



Practical Assignment Report

SWE30011 – IoT Programming

The Son Ngo - 103827804

SWE30011 – IoT Programming

I. Table of Contents

<i>I. Table of Contents.....</i>	<i>1</i>
<i>II. Summary.....</i>	<i>2</i>
1) Brief introduction about IoT in healthcare.....	2
2) ECG heart rate monitoring.....	2
3) Arduino-based implementation	2
<i>III. Conceptual Design and Description.....</i>	<i>4</i>
1) AD8232 sensor.....	4
2) Buzzer	8
3) LCD I2C display screen.....	9
4) Arduino Board	11
5) MacOS computer	11
6) Use case diagram.....	13
<i>IV. Resources.....</i>	<i>15</i>
<i>V. Appendix.....</i>	<i>16</i>
1) heart_monitor.ino.....	16
2) heart_beat.py.....	16
<i>VI. Conclusion.....</i>	<i>17</i>
<i>VII. References</i>	<i>18</i>

II. Summary

1) Brief introduction about IoT in healthcare

The Internet of Things has been applied widely in various industries due to its capability to connect objects and machines. This connectivity is essential in today's world, where organisations rely on the Internet for services and internal workflow. When it comes to healthcare, applications of IoT are highly prevalent. To provide patients with a timely service, which includes health indicators measurement and diagnosis, hospitals and clinics must implement a network of devices inter-connected. Hence, connectivity enables the accurate collection and high-speed transmission of real-time data, which is then utilised by healthcare practitioners. Without these invaluable data, doctors and nurses would not be able to understand the health condition of patients, thus unable to give precise diagnoses and suitable treatments.

2) ECG heart rate monitoring

Most hospitals nowadays deploy Electrocardiography (ECG) monitors to visualise how a patient's heart operates. From observing the ECG waveform and realising abnormal patterns, doctors can identify any symptoms of potential cardiovascular diseases, such as heart attack, cardiomyopathy or congenital heart defect. Moreover, ECG monitors prove to be highly useful in the cases of. ECG detects the muscular and electrical functions of the heart by pulsating electrical waves towards the skin. Subsequently, the electrodes' waves interact with electrical pulses generated by heart muscle depolarisations, resulting in a new waveform for observation. With the rise of telemedicine and home care, effective and easy-to-use heart rate monitoring systems are increasingly in demand.

3) Arduino-based implementation

The Arduino platform is well-known for its adaptability, and portability in building electronic prototypes, including medical devices. In addition, Arduino also brings affordability and functionalities, which spur innovation in biotechnology around the world, especially in underdeveloped regions. It ensures that electronics and mechatronics are accessible to everyone, including non-professionals.

This project aims to simulate an electrocardiogram monitor using the Arduino toolkit and the AD8232 ECG heart rate sensor. AD8232 is a hybrid sensor, which can send both analog and digital signals. While the digital signals notify about malfunctioning, analog signals are

generated by the patient's heartbeat itself. In addition, this prototype also includes serial communication with an edge computer – Macbook Air M1. MacOS systems are somewhat similar to Linux OS, in terms of the terminal syntax and serial communication capability. Hence, MacOS computers can replace Raspberry Pi in terms of serial communication with Arduino board, while being a power supplier for Arduino board in the meantime. Moreover, this Arduino-based heartbeat monitor integrates an LCD I2C screen and a buzzer as actuators. While the buzzer plays sound according to the heartbeat, the LCD I2C screen acts as an actuator and a user interface at once. The LCD screen will display the BPM value calculated by MacOS so that users can know their specific heart rate.

This simulation illustrates some enhancements to the Arduino's capabilities, thus delivering a mobile and portable device used in various environments, from hospitals, and clinics to at-home patient care. Electrocardiograms and any similar advanced healthcare technology would no longer be confined to the boundaries of traditional medical facilities.

