MANCHESTER 1824
The University of Manchester

Development of medical bracelet measuring heart rate and producing an ECG

Third Year Individual Project – Final Report

February 2020

Alkinoos Sarioglou

10136315

Supervisor: Dr Emad Alsusa

School of Electrical and Electronic Engineering

Contents

Total word count: 1825

1. Methods

1.1. Electrical Components

For the implementation of the medical bracelet the following four electrical components were selected:

- DFRobot Gravity Heart Rate Monitor Sensor for Arduino
- Gravity Analog ECG Sensor for Arduino
- ESP32-DevKitC Wi-Fi module
- Lithium Ion Polymer Battery

1.1.1. Heart Rate Sensor

The Heart Rate Sensor uses the technique of Photo-Plethysmography (PPG) to record the heart activity. Its operation is based on two green LEDs which emit light on the skin of the patient and on a photodiode in the sensor which measures the reflected light from the skin tissues. The photocurrent produced is then converted to an output voltage through a loading resistor. Appropriate calculations in the code allow conversion of the output voltage to the value of heart rate of the patient in BPM.

Two Heart Rate Sensors were tested in order to decide on the most suitable one and in the end the DFRobot Gravity Heart Rate Monitor Sensor for Arduino was selected for this application.

The IC of the sensor incorporates the SON1303 Integrated Heart Rate Sensor, which includes the LEDs and the photodiode. Additionally, the SON3130 chip is included which performs amplification of the photodiode's voltage and provides the following advantages:

- Low input bias current of 10 pA
- Low supply current of 60 μA

This sensor satisfies the requirement of small size as its dimensions are 24 mm x 28 mm, which means it can fit on an average person's wrist and is compatible with the size of a bracelet. Furthermore, it demonstrated more accurate measurements than the MAXREFDES117# sensor, which lay closer to the data extracted from the benchmark device as shown in the Results section. Finally, the cost was only 30p more expensive than the competitive sensor, therefore offering more advantages for a negligible difference in price.

Some of its electrical characteristics include [1]:

Input Voltage	3.3 V – 6 V (5 V recommended)
Output Voltage	0 V – Vin (Analogue Mode) , 0 V / Vin (Digital Mode)
Operating Current	<10 mA

Table 1: Electrical Characteristics of the DFRobot Gravity Heart Rate Monitor Sensor

1.1.2. ECG Sensor

As seen in the Theoretical Development section, the operation of the ECG sensor is based on the monitoring of the electric potential from the different electrodes attached in separate parts of the body which are then combined to give the output ECG graph.

Even though there were very limited choices for ECG sensors in the market, the Gravity Analog ECG Sensor for Arduino can perform a satisfactory ECG test on a patient which lies close to the output graph from the benchmark device as seen in the Results section.

This sensor incorporates the AD8232 Heart Rate Monitor Chip by Analog Devices, which offers great benefits [2] such as:

- Low supply current of 170 µA
- Common-mode rejection ratio of 80 dB
- High signal gain (G=100)

Additionally, it combines the voltage input of three electrodes attached to the appropriate positions on the human body to extract a 3-lead ECG graph, which provides greater accuracy than a 1-lead ECG.

As the size is an important factor of the device as well, this IC's dimensions are 35 mm x 22 mm, which fits inside the range for the medical bracelet.

Finally, its electrical characteristics include [3]:

Input Voltage	3.3 V – 6 V (5 V recommended)
Output Voltage	0 V – 3.3 V

Operating Current	<10 mA

Table 2: Electrical Characteristics of the Gravity Analog ECG Sensor

1.1.3. Wi-Fi Module

A Wi-Fi module is used for the communication of the Heart Rate data and the ECG output voltage to an IoT Cloud Service, where the data is displayed in a user-friendly environment.

For this application, the ESP32-DevKitC Wi-Fi module was selected as it provides Wi-Fi and Bluetooth connectivity, multiple ADC channels for sampling the analogue voltages of the sensors and GPIO pins.

