Medical Monitoring device*

1st t Evrard Leuteu Feukeu
Hochschule Hamm- Lippstadt
Electrical Engineering
Hamm, Germany
evrard.leuteu-feukeu@stud.hshl.de

***Abstract*—This document is an overview and description for monitoring medical device which can be use in hospitals and clinics. Health monitoring device opens endless opportunities to develop man to machine and machine to man systems that can learn and adapt without continues assertion of support , explicit instructions of data patterns with the help of algorithm that process, analyse and interprets data into the computer system. A brief tour on how the Medical monitoring device functions will be explain and seen in detail in this document.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In recent days the data from wearable devices has helped in saving the lives of people by alerting them about health issues such as abnormal heart condition, high blood pressure and low blood oxygen etc. These devices not only help to keep track of the health of the user but can also make the management of user's health data more streamlined and efficient. This technology is relatively a new addition to the health sector and requires a solid back-end and communication structure to support it. These Wearable devices would then gather vital patient information and will transmit it to a cloud based server where we would store this information as patient record, which would be accessible by the medical staff. This would give doctors easy access to the information as the can download the information from anywhere through the cloud. This technology requires robust communication, which provides high speed data sharing and is reliable. one can implement protocols such as Bluetooth, WiFi and MQTT to achieve this. For my project we used Mqtt protocol for low latency low bandwidth communication where the connection is maintained through out. The next section discusses the general approach of my prototype

II. Context

At first glance it is quite difficult to conceptualise a health monitoring device to operate in real time. But thanks to recent technologies, we are able to achieves this. On the behalf of a patient, data can be a huge source of information which at the same time can be used to serve as a grate reference to determine the most effective and biased approach to an efficient cure. This device comprises all of the above and serve as a middle man between a patient and a medical expert. Explicitly, data is collected from the patient thanks to sensors and send to the monitoring device which in turn provide not only visual output but also the opportunity to store or retrieve information previously stored in a cloud. Furthermore the medical body or medical experts are couple of authorised expert certified tor perform medical duties. These experts has direct access to the patient data.

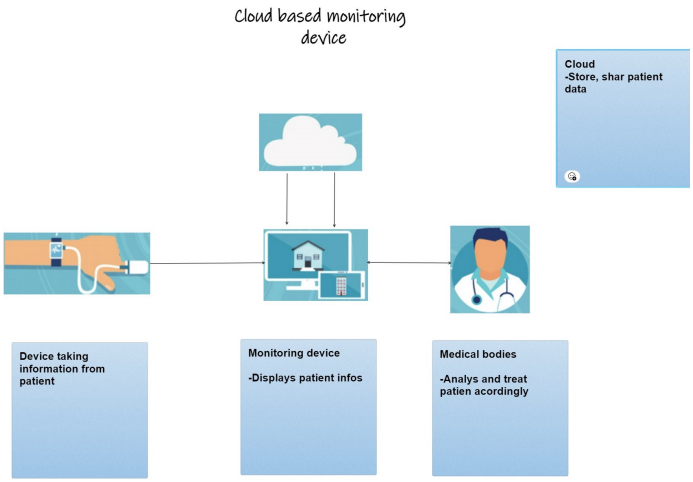

Fig. 1. components of the prototype

III. Use case

As seen above the monitoring device serve as a middle man between the patient and the medical bodies. But this goes beyond a single patient, as in a hospital for instance,there are often a huge number of patients. So with that in mind this device provide a cascading access to multiple patient by multiple doctors. It is possible to switch between patient and have access to multiple datas through this device so as to save more lives with minimum effort on patient to patient data fetching.

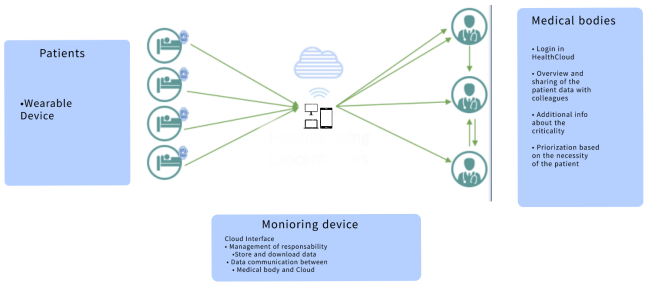

Fig. 2. Use case

IV. Users

A sample of the user interface is shown on the figure below. This is the concept of hoe the user interface is supposed to look like. An example of how the data is displayed is shown below that is a patient id picture, a biography to have more insight of the patient day to day activities, information about the patient for example age, location, work and so on. furthermore we can see a red box which shows pertinent informations about the patient. These informations are crucial to have a background history of a patient so as to know the patient allergies , former symptoms and so on so as to inform the medical bodies of how different each patient can be and for them to act accordingly.