III. Conceptual Design and Description

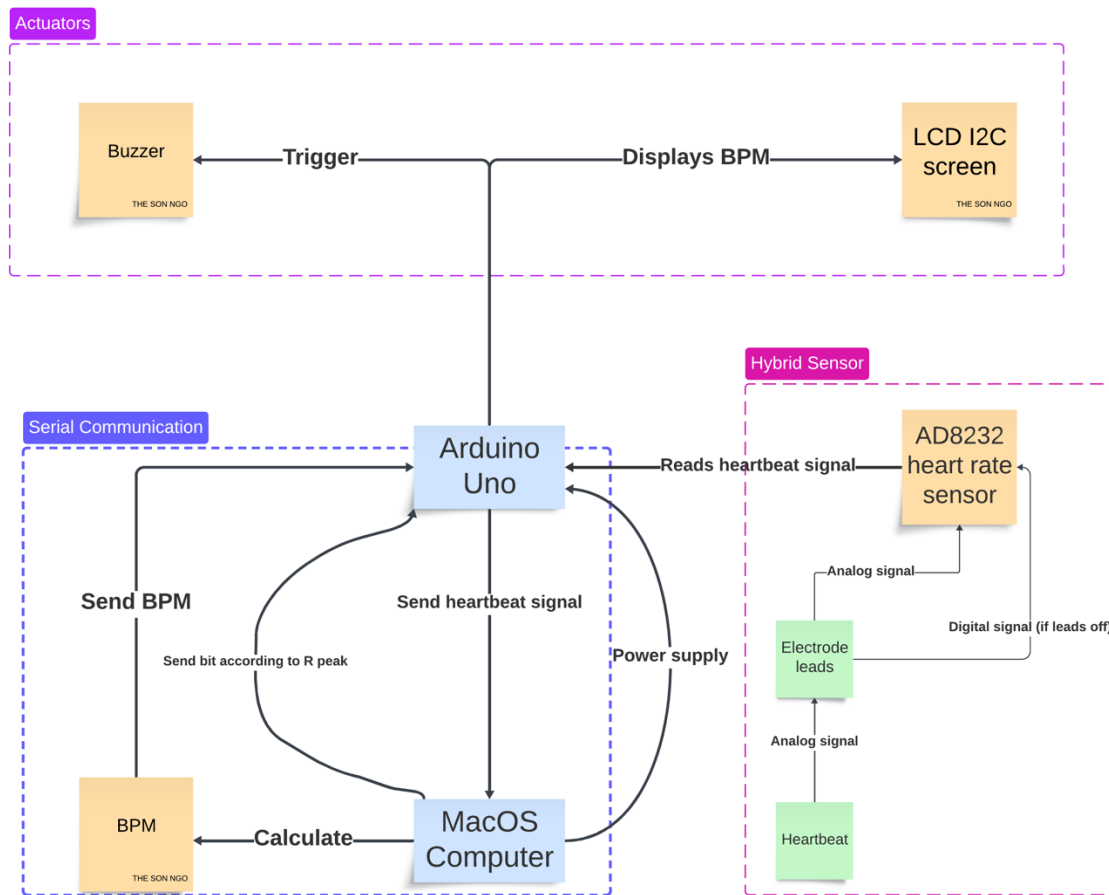


Fig1. Block diagram of Arduino-based ECG monitor design

Figure 1 is a block diagram that shows the architecture of this ECG heartbeat Arduino-based project. Below is a description of the main components of the design.

1) AD8232 sensor

AD8232 sensor is a portable sensor for detecting electrical changes resulting from the muscular movements of the heart. With a price of \$28, which is relatively cheap compared to other advanced medical devices, AD8232 provides a cost-effective solution to build a monitoring system for heart activities. Let's dive into the mechanism behind ECG technology.