This development board incorporates the ESP32-WROOM-32U microcontroller, which offers the following features:

SPI Flash	32 Mbits
Crystal Oscillator Frequency	40 MHz

Table 3: Features of the ESP32-WROOM-32U microcontroller

The Development Board overall provides a micro-USB port, in order to be able to program the board directly. Furthermore, the board is compatible with the programming environment (Arduino IDE, Version 1.8.11, Arduino, Italy) and software support and libraries are provided.

The Wi-Fi module supports TCP/IP and a full 802.11 b/g/n Wi-Fi MAC protocol providing speed of up to 150 Mbps, transmitting power of up to 20.5 dBm and the frequency range is between 2.4 GHz and 2.5 GHz [4].

The board provides advantages in size as well, as its dimensions are 48 mm x 28 mm.

1.1.4. Battery

The medical bracelet is powered by a Lithium Polymer Rechargeable Battery with nominal voltage of 3.7 V and rated capacity of 1800 mAh. Its main advantages include [5]:

- Rechargeability
- Size (53.5 mm x 35 mm)
- Twice as much energy as NiMh cells with low self-discharge

- Low weight
- Thermally stable

1.2. Software Components

Two software components are necessary to be able to receive the information of heart rate and ECG from the Wi-Fi module and display the results in a user-friendly environment. One of the aims of the project is to provide flexibility in the way that users and doctors view the results, therefore two means of displaying information were developed, an IoT Cloud Service (for computer users) and a mobile Application (for smartphone users). These were created by using the:

- Ubidots IoT Cloud Service and the
- Blynk App Development Platform

On one hand, Ubidots is an IoT platform in which developers can easily communicate data from sensors and display them using different means of presentation, such as Line Charts, Graphs, Histograms and more. It provides support with Arduino libraries for multiple Wi-Fi modules such as the ESP32-DevKitC used in this application.

On the other hand, the Blynk Development Platform aids in the development of mobile applications compatible with Android OS version 4.2+ and iOS version 9+ [6]. Furthermore, Blynk applications can receive data using Wi-Fi connection from a diverse set of Wi-Fi modules, including the ESP32 module used here.

1.3. Mechanical Component

The electrical components on the PCB were assembled with the mechanical component of the system to provide ease of use by the patients. This is a:

- 3D-printed bracelet

The bracelet has a compact design in order to fit all of the components in a small area.

1.4. Schematic Diagram

The following schematic diagram shows how the electrical components of the system were connected:

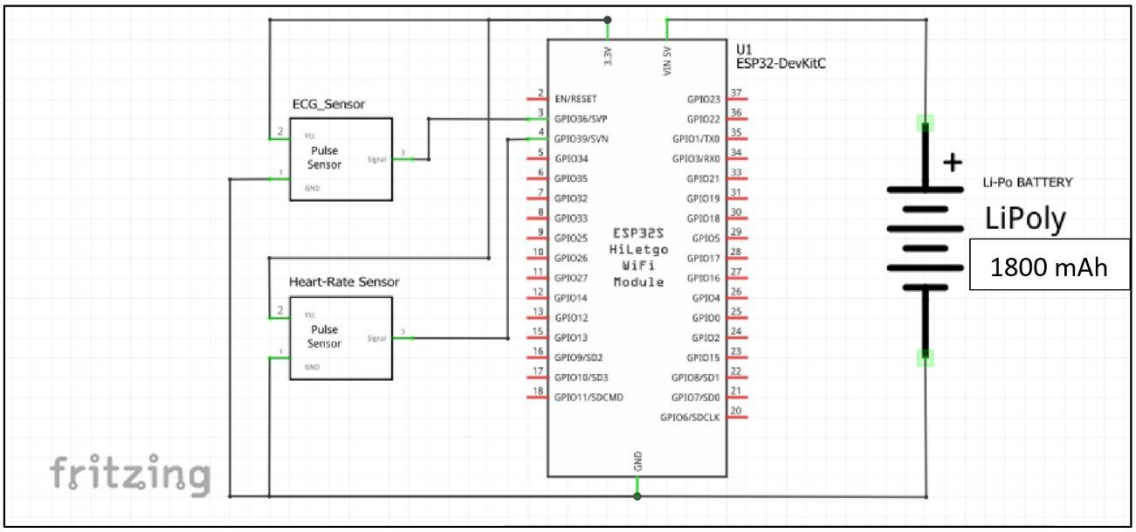

Figure 1: Schematic Diagram of the system

The schematic diagram is designed using an open-source EDA tool (Fritzing, Version 0.9.4, Interaction Design Lab, Potsdam, Germany).

1.5. Wiring Diagram

The components were then physically connected according to the following wiring diagram:

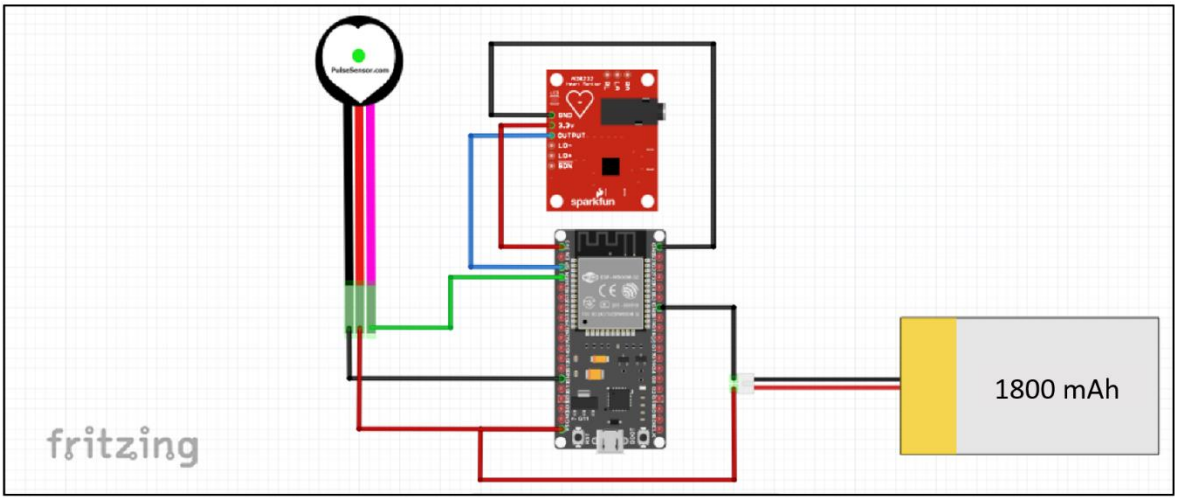

Figure 2: Wiring Diagram of the system

The following connections were made:

ESP32-DevKitC	Heart Rate Sensor
3.3 V or 5 V	Vcc
GND	GND

VN	Vout
ESP32-DevKitC	ECG Sensor
3.3 V or 5 V	Vcc
GND	GND
VP	Vout
ESP32-DevKitC	Battery
5 V	Positive
GND	Negative

Table 4: Connections between the Wi-Fi module and the Sensors

1.6. Procedure

The code of the system was developed in an Integrated Development Environment compatible with the Wi-Fi module, the IoT Cloud Service and the Mobile App Development Platform (Arduino IDE, Ver. 1.8.11, Arduino, Italy).