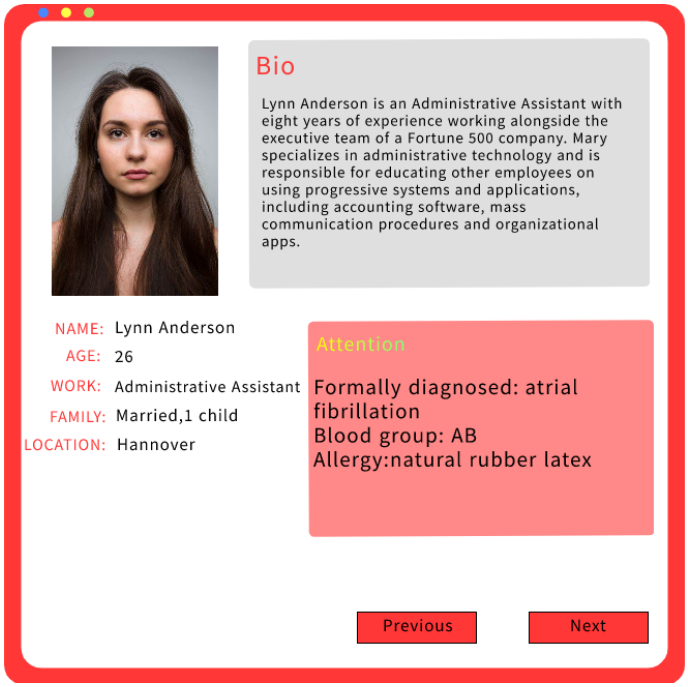

Fig. 3. Patient persona

V. General approach

To show the implementation of how communication protocol is archieved, a high fidelity, low resolution prototype is designed where information is send from the sensors like temperature and heartbeat to the monitoring device such as a tablet or a mobile phone. The prototype consists of three components as follows:

- Sensor node which consist of the the sensors that are connected to Arduino UNO wife rev 2
- Raspberry pi
- monitoring device

In this approach the sensor node acts as a publisher. A publisher generates the data and sends it to the MQTT broker. MQTT broker acts as a server that collects the messages from all the publishers, save the data and distributes it to the right subscriber. Raspberry pi servers as the MQTT broker for our project. The data is then displayed on the monitoring device that has subscribed to the MQTT broker. The digital device in our prototype can also be upgraded to act as a publisher for advanced features such as adding information to the patient data.

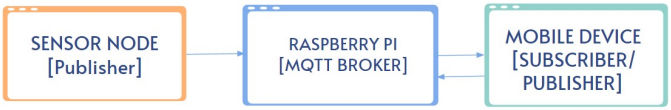

Fig. 4. components of our prototype

VI. Measurement Sensors

One of the monitoring device measurements is the Blood pressure, Blood pressure refers to the amount of force applied to the artery walls. High blood pressure, also known as hypertension, is an incredibly common health problem. as shown in Fig. 2

Damaged and narrowed arteries, aneurysms, coronary artery disease, an enlarged heart, renal failure and scarring, stroke, heart failure, and other complications can all be caused by high blood pressure. High blood pressure is known as the ”silent killer” since the harm it does has no warning signs or symptoms, which is why it’s critical to take your blood pressure directly on a regular basis.

For measuring the blood pressure, we will use the Ky-039 heartbeat sensor in Fig. 3

