HARDWARE REPORT

24AIM113 & 24AIM114

Introduction to NN, CNN and GNN

Analog system design

Team Members

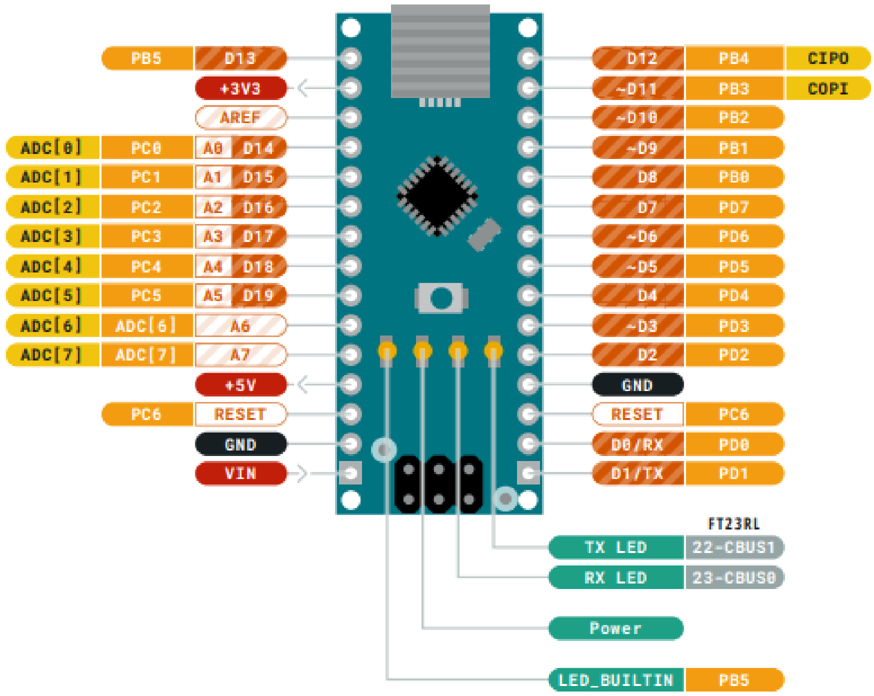
Group 7

CB.AI.U4AIM24004	ANITRA R
CB.AI.U4AIM24028	NARESH L
CB.AI.U4AIM24029	NIMISHA PATEL
CB.AI.U4AIM24050	YATISH S

Faculty In-Charge: Dr. Amruta V & Dr. Snigdhatanu Acharya

Components used and their specifications

1. Arduino Nano



Operating voltage	Input voltage	Power consumption	Flash Memory	Clock Speed	DC per I/O Pins	Analog IN pins	Digital I/O Pins
5V	7-12V	19 mA	32 KB	16 MHz	20 mA	8	22 (6 are PWM)

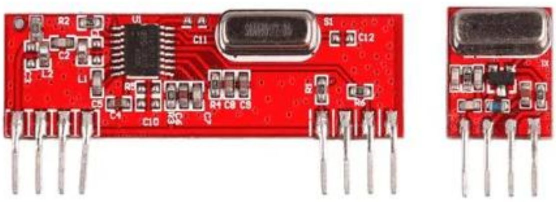
Arduino Nano is a small, complete and breadboard-friendly board based on the ATmega328. It lacks only a DC power jack and works with a Mini-B-USB cable instead of a standard one.

2. Silicon Tube Peristaltic Pump

Voltage	Operating current	Operating Temperature (°C)	Noise Level (Db)	Length (mm)	Width (mm)	Height (mm)	Flow rate (ml/min)
6 VDC	0.35	0-40	<40	67	55	41	10.5

A **peristaltic pump**, also commonly known as a **roller pump**, is a type of positive displacement pump used for pumping a variety of fluids. The fluid is contained in a flexible tube fitted inside a circular pump casing. Most peristaltic pumps work through rotary motion, though linear peristaltic pumps have also been made. The rotor has a number of "wipers" or "rollers" attached to its external circumference, which compress the flexible tube as they rotate by. The part of the tube under compression is closed, forcing the fluid to move through the tube.

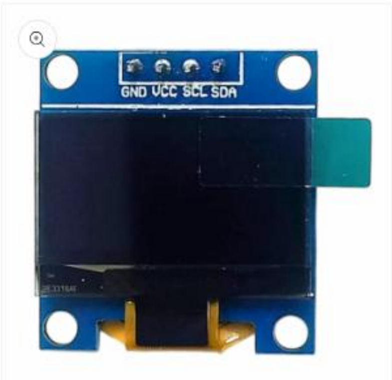
3. 433 MHz RF Transmitter and RF Receiver:



Range in open space (meters)	RX receiver frequency (MHz)	RX Typical Sensitivity (Dbm)	RX supply current (mA)	RX Operating Voltage (V)	TX Frequency Range (MHz)	TX Supply voltage (V)	TX Out Put Power (Dbm)
100	433	105	3.5	5	433.92	3-6	4-12

The **RF transmitter** module is very simple to operate and offers low current consumption (typical. 11mA). Data can be supplied directly from a microprocessor or encoding device, thus keeping the component count down and ensuring a low hardware cost.

4. SSD1306 OLED Display Module



Resolution	V_{in} (or) V_{cc}	Display size	Display type	Dimensions
128 x 64 pixels	3.3-5 V	0.96 inch	OLED Display	2.7cm x 2.8cm

This is the **0.96" OLED display** module. It has 4 pins, amd it is made of 128x64 individual blue OLED pixels, and each one is turned on or off by the controller chip.

