The IOT Based Jacket

Eng. Elie El Arou
Eng. Atieh Atieh

Outline

INTRODUCTION

STATISTICS (1)

ONE IN FOURTH

one in fourth of the elderly population report falling each year

300,000 HOSPITALIZED

There is 300,000 elderly individuals hospitalized annually due to broken hips

$ 6,000,000,000 COSTS

medical costs for broken hips in the U.S. is six billion dollars per year

STATISTICS (2)

33%

33% of elderly individuals do not survive beyond one year after experiencing a hip fracture

31.8%

31.8% of elderly mortality in the US is cause by heart disease

WE HAVE THE SOLUTION!

PROJECT REQUIREMENTS

ELECTROCARDIOGRAM

Monitor live ECG

O2 LEVEL & HEART RATE

Monitor heart rate and oxygen level with 85 to 90% accuracy

TEMPERATURE

Track body and room temperature with 98% accuracy

FALL DETECTION

Track any fall and deploy airbag

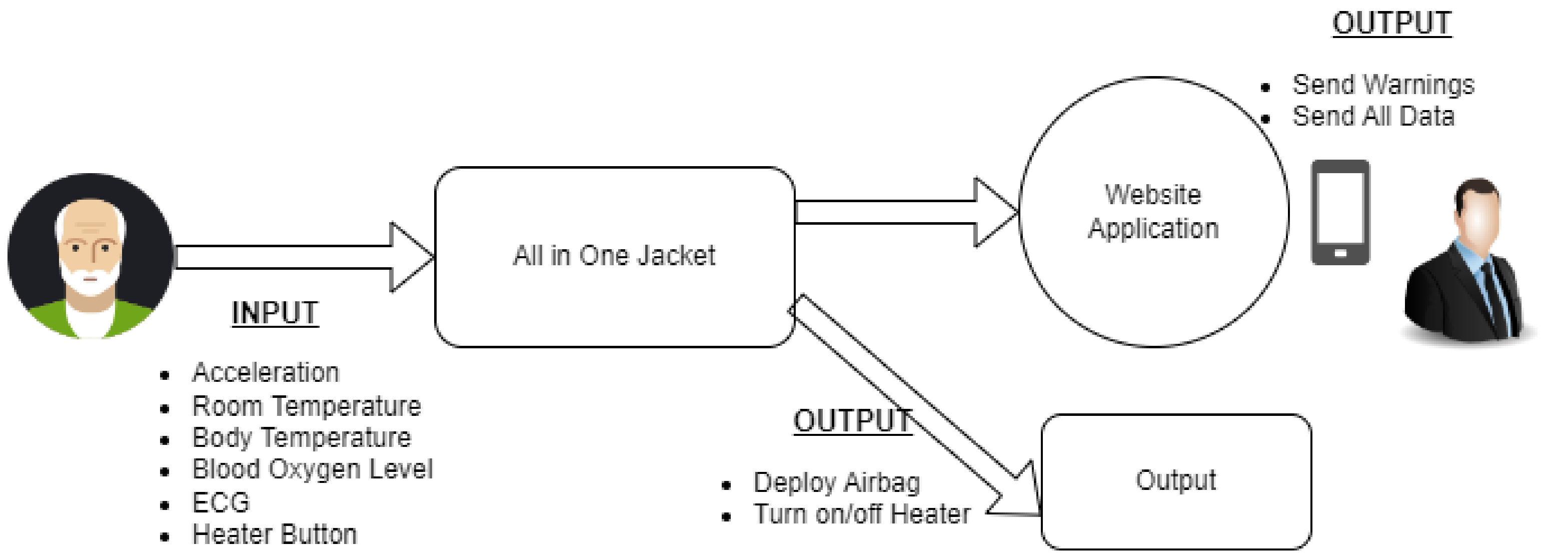
WARMING MODULE

Give warmth through heated sheets

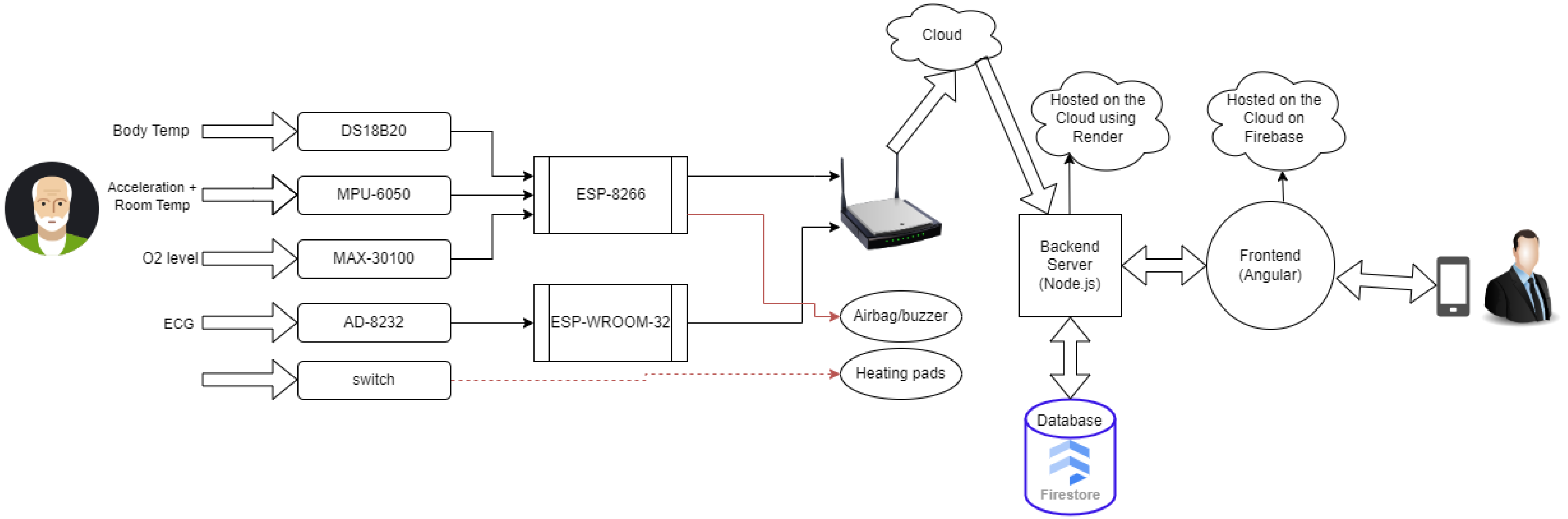
INTERFACE

Displayed and send notification on a web and PWA application

FUNCTIONAL DECOMPOSITION LEVEL 0



FUNCTIONAL DECOMPOSITION LEVEL 1



HARDWARE COMPONENTS

MICROCONTROLLERS
ESP-8266 & ESP-WROOM-32

ELECTROCARDIOGRAM
AD-8232

BLOOD OXYGEN LEVEL
MAX-30100

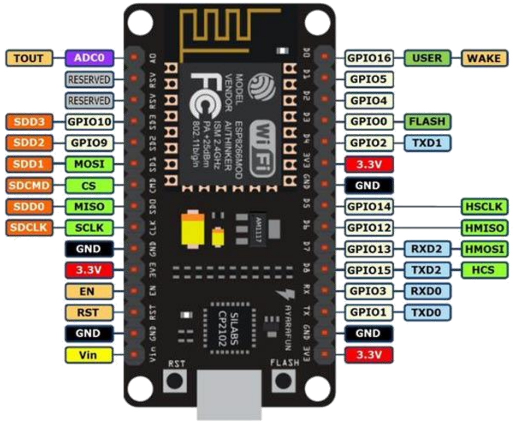
FALL DETECTION
MPU-6050

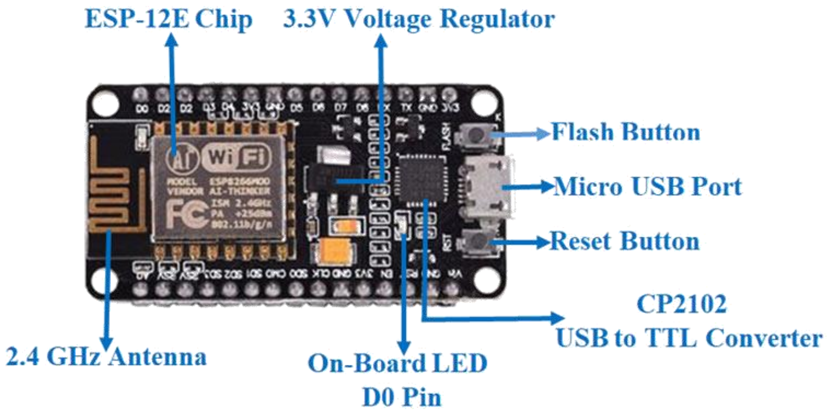
BODY TEMPERATURE
DS18B20

Heating pads
Heating pads

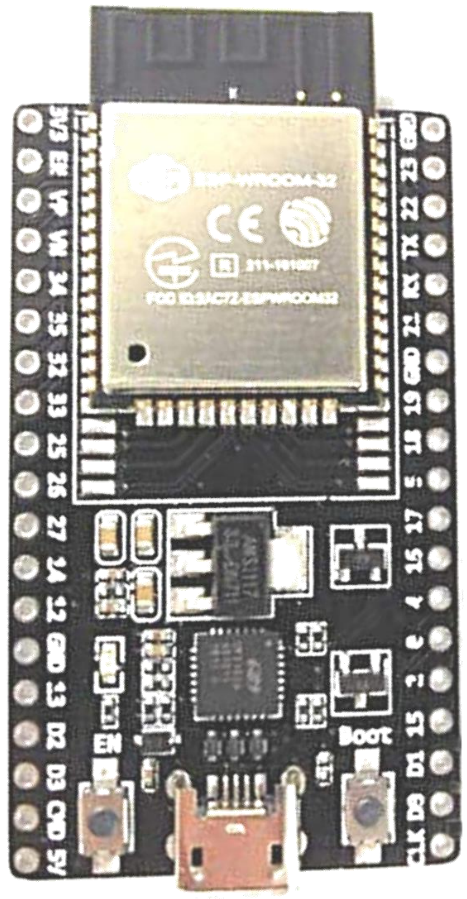
ESP-8266 Specifications

- Price: 3.5$
- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- PCB Antenna



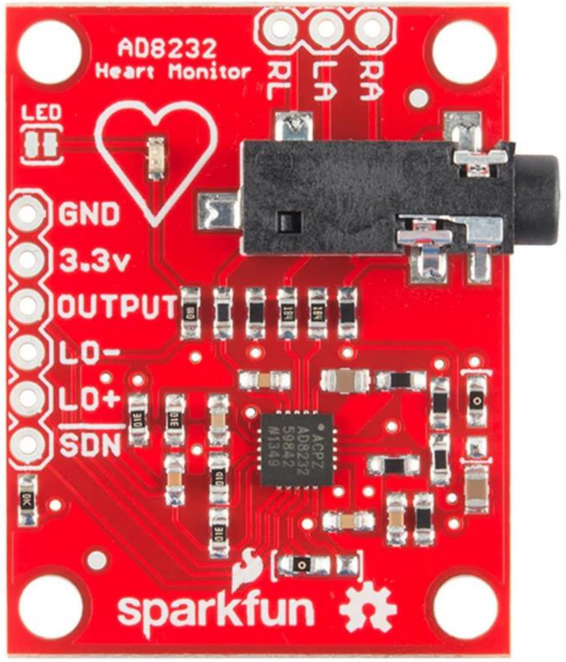


ESP-WROOM-32 Specifications



- Microprocessor: Tensilica Xtensa LX6
- Maximum Operating Frequency: 240MHz
- Operating Voltage: 3.3V
- Analog Input Pins: 12-bit, 18 Channel
- DAC Pins: 8-bit, 2 Channel
- Digital I/O Pins: 39
- DC Current on I/O Pins: 40 mA
- DC Current on 3.3V Pin: 50 mA
- SRAM: 520 KB
- Communication: SPI(4), I2C(2), I2S(2), CAN, UART(3)
- Wi-Fi: 802.11 b/g/n
- Bluetooth: V4.2 – Supports BLE and Classic Bluetooth

AD-8232 Specifications

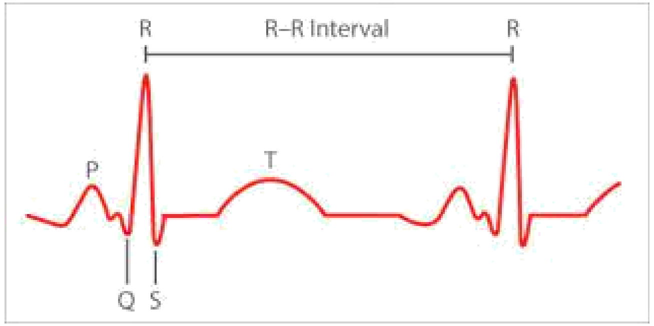


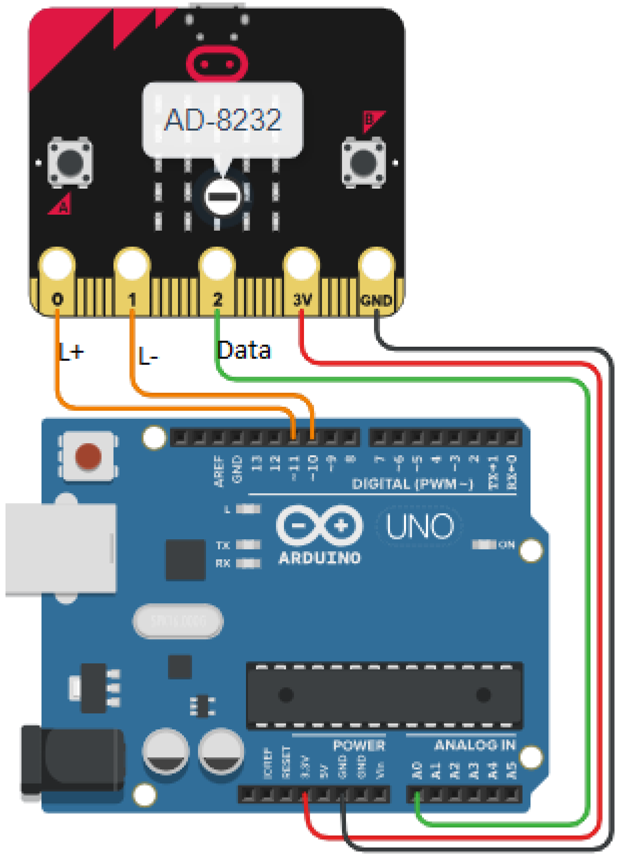
- Price: 11$
- Fully integrated single-lead ECG front end
- Supply current: 170 µA
- Common-mode rejection ratio: 80 dB
- Two or three electrode configurations
- Accepts up to ±300 mV of half cell potential
- Fast restore feature improves filter settling
- Leads off detection: ac or dc options
- Integrated right leg drive (RLD) amplifier
- Single-supply operation: 2.0 V to 3.5 V

AD-8232 Specifications

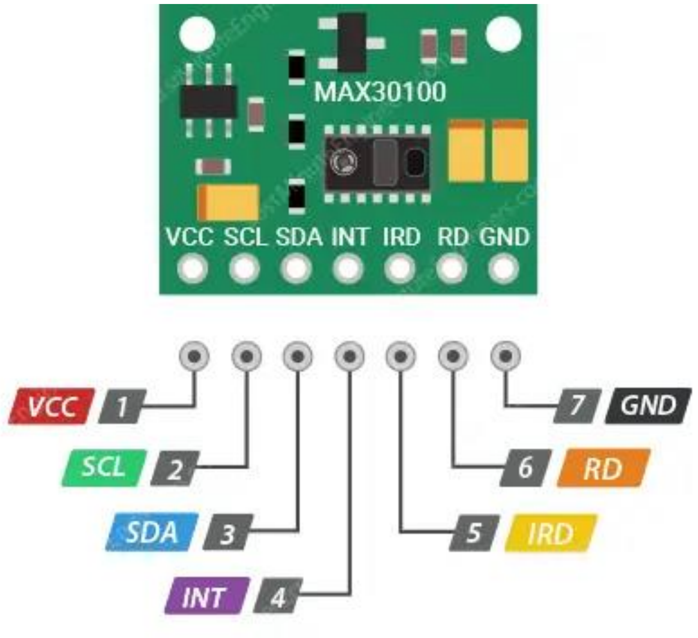
PINS:

- SDN
- LO+
- LO-
- OUTPUT
- 3.3V
- GND
- right arm (RA)
- left arm (LA)
- and right leg (RL)





MAX-30100 Specifications

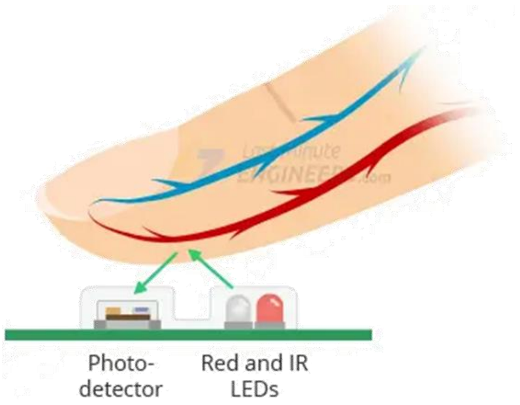


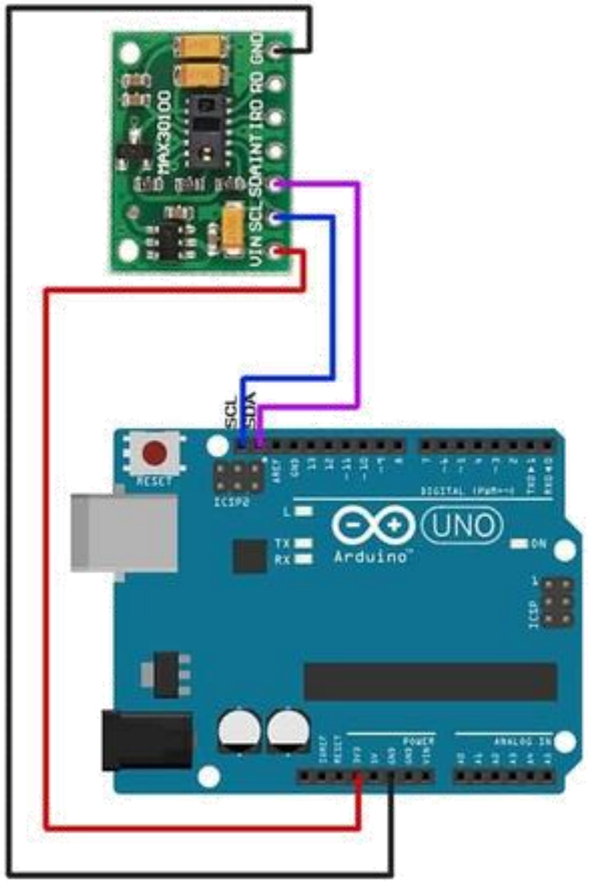
- Two light-emitting diodes
- Two photo detectors, optimized optics and low noise simulation.
- Communication modes: standard IIC protocol
- Power supply: 3.3V to 5.5V
- Current draw: ~600μA (during measurements)
- Current draw: ~0.7μA (during standby mode)
- Red LED Wavelength: 660nm
- IR LED Wavelength: 880nm
- Temperature Range: -40˚C to +85˚C
- Temperature Accuracy: ±1˚C

MAX-30100 Specifications

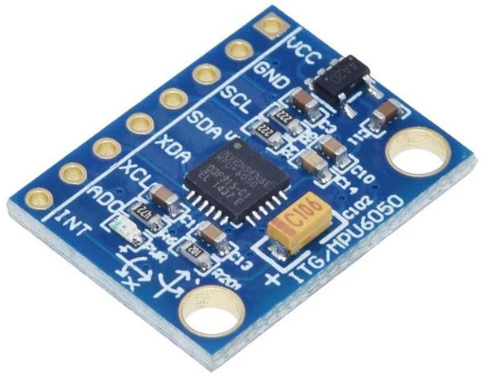
PINS:

- VCC
- GND
- SCL
- SDA
- INT
- IRD
- RD





MPU-6050 Specifications

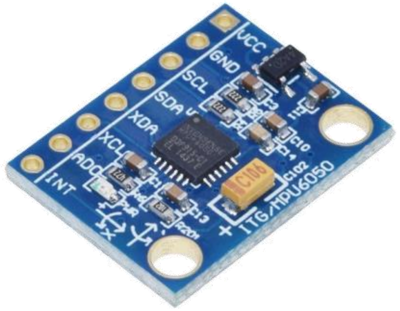


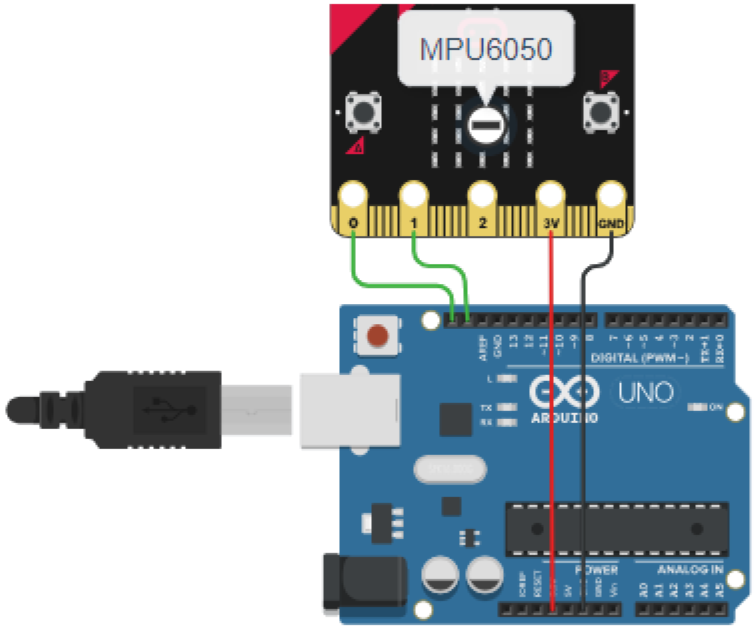
- Power supply :3-5v
- Communication modes: standard IIC communications protocol
- DEGREES OF FREEDOM : 6 x
- Chip built-in 16bit AD converter, 16-bit data output
- Acceleration range: ± 2 ± 4 ± 8 ± 16g
- Pin spacing 2.54mm

MPU-6050 Specifications

PINS:

- VCC
- GND
- SCL
- SDA
- XCL
- XDA
- ADD
- INT





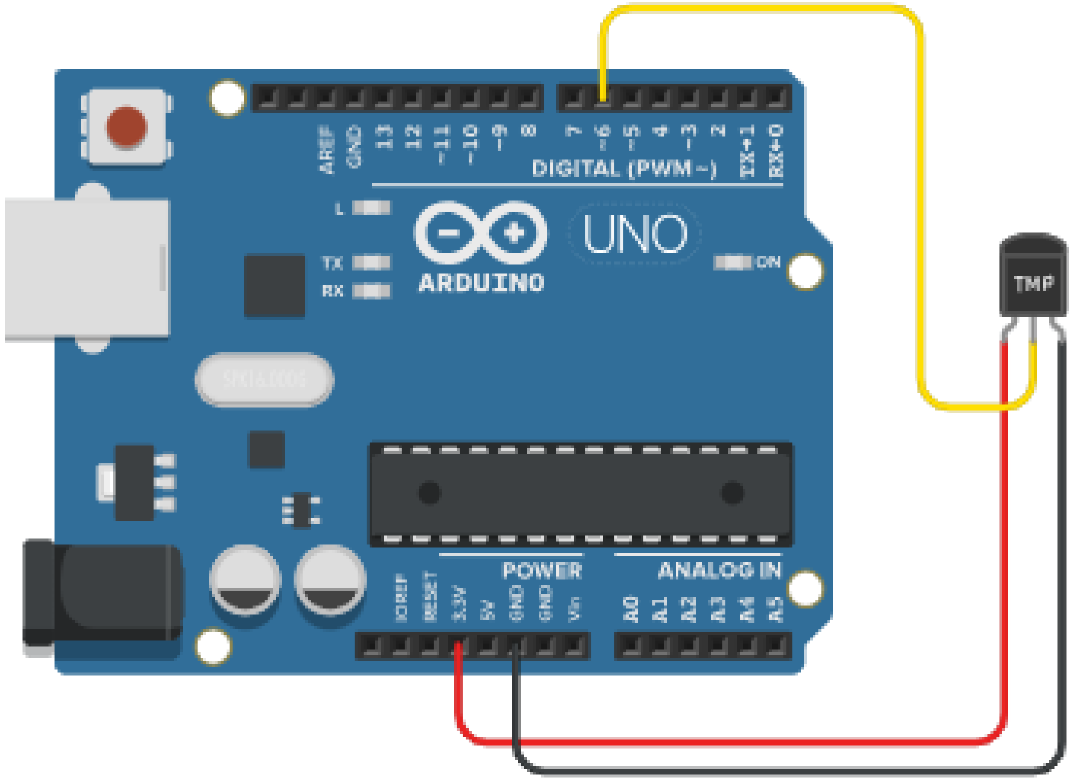
DS18B20 Specifications

- 1-Wire interface requires only one port pin for communication
- Requires no external components
- Power supply range is 3.0V to 5.5V
- Zero standby power required
- Measures temperatures from -55°C to +125°C(-67°F to +257°F)
- ±0.5°C accuracy from -10°C to +85°C

DS18B20 Specifications

PINS:

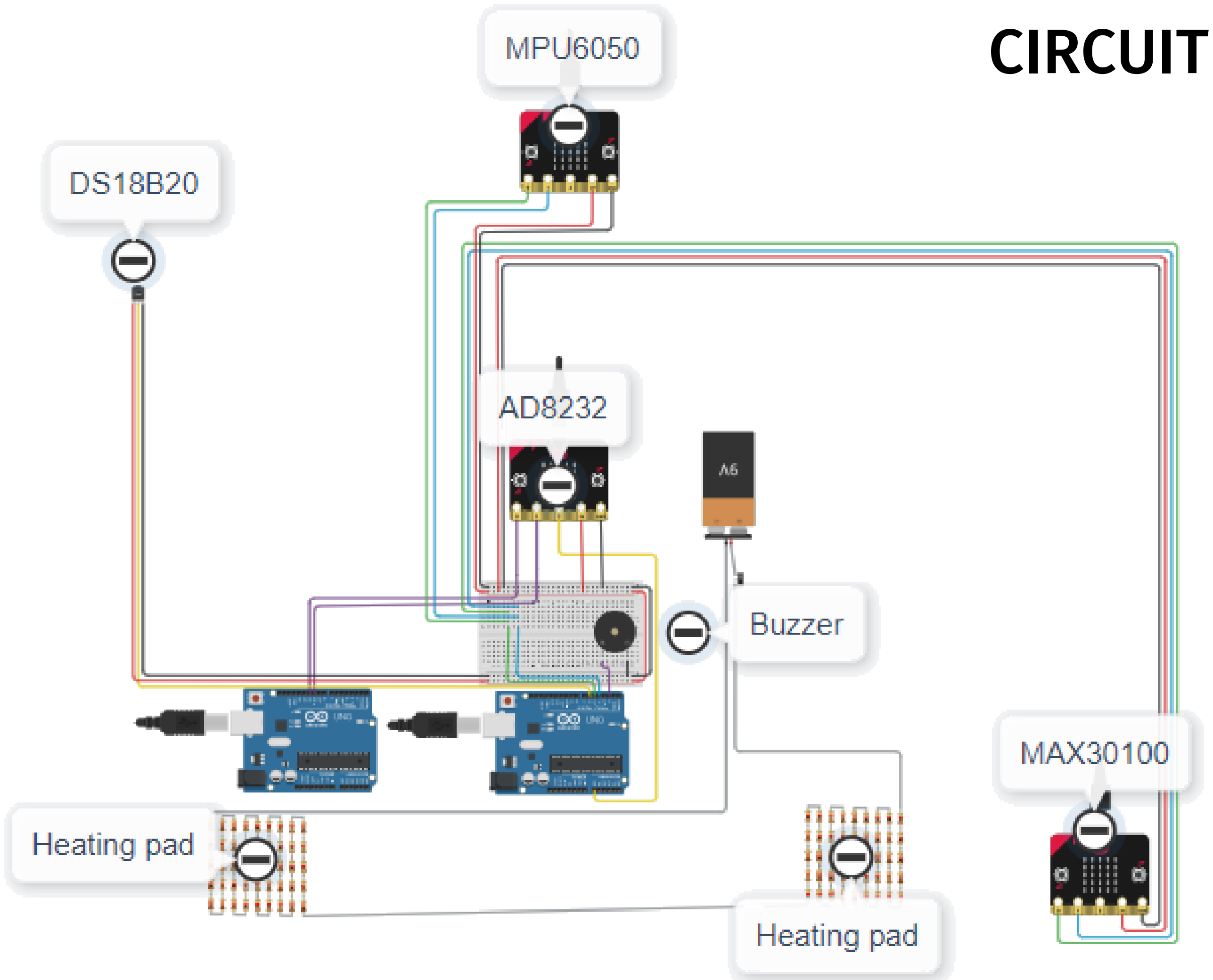
- VCC
- GND
- DATA



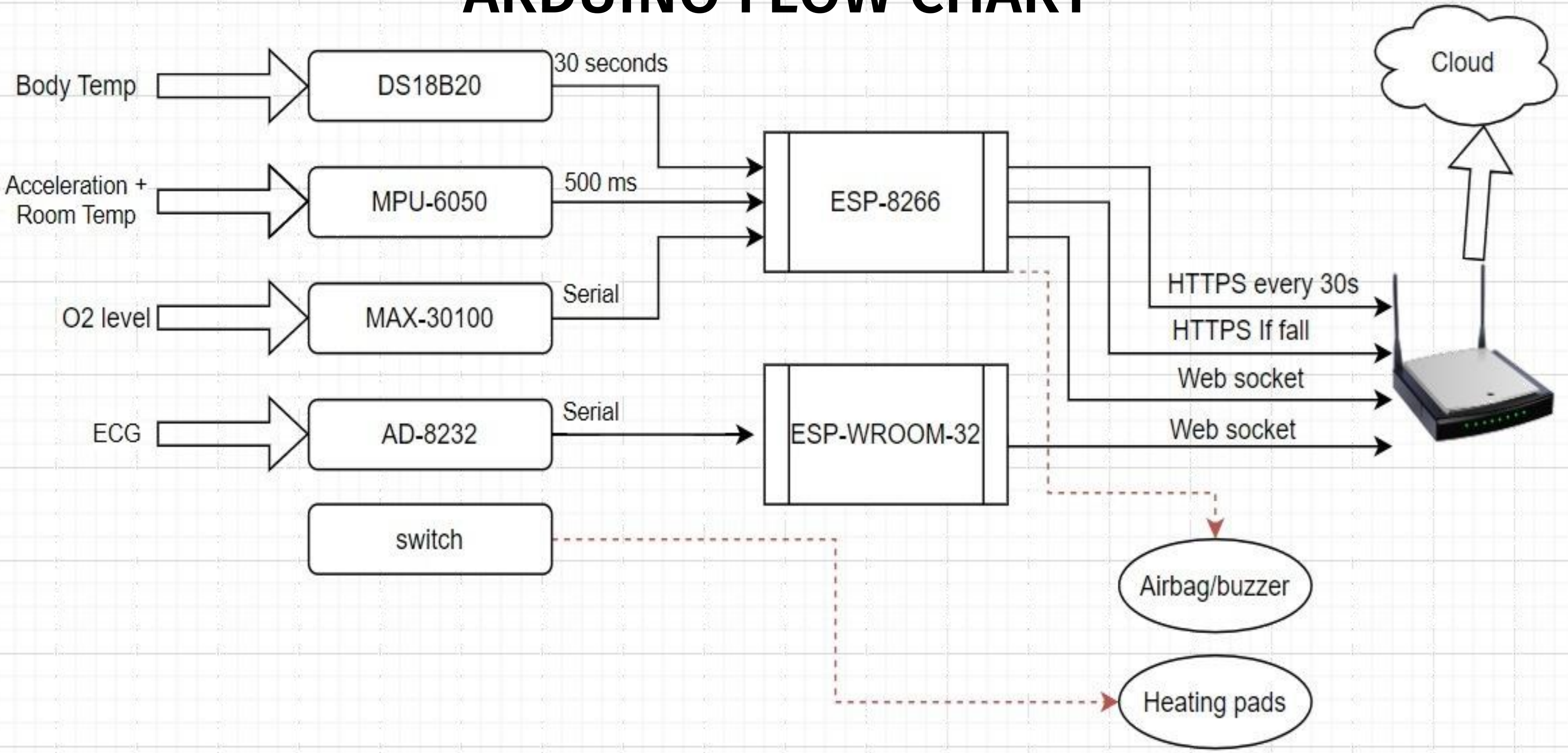
AD-8232 Specifications

- Material: Silicon Rubber
- Dimensions: 5x5cm
- Voltage input: 12VDC
- Power consumption: 10W