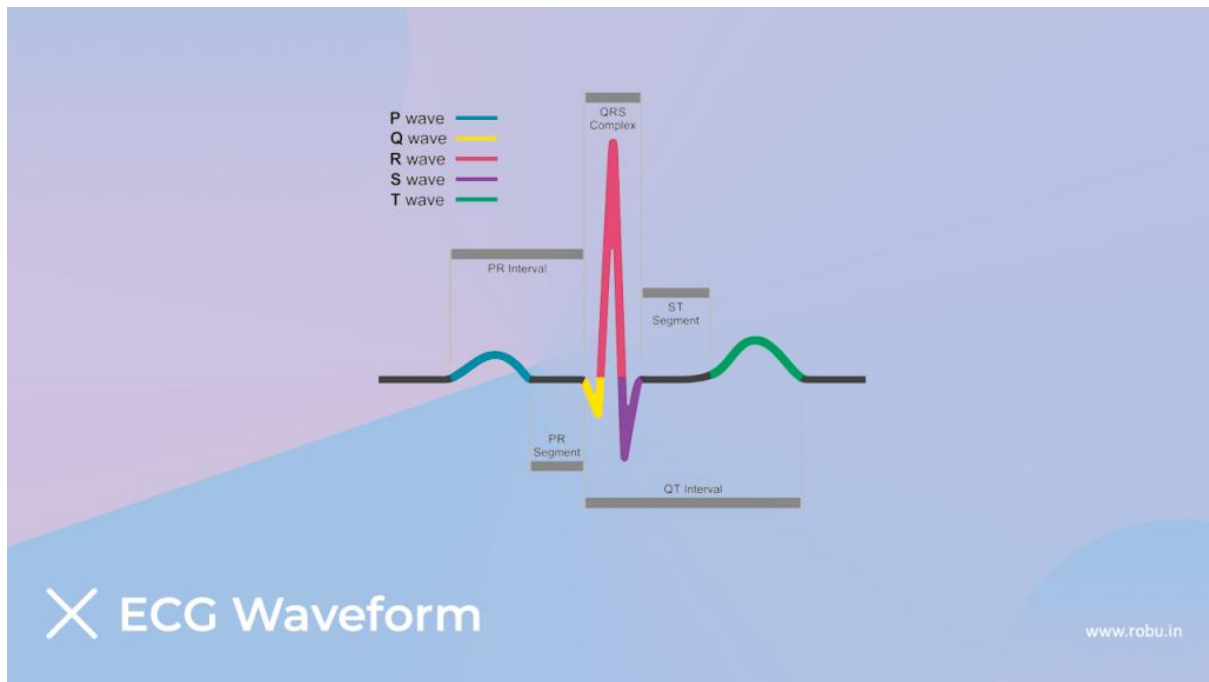


Fig2. ECG waveform. Source: www.robu.in

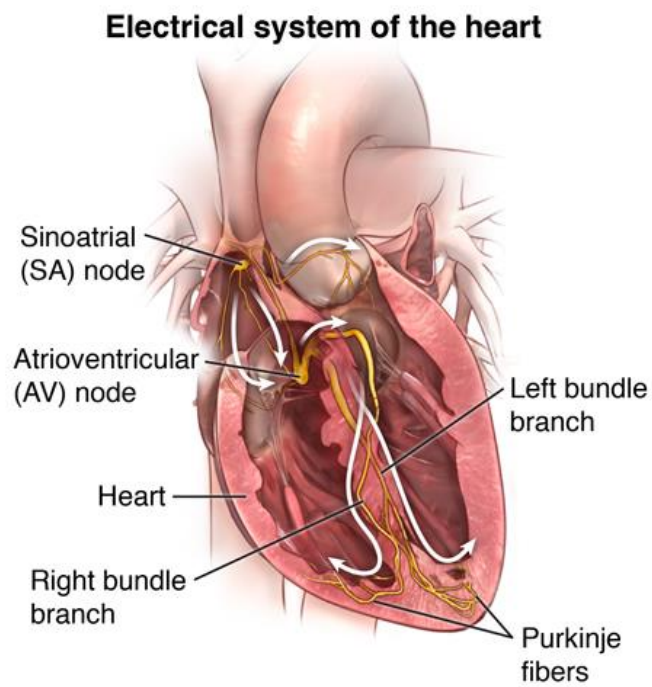


Fig3. Heart's electrical system. Source: John Hopkins Medicine

Non-invasive technology, which ECG is based on, does not physically infiltrate the skin to measure cardiovascular indicators, so it is relatively safe to implement. However, ECG might pose potential risks regarding the electricity it generates towards the skin. Operators should be

careful in choosing the right power supply source, as excessive electricity supply might cause electrical shocks.

The output of ECG is a waveform consisting of 5 intervals: P, Q, R, S, T (Figure 2):

- The P wave is produced by the sinoatrial node (Figure 3). The node itself generates electrical pulses to activate the atria (right upper chamber of the heart).
- QRS wave segment results from the activity of the atrioventricular node. This node is responsible for the pumping activities of the ventricles (lower chambers of the heart). The peak R wave from this segment represents one heartbeat.
- Finally, the T wave represents the repolarisation of ventricles. Repolarisation is necessary for the heart to repeat its blood-pumping cycle.

Nevertheless, ECG waves are often really noisy due to the external waves produced by other organs (e.g.: lungs, digestive organs, etc). Fortunately, the AD8232 sensor introduces filter, amplification and extraction functionalities, which lead to the accurate and clear identification of heart signals. AD8232 has a total of 6 pins: GND (ground pin, acts as - polar), 3.3V pin (power pin, acts as + polar), OUTPUT pin (outputs heartbeat analog signal), LO+ and LO- pins (digital output pins to detect leads-off), and SDN pin (shutdown, this is not used). Figure 4 shows how to wire AD8232 to the Arduino board.

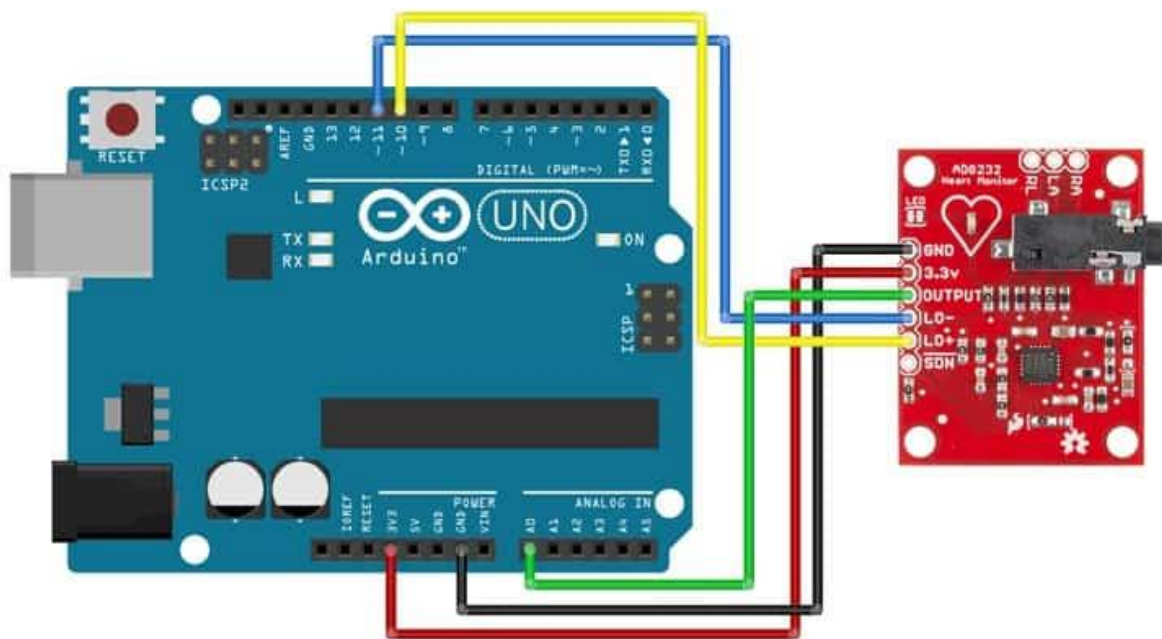


Fig 4. Circuit diagram to connect AD8232. Source: [how2electronics](https://www.how2electronics.com)

