Bluetooth and is too expensive, starting at 400$ the basic, being an educational platform.

Nowadays, there is a tendency to create open source IoT devices. The possible design of efficient medical IoT sensor nodes in terms of low-cost, low power-consumption, and increased data accuracy based on open-source platforms is covered by Nikos Petrellis et al. [3]. Their solution is performed with an Arduino UNO attached to a Raspberry Pi, but the costs of these elements (together more than 60$ nowadays) are still high compared to this project objective. And Sugathan et al. [4] create a generic clothing technology to measure SPO2, electro dermal activity, and body temperature implemented with an Arduino Lily-pad, although they send data through serial communication and they don't use a wireless model, indeed it's one of their future enhancements.

From a broad point of view, the final destination of this project is the design and implementation of a platform based in IoT, improving the requirements followed by Nikos Petrellis, to monitor patients of a Hospital or old people's home in a non-intrusive way when it's not necessary.

2. Technical Proposal

The main purpose is to develop a platform that reaches the following objectives:

- First, a wearable small size device with a wireless connection to transmit the data collected from its sensors, focusing on a TCP/IP model for that purpose.

- Second, a low cost system, using components easy to buy nowadays on the Internet and cheap compared with others actual similar models.

- Third, a concentrated multi-sensor device capable of measuring different physiological signals.

- Fourth, a scalable IoT project. Sensors could be replaced or added depending on pins available and the protocol connection used.

To fulfill these mentioned requirements, a system is designed based on the development of an IoT device to acquire physiological and internal device data by sensors;

then the packeting (JSON) and transfer of them with a TCP/IP model and finally their reception in an extern server.

The system process is as follows:

1. Sensorization: This device works with 3 external sensors (temperature sensor, pulse sensor and accelerometer) and 1 internal sensor (MCU temperature).

The internal sensor helps to know how is the IoT device working and to control it. Knowing the temperature a problem with the device can be detected if the temperature grows extremely.

2. Data collection: sensors send physical analog signal to the microcontroller which are sampled and quantified to transform them into manageable digital data. In some cases the signal is digitalized in an external module where the sensor is placed rather than in the microcontroller, in this case the microcontroller only has to collect the result following the communication protocol used by the module in question (SPI, I2C, UART, or one of its own).

3. Formatting and Transfer: Once the microcontroller has the data, it must prepare them for shipment. It is important to detail a format for the shipment that is distinguishable later in the reception, clearly marking the distinction between data collected from different sensors. JSON, a text format for sending data, widely used, easy to handle both at source and destination, is used to send the data collected to the destination.

The data is packeting using TCP/IP headers and then sending to the network through Wi-Fi.

4. Reception: At the destination the data is unformat and stored differently depending on the sensor to which they belong. Subsequently, these data can be represented in many ways, using an array, a graph… In this case is represented using a real time graph.

3. Architecture

The system architecture can be described from a topological network point of view, shown in Figure 1.

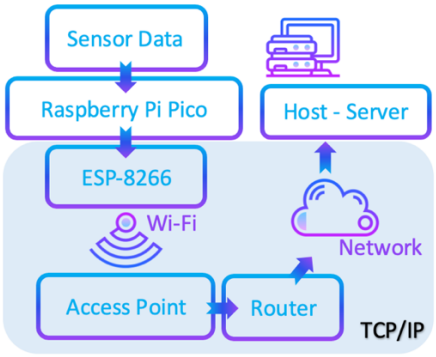

Figure 1. Network topological architecture of the whole system.

The device is setting with an IP address and a port number. It transmits the information collected to an access point through Wi-Fi, and the access point sends the information to a router which decides where send it. Once the information is sent through the Internet or a local network and then it arrives to the destination, a server.

This system uses TCP (Transmission Control Protocol) rather than UDP (User Datagram Protocol) to send the information to the remote host or server. The main differences between them are that TCP is a connection oriented and reliable protocol; and otherwise UDP is a connectionless and unreliable protocol. The data handled demands a reliable protocol because physiological data is dealt(it can be considered sensitive information). And an oriented connection protocol is needed because before to send the information the origin has to make sure that the destination is ready to receive it, although this kind of detail is less important than the reliable feature. Nevertheless, these TCP features pay its advantage in time, losing a little real time experience. It's true that UDP can be used implementing the TCP features in the application layer of TCP/IP model improving times, but also gaining more complexity, but this is out of the scope of this work.

An implementation detail to take account in the transfer step is the format of the information. All is encapsulated in a JSON format with the following shape:

```
{"Temp":"_","TempMcu":"_","PulseSig":"_",
"Acel_x":"_","Acel_y":"_","Acel_z":"_"}
```

Being the "_" the information of each field measured. Then, the destination knows this format and extract the information of each field to represent it properly.

When the information finally reaches the destination, it must be represented. For that purpose, a server code is written in Python that receive the data and represent it in a graphical format with the Matplotlib python library in real time. The server works with two threads, one has the task of receiving the data and save it, and the other thread represents the data in graphs, which are shown in the figure 2.