CIRCUIT IMPLEMENTATION

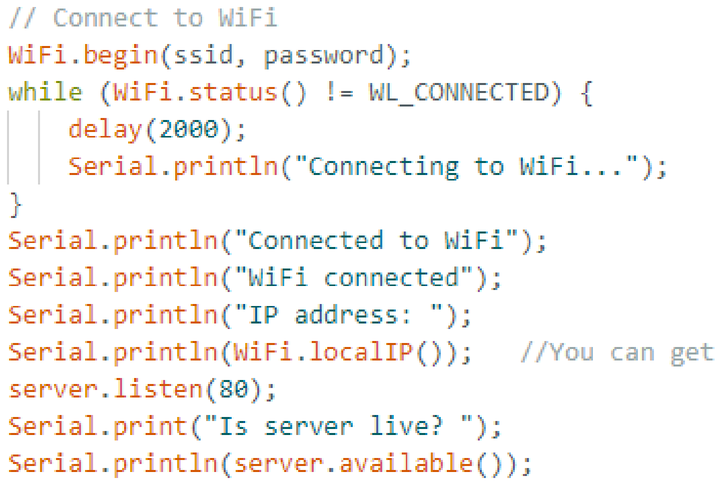


ARDUINO FLOW CHART



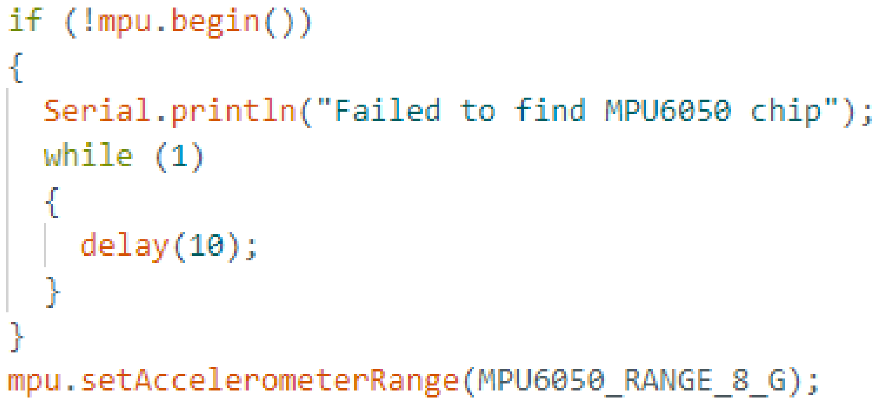
ARDUINO SETUP

Wi-Fi connection:



```
// Connect to WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(2000);
    Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());   //You can get
server.listen(80);
Serial.print("Is server live? ");
Serial.println(server.available());
```

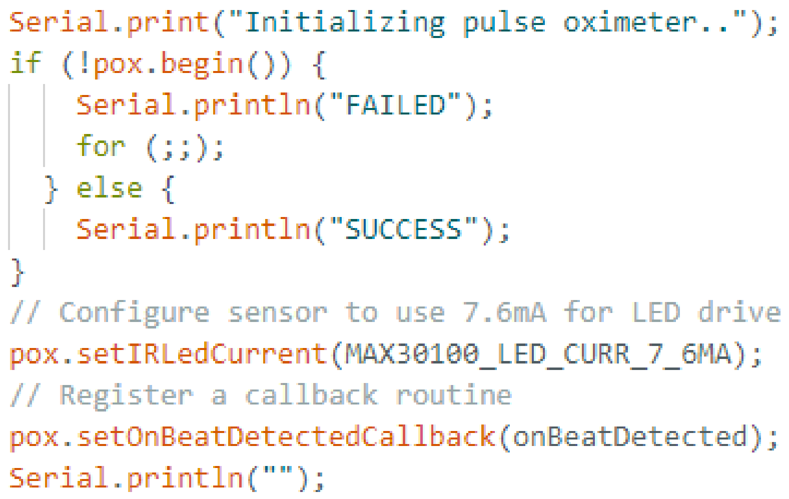
MPU-Gyroscope:



```
if (!mpu.begin())
{
  Serial.println("Failed to find MPU6050 chip");
  while (1)
  {
    delay(10);
  }
}
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
```

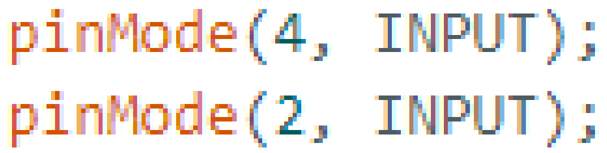
ARDUINO SETUP (2)

MAX-30100 oxygen level:



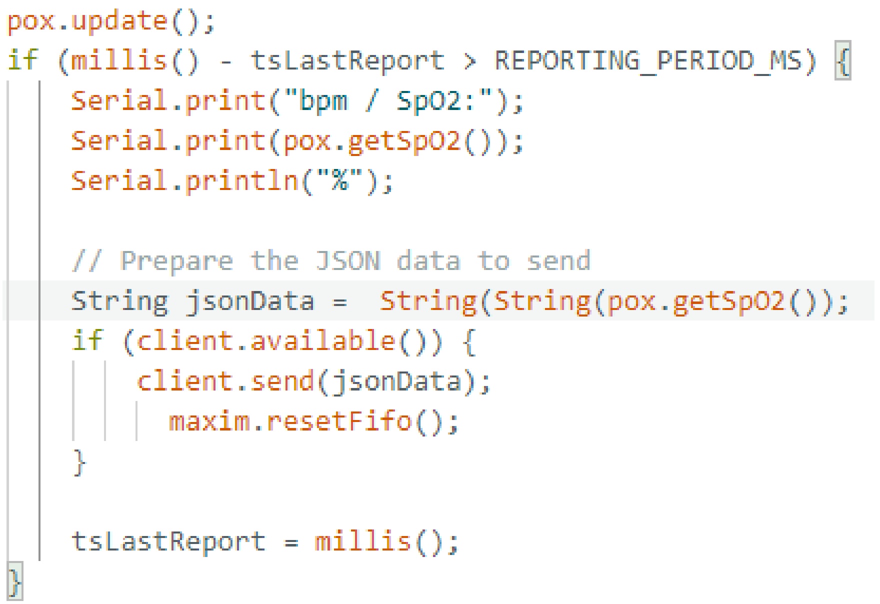
```
Serial.print("Initializing pulse oximeter..");
if (!pox.begin()) {
    Serial.println("FAILED");
    for (;;);
  } else {
    Serial.println("SUCCESS");
}
// Configure sensor to use 7.6mA for LED drive
pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
// Register a callback routine
pox.setOnBeatDetectedCallback(onBeatDetected);
Serial.println("");
```

AD-8232 ECG

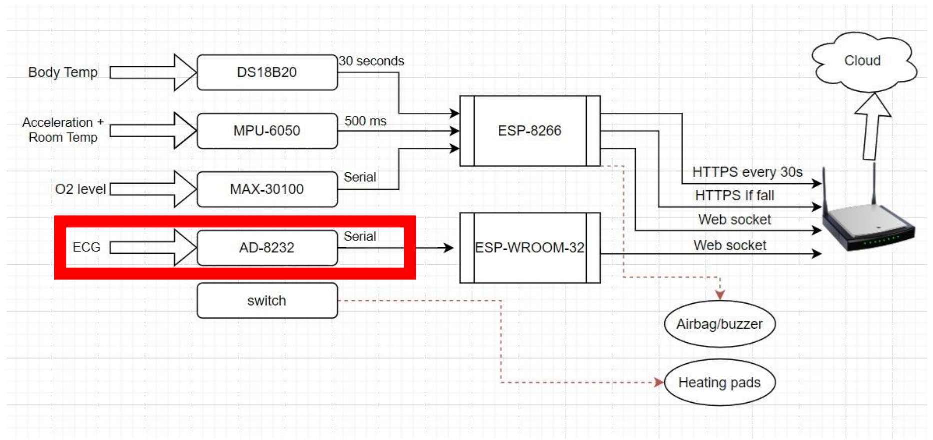


```
pinMode(4, INPUT);
pinMode(2, INPUT);
```

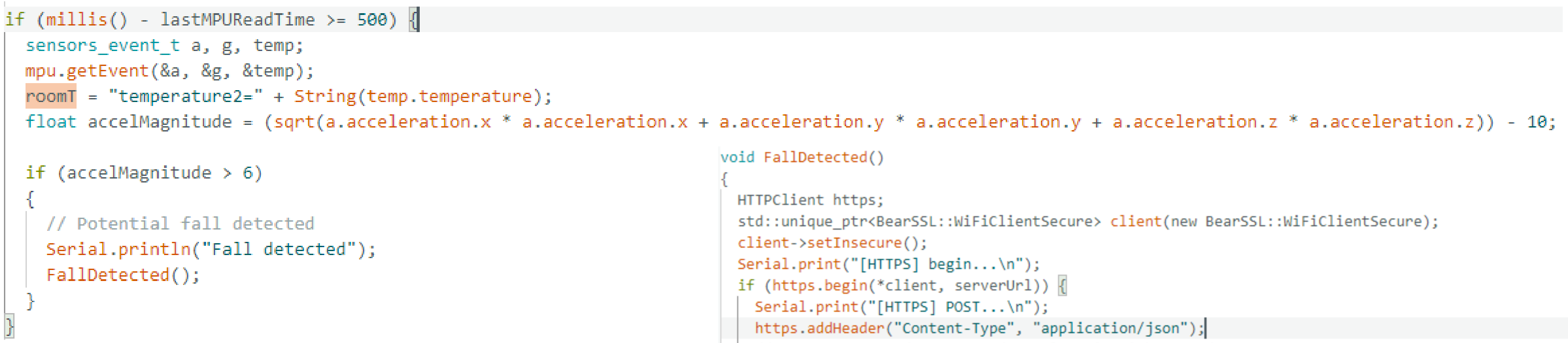
OXYGEN LEVEL MAX-30100



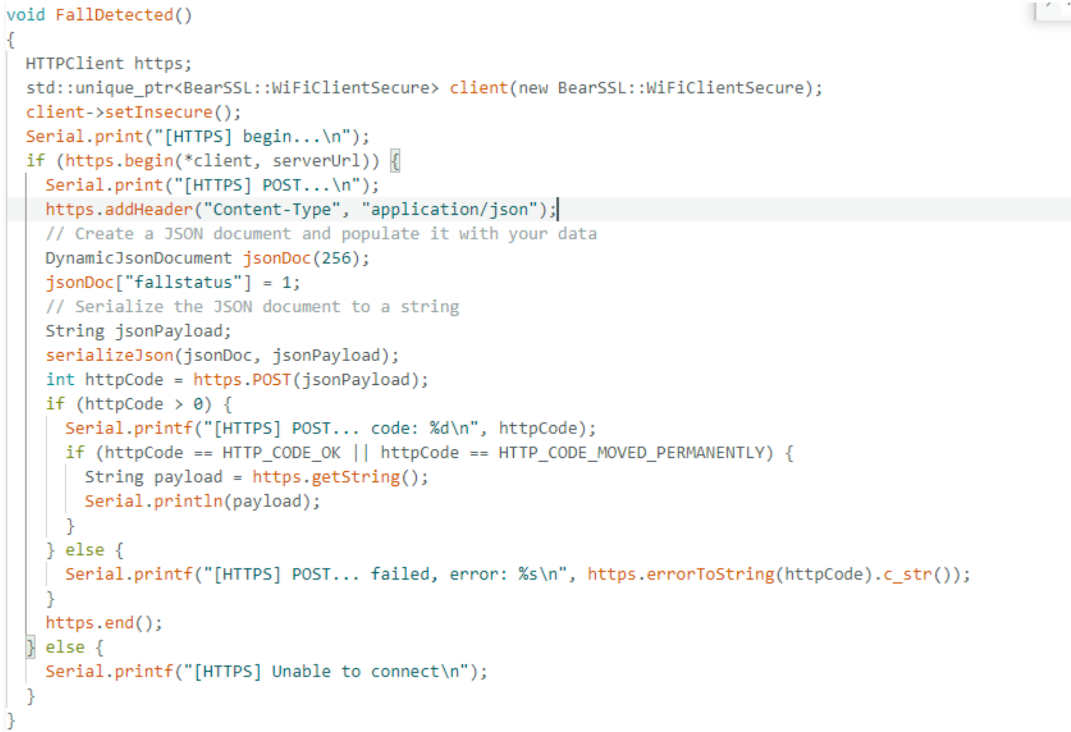
```
pox.update();
if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
    Serial.print("bpm / SpO2:");
    Serial.print(pox.getSpO2());
    Serial.println("%");

    // Prepare the JSON data to send
    String jsonData =  String(String(pox.getSpO2());
    if (client.available()) {
        client.send(jsonData);
          maxim.resetFifo();
    }

    tsLastReport = millis();
}
```



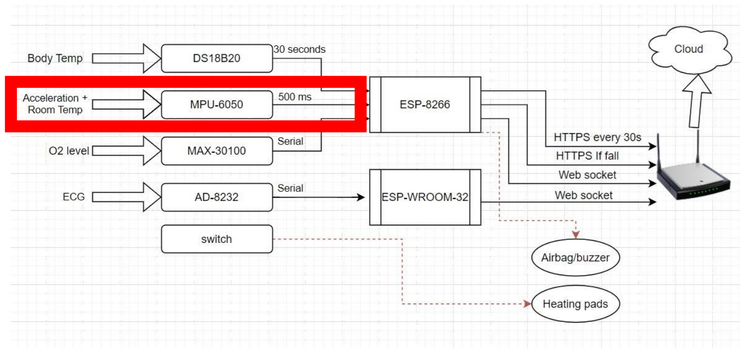
FALL DETECTION MPU-6050



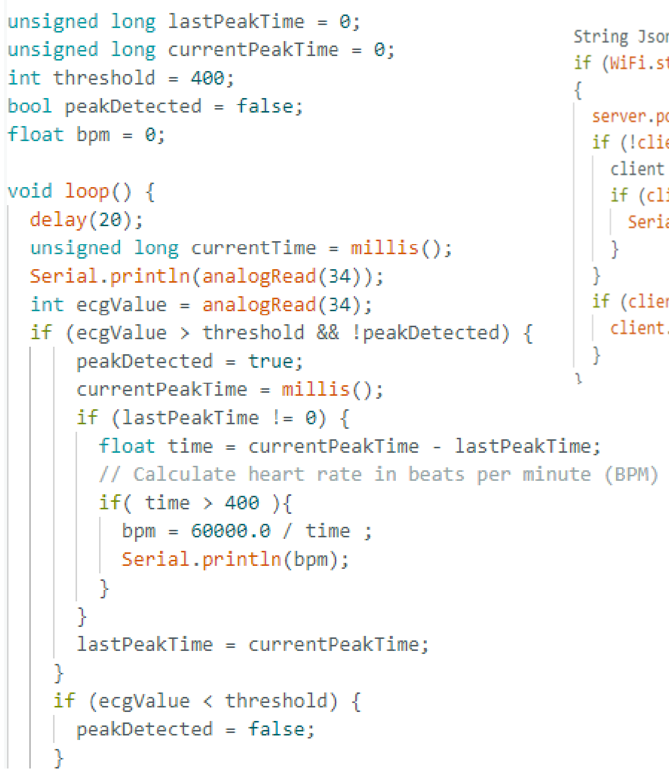
```
if (millis() - lastMPUReadTime >= 500) {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  roomT = "temperature2=" + String(temp.temperature);
  float accelMagnitude = (sqrt(a.acceleration.x * a.acceleration.x + a.acceleration.y * a.acceleration.y + a.acceleration.z * a.acceleration.z)) - 10;

  if (accelMagnitude > 6)
  {
    // Potential fall detected
    Serial.println("Fall detected");
    FallDetected();
  }
}
```