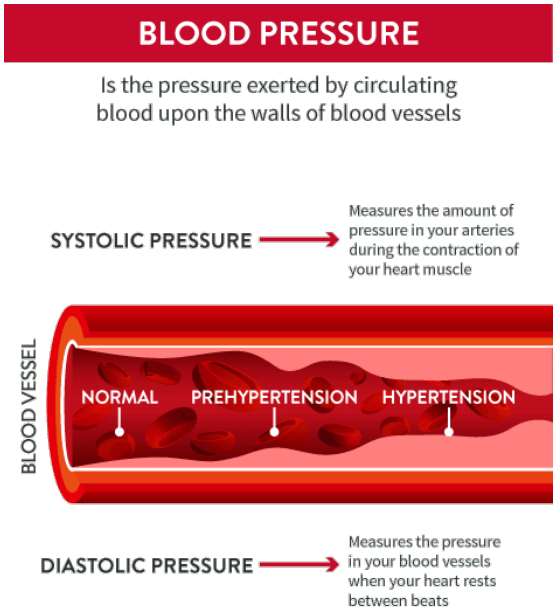

Fig. 5. Blood-Pressure Infographic

Fig. 6. KY-039 Heartbeat Sensor

If a finger is held between the infrared light-emitting diode and the photo transistor, the pulse can be detected at the signal output. This sensor module's measurement configuration, which includes an infrared diode and a phototransistor, now allows us to measure the pulse by placing a finger between the diode and the transistor. Explanation: The flesh on the hand can be shone through, just like the flashlight. You can see the blood pumping if you hit a blood vessel. Because blood has a variable density at different sites in the vein, this pumping can be identified by changes in brightness in the blood flow. The sensor module can precisely record these brightness variations, allowing the pulse to be identified.

To get an optimal result, we recommend to prepare the module for measuring as shown in the following pictures: as shown in Fig. 4

Fig. 7. Preparing for measurement 1

To increase the sensitivity of the sensor, we recommend fixing the sensor to the fingertip of the little finger using a plaster / adhesive tape / insulating tape. as shown in Fig. 5

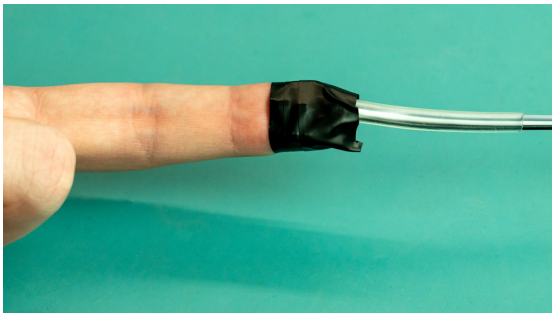

Fig. 8. Preparing for measurement 2

When the sensor is placed over a bigger blood vessel, the heartbeat is recorded/registered especially well. If necessary, we recommend moving the sensor to a different location on the fingertip to optimize the signal.

After setting the heart beat sensor with the Arduino environment as shown in Fig. 10

We get the measurement of the heartbeat sensor as shown in Fig. 8:

Fig. 9. Measurement by heartbeat sensor

The second measurement from our device is the temperature and we did it via the KY-015 COMBI-SENSOR (TEMPERATURE and HUMIDITY) : This sensor is a mixture of temperature sensor and humidity sensor in a compact design. shown in Fig. 7 at the end we got the values as has been

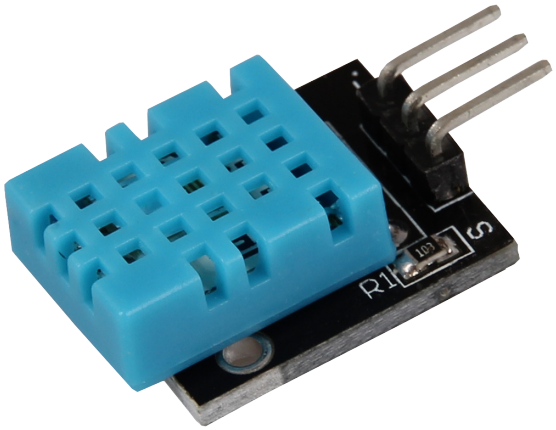

Fig. 10. COMBI-SENSOR (TEMPERATURE and HUMIDITY)

shown in a physical way during the presentation. as shown in Fig. 12

VII. MQTT Hardware Configuration

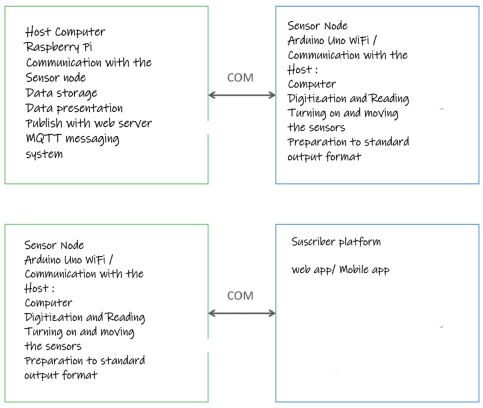

Fig. 11. Data communication structure

The stucture of the network is comprised as follows, a central broker persed by a raspberry pi, a subscriber and a publisher module which are a smart phone and aduino Uno wifi rev 2 respectively. The later recieves information from the patient with the help of Temperature and heartbeat sensors and process it to the medical body. The MQTT (Message Queuing Telemetry Transport) is mainly used.