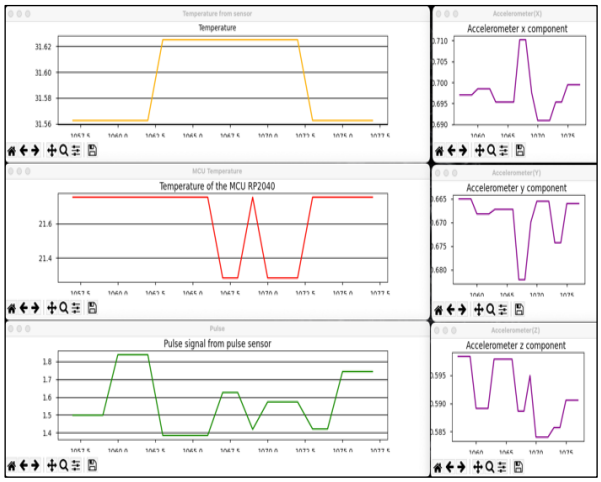

Figure 2. Data representation with graphs.

There are 6 graphs, one of the pulse sensor (green in Figure 2), another for the temperature (yellow in Figure 2), 3 for each of the axes of the accelerometer (purple in Figure 2) and other for the temperature of the mcu (red in the Figure 2). Each graph is represented in different

windows, so the final user of the data can move and resize each graph depending on the screen size and which graph or graphs that are considered more important. Also, the Matplotlib frame representation gives some useful tools to manipulate the graphs. The server code is upload in [5].

4. Implementation

The IoT device developed in this work uses

Temperature Sensor: a sensor DS18B20 is used to measure the temperature which is connected to the Raspberry Pi Pico (GPIO 16) through a resistance of 4,7 kΩ (Figure 3 and Figure 6). It's price is lower than 1$.

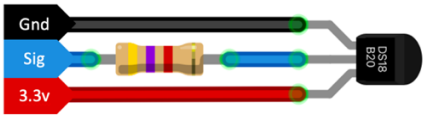

Figure 3. DS18B20 connections with mcu.

Pulse Sensor: to obtain the pulse a module called "Pulse Sensor for Arduino", built with an APDS-9008 photo sensor and MCP6001 Op-Amp, is connected directly to the pin 31 (ADC0) of the Raspberry Pi Pico. It sends an analogical signal to it, wire pink of the Figure 4. It's price is around 1$ and 3$.

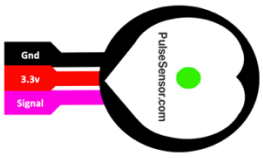

Figure 4. Pulse Sensor connections with mcu.

Accelerometer: To obtain the movements the MPU5060 board is used. It employs an I2C communication protocol, so it needs two I2C pins. The Slave Data Address pin (SDA) is connected to the pin SDA 19, and the Slave Clock pin (SCL) is connected to the pin SDA 20 (Figure 5 and 6). Its price is around 2$.

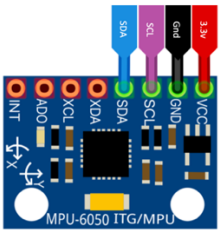

Figure 5. MPU5060 connections with mcu.

Microcontroller (mcu): Raspberry Pi Pico has a dual core microcontroller chip (Rp2040) with a flexible clock of 133MHz, a SRAM of 264KB, 26 pins, 4 of it analog, two UART, two I2C and two SPI connections available. Also, there is a temperature sensor inside accessible with software to control mcu temperature. The price of this microcontroller is around 5$. All the connections are shown in the figure 6.

The Raspberry Pi Pico can be programmed with Micropython, C or C++. In this case the microcontroller has been programmed in Micropython, a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.

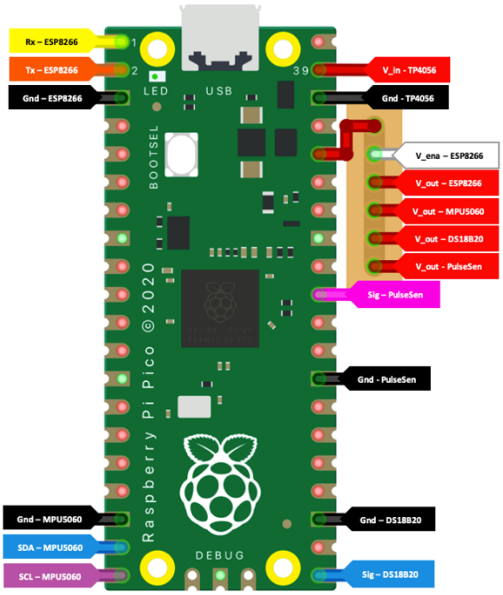

Figure 6. Raspberry Pi Pico connections diagram.

Wi-Fi Module: The ESP8266 Module is used as a slave of the mcu through a UART connection. The connections of the wires must be done connecting Rx pin of the ESP8266 with the mcu Tx pin, and Tx of the ESP8266 with Rx mcu pin. Is necessary in some boards uses one pin of the ESP8266 fed with 3.3V to enable the UART mode (Figure 7). Its price is 1$.