In order to set up the Arduino IDE to be compatible with the ESP32-DevKitC and communicate the data to the Cloud, the following additional equipment/tools were needed:

- USB A to mini USB B cable
- Computer with installed Arduino IDE, Ver. 1.8.11.
- Ubidots IoT Cloud Service Account

The steps required to successfully implement the functionality of the system were the following:

- The Arduino IDE environment was initiated and the ESP32 support for the Arduino environment was installed so that the ESP32 Wi-Fi module could be accessed [7].
- The “ESP32 Dev Module” from the “Board” options in the “Tools” drop-down menu was selected.
- The .zip file for the “PubSubClient.h” file was downloaded and included it as a library in the Arduino project along with the “WiFi” Arduino library [8].
- The strap was attached on the heart-rate sensor, so that it took the form of a bracelet.
- The heart-rate sensor was worn on the left wrist very tightly with the sensor touching the upper side of the wrist.
- The biomedical pads with the conductive gel were connected to the ECG electrode cables.

- The three ECG electrodes on the user's body were attached according to the following figure:

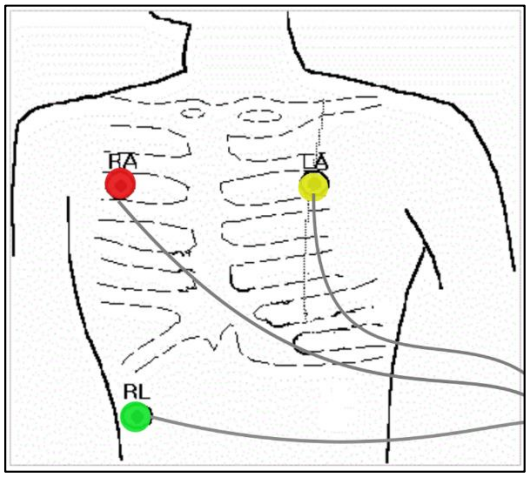

Figure 3: Placement of 3-lead ECG Electrodes [9]

The connections made are summarized in the following table:

ECG Electrodes	Body parts
RA or R	Right part of the chest or Right arm
LA or L	Left part of the chest or Left arm
RL or F	Right part of the abdomen or Right leg

Table 5: Connections of ECG Electrodes on the body

- The Wi-Fi module was connected to the Computer using the USB A to mini USB B cable.
- The main loop of the code used to implement the full functionality is shown below:

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }

  if (turn == true){
      sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
      sprintf(payload, "%s", ""); // Cleans the payload
      sprintf(payload, "{\"%s\":", VARIABLE_LABEL); // Adds the variable label

      uint8_t rateValue;
      heartrate.getValue(heartratePin); ///< A1 foot sampled values
      rateValue = heartrate.getRate(); ///< Get heart rate value
      if(rateValue)  {
        Serial.println(rateValue);
        dtostrf((float)rateValue, 4, 0, str_sensor);
        sprintf(payload, "%s {\"value\": %s}}", payload, str_sensor); // Adds the value
        Serial.println("Publishing BPM data to Ubidots Cloud");
        turn = false;
        client.publish(topic, payload);
        client.loop();
      }
      delay(20);
  }
```

```
  else if (turn == false) {
      sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
      sprintf(payload, "%s", ""); // Cleans the payload
      sprintf(payload, "{\"%s\":", VARIABLE_LABEL_ECG); // Adds the variable label

      float sensor = analogRead(SENSOR);

      /* 4 is mininum width, 2 is precision; float value is copied onto str_sensor*/
      dtostrf(sensor, 4, 2, str_sensor);

      sprintf(payload, "%s {\"value\": %s}}", payload, str_sensor); // Adds the value
      Serial.println("Publishing ECG data to Ubidots Cloud");
      turn = true;
      client.publish(topic, payload);
      client.loop();
      delay(100);
  }