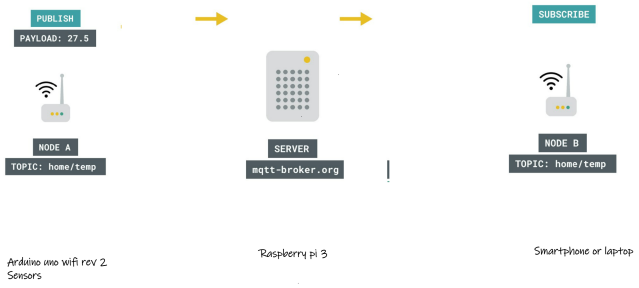

Fig. 12. Diagram of MQTT based

VIII. Implementation

A. Message Queuing Telemetry Transport (MQTT)

The MQTT protocol was first released in 1999 as a simple publish-and-subscribe system. It's especially handy for low-bandwidth devices, as it allows us to communicate commands, sensor values, and messages across the Internet with minimal effort.[9]

A simple explanation of how it works is that a node transmits a payload to a broker, such as an Arduino with a Wi-Fi module. A broker is a type of "middle-point" server that, in essence, keeps payloads given to it in "topics." A topic is a description of the data it includes; it may be "basement humidity" or "living room temperature," for example. The information from the broker can then be subscribed to by another node, and voila, data has been [9]transferred from Node A to Node B via the Internet.

1) Mosquitto Broker and Client on Raspberry Pi: [ht!] pi@raspberry: $sudoaptinstall - ymosquittomosquitto - clients$

pi@raspberry: $mosquitto - d$

pi@raspberry: $mosquitto_sub - d - ttestTopic$

pi@raspberry: $mosquitto_pub - d - ttestTopic - m"Publishingthefollowing Helloworld!text"$

pi@raspberry: $mosquitto_sub - d - ttestTopic$

2) Smartphone as MQTT Client: To access data from the Arduino, we need to create a panel such as SSH panel in the Web App or Mobile App Panel so as to access a subscriber and a publisher data communication. After creating the panel, we can run a test using the raspberry Pi and the Web App or Mobile app!

B. Raspberry Pi - App Communication

The Arduino UNO WiFi Rev.2 capabilities can be used with the following library. It can act as both a server and a client, accepting incoming connections and sending outgoing ones. WEP, WPA2 Personal, and WPA2 Enterprise encryptions are supported by the library. This library includes all of the original WiFi library's methods as well as connectSSL (). Many of the function calls in the WiFiNINA library are comparable to those in the Ethernet and WiFi libraries.[10] [ht!] include ¡SPI.h¿ include ¡WiFiNINA.h¿

Aditional Libraries Properties [ht!] name=ArduinoMqttClient version=0.1.5 author=Arduino maintainer=Arduino ¡info@arduino.cc¿ sentence=[BETA] Allows you to send and receive MQTT messages using Arduino. paragraph= category=Communication url=https://github.com/arduino-libraries/ ArduinoMqttClient architectures=* includes=ArduinoMqttClient.h [10]

C. Arduino WiFi - Raspberry Pi communication

In order to send the data received from our sensors(i.e Arduino) and transmit them to the Raspberry Pi, there need to be a communication protocol established and clearly implemented. As we decided to implement the MQTT protocol, the Arduino is identified as a publisher (supposed to send data to the Broker) and the Raspberry Pi who is identified as the broker (Working as a server and from where all the data is going through).

Let's have a look on how the communication is done within the Arduino:

First we have to include the MQTT Library: [language=Arduino] include ¡ArduinoMqttClient.h¿ Then we have to connect to a WiFi Network with the credentials as the SSID and the password [language=Arduino] char ssid[] = SECRET$_SSID; charpass[] = SECRET_PASS$; After that we need to initialise the MQTT client as a WiFi client with the following command: [language=Arduino] WiFiClient wifiClient; MqttClient mqttClient(wifiClient); And after that we get into the Loop function where we will now have to get the values from the sensors. The following lines of code explains that: [language=Arduino] mqttClient.beginMessage(topic); mqttClient.print("The heartbeat is: "); mqttClient.print(heartRateBPM); mqttClient.print(" and the temperature is: "); mqttClient.print(DHT.temperature); mqttClient.print("*C"); mqttClient.endMessage(); At the first line we begin by connecting to the Raspberry Pi by sending a message (and then connect to the topic set at the beginning) end we finish by exiting at the last line the message sending. The final code of the project can be found in the Appendix.

