Table of Contents

1. Introduction 3

The AD8232 Sensor and Biopotential Signal Processing 3

What is ECG? 3

2. Circuit Configuration 4

I. Materials 4

II. Configuration 5

AD8232 ECG Sensor Configuration 5

OLED Display Configuration 6

Power Distribution Strategy 6

3. Source Codes 6

Core Libraries 6

Program Structure and Logic 6

Key Algorithms 7

4. Construction and Testing 10

I. Construction 10

II. Testing 10

Testing Methodology 10

Peak Detection and BPM Calculation Performance 10

5. Conclusion 13

6. REFERENCES 13

1. Introduction

Monitoring cardiac health is of critical importance in modern medicine and personal wellness applications. Electrocardiography (ECG) is a fundamental method used for diagnosing various cardiac conditions by recording the heart's electrical activity. This project aims to develop a portable, real-time ECG monitoring system.

Within the scope of this project, a high-sensitivity AD8232 ECG sensor is interfaced with a powerful STM32F446RE microcontroller. The ECG signal, processed by the sensor, is visualized in real-time for the user on a 128x64 resolution OLED display.

The AD8232 Sensor and Biopotential Signal Processing

The AD8232 is an integrated signal conditioning block specifically designed for ECG and other biopotential measurement applications. Its primary function is to extract, amplify, and filter small biopotential signals, even in the presence of noisy conditions such as those created by motion or remote electrode placement. This advanced design allows for the processed output signal to be easily acquired by an ultralow-power analogue-to-digital converter (ADC) or an embedded microcontroller.

What is ECG?

An electrocardiogram is a characteristic waveform that represents the electrical cycle of the heart. By analysing the components of this waveform, one can obtain information about the heart's physiological condition:

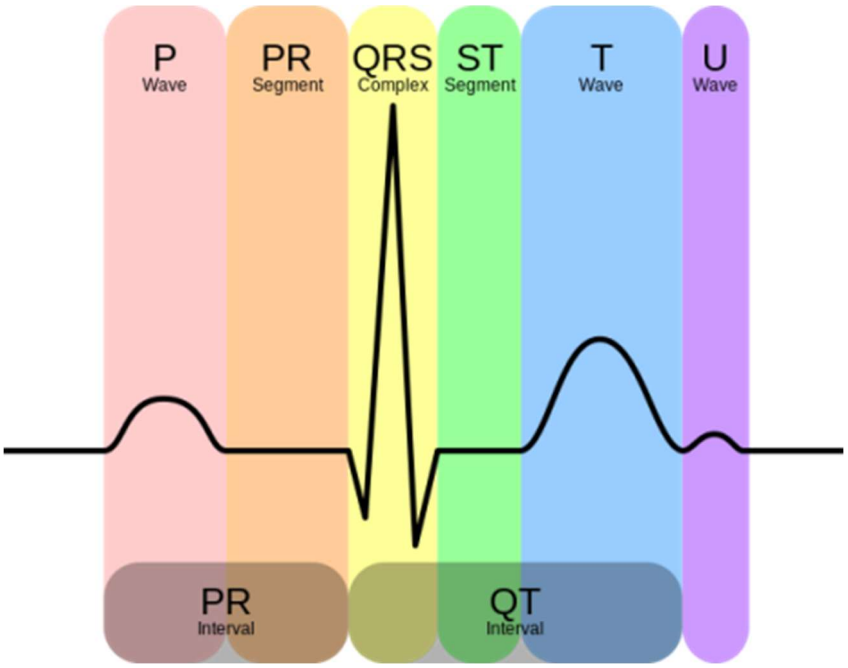

Figure 1: ECG signal

P Wave: The first upward deflection of the ECG tracing, indicating the contraction of the atria.

QRS Complex: Begins with a small downward deflection (Q), followed by a large upward peak (R), and concludes with a downward S wave. This complex signifies ventricular depolarization and contraction.

T Wave: Typically, a smaller, upward-sloping wave that represents ventricular repolarization (relaxation).