}
```

- In the Arduino code the WIFISSID, PASSWORD variables were replaced according to the local network and a random MQTT_CLIENT_NAME was inserted for the device.
- The Arduino code was downloaded to the Wi-Fi module, which communicated the data to the Ubidots Cloud Service.

In order to set up the Ubidots IoT Cloud environment and display the results the following steps were essential:

- A Ubidots Account was created at https://ubidots.com/.
- On the “Devices” screen, a new device was added named “esp8266”.
- Inside the device options, two new variables were created called “Heart-rate” and “ECG”, which were assigned the instantaneous values of heart rate in BPM and the electric potential measured by the ECG device in order to produce the ECG graph respectively.
- On the “Dashboard” screen, two new widgets were created. One of them was a Metric and displayed the live measurement of the heart rate in BPM. The other one was a Double Axis graph, which demonstrated the live ECG graph of the user.
- The Heart-rate Metric widget was linked with the “Heart-rate” variable created before and the most recent value was displayed.
- The ECG Double Axis widget was linked with the “ECG” variable created before and it included “Time” as the x-axis and the “Amplitude” of the output voltage by the ECG sensor as the y-axis.

In order for the correct values to be displayed some changes were needed on the Arduino code:

- The Ubidots TOKEN was extracted from the website, by selecting “API Credentials” from the drop-down menu after clicking on the User image, and it was copied to the corresponding variable in the code.
- The variable labels were specified to be the same in the code as in the Ubidots cloud.
- The device label was specified to be the same as the one in the Ubidots cloud.

After the successful completion of this procedure, the system was ready to take measurements from the user, communicate them to the IoT Cloud and display them for analysis by a doctor.

An example of the dashboard displaying the Heart Rate measurement in BPM and the ECG graph is shown:

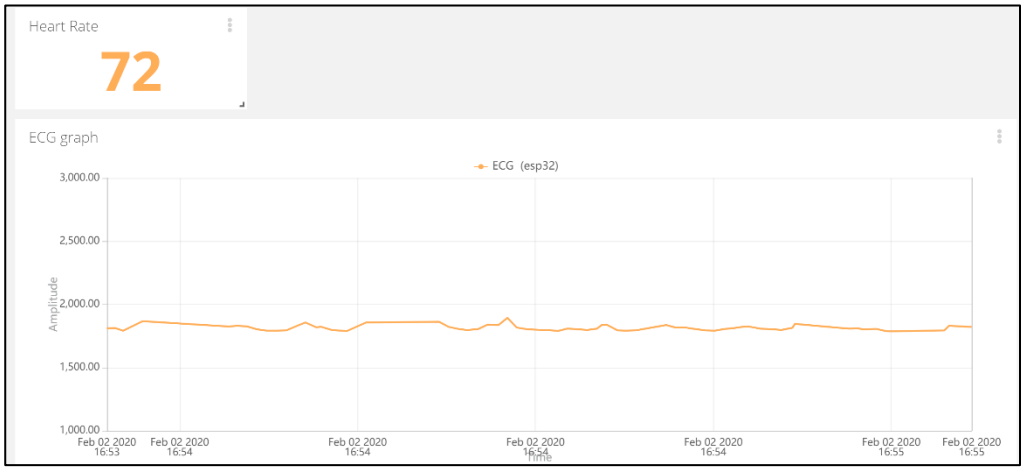

Figure 4: Sample Measurement

2. References

[1] Available: https://wiki.dfrobot.com/Heart_Rate_Sensor_SKU__SEN0203, [Accessed: 06- February- 2020]

[2] Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf, [Accessed: 06- February- 2020]

[3] Available: https://wiki.dfrobot.com/Heart_Rate_Monitor_Sensor_SKU__SEN0213, [Accessed: 06- February- 2020]

[4] Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf, [Accessed: 06- February- 2020]

[5] Available: https://uk.rs-online.com/web/p/speciality-size-rechargeable-batteries/1449405/, [Accessed: 06- February- 2020]

[6] Available: https://blynk.io/en/getting-started, [Accessed: 06- February- 2020]

[7] Available: https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/windows.md, [Accessed: 06- February- 2020]

[8] Available: https://github.com/knolleary/pubsubclient, [Accessed: 06- February- 2020]

[9] Available: https://www.dfrobot.com/product-1510.html, [Accessed: 06- February- 2020]