GND is connected to the GND port of the Arduino board, while the 3.3V pin should be connected to the 3.3V power source of the board. Because the output is in analog form, so OUTPUT pin will be connected to an analog port (A0 is the most popular choice). LO+ and LO- are digital pins, so they have to be connected to 2 digital ports (11 and 12 in this case). The sensor will detect electric pulses from the heart via 3 electrode pads, then it sends the received analog signal to port A0 of Arduino Uno.

There are two ways to mount the ECG electrode pads on the body, this is shown in Figure 5.

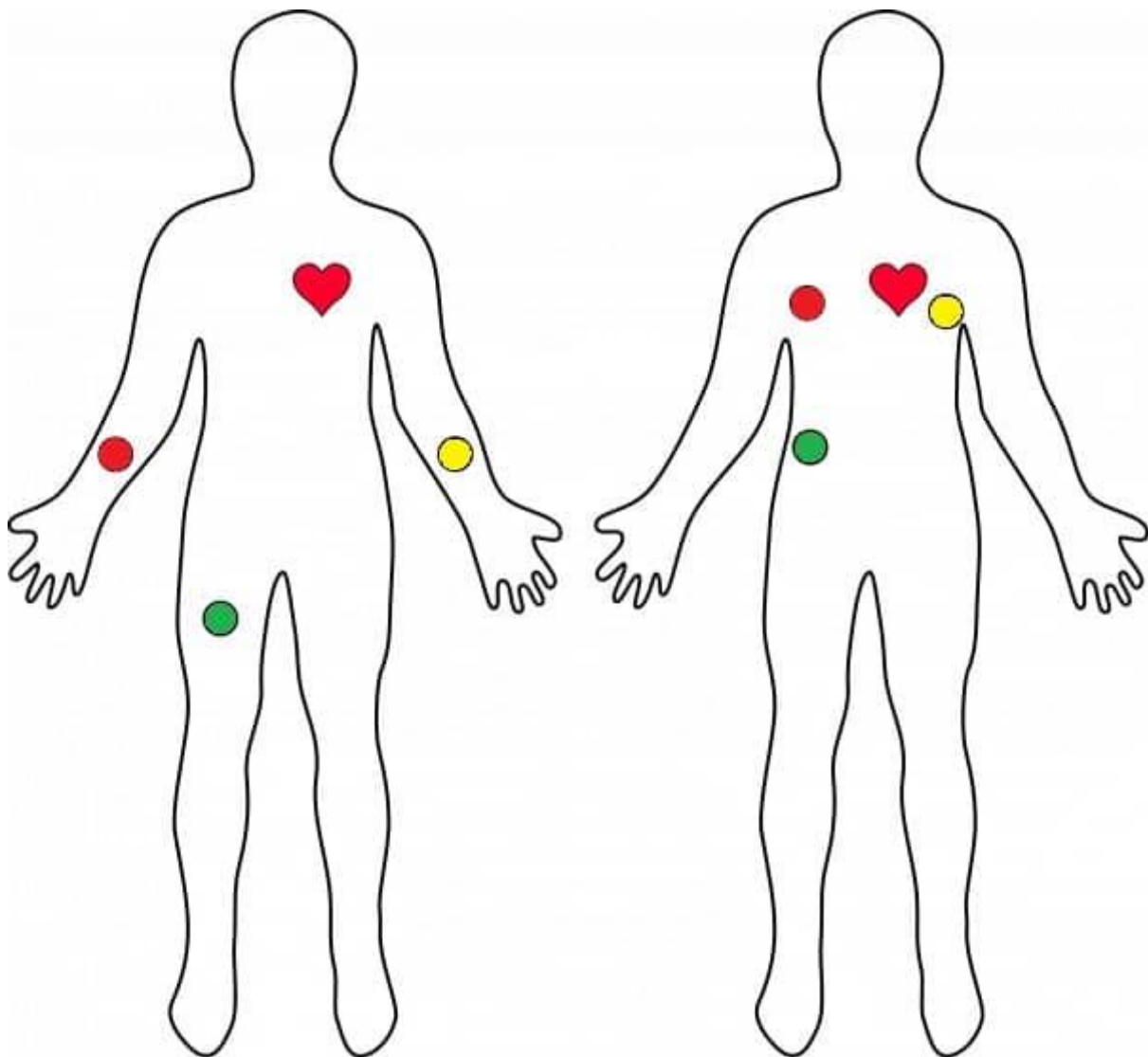


Fig 5: Two ways of placements. Source: how2electronics

There are 3 electrode pads in total, each has different colours: Red, Yellow, and Green. Each pad must be placed in its assigned position on the image (Green for the right leg, red for the

right arm, and yellow for the left arm). The closer the pads are to the heart, the more accurate the measurement is.