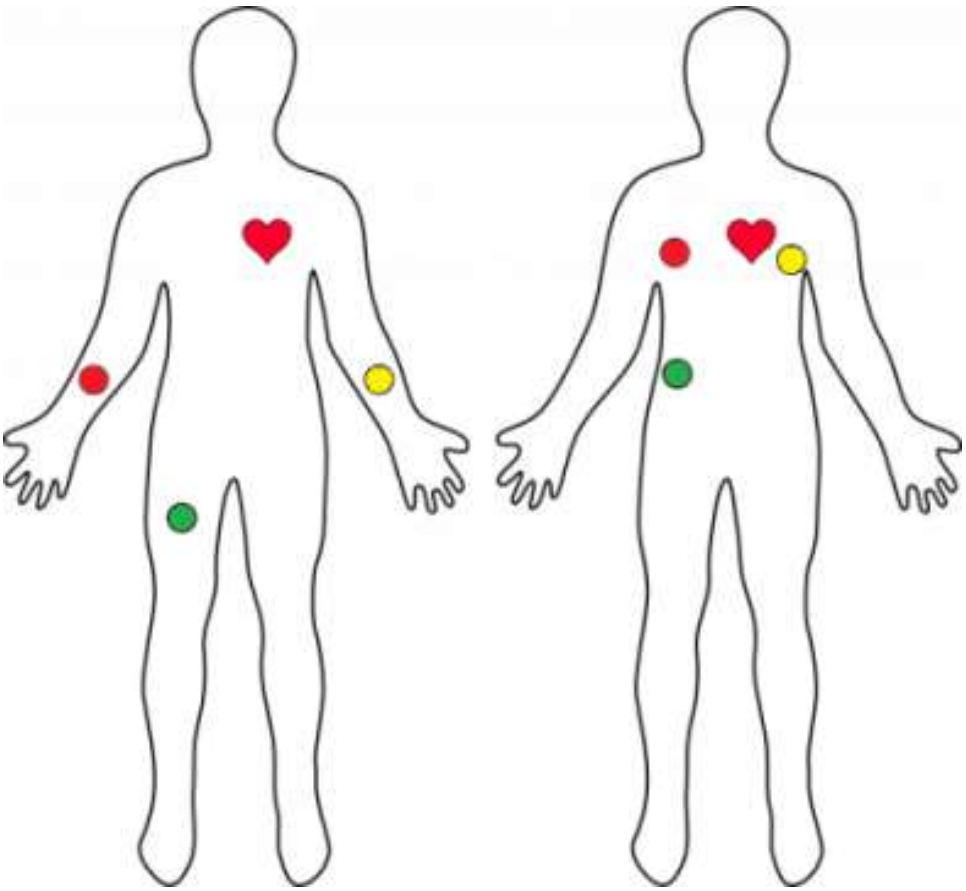

Figure 2: ECG probe placement

It is recommended to snap the sensor pads on the leads before sticking to the body, the closer to the heart, the better the measurement would be. The cables are color-coded to help you identify a proper placement.

2. Circuit Configuration

I. Materials

- STM32F446RE
- AD8232 ECG Sensor
- A0364 | 0.96 inch 128x64 OLED LCD (SSD1306)
- Jumper wires