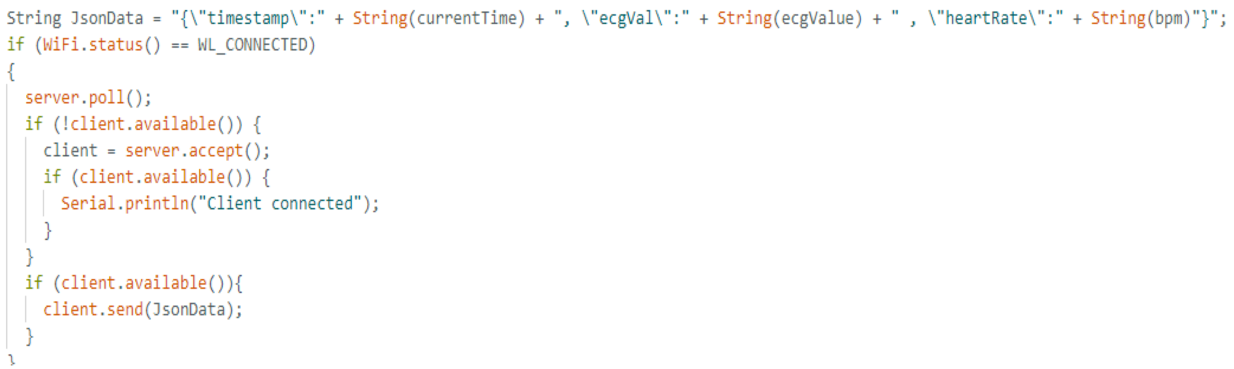
```
void FallDetected()
{
  HTTPClient https;
  std::unique_ptr<BearSSL::WiFiClientSecure> client(new BearSSL::WiFiClientSecure);
  client->setInsecure();
  Serial.print("[HTTPS] begin...\n");
  if (https.begin(*client, serverUrl)) {
    Serial.print("[HTTPS] POST...\n");
    https.addHeader("Content-Type", "application/json");
    // Create a JSON document and populate it with your data
    DynamicJsonDocument jsonDoc(256);
    jsonDoc["fallstatus"] = 1;
    // Serialize the JSON document to a string
    String jsonPayload;
    serializeJson(jsonDoc, jsonPayload);
    int httpCode = https.POST(jsonPayload);
    if (httpCode > 0) {
      Serial.printf("[HTTPS] POST... code: %d\n", httpCode);
      if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
        String payload = https.getString();
        Serial.println(payload);
      }
    } else {
      Serial.printf("[HTTPS] POST... failed, error: %s\n", https.errorToString(httpCode).c_str());
    }
    https.end();
  } else {
    Serial.printf("[HTTPS] Unable to connect\n");
  }
}
```



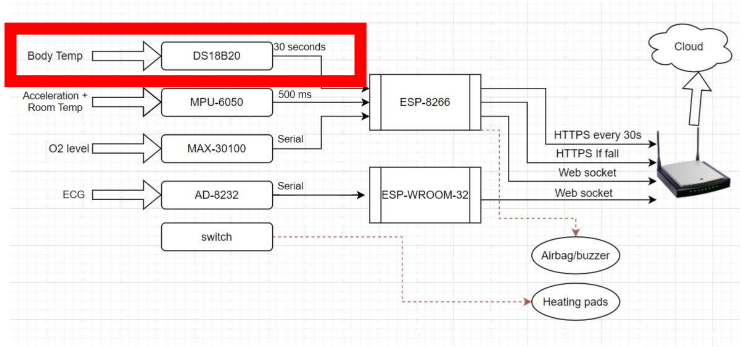
ECG AND HEART RATE AD-8232



```
unsigned long lastPeakTime = 0;
unsigned long currentPeakTime = 0;
int threshold = 400;
bool peakDetected = false;
float bpm = 0;

void loop() {
  delay(20);
  unsigned long currentTime = millis();
  Serial.println(analogRead(34));
  int ecgValue = analogRead(34);
  if (ecgValue > threshold && !peakDetected) {
      peakDetected = true;
      currentPeakTime = millis();
      if (lastPeakTime != 0) {
        float time = currentPeakTime - lastPeakTime;
        // Calculate heart rate in beats per minute (BPM)
        if( time > 400 ){
          bpm = 60000.0 / time ;
          Serial.println(bpm);
        }
      }
      lastPeakTime = currentPeakTime;
    }
    if (ecgValue < threshold) {
      peakDetected = false;
    }
```

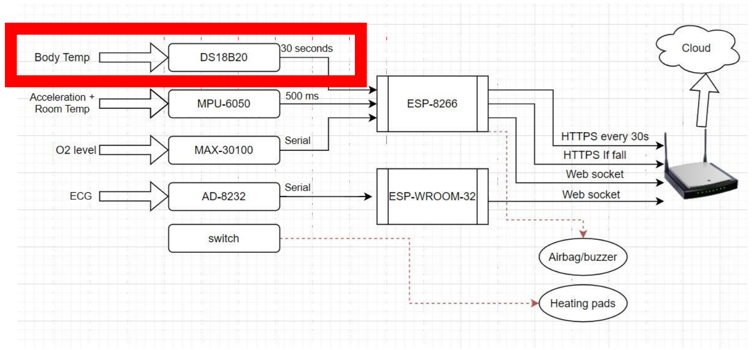


```
String JsonData = "{\"timestamp\":" + String(currentTime) + ", \"ecgVal\":" + String(ecgValue) + " , \"heartRate\":" + String(bpm)"}";
if (WiFi.status() == WL_CONNECTED)
{
  server.poll();
  if (!client.available()) {
    client = server.accept();
    if (client.available()) {
      Serial.println("Client connected");
    }
  }
  if (client.available()){
    client.send(JsonData);
  }
}
```

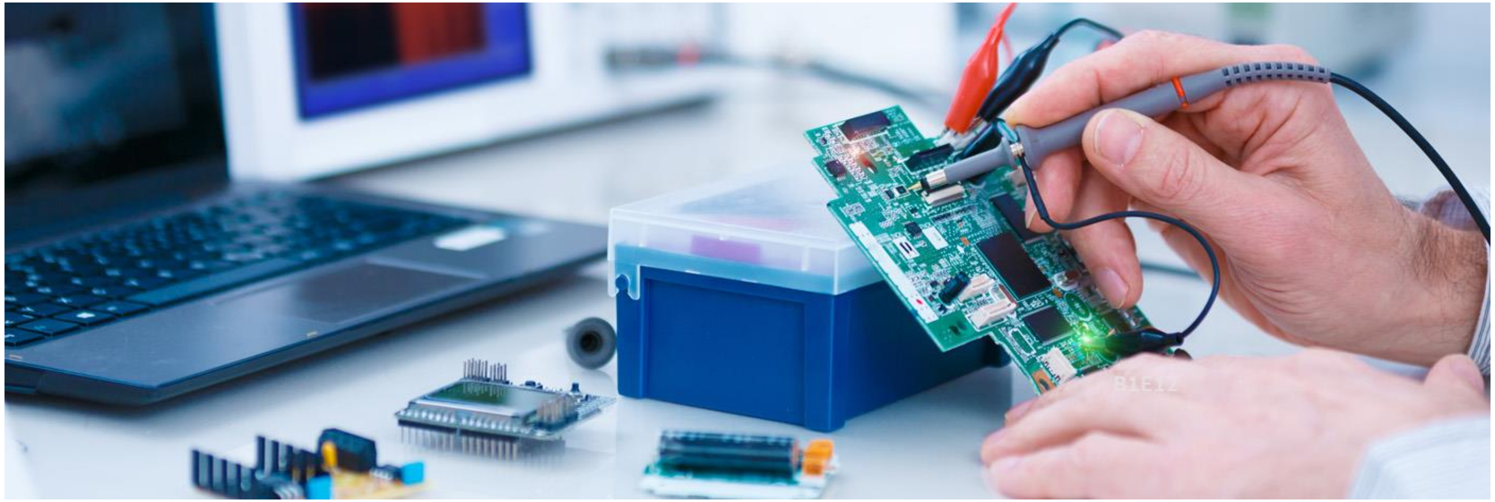


TEMPERATURE SENSOR DS18B20

```
 unsigned long currentMillis = millis();
 if (currentMillis - lastTemperatureUpdateTime >= temperatureUpdateInterval)
 {
   // It's time to update temperature
   sensors.requestTemperatures();
   float temperature = sensors.getTempCByIndex(0);
   sendTemperatureToServer(temperature);
   lastTemperatureUpdateTime = currentMillis; // Reset the timer
 }
void sendTemperatureToServer(float temperature)
{
  String data = "temperature=" + String(temperature) + "&" +roomT;
  std::unique_ptr<BearSSL::WiFiClientSecure> client(new BearSSL::WiFiClientSecure);
  client->setInsecure();
  HTTPClient https;
  Serial.print("[HTTPS] begin...\n");
  if (https.begin(*client, "https://JackBack.onrender.com/api/update")) {
    Serial.print("[HTTPS] POST...\n");
    https.addHeader("Content-Type", "application/x-www-form-urlencoded");
    int httpCode = https.POST(data);
    if (httpCode > 0) {
      Serial.printf("[HTTPS] POST... code: %d\n", httpCode);
      if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
      {
        String payload = https.getString();
        Serial.println(payload);
      }
```



HARDWARE TESTING



FALL DETECTION MPU-6050

Natural Movements

```
-9.33 0.51 0.88  0.03 0.27  0.28 1.36  0.71 1.46  1.01  1.15
0.76  0.78 0.84  0.72 0.91  0.73 0.81  0.75 0.29  0.82  0.54
0.39  0.69 1.16  0.63 0.63  1.21 0.53  1.61 0.52  1.11  0.64
1.02  0.69 -0.21 1.47 1.29  0.33 -0.16 1.26 1.77  0.92  0.52
1.23  0.35 0.70  0.94 1.02  0.43 0.89  1.18 0.70  1.46  0.68
1.21  1.30 1.12  1.48 -0.17 1.38 -0.02 2.17 -0.11 1.31  0.47
0.58  0.06 0.34  2.36 1.79  0.69 1.60  1.17 1.11  -0.86 0.48
-0.45 1.22 0.93  0.25
```

FALL DETECTION MPU-6050

Fall Detection with threshold [-2,+2]

```
0.80 0.86  2.04 Fall detected
0.96 0.05  1.27 1.33  2.11 Fall detected
0.72 0.13  23.85 Fall detected
2.39 Fall  detected
0.33 0.45  0.71 2.10 Fall detected
0.62 0.72  0.42 0.83  0.56 1.14 .144 0.53 1.76 1.93
0.47 2.46  Fall detected
0.76 1.47  0.88 0.80  -2.50 Fall detected
0.99 1.28  1.65 -5.27 Fall detected
0.69 -0.62 1.60 -0.13 -2.18 Fall detected
0.10 0.16  0.41 0.42  -3.55 Fall detected
1.11 1.10  1.11 1.01  1.08 1.01 1.15 1.09 0.99 1.13
1.07 9.84  Fall detected
0.78 0.96  1.23
```

FALL DETECTION MPU-6050

```
0.39  0.43
0.37  0.41  0.34  0.40  0.41  0.40 0.42 0.43 0.39 0.41 -0.27
-0.47 0.43  0.90  -0.59 30.74 Fall detected
0.11  -0.40 -4.33 Fall detected
-2.07 -1.40 16.54 Fall detected
-0.48 4.90  Fall detected
-0.48 0.04  -1.62 -0.62 -0.50 4.45 Fall detected
-3.56 -1.68 -1.17 0.40  0.29
```

Fall Detection with threshold [-4,+4]

Fall Detection with threshold [-6,+6]

```
              1.05 0.88 1.21 1.30 0.81 0.07 11.73 Fall detected
             0.26 1.93 2.26 -0.74 0.84 1.07 -6.54 Fall detected
                   1.18 0.79 0.53 0.91 0.50 10.86 Fall detected
        1.30 2.13 1.87 1.14 1.76 2.60 1.52 2.56 2.07 -0.22 1.92
         3.99 0.96 2.31 2.71 2.81 3.85 1.12 0.85 2.77 1.30 1.36
                         3.77 0.70 2.08 1.15 7.59 Fall detected
1.17 0.98 1.06 1.70 2.16 0.59 0.61 0.44 9.90 Fall detected 0.06
                           4.93 -1.20 -0.64 19.29 Fall detected
                                            -6.84 Fall detected
                                                           0.64
```

AIRBAG

1 **Fall Detection Delay**
500 milliseconds

2 **Airbag Delay**
50 milliseconds

3 **Total Delay**
550 milliseconds

4 **Equation**

$$time = \sqrt{\frac{2h}{g}} = 580\ milliseconds$$

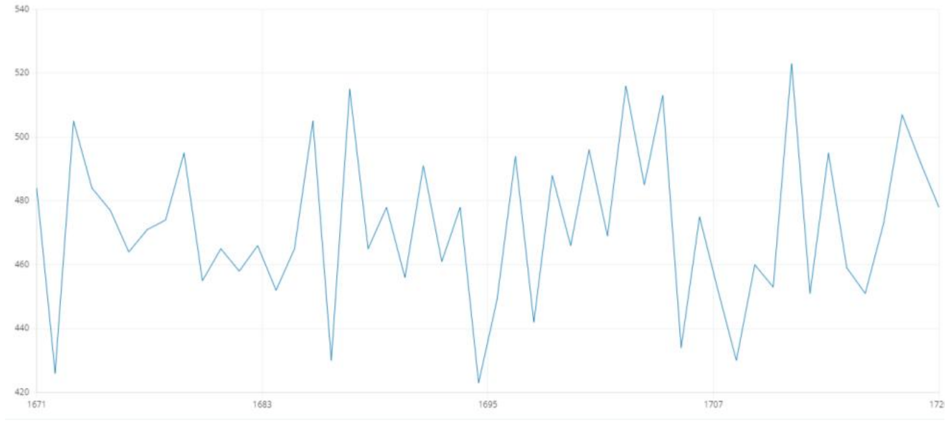
HEATING PADS

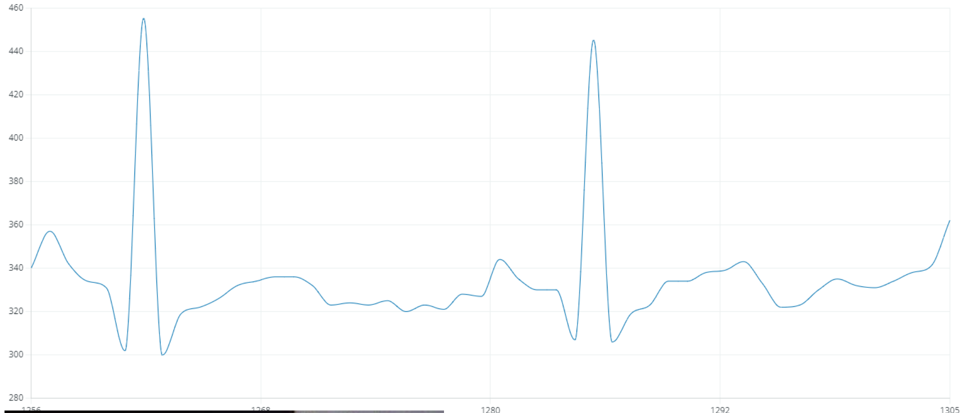
1 **Standard temperature**
According to Cardinal Health, the appropriate and safe temperature range is 40-45°C.