IX. Conclusion

Part of telehealth includes remote patient monitoring, which is done using so-called wearable devices and monitors. These include gadgets that can continuously or at intervals track a patient's health indicators, such as cardiac activity or heart beat .

The MQTT protocol is the most common protocol for transmitting data from wearable devices and sensors because it's easy and convenient. That's why it can be found not only in wearable devices, but also in almost any smart gadget. [7]

Acknowledgment

This work has been made possible at the Hochschule Hamm-Lippstadt under the course Industrial Communications at the Electronics Engineering department and supervised by Pr.Dr Joao Paulo Javidi da Costa. Thanks also to the authors in the reference for their papers that have contributed to make this paper better.

References

[1] https://www.researchgate.net/figure/Different-types-of-wearable-technology-fig5-322261039.

[2] https://www.designnews.com/sites/designnews.com/files/W6XXNX.jpg

[3] https://www.allheart.com/b-sphygmomanometers.html. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] https://hbr.org/resources/images/article-assets/2019/05/May19-26-647373390.jpg.

[5] https://content.u-blox.com/sites/default/files/Solution-architecture-for-portable-and-wearable-health-devices.jpg.

[6] https://sensorkit.joy-it.net/en/sensors/ky-039 .

[7] https://sensorkit.joy-it.net/en/sensors/ky-015.

[8] https://www.kaspersky.com/about/press-releases/2022-tracking-your-heartbeatand-payment-data-33-vulnerabilities-found-in-the-data-transfer-protocol-for-wearable-devices.

[9] https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-devicehardware–software-needed

https://www.arduino.cc/reference/en/libraries/wifinina/

Appendix

Here is the final code:

[language=Arduino][ht!] include ¡dht.h¿ include ¡ArduinoMqttClient.h¿ include ¡WiFiNINA.h¿ include ¡SPI.h¿ define dht$_apinA0$

int rawValue; char ssid[] = SECRET$_SSID; charpass[] = $SECRET_PASS; WiFiClientwifiClient; MqttClientmqttClient(wi

const char broker[] = "test.mosquitto.org"; int port = 1883; const char topic[] = "testTopic"; int status = WL$_IDLE_STATUS; //theWiFiradio'sstatusdhtDHT; intanalogPin = 1; constintdelayMsec = 60; //100msecpersampleintBPM = 0; intj = 0;$

void setup() Serial.begin(9600); while (!Serial) Serial.println ("Heartbeat detection sample code."); while (status != WL$_CONNECTED)Serial.print("AttemptingtoconnecttoWPASSI$

if (!mqttClient.connect(broker, port)) Serial.print("MQTT connection failed! Error code = "); Serial.println(mqttClient.connectError());

while (1);

void loop() static int beatMsec = 0; int heartRateBPM = 0; if (heartbeatDetected (analogPin, delayMsec)) heartRateBPM = 60000 / beatMsec; BPM+=heartRateBPM; j=(millis()/1000)if(j¿=1) BPM=BPM/60; BPM=0; Serial.print ("Pulse detected:"); Serial.println (heartRateBPM);

beatMsec = 0;

delay (delayMsec); beatMsec +=delayMsec; delay(1000); printCurrentNet(); DHT.read11(dht$_apin); mqttClient.beginMessage(topic); mqttClient.p$ "); $mqttClient.print(heartRateBPM); mqttClient.print("andtheten$ "); $mqttClient.print(DHT.temperature); mqttClient.print("*C"); mqttClient.endMessage();$

void printWifiData() // print your board's IP address: IPAddress ip = WiFi.localIP(); Serial.print("IP Address: "); Serial.println(ip); Serial.println(ip);

// print your MAC address: byte mac[6]; WiFi.macAddress(mac); Serial.print("MAC address: "); printMacAddress(mac);

void printCurrentNet() // print the SSID of the network you're attached to: Serial.print("SSID: "); Serial.println(WiFi.SSID());