II. Configuration

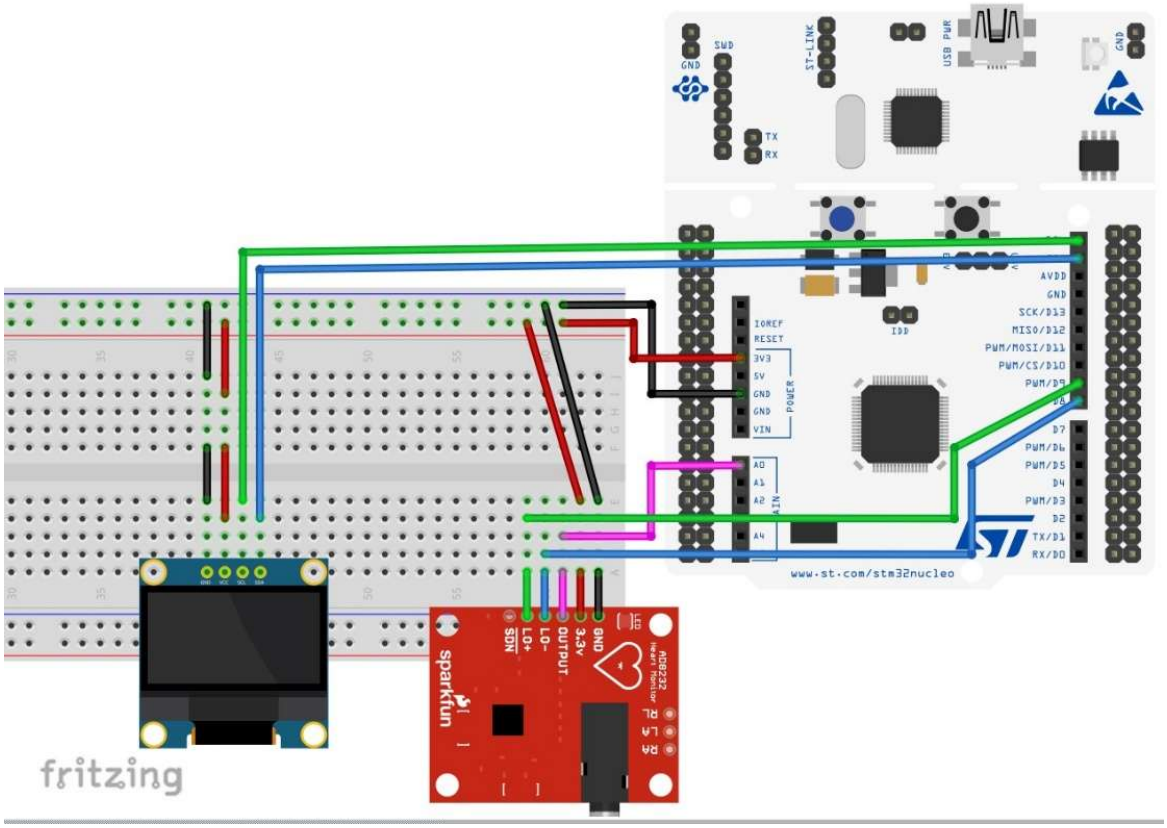

Figure 3: Circuit configuration

AD8232 ECG Sensor Configuration

The AD8232 sensor is the primary data acquisition module. Its five pins are connected to the STM32 Nucleo board to handle power, analogue signal output, and lead-off detection. The pin-out is as follows:

AD8232 Pin	STM32 Pin	Function
GND	GND	Common Ground Reference
3.3V	3.3V	Power Supply Input
OUTPUT	PA0 (A0)	Analog ECG signal output to the ADC
LO+	PA9 (D11)	Lead-Off Detection (Positive Electrode)
LO-	PA8 (D12)	Lead-Off Detection (Negative Electrode)

Table 1

The LO+ and LO- pins are connected to digital inputs on the STM32 to monitor the connection status of the electrodes. If an electrode is not making proper contact with the body, these pins go HIGH, allowing the software to detect the fault and display a warning.

OLED Display Configuration

A 128x64 pixel monochrome OLED display is used for real-time visualization of the ECG waveform. It communicates with the STM32 via the I²C protocol.

OLED Pin	STM32 Pin	Function
GND	GND	Common Ground Reference
VCC	3.3V	Power Supply Input
SCL	PB8 (SCL)	I²C Serial Clock Line
SDA	PB9 (SDA)	I²C Serial Data Line

Table 2

Power Distribution Strategy

Both the AD8232 sensor and the OLED display operate on a 3.3V logic level. A shared power rail was established using the 3.3V output from the STM32F446RE board.

An initial attempt was made to power the OLED display from the 5V output via a resistive voltage divider. However, this approach was not appropriate, as the display's current draw caused the voltage to collapse to an insufficient 0.9V. The final, stable configuration powers both modules from the same 3.3V line, which was measured to provide a reliable **3.3V** to the AD8232 and **3.2V** to the OLED display, ensuring correct operation.

3. Source Codes

The coding for the STM32F446RE microcontroller was developed using the **Arduino IDE**.

Core Libraries

The project relies on several key libraries to interface with the hardware components:

- Wire.h: The standard Arduino library for I²C communication. It is essential for communicating with the OLED display.
- Adafruit_GFX.h: A powerful graphics library by Adafruit that provides the core functions for drawing pixels, lines, shapes, and text.
- Adafruit_SSD1306.h: A specific hardware driver library for the SSD1306 controller used in the 128x64 OLED display. It works in conjunction with the GFX library.

Program Structure and Logic

The program operates on the standard Arduino setup() and loop() structure.

- setup() Function: This function runs once upon startup or reset. It handles all initial hardware configurations, including:
    - Initializing the I²C bus and the OLED display.
    - Setting the pin modes for the EKG analogue input (EKG_PIN) and the digital lead-off detection pins (LO_PLUS_PIN, LO_MINUS_PIN).
    - Clearing the display and preparing it for the main loop.
- loop() Function: This function runs continuously and contains the main logic for the EKG monitor. It is divided into two primary operational states:

Warning State: If the lead-off detection pins indicate that the electrodes are not properly connected, the system displays a "Check Electrodes" message.