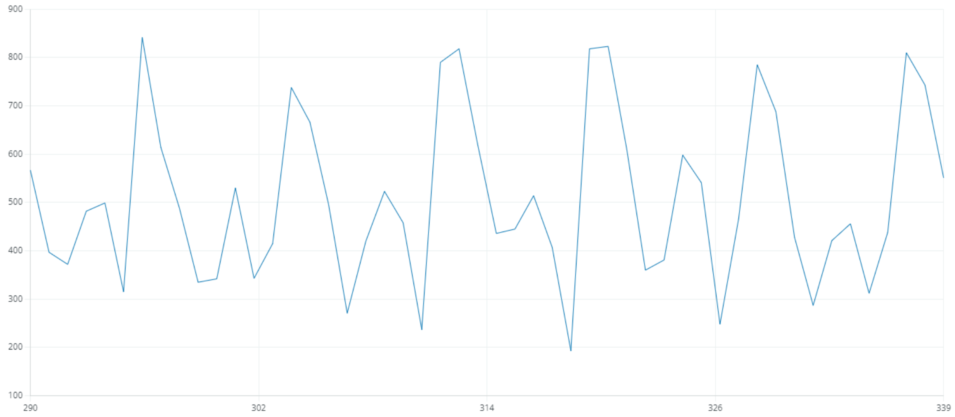
2 **At 9 volts**
Temperature is 35.63°C

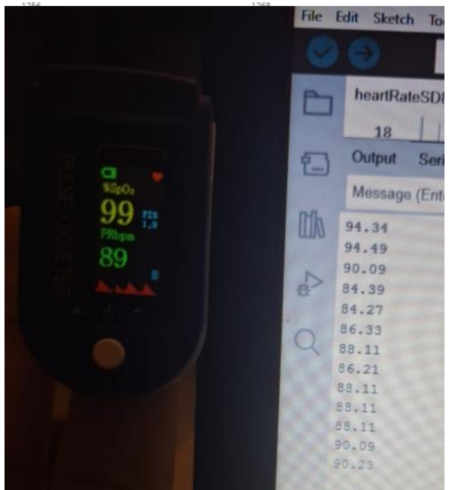
3 **At 12 volts**
Temperature is 43.94°C

ECG AND HEART RATE AD-8232





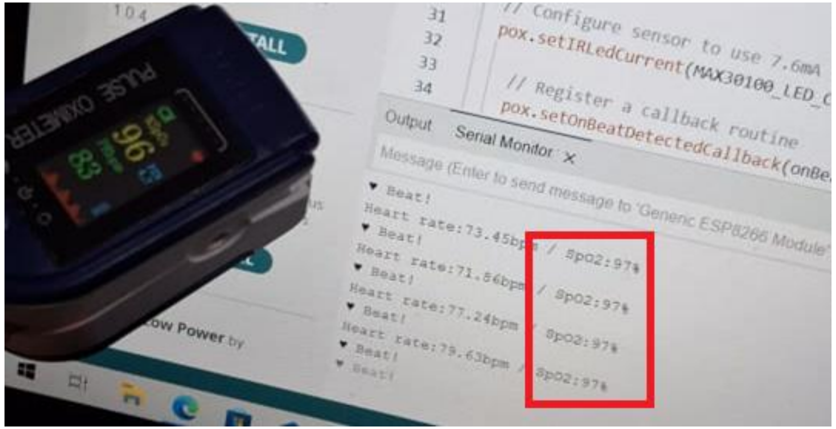




```
94.34
94.49
90.09
84.39
84.27
86.33
88.11
86.21
88.11
88.11
88.11
90.09
90.23
```

1% and up to 8% error

OXYGEN LEVEL MAX-30100



```
// Configure sensor to use 7.6mA
pox.setIRLedCurrent(MAX30100_LED_C

// Register a callback routine
pox.setOnBeatDetectedCallback(onBe
```

```
Message (Enter to send message to 'Generic ESP8266 Module'
♥ Beat!
Heart rate:73.45bpm / SpO2:97%
♥ Beat!
Heart rate:71.56bpm / SpO2:97%
♥ Beat!
Heart rate:77.24bpm / SpO2:97%
♥ Beat!
Heart rate:79.63bpm / SpO2:97%
♥ Beat!
♥ Beat!
```

1% error

ROOM TEMPERATURE USING MPU-6050

1

22.4°C

Room temperature is 22.4°C

2

22.18 °C

On MPU-6050 is 22.18 °C

3

1.2%

1.2% error calculated

BODY TEMPERATURE USING DS18B20

1. **22.4°C**

   Body temperature is 22.4°C

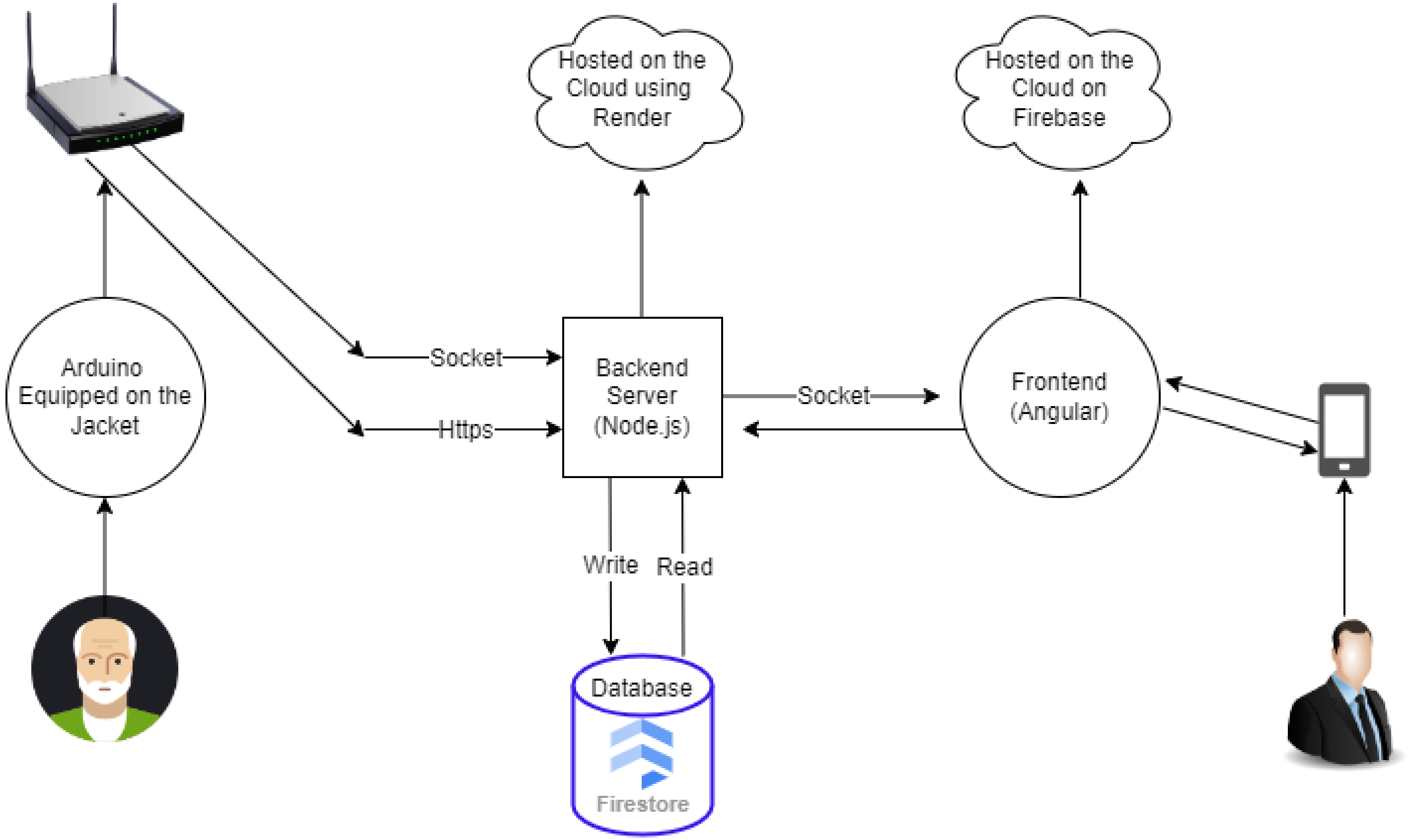
2. **22.31 °C**

   On DS18B20 is 22.31 °C

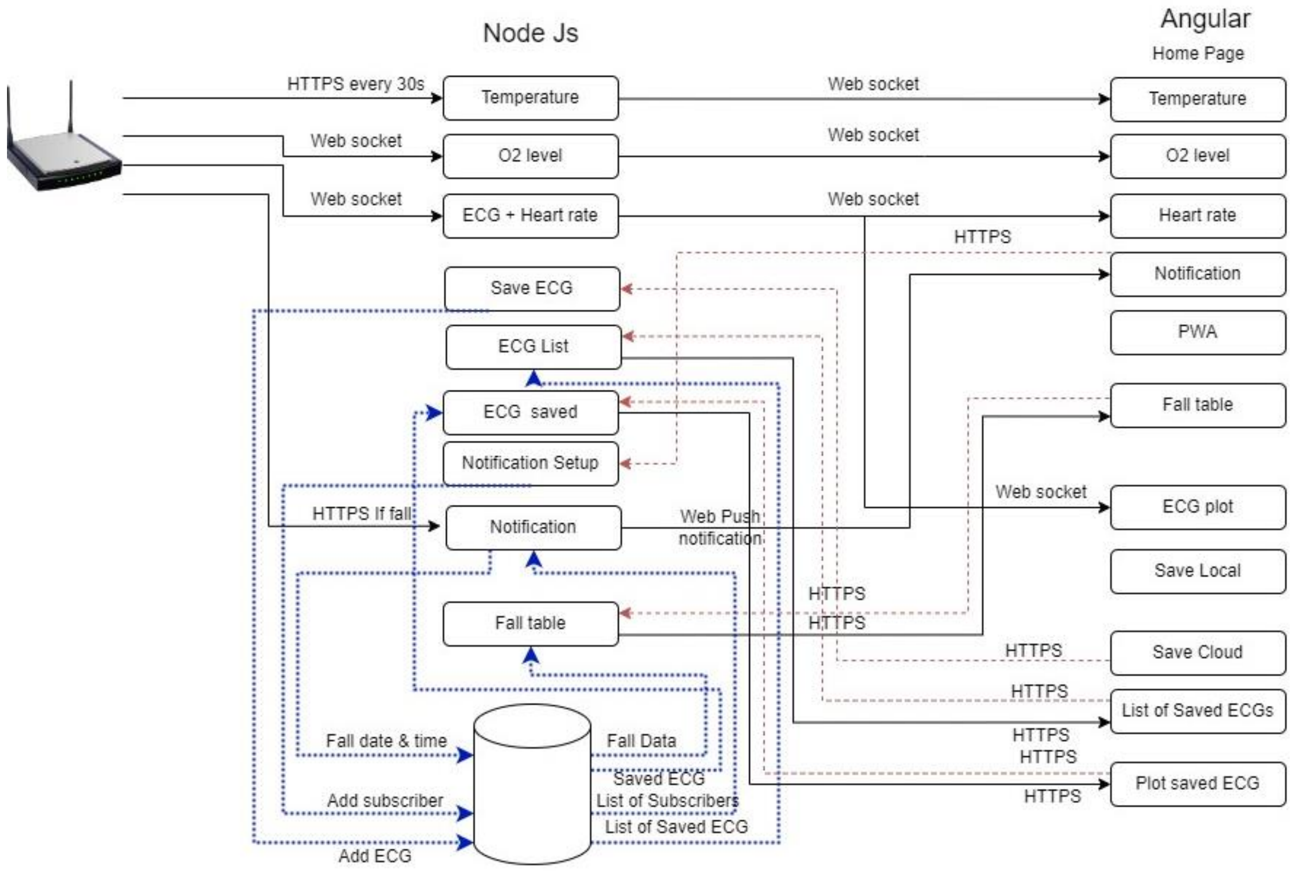
3. **0.66%**

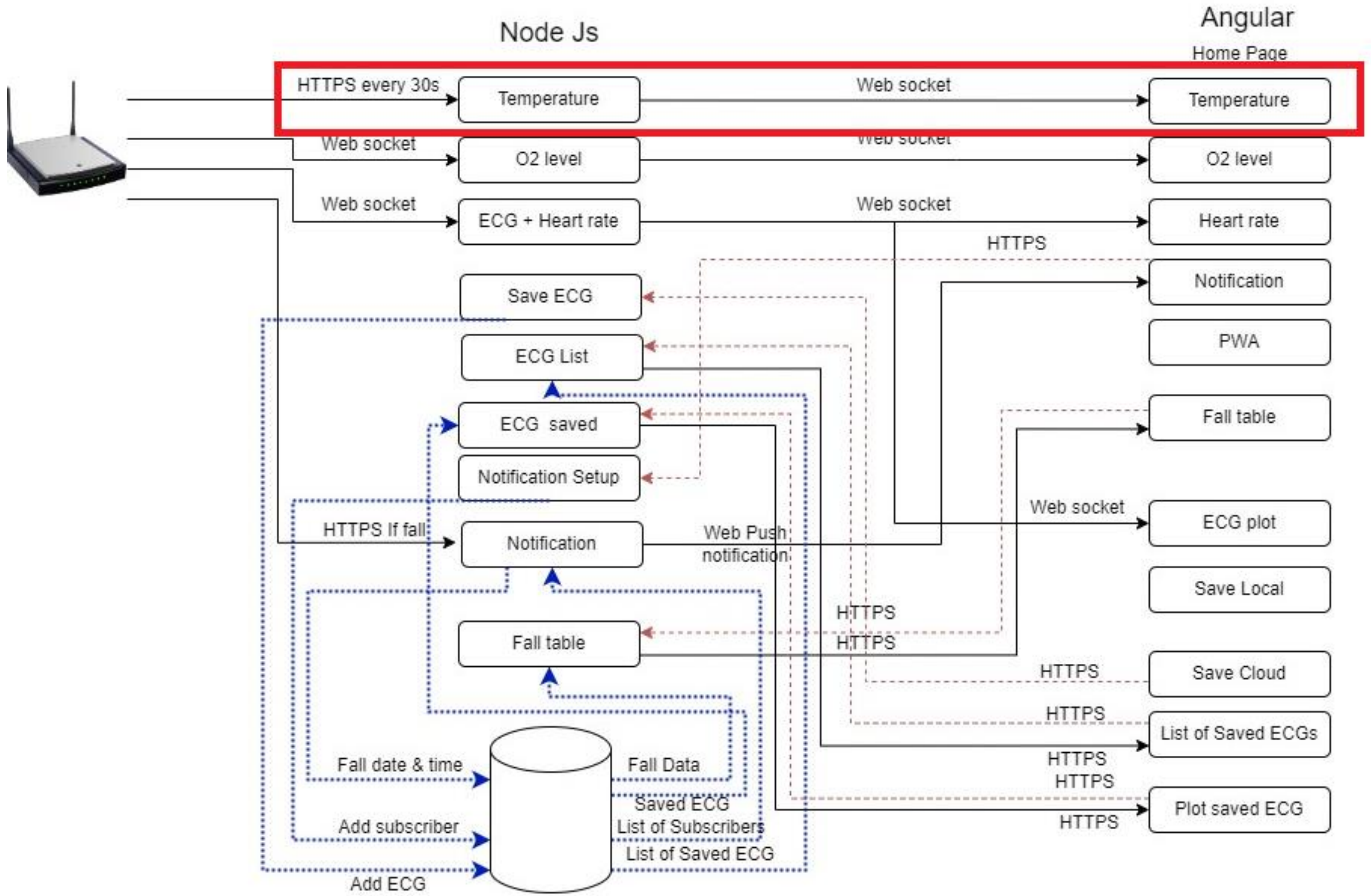
   0.66% error calculated

APPLICATION CHART

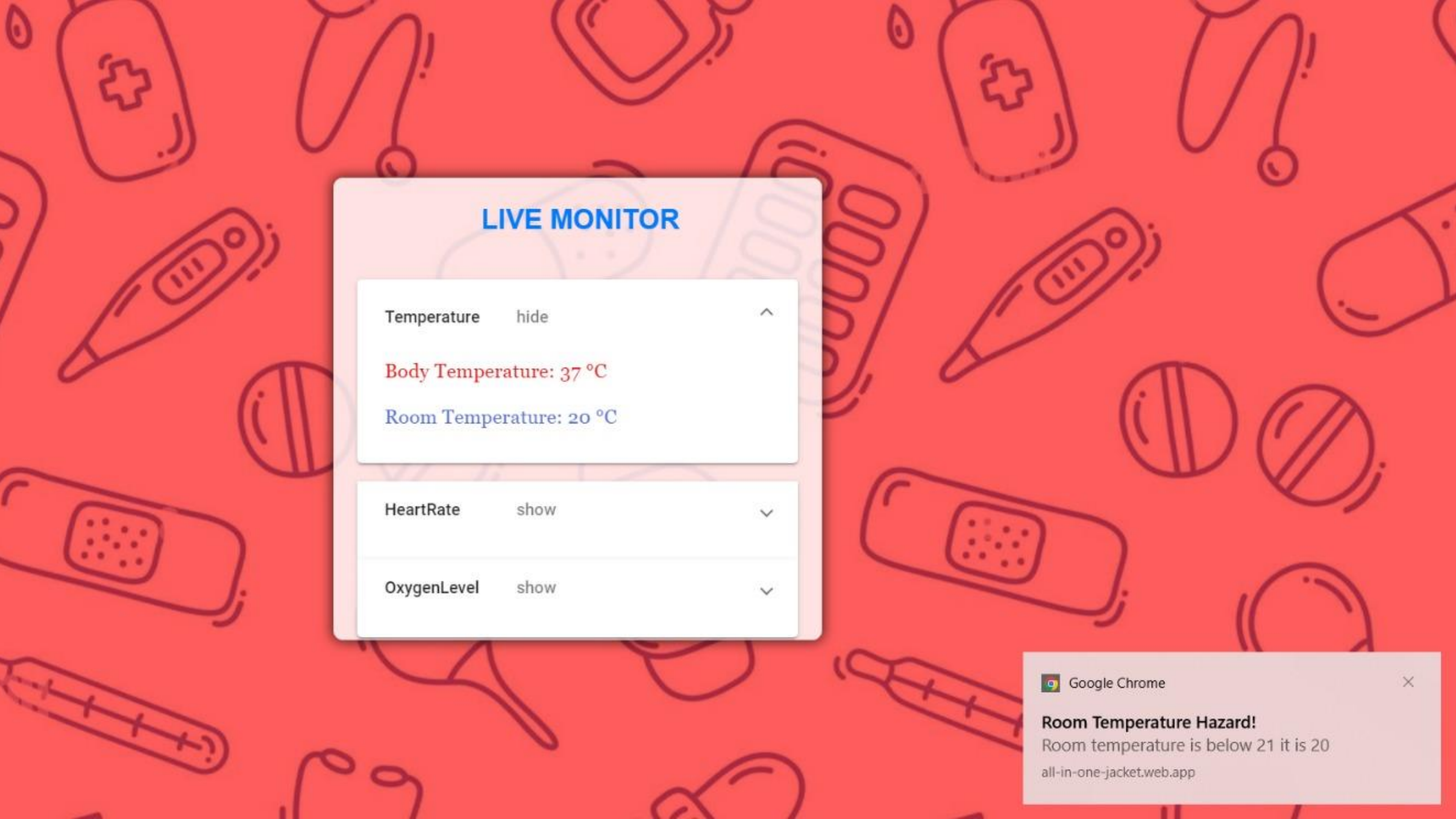

Hosted on the Cloud using Render
Hosted on the Cloud on Firebase
Socket
Https
Arduino Equipped on the Jacket
Backend Server (Node.js)
Socket
Frontend (Angular)
Write
Read
Database
Firestore