2) Buzzer

Buzzer is one of the actuators of this ECG monitor system. It is intended to play sounds according to the heartbeat detected, like a real ECG machine used in hospitals. To be more specific, it will play a sound every time the Edge computer sends a bit to the Arduino board via serial communication. The Edge computer will only send bits after it has read a heartbeat signal (a signal with a magnitude larger than 450) sent from the Arduino, which is again an example of serial communication. This signal is the analog data collected by the AD8232 sensor. After receiving the bit, Arduino will send a DC into the buzzer to make a “beep” sound. The buzzer negative polar is connected to the GND port of the board, while the positive polar will be connected to digital port 12. The code below shows how serial communication works between macOS and Arduino Uno to trigger the buzzer.

```
if ser.in_waiting > 0:
    line=ser.readline().decode('utf-8').strip()
    if line.isdigit() and int(line) > 450:
        ser.write(b'B')
```

Fig 6. Serial communication on macOS (using Python)

```
int heartbeatSignal = analogRead(sensorPin);
Serial.println(heartbeatSignal);

if (Serial.available() > 0) {
    char received = Serial.read();
    String bpm = Serial.readStringUntil('\n');
    if (received == 'B') {
        digitalWrite(buzzerPin, HIGH);
        delay(50);
        digitalWrite(buzzerPin, LOW);
    }
}
```

Fig 7. Arduino Uno triggers a buzzer. This results from serial communication between macOS and the Arduino board.

To develop a multi-media IoT-based system that is accessible by everyone, visualisation should not be the only way the represent data. Audio representations prove to be useful in helping stakeholders interpret data patterns, especially for people who are blind or have little experience with healthcare data visualisation.

3) LCD I2C display screen

The LCD screen is another actuator of this Arduino-based medical system, and in the meantime acts as a user interface to display BPM value – a quantity to measure the heart rate. An LCD consists of several layers. At the back is the light source (either a set of LEDs or a fluorescent lamp), followed by a layer of polarizing filters, and two layers of glass containing liquid crystal molecules, electrodes that shape into a grid. The displays on the LCD screen are controlled by electrical sent by Arduino, which alters the electrodes and shapes the pixels. In this project, LCD I2C is utilised for the interface. The difference between LCD I2C and conventional LCD is that I2C protocol allows the screen to use much fewer ports on the Arduino board. The image below illustrates the difference in complexity between wiring a normal LCD and an I2C LCD.

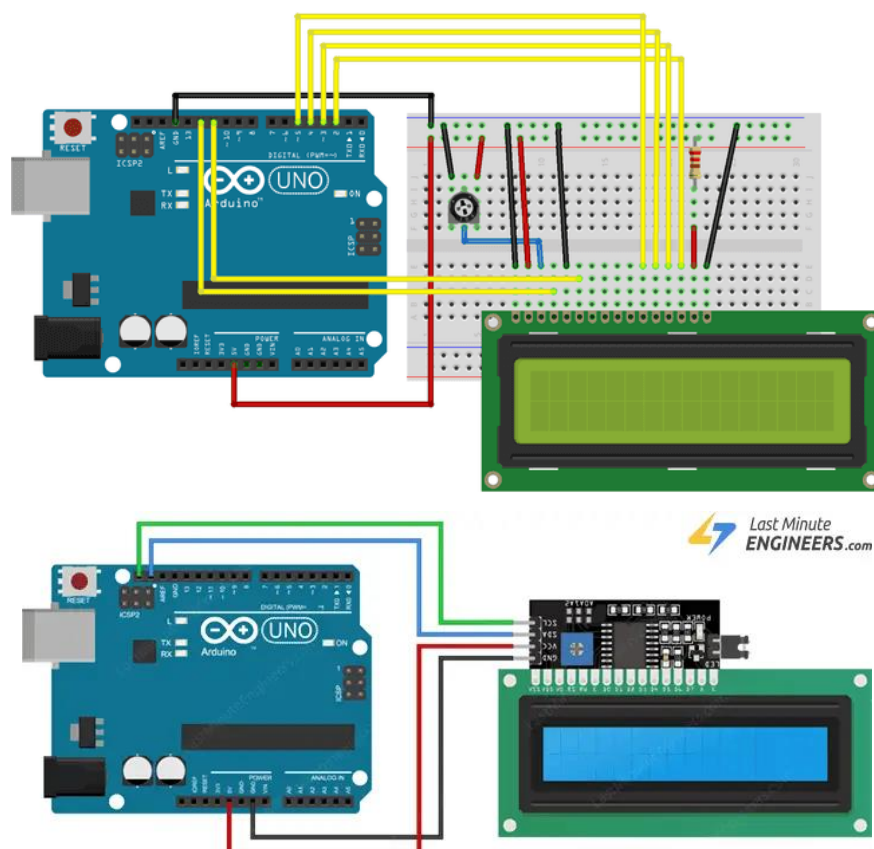


Fig 8. Up: Conventional LCD circuit diagram (Source: Arduino Docs) / Down: I2C LCD circuit diagram (Source: Last Minute Engineers)

I2C LCD has 4 inputs: GND, VCC, SDA and SCL. While GND and VCC are for power supply (GND for negative and VCC for 5V positive), SDA and SCL, which are data line and clock line respectively, are connected to the ports with the same names on the Arduino Board.