Normal Operation State: If electrodes are connected, the system proceeds with data acquisition, processing, and display.

Key Algorithms

- Lead-Off Detection: The system constantly monitors the LO+ and LO- digital pins. A bool variable, isInWarningState, acts as a state flag. This flag ensures that the "Check Electrodes" message is only drawn once when the leads are first disconnected, and the screen is only cleared once when they are reconnected. This prevents screen flicker and makes the transition between states seamless.
- Beat Detection and BPM Calculation: Heart rate is calculated using a peak-detection algorithm over a fixed interval:
    - An analogue reading from the AD8232's OUTPUT pin is compared against a predefined beatThreshold.
    - To prevent a single heartbeat from being counted multiple times, a time-based debounce mechanism (minTimeBetweenBeats) ensures that a new beat is only registered after a minimum time has elapsed since the last one.
    - Detected beats are counted in a beatCounter variable over a 15-second interval (bpmCalcInterval).
    - At the end of the interval, the Beats Per Minute (BPM) is calculated using the formula: BPM = beatCounter * 4. The result is then stored and displayed for the next 15 seconds, providing a stable reading.
- Real-Time ECG Graphing: The ECG waveform is drawn on the OLED display in real-time:
    - The raw analog sensor value is scaled to fit the vertical resolution of the display using the map() function.
    - The display.drawLine() function draws a line segment from the last known coordinate to the newly calculated coordinate.
    - A variable, plotX, is incremented with each loop cycle to move the drawing position horizontally across the screen. When it reaches the right edge, the screen is cleared, and the drawing position resets to the left, creating a continuous scrolling effect.

The code also has line to line comment.

```
//Arduino i2c libary
#include <Wire.h>
//Screen libaries
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// --- Screen Settings and Analog Pin denifation ---
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1 //no reset pin
#define EKG_PIN PA0      //A0 Pin as input

// Lead-Off pins defination
const int LO_PLUS_PIN = D8;    //LO+ pin
const int LO_MINUS_PIN = D9;   //LO- pin

// --- BPM Calculating Limits Settings ---
int beatThreshold = 800;                    //Limit point to be taken as a Pulse
unsigned long minTimeBetweenBeats = 400;    // Debounce delay for beat detection to avoid false triggers.

// --- Screen defination ---
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// 15 second counting veriables
const long bpmCalcInterval = 15000; // 15 Second in ms
unsigned long lastBpmCalcTime = 0;  // Last bpm calculation time
int beatCounter = 0;                // bpm counter in 15 sec
int displayBpm = 0;                 // Bpm record for secreen dispilaying

bool isInWarningState = false;    //Lead-off connection bool
unsigned long lastBeatTime = 0;   //Protection for debounce calculation
int lastPlotX = 0;                //X cordination for plotting
int lastPlotY = 0;                //Y cordination for plotting
int plotX = 0;                    //time interval for plotting (screen flowing)
```

Figure 4

```
void setup() {
  Serial.begin(115200);
  pinMode(EKG_PIN, INPUT);       //Analog pin as input
  pinMode(LO_PLUS_PIN, INPUT);    //LO+ Digital input pin
  pinMode(LO_MINUS_PIN, INPUT);   //LO- Digital input pin

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { //System protection for screen working. If the screen adress wrong infinite loop
    while(true);
  }

  display.clearDisplay();
  lastPlotY = 30;              //Almost middle point to start screening
  display.display();           //Update the screen
  lastBpmCalcTime = millis();   //reset bpm time to now
}
```