// print the MAC address of the router you're attached to: byte bssid[6]; WiFi.BSSID(bssid); Serial.print("BSSID: "); printMacAddress(bssid);

// print the received signal strength: long rssi = WiFi.RSSI(); Serial.print("signal strength (RSSI):"); Serial.println(rssi);

// print the encryption type: byte encryption = WiFi.encryptionType(); Serial.print("Encryption Type:"); Serial.println(encryption, HEX); Serial.println();

void printMacAddress(byte mac[]) for (int i = 5; i ¿= 0; i–) if (mac[i] ¡ 16) Serial.print("0"); Serial.print(mac[i], HEX); if (i ¿ 0) Serial.print(":"); Serial.println();

bool heartbeatDetected (int IRSensorPin, int delay) static int maxValue = 0; static bool isPeak = false;

bool result = false;

rawValue = analogRead (IRSensorPin); // Here the current voltage value at the photo transistor is read out and stored temporarily in the rawValue variable rawValue *=(1000 / delay);

// Should the current value deviate too far from the last maximum value // (e.g. because the finger was put on again or taken away) // So the MaxValue is reset to get a new base. if (rawValue * 4L ¡maxValue) maxValue = rawValue * 0.8; // Detect new peak if (rawValue¿ maxValue - (1000 / delay)) // The actual peak is detected here. Should a new RawValue be bigger // as the last maximum value, it will be recognized as the top of the recorded data. if (rawValue¿ maxValue) maxValue = rawValue; // Only one heartbeat should be assigned to the recognized peak if (isPeak == false) result = true; isPeak = true; else if (rawValue ¡maxValue - (3000 / delay)) isPeak = false; // This is the maximum value for each pass // slightly reduced again. The reason for this is that // not only the value is otherwise always stable with every stroke // would be the same or smaller, but also, // if the finger should move minimally and thus // the signal would generally become weaker. maxValue -=(1000 / delay); return result;

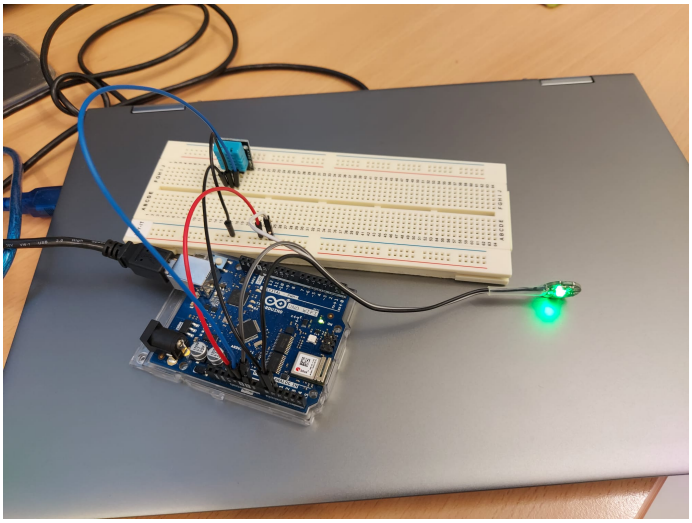

Fig. 13. Sensor node

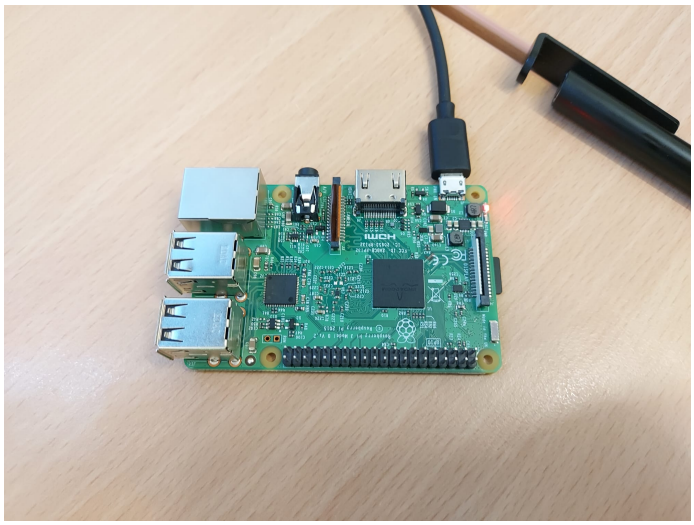

Fig. 14. Raspberry pi as the MQTT broker

Fig. 15. The digital device that display's the data