Let's delve into how the LCD screen is triggered. Before BPM is displayed, it is calculated by Edge computer. The calculation is done right after the edge device receives the peak R signal from Arduino. Through serial communication, the edge computer sends the BPM value in encoded string format. Arduino will read this encoding and output bits to the LCD screen.

LCD screen with I2C protocol would not be able to communicate well if Arduino does not integrate the LiquidCrystal_I2C library. This is a library developed by Marco Schwartz to ease the I2C communication of LCD, providing easy-to-use syntax and methods of activating LCD.

```
// initialize the serial communication:
Serial.begin(9600);
lcd.init();           // Initialize the LCD actuator
lcd.backlight();      // Turn on the backlight

lcd.setCursor(0,0);
lcd.print("Welcome");
delay(2000);
lcd.clear();
```

Fig 9. LCD initialisation

```
lcd.setCursor(0,0);
lcd.print("BPM:");
lcd.setCursor(0,4);
lcd.print(bpm); // Display the BPM value from Python
delay(2000);
lcd.clear();
```

Fig 10. Write the BPM value calculated by the edge device to the LCD and keep it for 2 seconds

4) Arduino Board

This is the most important component of the architecture. Acting as the central processing unit of the system, it plays many roles at once:

- Power supply: it acts as the power supply for the sensor and the actuators
- Serial communication: Arduino is connected with the edge computer via the “/dev/cu.usbserial-110” port, also known as the USB-C port of the Macbook. Arduino can send data to the edge device for analytics and receive processed data thanks to the capability of serial communication.
- Signal receiver: Arduino receives heartbeat signals from the AD8232 sensor, as well as the buzzer bit triggers and BPM values from the edge device.
- Trigger actuators: Arduino board explicitly triggers the buzzer by sending DC and writing characters on LCD with the assistance of the LCD_I2C library.

```
// read from sensor and record time
int heartbeatSignal = analogRead(sensorPin);
Serial.println(heartbeatSignal);
```

Fig 11. Arduino reads heart signals from the sensor and outputs to serial communication

5) MacOS computer

The Macbook laptop acts as an edge computer in this IoT-based system. It is an indispensable component for the serial communication capability of the system. The serial communication itself is initiated on the Macbook using the “pyserial” library.

```
ser = serial.Serial('/dev/cu.usbserial-110', 9600)
```

Fig 12. Serial communication initialisation

```
line=ser.readline().decode('utf-8').strip()
ser.write(b'B')
ser.write(f'{bpm:.2f}\n'.encode())
```

Fig 13. I/O operations using pyserial

Edge computer takes responsibility for most of the calculation, identification and data analytics tasks. In addition, it is also the main power supply for the Arduino board, which then distributes

the power among the other components. The code below shows how the edge computer calculates the BPM value from the received heart signal.

```
if line.isdigit() and int(line) > 450:
    current_time = time.time()
    if last_time != 0:
        bpm = 60 / (current_time - last_time)
        print(f"BPM: {bpm:.2f}")
        ser.write(b'B')
        ser.write(f"{bpm:.2f}\n".encode())
```

Fig 14. BPM value calculation

A Python library called “time” is integrated into the edge device’s program (a Python script) to record time stamps. In general, the time difference in seconds between the current beat and the last recorded beat is deducted, and then 60 is divided by this difference. The result is the BPM quantity, also known as “beats per minute”.

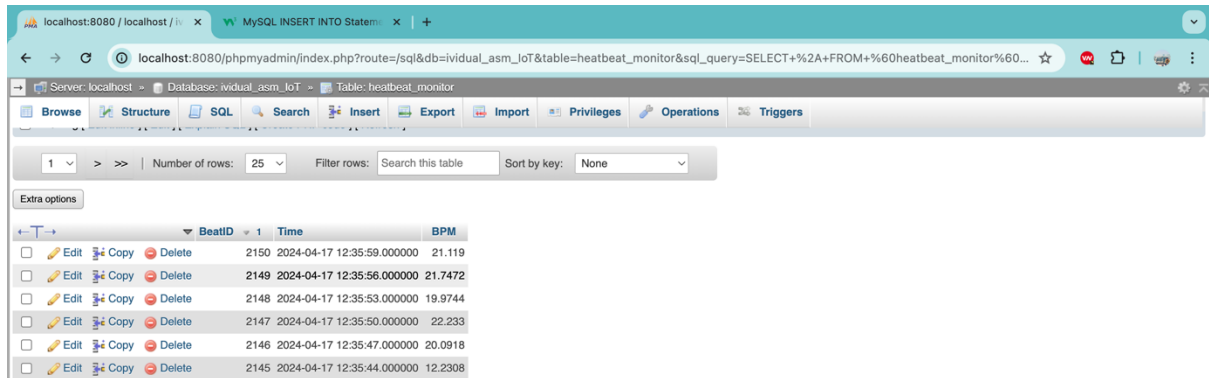
Nevertheless, the edge computer does not stop at numeric calculations. With the assistance of the “pymysql” library, edge device can record BPM values on the phpMyAdmin MySQL database. “pymysql” provides methods resembling functionalities of MySQL language, which enables database connection, creation and alteration. In this project, the database is called “ividual_asm_IoT” (sorry for the typo during creation ☺), which consists of a table named “heartbeat_monitor”. This table provides storage for BPM values from patients together with the time at the moment the value is recorded. The whole database itself is locally hosted on the edge device so that connections and queries are easy to deploy. Because the phpMyAdmin endpoint has already been installed on this Macbook edge computer, we can use this webpage as a platform to observe, analyse and interact with the database.

```
date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
cur.execute('insert into heartbeat_monitor (BPM, Time) values (%s, %s);', (bpm, date))
conn.commit()
```