Figure 5

```
52  void loop() {
53    bool leadsAreOff = (digitalRead(LO_PLUS_PIN) == HIGH || digitalRead(LO_MINUS_PIN) == HIGH); //bool of the elektrods are connected or not
54
55    if (leadsAreOff) {                //Electrod connection correction
56      if (!isInWarningState) {        //Check warning state
57        isInWarningState = true;      //Yes we are in warning state
58        display.clearDisplay();
59        display.setTextSize(1);
60        display.setTextColor(SSD1306_WHITE);
61        display.setCursor(15, 20);
62        display.println("ELEKTROTLARI");
63        display.setCursor(15, 35);
64        display.println(" KONTROL EDIN");
65        display.display();          //Setting screen warning state wrate text, cordinates and update screen
66
67        displayBpm = 0;
68        beatCounter = 0;
69      }
70    }
71    else { // Electrods are connected
72      if (isInWarningState) {         //If we were previously in the warning state, this block resets everything for normal operation.
73        isInWarningState = false;     // We are no longer in a warning state, so set the flag to false.
74        display.clearDisplay();       //Clear screen
75        lastBpmCalcTime = millis();   // Reset the 15-second BPM calculation timer.
76        plotX = 0;                    //Reset plot positioning left
77        lastPlotX = 0;
78        beatCounter = 0;              // Reset the beat counter and the displayed BPM value to zero.
79        displayBpm = 0;
80      }
81
```

Figure 6

```
82      // --- EKG Part ---
83      unsigned long currentTime = millis();       // Get the current timestamp for this loop cycle.
84      int sensorValue = analogRead(EKG_PIN);      // Read the raw analog value from the EKG sensor pin.
85      Serial.println(sensorValue);                // Send the raw sensor value to the Serial Plotter for debugging.
86
87      //  Detect peek point and increase bpm
88      if (sensorValue > beatThreshold && currentTime - lastBeatTime > minTimeBetweenBeats) {
89        beatCounter++;                // If the signal is high enough to be a pulse and bumper time passed then increment our count of bpm
90        lastBeatTime = currentTime; // Record the new bpm time
91      }
92
93      // Check 15 second is passed
94      if (currentTime - lastBpmCalcTime > bpmCalcInterval) { //If passed
95        displayBpm = beatCounter * 4;     // Calculate new bpm for sceen(1 minute)
96        beatCounter = 0;                  // Reset counter
97        lastBpmCalcTime = currentTime;    // Reset time
98      }
99
```

Figure 7

```
100     // --- Plotting ---
101     if (plotX >= SCREEN_WIDTH) {        // If the drawing position has reached the right edge of the screen
102       plotX = 0;
103       lastPlotX = 0;
104       display.clearDisplay();           // reset the horizontal position to the left and clear the entire screen
105     }
106     //4085 - 2047 -1023
107     int plotY = map(sensorValue, 0, 4095, 45, 5);                         // Map the raw sensor value (0-2048) to a vertical screen coordinate (45-5)
108     display.drawLine(lastPlotX, lastPlotY, plotX, plotY, SSD1306_WHITE);    // Draw a line from the previous point to the new current point
109
110     lastPlotX = plotX;        // Remember the current point's coordinates for the next line segment
111     lastPlotY = plotY;
112
113     // --- Calculate 15 second interval ---
114     int elapsedSeconds = (currentTime - lastBpmCalcTime) / 1000;  // Calculate how many seconds have passed since the last BPM update
115
116     display.fillRect(108, 0, 20, 8, SSD1306_BLACK);    // Clear the small area in the top-right corner where the countdown is displayed
117     display.setTextSize(1);           // Print the new countdown value in the top-right corner
118     display.setTextColor(SSD1306_WHITE);
119     display.setCursor(110, 0);
120     display.print(elapsedSeconds);
121
122
123     display.fillRect(0, 52, SCREEN_WIDTH, 12, SSD1306_BLACK);     // Clear the bottom area of the screen where the BPM value is shown
124     display.setTextSize(1);                         // Print the current BPM value
125     display.setCursor(35, 54);
126     display.setTextColor(SSD1306_WHITE);
127     display.print(displayBpm);
128     display.print(" BPM");
129
130     display.display();        //Update screen
131
132     plotX++;                  // Move the horizontal drawing position one pixel
133   }
134   analogReadResolution(12);
135   Serial.print(", 12-bit : ");
136   Serial.print(analogRead(A0));
137   delay(5);                   // Wait for 5 milliseconds to stabilize the loop and analog readings
138 }
```