IN DEPTH APPLICATION CHART





Node Js
Angular
Home Page
HTTPS every 30s
Temperature
Web socket
Temperature
Web socket
O2 level
Web socket
O2 level
Web socket
ECG + Heart rate
Web socket
Heart rate
HTTPS
Notification
Save ECG
PWA
ECG List
Fall table
ECG saved
Notification Setup
Web socket
ECG plot
HTTPS If fall
Notification
Web Push notification
Save Local
HTTPS
HTTPS
Fall table
HTTPS
Save Cloud
HTTPS
List of Saved ECGs
HTTPS
HTTPS
Plot saved ECG
HTTPS
Fall date & time
Fall Data
Saved ECG
Add subscriber
List of Subscribers
List of Saved ECG
Add ECG



LIVE MONITOR

Temperature hide

Body Temperature: 37 °C

Room Temperature: 20 °C

HeartRate show

OxygenLevel show

Google Chrome

Room Temperature Hazard!

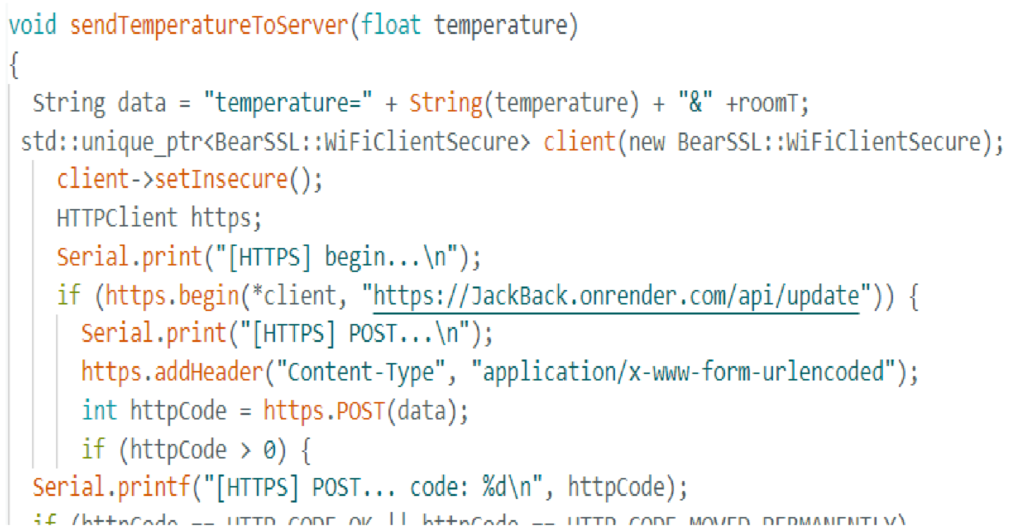
Room temperature is below 21 it is 20

all-in-one-jacket.web.app

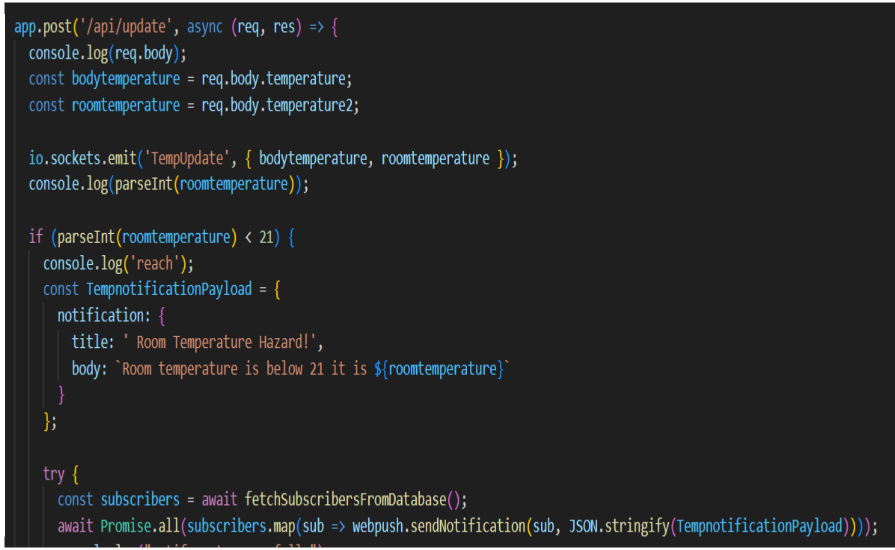
TEMPERATURE

Arduino Side

Node.js Side



```
void sendTemperatureToServer(float temperature)
{
  String data = "temperature=" + String(temperature) + "&" +roomT;
 std::unique_ptr<BearSSL::WiFiClientSecure> client(new BearSSL::WiFiClientSecure);
    client->setInsecure();
    HTTPClient https;
    Serial.print("[HTTPS] begin...\n");
    if (https.begin(*client, "https://JackBack.onrender.com/api/update")) {
      Serial.print("[HTTPS] POST...\n");
      https.addHeader("Content-Type", "application/x-www-form-urlencoded");
      int httpCode = https.POST(data);
      if (httpCode > 0) {
  Serial.printf("[HTTPS] POST... code: %d\n", httpCode);
  if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
```

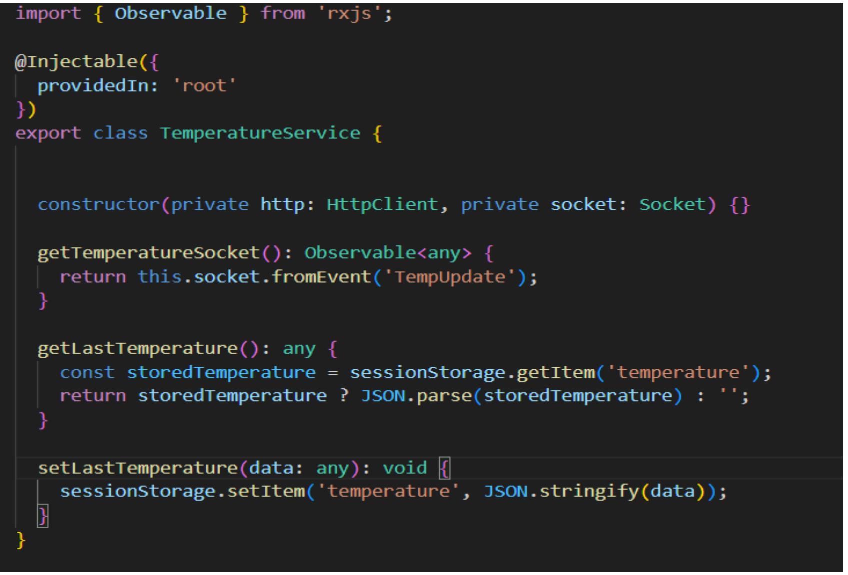


```
app.post('/api/update', async (req, res) => {
  console.log(req.body);
  const bodytemperature = req.body.temperature;
  const roomtemperature = req.body.temperature2;

  io.sockets.emit('TempUpdate', { bodytemperature, roomtemperature });
  console.log(parseInt(roomtemperature));

  if (parseInt(roomtemperature) < 21) {
    console.log('reach');
    const TempnotificationPayload = {
      notification: {
        title: ' Room Temperature Hazard!',
        body: `Room temperature is below 21 it is ${roomtemperature}`
      }
    };

    try {
      const subscribers = await fetchSubscribersFromDatabase();
      await Promise.all(subscribers.map(sub => webpush.sendNotification(sub, JSON.stringify(TempnotificationPayload))));
```

TEMPERATURE

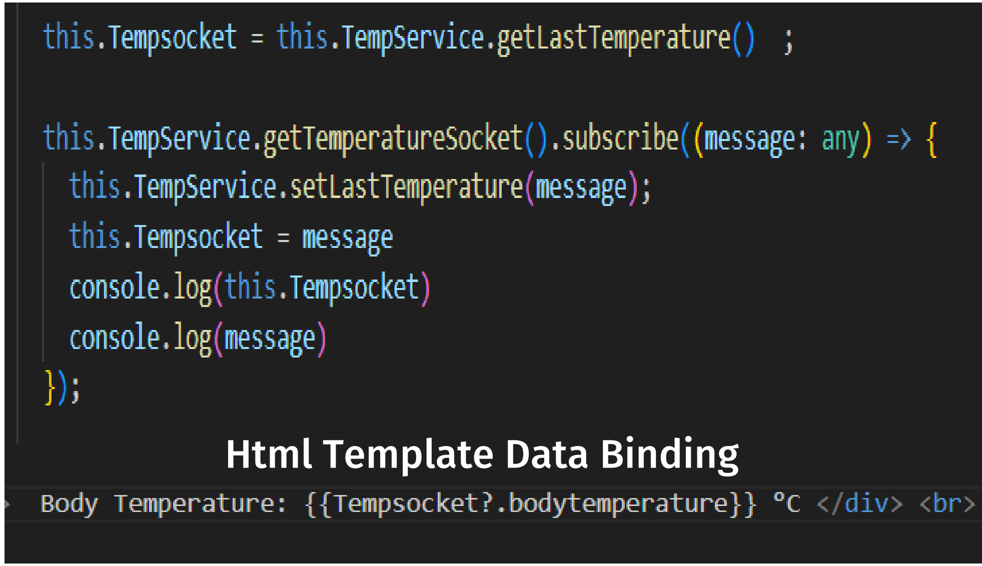
Subscribe

Temperature Service



```
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class TemperatureService {


  constructor(private http: HttpClient, private socket: Socket) {}

  getTemperatureSocket(): Observable<any> {
    return this.socket.fromEvent('TempUpdate');
  }

  getLastTemperature(): any {
    const storedTemperature = sessionStorage.getItem('temperature');
    return storedTemperature ? JSON.parse(storedTemperature) : '';
  }

  setLastTemperature(data: any): void {
    sessionStorage.setItem('temperature', JSON.stringify(data));
  }
}
```

Home-Page Component

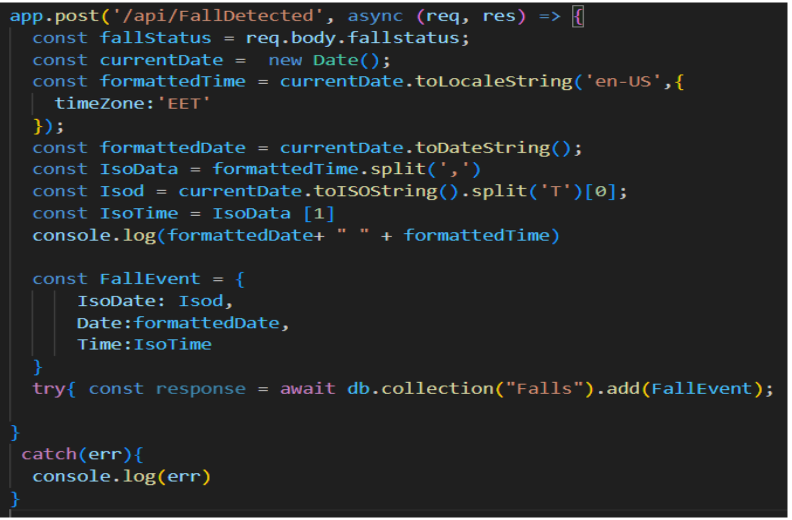


```
this.Tempsocket = this.TempService.getLastTemperature()  ;

this.TempService.getTemperatureSocket().subscribe((message: any) => {
  this.TempService.setLastTemperature(message);
  this.Tempsocket = message
  console.log(this.Tempsocket)
  console.log(message)
});
```

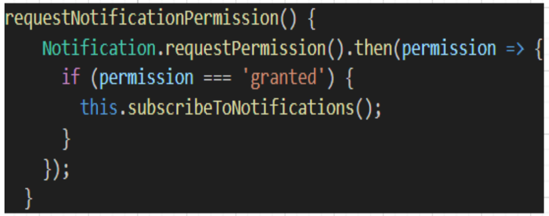
Html Template Data Binding

```
Body Temperature: {{Tempsocket?.bodytemperature}} °C </div> <br>
```

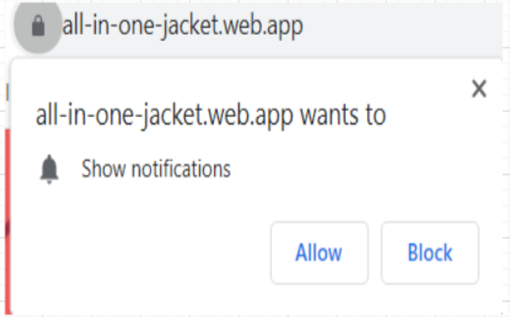
FALL DATA + NOTIFICATION



```
app.post('/api/FallDetected', async (req, res) => {
  const fallStatus = req.body.fallstatus;
  const currentDate =  new Date();
  const formattedTime = currentDate.toLocaleString('en-US',{
    timeZone:'EET'
  });
  const formattedDate = currentDate.toDateString();
  const IsoData = formattedTime.split(',')
  const Isod = currentDate.toISOString().split('T')[0];
  const IsoTime = IsoData [1]
  console.log(formattedDate+ " " + formattedTime)

  const FallEvent = {
      IsoDate: Isod,
      Date:formattedDate,
      Time:IsoTime
  }
  try{ const response = await db.collection("Falls").add(FallEvent);

}
 catch(err){
  console.log(err)
}
```



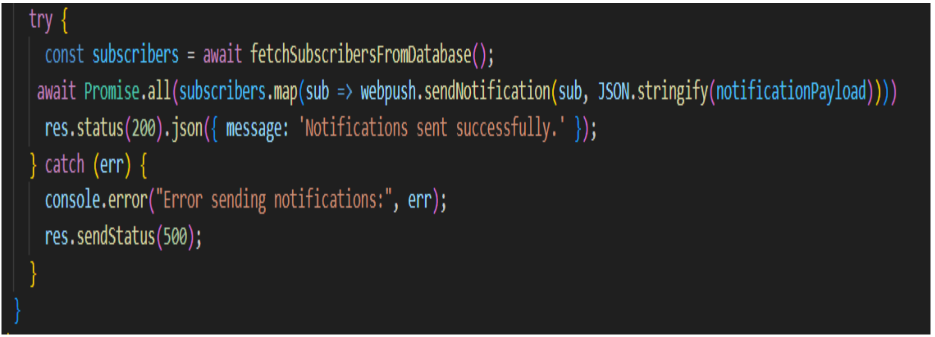
```
requestNotificationPermission() {
    Notification.requestPermission().then(permission => {
      if (permission === 'granted') {
        this.subscribeToNotifications();
      }
    });
  }
```