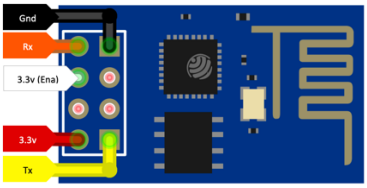

Figure 7. ESP8266 connections with mcu, white wire enable UART mode.

The Wi-Fi module works with AT commands, which consists of a series of short text strings which can be combined to produce commands for operations such as dialing, hanging up, and changing the parameters of a connection; transmitted through the UART communication, so Micropython handles the commands in the form of strings to implement the corresponding communication protocol; some useful AT commands employed are:

- ***AT***: To test AT startup, also used to test the connection between the Raspberry Pi Pico and ESP8266.
- ***AT+CWJAP="Network_name","Password"***: To Search the Wi-Fi access point with the name "*Network_name*" and to connect to it with the password "*Password*".
- ***AT+CIPSTART="TCP","192.168.12.147",9999***: To establish a TCP connection with a host or server with the IP address "*192.168.12.147*" and in the port "*9999*".

- ***AT+CIPSEND="Info_String"***: To send the information "*Info_String*" in the form of string through the connection established.

Battery: LiPo battery of 3.7 V, 1000 mAh, 10x20x50 mm size and its prize is around 5$. These features are very relevant, 3.7 V is the voltage that the microcontroller needs, dimensions match with its dimensions too, and the mAh will determine the durability of the battery depending on the total consume.

Battery Charger: The TP4056 module is used as an intermediary between the battery, the microcontroller and the charger. It has a micro-usb plug, and two leds, red and blue. If red is blinking that means that the battery is charging, and if blue bright and red extinguish means charge termination. Two wires will be connected to the battery (Bat) and the other two will be connected to the mcu, one to power system pin (VSYS) and the other to ground (Gnd), showed in the Figure 8. Its price is 0.3$.

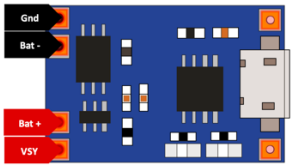

Figure 8. TP4056 connections with mcu and battery.

5. Results

The final prototype of the device built is as the Figure 9AB shows, whose size is 20x40x50 mm. The global consume is 110 mA, so with a battery of 1000 mAh full charged the autonomy could be of more or less 9 hours, at best. The coverage area of the device depends on the coverage area of the ESP8266, and it depends on the access point technology and antenna, so it is conditional on each circumstance.

A case is designed and printed in 3D with PLA to protect the device, and two straps have been added to attach the device to the arm, in the Figure 9CD.

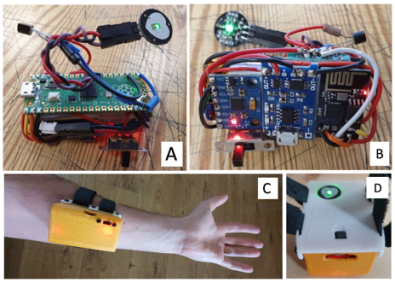

Figure 9. Final device result.

The final price of whole device is, looking up to, 17,3$ and with a conservative estimate 15$, depending on where the components are bought.

To test the system an access point is enabled with a smartphone where a computer and the IoT device are connected. First of all, the server must be running before to start the device, once it is, the device can be switched on, and after a little time of connection, between 3 or 5 seconds, the device begins to send the information to the server and when it receives it, the information is plotted in the graphs.

6. Conclusion

A prototype system have been developed fulfilled the objectives of low cost (15-17$), wireless, small size, scalable and capable of measure different physiological signals to send them after to a server where will be plotted.

This system may help in the future to monitor patients in a non-intrusive way by being placed in hospitals and complex care such as nursing homes, making real-time monitoring easier, cheaper and conserving resources, thus improving today's healthcare.

Future improvements encompass the utilization of MQTT protocols (MQTT message headers are small to optimize network bandwidth), also to publish each measure as a topic in a MQTT broker to permit other devices or services access to the information. It could improve the security with the encryption of the communication and the authentication of the IoT device with the server to avoid third entity in the client-server relationship.

The project is upload in Github as open-source [6].

Acknowledgements

Finally, I would like to express my gratitude to Eloy José Urendes Jiménez to help me to improve some details of this work.

References

[1] Web of AWS where explain the terms around IoT: https://aws.amazon.com/es/what-is/iot/

[2] Cisco, “Cisco Annual Internet Report (2018–2023) White Paper”, March 9 2020 : https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] Petrellis N, Birbas M, Gioulekas F. On the Design of Low-Cost IoT Sensor Node e-Health Environments. *Electronics*. 2019; 8(2):178. https://doi.org/10.3390/electronics8020178

[4] A. Sugathan, G. G. Roy, G. J. Kirthyvijay and J. Thomson. Application of arduino based platform for wearable health monitoring system, 2013 IEEE 1st International Conference on Condition Assessment Techniques in Electrical Systems (CATCON), 2013, pp. 1-5 (ISBN: 978-1-4799-0083-1).

[5] Code in Github for the server that plots the information: https://github.com/bermejo4/IoT_Medical_Device/blob/main/Server/tcp_graficador_pico.py

[6] All the documentation and code of this project: https://github.com/bermejo4/IoT_Medical_Device