Figure 8

4. Construction and Testing

I. Construction

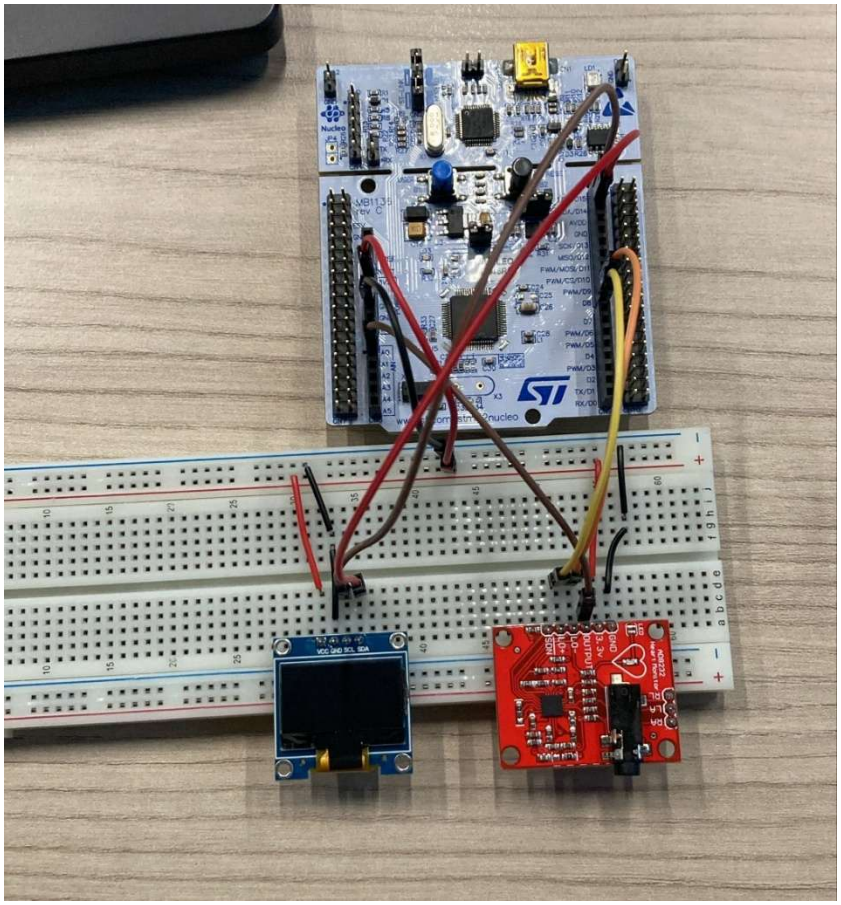

Figure 9: Circuit

II. Testing

Testing Methodology

The testing was conducted by connecting the AD8232 sensor to a test subject using three standard ECG electrodes. The system's output was monitored through two primary channels:

Real-Time Visual Feedback: The live ECG waveform and calculated Beats Per Minute (BPM) were observed on the 128x64 OLED display.

Numerical Data Analysis: The raw analogue-to-digital converter (ADC) values from the sensor were streamed to the Arduino IDE's Serial Monitor for quantitative analysis.

The primary objective of this phase was to confirm that the beat detection algorithm could reliably identify heartbeats from a live signal and produce a stable BPM reading.

Peak Detection and BPM Calculation Performance

The core of the firmware's logic is its ability to detect the R-wave peak of the QRS complex. This was achieved by implementing a two-factor algorithm:

- Signal Threshold (beatThreshold): Through empirical testing, a threshold value of 800 was determined to be optimal. During operation, raw ADC values for R-wave peaks consistently reached approximately 1003, making the 800-level threshold effective for distinguishing a true heartbeat from baseline noise and other waveform components like the P or T waves.
- Time-Based Debounce (minTimeBetweenBeats): To prevent a single, wide R-wave from being counted multiple times, a time-based filter was implemented. A value of 400 milliseconds was set as the minimum time that must pass before the system can register a new beat. This effectively sets a physiological maximum detectable heart rate of 200 BPM (60,000 ms / 300 ms), which is a robust limit for avoiding false detections from noise or T-wave interference.

Despite this "corrupted" signal, the system successfully demonstrated its robustness. The combination of the AD8232's powerful internal filters and the firmware's thresholding algorithm was sufficient to isolate the prominent R-wave peaks from the underlying noise. This confirmed that even without a "pure" signal, the system is capable of performing its primary function of calculating BPM.

Our code is worked to count the peak value. We determine the count limit is 800. When a signal passes the 800 our code thinks the detect a hearth pulse. Normally usual and max input value is almost 1003. We use time bumper to avoid signal mistake 0.4 second.

Normally we need to determine a pure ECG signal. However, our signal is corrupted. But we can anyway detect the bpm.