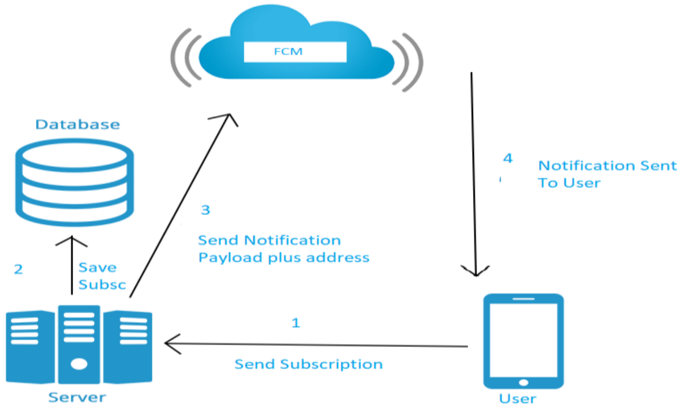
all-in-one-jacket.web.app

all-in-one-jacket.web.app wants to

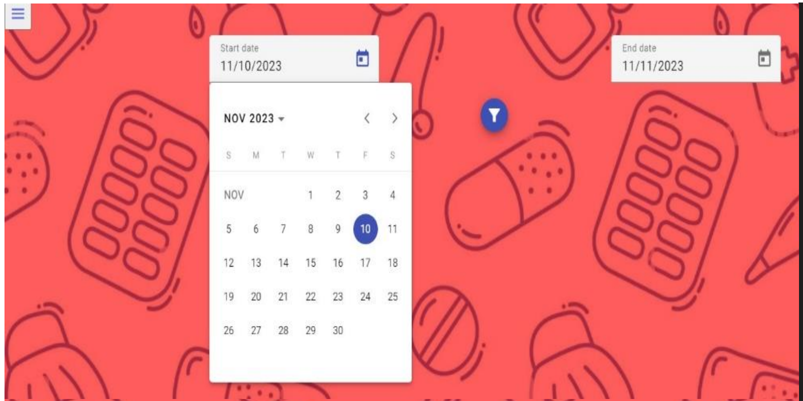
Show notifications



```
  try {
    const subscribers = await fetchSubscribersFromDatabase();
   await Promise.all(subscribers.map(sub => webpush.sendNotification(sub, JSON.stringify(notificationPayload))))
    res.status(200).json({ message: 'Notifications sent successfully.' });
  } catch (err) {
    console.error("Error sending notifications:", err);
    res.sendStatus(500);
  }
}
```

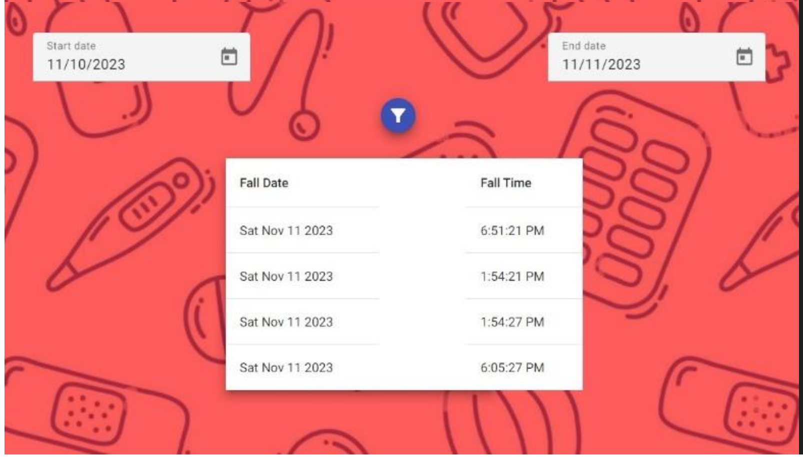


FALL DATA QUERY



Start date
11/10/2023

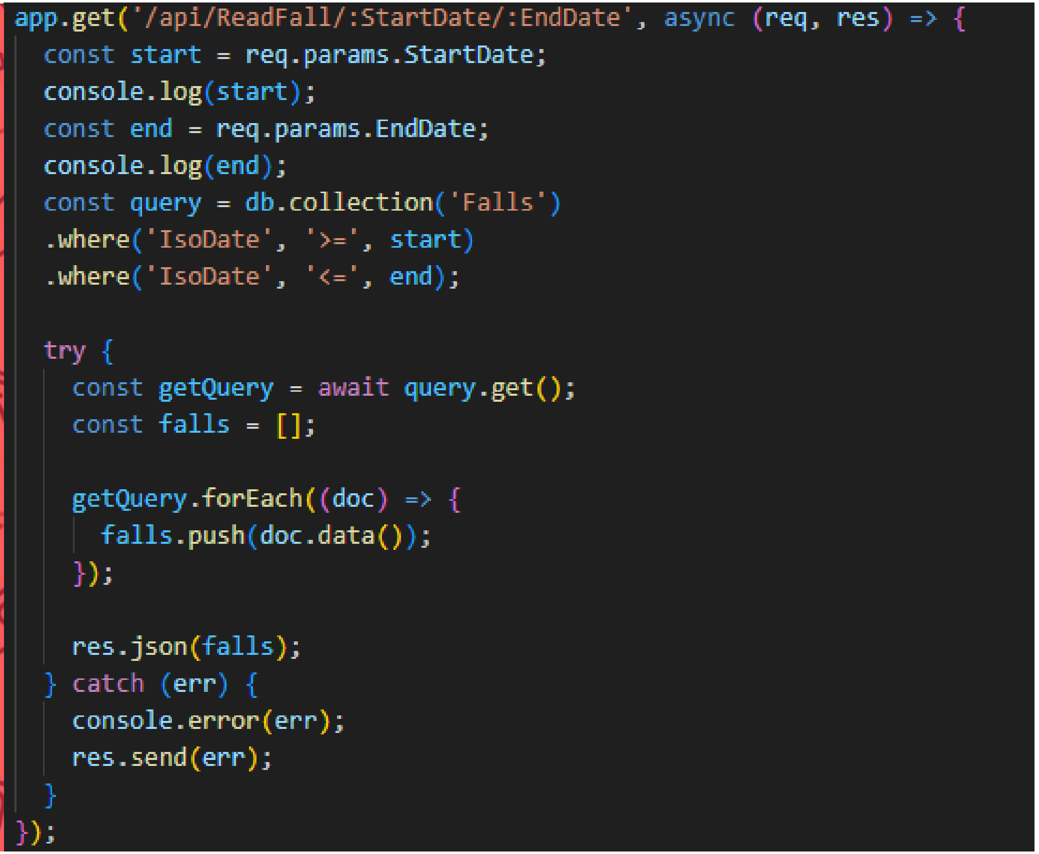
End date
11/11/2023



Start date
11/10/2023

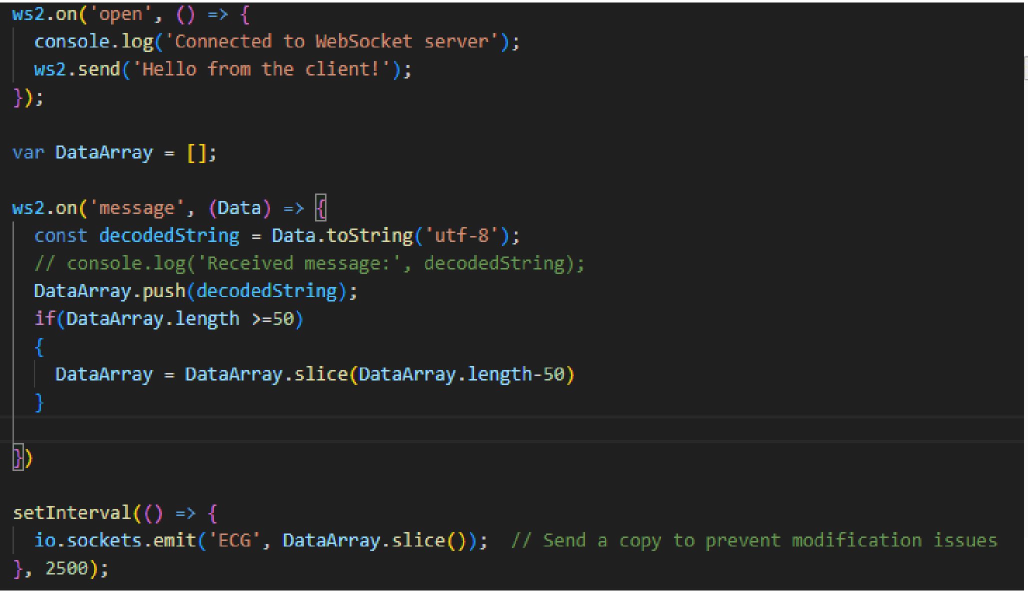
End date
11/11/2023

Fall Date	Fall Time
Sat Nov 11 2023	6:51:21 PM
Sat Nov 11 2023	1:54:21 PM
Sat Nov 11 2023	1:54:27 PM
Sat Nov 11 2023	6:05:27 PM

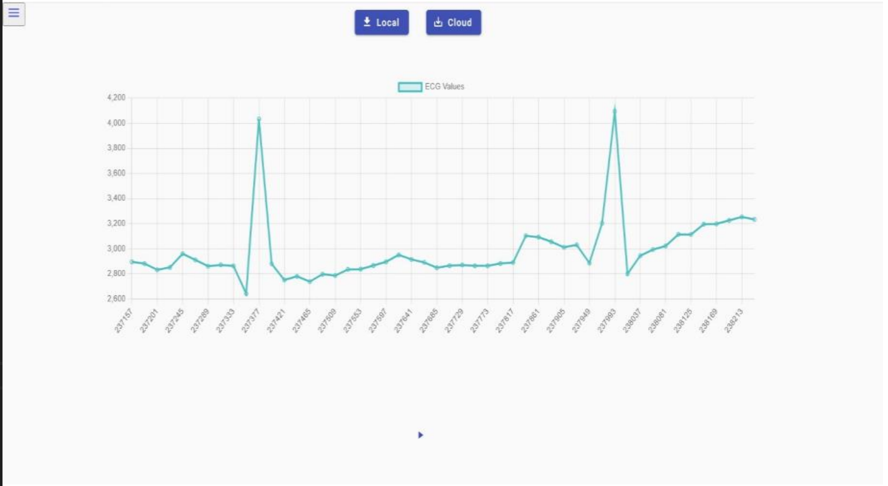


```
app.get('/api/ReadFall/:StartDate/:EndDate', async (req, res) => {
  const start = req.params.StartDate;
  console.log(start);
  const end = req.params.EndDate;
  console.log(end);
  const query = db.collection('Falls')
  .where('IsoDate', '>=', start)
  .where('IsoDate', '<=', end);

  try {
    const getQuery = await query.get();
    const falls = [];

    getQuery.forEach((doc) => {
      falls.push(doc.data());
    });

    res.json(falls);
  } catch (err) {
    console.error(err);
    res.send(err);
  }
});
```

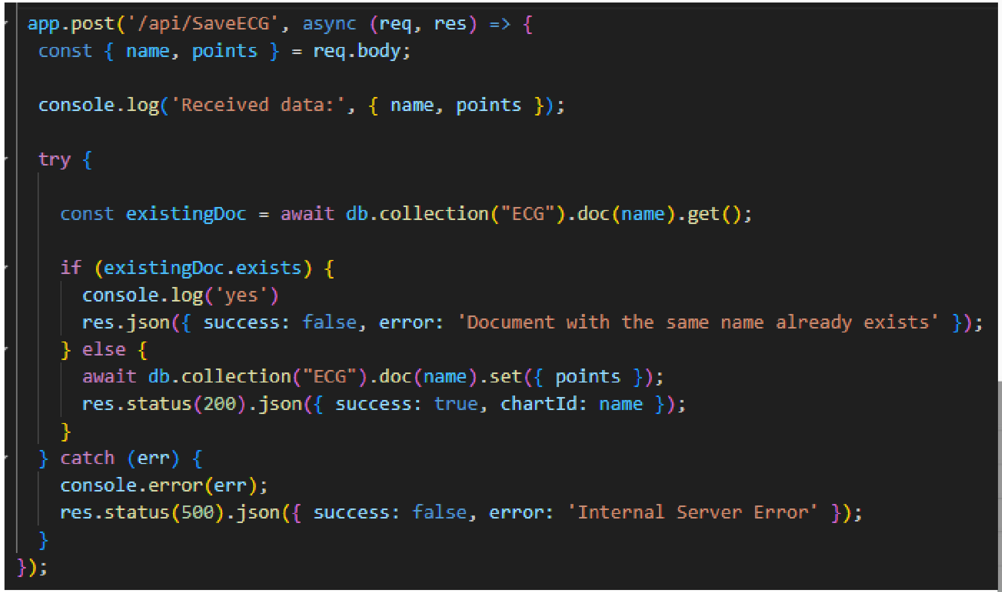
ECG PLOTTING



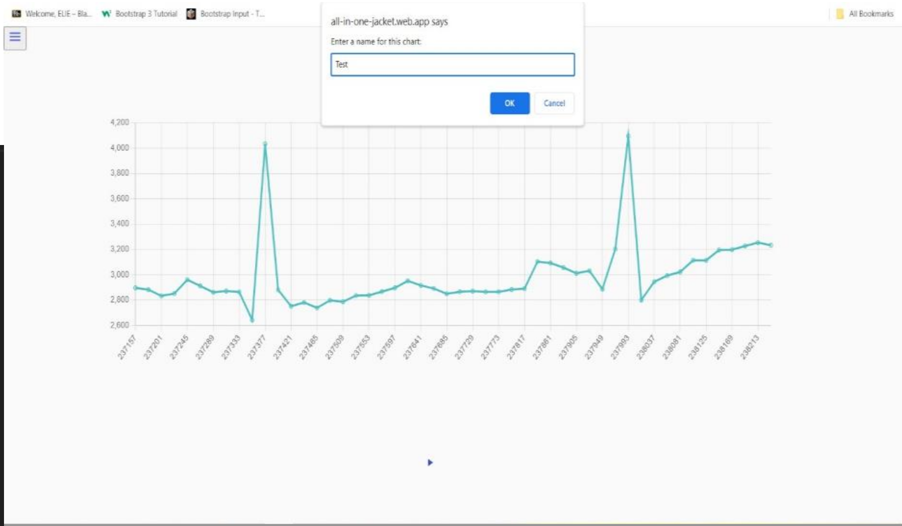
```
ws2.on('open', () => {
  console.log('Connected to WebSocket server');
  ws2.send('Hello from the client!');
});

var DataArray = [];

ws2.on('message', (Data) => {
  const decodedString = Data.toString('utf-8');
  // console.log('Received message:', decodedString);
  DataArray.push(decodedString);
  if(DataArray.length >=50)
  {
    DataArray = DataArray.slice(DataArray.length-50)
  }

})

setInterval(() => {
  io.sockets.emit('ECG', DataArray.slice());  // Send a copy to prevent modification issues
}, 2500);
```