Fig 15. Data is recorded to MySQL database

```
conn=pymysql.connect(user="sonngo",password="",host="localhost",port=3306,database="ividual_asm_IoT")
```

Fig 16. Database connection initialization



BeatID	Time	BPM
2150	2024-04-17 12:35:59.000000	21.119
2149	2024-04-17 12:35:56.000000	21.7472
2148	2024-04-17 12:35:53.000000	19.9744
2147	2024-04-17 12:35:50.000000	22.233
2146	2024-04-17 12:35:47.000000	20.0918
2145	2024-04-17 12:35:44.000000	12.2308

Fig 17. Database displayed on phpMyAdmin

Last but not least, the edge computer can detect leads-off electrodes by receiving digital signals from Arduino, which was previously generated by the sensor.

```
elif line == "!":
    print("Leads off detected!")
```

Fig 18. Leads off detection

6) Use case diagram

From the information collected above, the following use case diagram can be deduced.

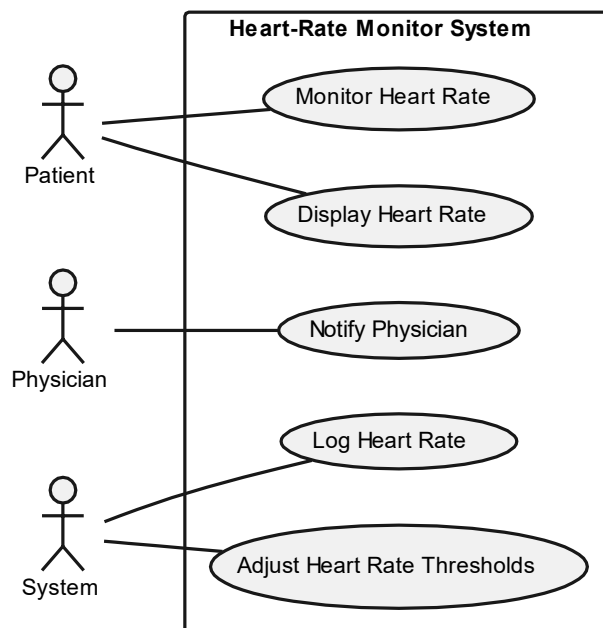


Fig 19. Use case diagram for Arduino-based ECG monitor system.

Overall, the system has a total of three actors: the patient, the physician and the system itself; and 5 use cases: Monitoring heart rate, heart rate visualising, assisting physicians in diagnosis, storing heart rate values and adjusting heart rate baseline. The diagram classifies each use case to the corresponding actors, thus giving readers an overview of the system's applicability and functionalities. Let's describe each use case:

- Monitor heart rate: This is the most important functionality of this system. Without it, the whole system becomes useless. The mechanisms behind this use case have been analysed in the previous section.
- Heart rate visualisation: This can be done in many ways. The first form is the tabular visualisation, which is the table hosted on phpMyAdmin. Heart rate visualisation can also be seen from the "Serial Plotter" integrated into Arduino IDE. To access it, go to Tools > Serial Plotter. Moreover, heart rate can be displayed in the form of BPM values on the LCD screen alone. Finally, heart rate can also be illustrated on a webpage or mobile software. This is out of the scope of this project.
- Diagnosis: Given the heart rate BPM and plots, physicians can easily observe the activities of the heart and identify any abnormal patterns, thus diagnosing the diseases and coming up with treatments.
- Log heart rate: The system logs all the data collected from the sensor to the RSDM database hosted locally on phpMyAdmin. The internal workflow of this process has been discussed in section 5 about MacOS edge computer.
- Adjust baseline: This is an advanced functionality that has not been deployed in the IoT-based system. This use case requires more complex data analytics to calculate the mean average heart rate. Nevertheless, if it is deployed successfully for Arduino, it will be a milestone in bringing IoT closer to Data Science.