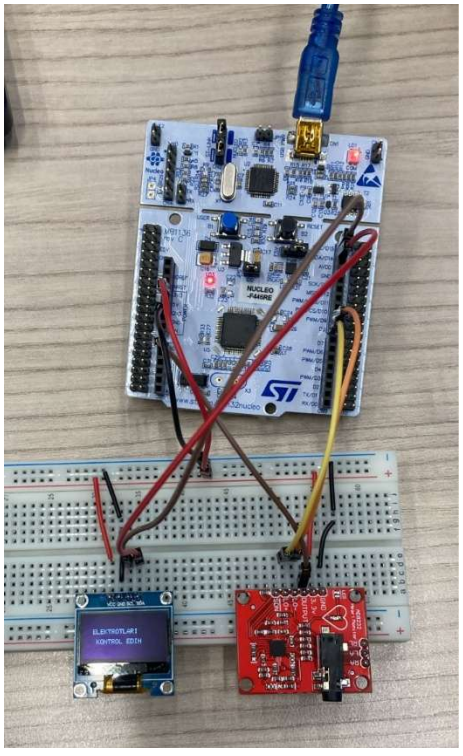

Figure 10: Lead-off condition

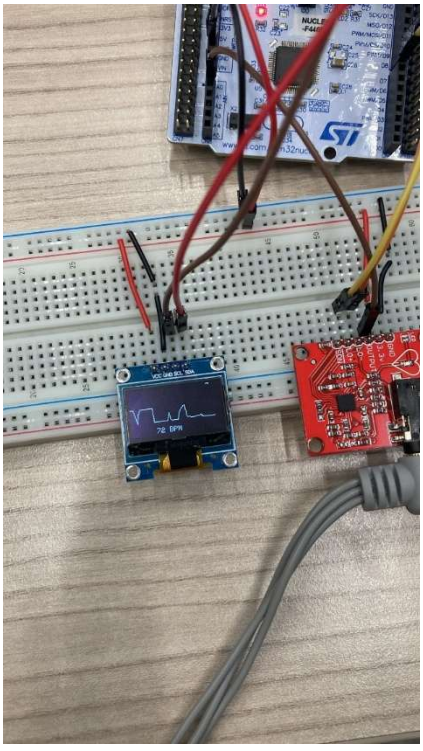

Figure 11

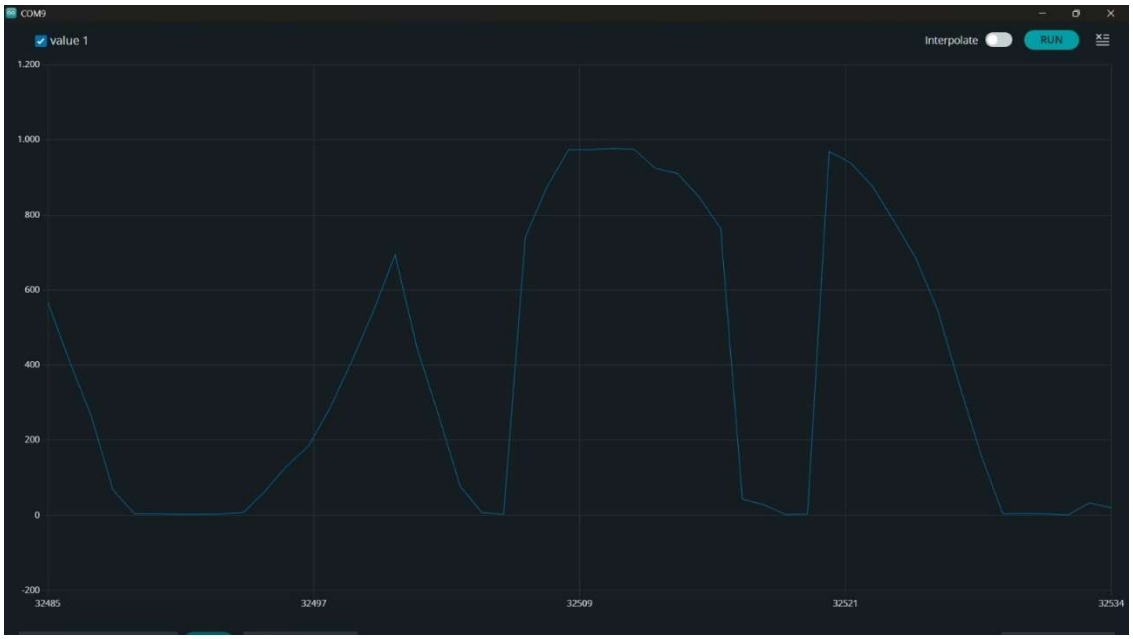

Figure 12: Ardunio serial plotter

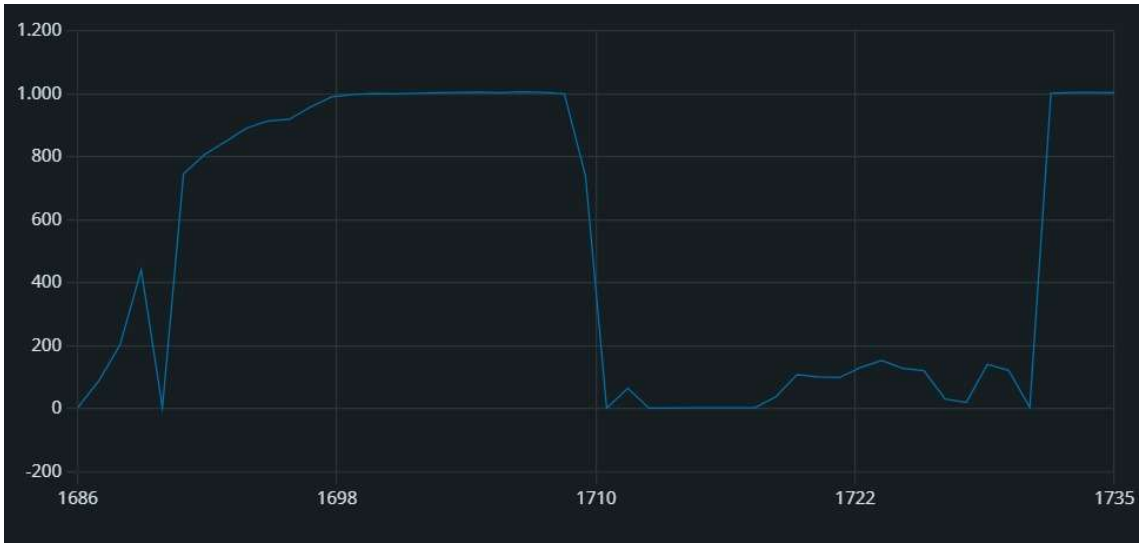

Figure 13

5. Conclusion

This project set out to design and implement a portable, real-time ECG monitoring system by interfacing an AD8232 sensor with an STM32F446RE microcontroller and an OLED display. The primary objective was to successfully acquire, process, and visualize a live electrocardiogram signal. The developed prototype is capable of displaying a continuous ECG waveform and calculating a user's heart rate in Beats Per Minute (BPM).

A significant challenge encountered during development was the quality and integrity of the input signal. The raw data acquired from the AD8232 sensor contained a considerable amount of noise. After extensive testing, it was concluded that the specific sensor unit being used was likely faulty, as the noise level was higher than specified in the component's datasheet.