ECG SAVING



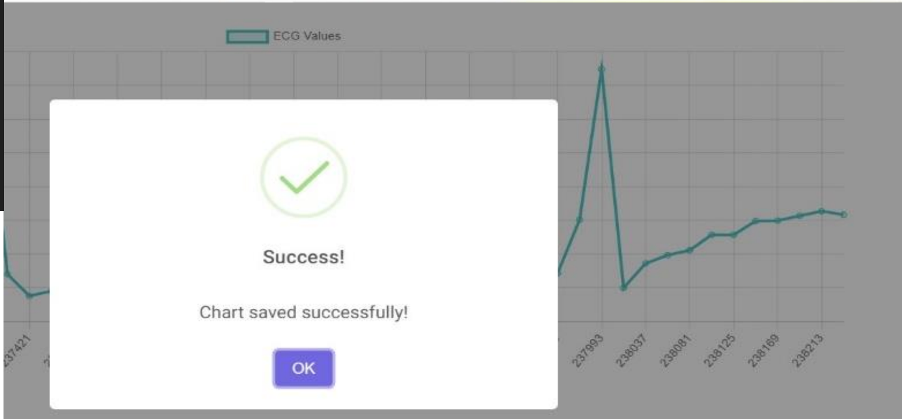
```
app.post('/api/SaveECG', async (req, res) => {
 const { name, points } = req.body;

 console.log('Received data:', { name, points });

 try {

   const existingDoc = await db.collection("ECG").doc(name).get();

   if (existingDoc.exists) {
     console.log('yes')
     res.json({ success: false, error: 'Document with the same name already exists' });
   } else {
     await db.collection("ECG").doc(name).set({ points });
     res.status(200).json({ success: true, chartId: name });
   }
 } catch (err) {
   console.error(err);
   res.status(500).json({ success: false, error: 'Internal Server Error' });
 }
});
```



all-in-one-jacket.web.app says

Enter a name for this chart:

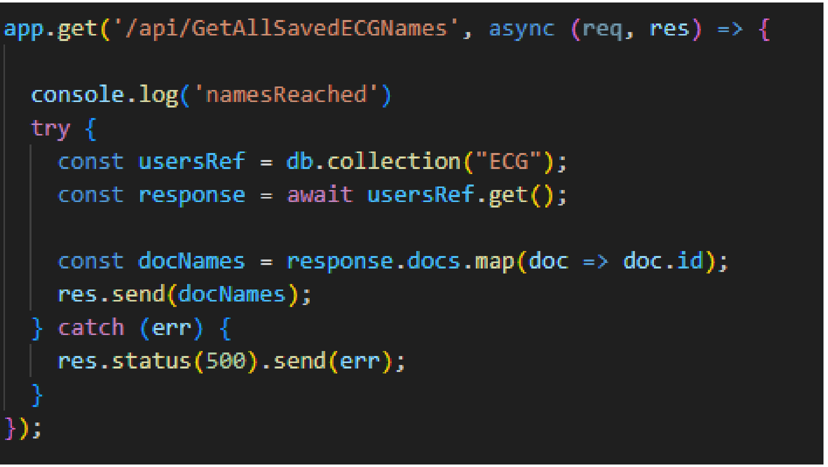
Test



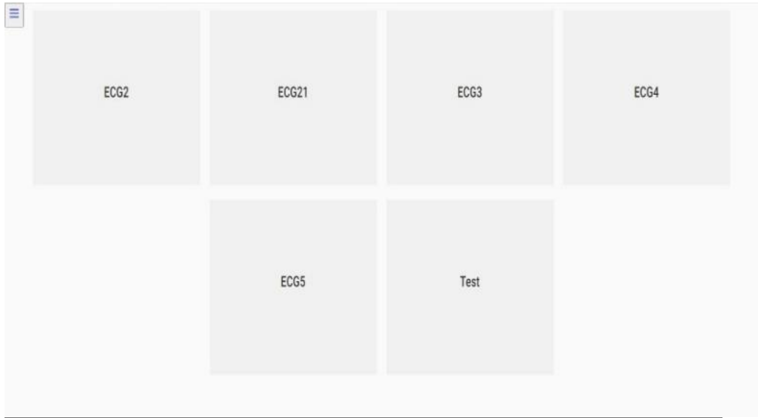
Success!

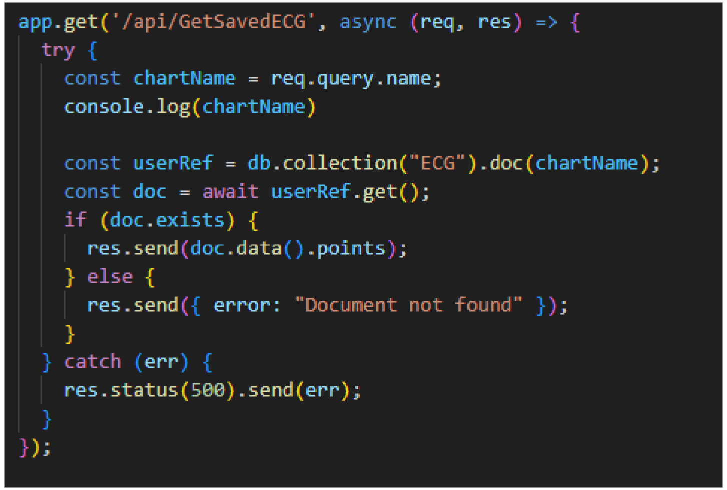
Chart saved successfully!

ECG PLOTTING & SAVING

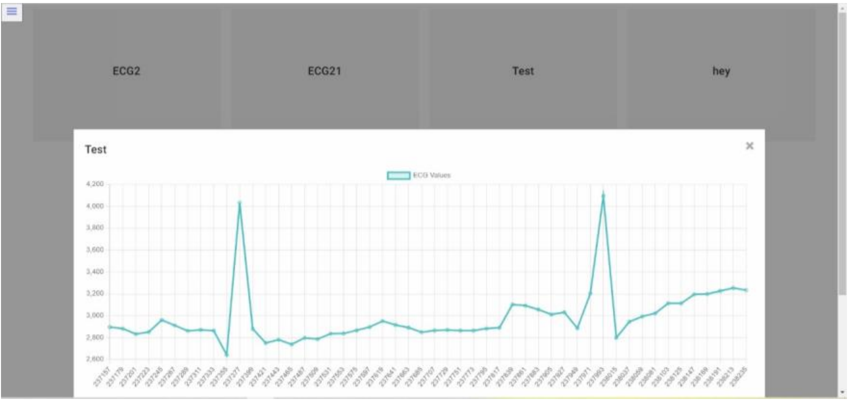


```
app.get('/api/GetAllSavedECGNames', async (req, res) => {

  console.log('namesReached')
  try {
    const usersRef = db.collection("ECG");
    const response = await usersRef.get();

    const docNames = response.docs.map(doc => doc.id);
    res.send(docNames);
  } catch (err) {
    res.status(500).send(err);
  }
});
```

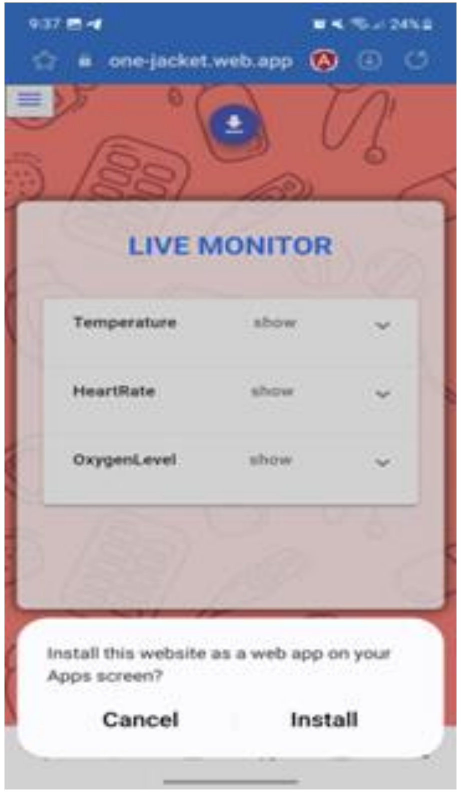




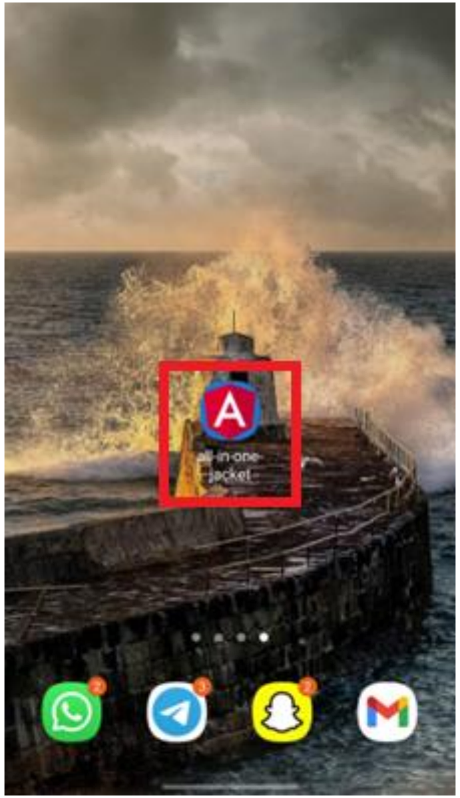
```
app.get('/api/GetSavedECG', async (req, res) => {
  try {
    const chartName = req.query.name;
    console.log(chartName)

    const userRef = db.collection("ECG").doc(chartName);
    const doc = await userRef.get();
    if (doc.exists) {
      res.send(doc.data().points);
    } else {
      res.send({ error: "Document not found" });
    }
  } catch (err) {
    res.status(500).send(err);
  }
});
```



ANGULAR PWA

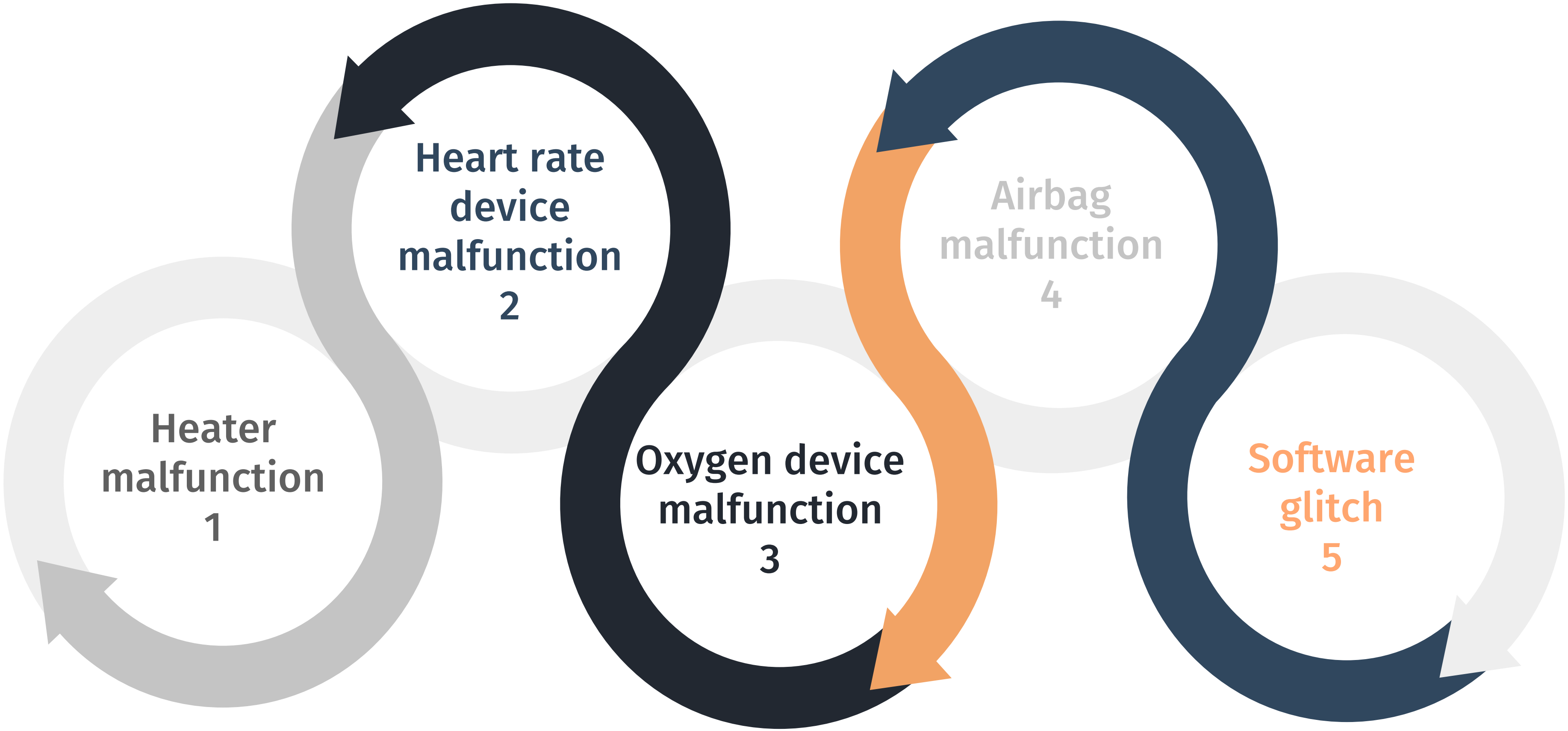


- Modern web technology to make websites feel more like the apps.
- Use functionalities that are only for application on a website like notification.
- Market of PWA will reach up to 10.77 billion dollars by 2027
- Google reports a significant increase of 270% in desktop installations of PWAs from the beginning of 2021.



TO INSTALL THE APP SCAN THE QR CODE

RISK ASSESSMENT USING FMEA



TYPES OF RATING	HEATER	ECG	OXYGEN	AIRBAG	SOFTWARE
SEVERITY RATINGS	9	6	3	10	7.5
OCCURRENCE RATINGS	5	1	1	6.5	5.5
DETECTION RATINGS	8	8	8	8	6
RPN	360	48	48	520	247.5

COMPONENTS	COST IN DOLLARS
ESP-8266	$3.50
ESP-WROOM-32	$7.77
AD-8232 + ELECTRODES	$11.00 + $6.00
MAX-30100	$7.00
MPU-6050	$4.00
DS18B20	$2.00
HEATING PADS	$6.00 * 2
JACKET	$15.00
POWER BANK	$20.00
ADDITIONAL MATERIAL (BREADBOARD, WIRES ETC..)	$20.00
TOTAL COST	$108.27

WORK BREAKDOWN

1.

Project Initiation

1.1. Collect all items
1.2. Test all collected items

2.

Hardware Development

2.1 Hardware connections
2.2 Hardware coding

3.

Tangible Implementation

3.1. Design jacket
3.2. Add all components

4.

Application Development

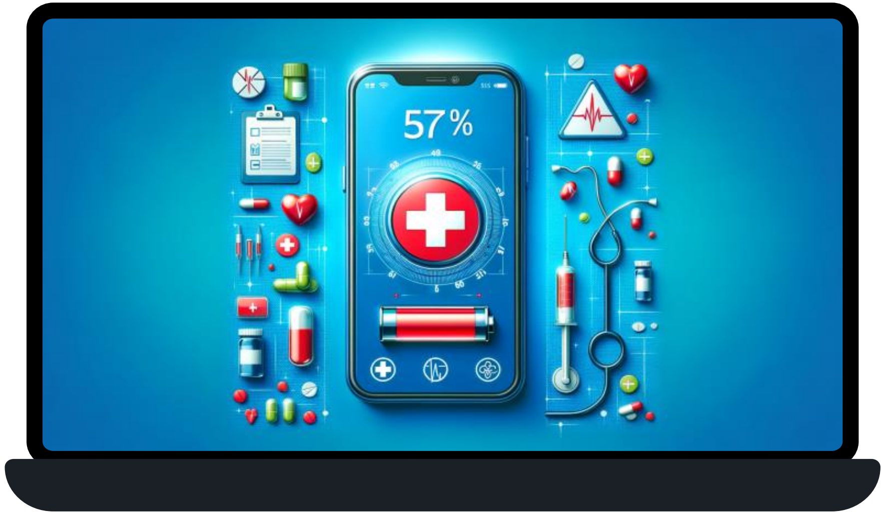
4.1.1. Front-end Development
4.1.2 Back-end Development
4.2. Deployment

5.

Testing

ACTIVITY	JUL	AUG	SEP	OCT	NOV
GET ALL COMPONENTS NEEDED	ELIE & ATIEH	ELIE & ATIEH			
PROGRAM THE HEART RATE			ELIE		
PROGRAM THE OXYGEN SENSOR			ATIEH		
PROGRAM THE TEMPERATURE SENSOR			ELIE		
PROGRAM FALL SENSOR			ATIEH		
DESIGN JACKET				ELIE & ATIEH	
ADD ALL COMPONENTS				ELIE & ATIEH	
CREATE THE FRONT-END				ELIE & ATIEH	
CREATE THE BACK-END				ELIE & ATIEH	
DEPLOY ON SERVER					ELIE
TEST ALL COMPONENTS					ELIE& ATIEH

THANK YOU!



KEEP YOUR ELDERLY SAFE