IV. Resources

- Github repository “kitloong/mac-homebrew-lamp.md”: This repository is a guideline to install phpMyAdmin on a MacOS M1 chip, together with launching Apache and installation of PHP and MySQL. By doing these tasks, turning a MacOS computer into an edge device is feasible.
- How2electronics - ECG Monitoring with AD8232 ECG Sensor & Arduino: It has both YouTube and webpage guidelines for building a basic Arduino circuit using AD8232 as the only sensor. Source code is taken into account.
- Arduino Serial Plotter: this is a tool provided by Arduino IDE to plot cardio graphs based on detected heartbeat signals. It is used as an additional means of visualisation. To access it, go to Tools > Serial Plotter.
- (Last Minute Engineers - In-Depth: Interfacing an I2C LCD with Arduino): This webpage provides a detailed guideline for deploying I2C LCD into Arduino circuits and how to program it. Source code to make displays on LCD is taken into account.
- Tronics Lk (2022) - How to Use I2C LCD with Arduino: Similar to Last Minute Engineer’s guide, this video shows how to wire and program I2C LCD easily.
- Arduino Docs - Liquid Crystal Displays (LCD) with Arduino: This manual introduces examples of programming with traditional LCD and how to wire it to Arduino. This is a complex approach, so it is only taken as a reference for comparison.
- Other references give additional scientific information to analyse more deeply about the system.

V. Appendix

Besides the hardware components of this IoT-based heartbeat monitor architecture, two programs are running behind the scenes:

1) `heart_monitor.ino`

This Arduino program is vital for low-level device controls and moderation between IoT components. It is responsible for explicit hardware interactions that ensure real-time data acquisition and control. Moreover, the Arduino program also enables coordination between other IoT components such as actuators and sensors. One of its main functionalities is to assign roles to different Arduino pins so that the resource is distributed optimally among the components. Using rule-based conditions, together with DC or AC pulses, the program can either command to trigger or shut down a component. These results in the high reliability and responsiveness of the system.

2) `heart_beat.py`

This Python script complements the Arduino program with capabilities in serial communication and data management. This script specialises in initiating and maintaining the data cycle between the Arduino board and the edge devices. The script can process the input data, perform necessary calculations to generate meaningful analytics, and then store these results in a relational database (MySQL in the context of this project). This not only assists real-time monitoring but also supports advanced data analysis for IoT-based systems, thus enabling long-term trend analysis and making informed decisions.

VI. Conclusion

This report has made a substantial analysis of the architecture of the ECG heartbeat monitoring system, which is based on the Arduino platform. The document has gone through the internal workflows of the main components, consisting of actuators and sensors, as well as how they are wired to the circuit. In addition, external libraries are also mentioned in the hardware descriptions. Serial communication is a highly essential segment for every IoT application, so it is also analysed and given in a plethora of examples. Given the fact that every technological system needs to have use cases specified, a UML use case diagram is also built to examine the practicability and applicability of this IoT monitoring system on a large scale. This ECG monitoring prototype is built beyond many online tutorials and manuals, by combining all of the original works. The result is a heartbeat monitor prototype that is multi-media, communicative and user-friendly. Nevertheless, the system needs programs working behind it to bridge the gap between hardware operations and data science tasks. If we image IoT systems as living creatures, hardware components would be the external body, and programs would be their central brains.

VII. References

- Github (n.d.). *Mac - Install Apache, PHP, MySQL + phpMyAdmin with Homebrew*. [online] Gist. Available at: <https://gist.github.com/kitloong/5f66a140f38b9698e8c3ed13b968ff47> [Accessed 10 Apr. 2024].
- How To Electronics (2019). *ECG Monitoring with AD8232 ECG Sensor and Arduino*. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=0yO3gqeoMJg&t=0s> [Accessed 9 Apr. 2024].
- How2electronics (2019). *ECG Monitoring with AD8232 ECG Sensor & Arduino*. [online] How To Electronics. Available at: <https://how2electronics.com/ecg-monitoring-with-ad8232-ecg-sensor-arduino/>.
- John Hopkins Medicine (n.d.). *Anatomy and Function of the Heart's Electrical System*. [online] www.hopkinsmedicine.org. Available at: <https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-and-function-of-the-hearts-electrical-system#:~:text=An%20electrical%20stimulus%20is%20generated>.
- Last Minute Engineers (2020). *In-Depth: Interfacing an I2C LCD with Arduino*. [online] Last Minute Engineers. Available at: <https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/#:~:text=Wiring%20an%20I2C%20LCD%20Display%20to%20an%20Arduino>.
- Söderby, K. (2021). *Liquid Crystal Displays (LCD) with Arduino | Arduino Documentation*. [online] docs.arduino.cc. Available at: <https://docs.arduino.cc/learn/electronics/lcd-displays/>.
- Tronics Lk (2022). *How to Use I2C LCD with Arduino | Very Easy Arduino LCD I2C Tutorial | Arduino 16x2 LCD I2C Tutorial*. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=CvqHkXeXN3M>.
- Upasani, V. and MACFOS (2021). *AD8232 Heart Rate Monitor Detail Guide; Interfacing with Arduino*. [online] Robu.in | Indian Online Store | RC Hobby | Robotics. Available at: <https://robu.in/ecg-sensor-ad8232-heart-rate-monitor-detail-guide-interfacing-with-arduino/>.