Despite this hardware limitation, a key success of this project was the development of robust firmware that could balance for the suboptimal signal. The software algorithm, which uses a combination of a signal-level threshold and a time-based debounce filter, proved effective at extracting the prominent R-wave peaks from the noisy data. This allowed for a accurate BPM calculation, demonstrating the software's handling.

6. REFERENCES

1. ECG with STM32 and AD8232 – Martinloren
2. Bluetooth-for-STM32-and-Android/README.md at master · viktorvano/Bluetooth-for-STM32-and-Android
3. Interferences between i2c oled screen and analog sensor - Projects / Networking, Protocols, and Devices - Arduino Forum
4. OLED ECG Circuit Not Displaying the Signal - Projects / Programming - Arduino Forum
5. Sistem Pemantauan Sinyal Jantung AD8232 Dengan LCD oled | SINAU PROGRAMMING
6. AD8232 Kalp Atış Hızı Sensörü Kullanımı - Arduino - Robolink Akademi
7. Arduino İle AD8232 Nabız-EKG Modülü Kullanımı - Arduino Destek
8. Arduino Based ECG & Heartbeat Monitoring Healthcare System : 8 Steps - Instructables
9. Oled 0.96 with pulse sensor fails to display properly - Other Hardware / Displays - Arduino Forum
10. AD8232 ECG Sensor & Arduino Interfacing with ECG Graph
11. Ona Kalp Atışlarınızı Hediye Edin - Lezzetli Robot Tarifleri