It works without a backlight, that is, in a dark environment, its brightness is higher compared to that of an LCD display.

5. AS7263 NIR Spectroscopy Sensor

Sensor	ADC Resolution	Commu-nication	V_{in} (or) V_{cc}	Channels (nm)	Wavelength Accuracy	AOI
Photodiode	16 bits	I2C	2.7-3.6V	610, 680, 730, 760, 810, 860.	±5 nm	20°

The **AS7263 spectrometer** detects wavelengths in the visible range at 610, 680, 730, 760, 810 and 860nm of light, each with 20nm of full-width half-max detection. The board also has multiple ways for us to illuminate objects that we will try to measure for a more accurate spectroscopy reading. There is an onboard LED that has been picked out specifically for this task, as well as two pins to solder our own LED into.

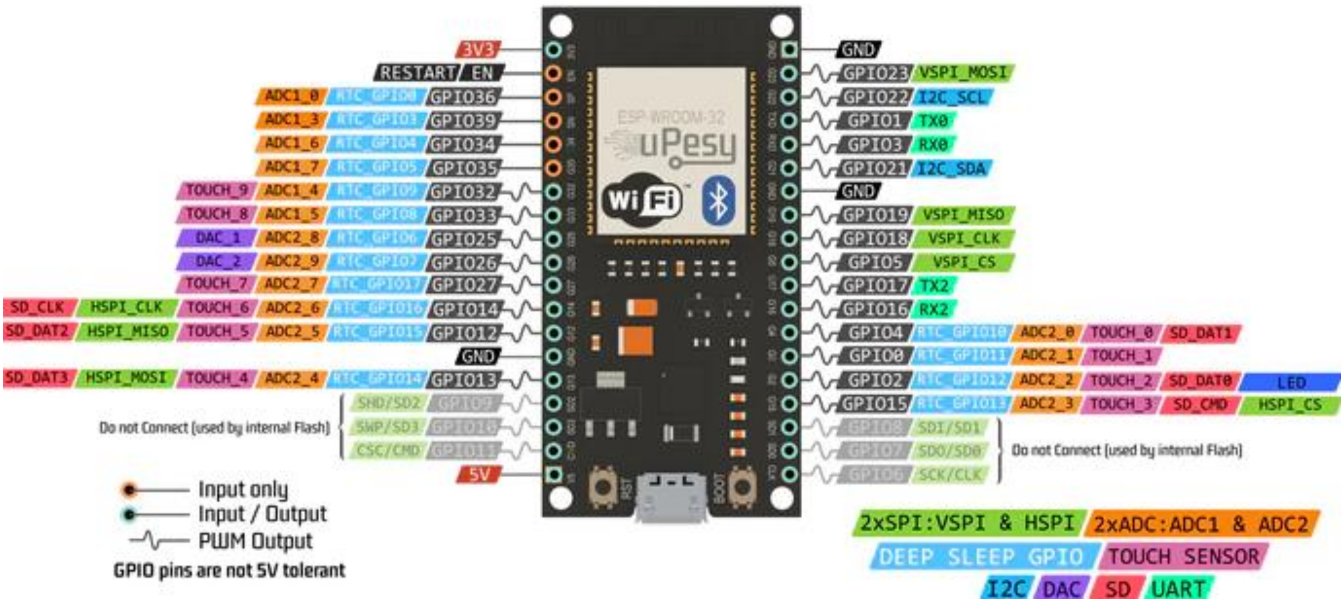
6. **MAX30100 Pulse Oximeter Heart Rate Sensor** (For testing)

Power supply	**Current Draw**	**Red LED Wavelength**	**IR LED Wavelength**	**Temperature Range**	**Temperature Accuracy**
3.3V to 5.5V	~600µA (during Measurements) ~0.7µA (during standby mode)	660 nm	880 nm	-40˚C to +85˚C	±1˚C

The **MAX30100** is a modern, integrated pulse oximeter and heart rate sensor IC, from Analog Devices. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry (SpO2) and heart rate (HR) signals.

On the right, the MAX30100 has two LEDs – a RED and an IR LED. And on the left is a very sensitive photodetector. The idea is that you shine a single LED at a time, detecting the amount of light shining back at the detector, and, based on the signature, you can measure blood oxygen level and heart rate.

7. **ESP32-WROOM-32**

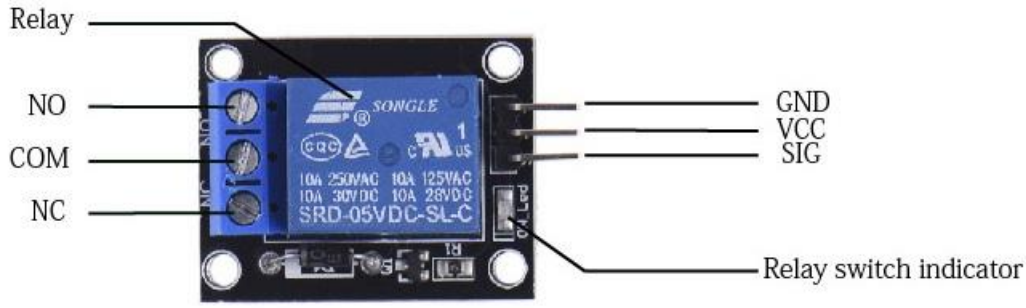


Flash Memory	Processor	Operating voltage (VDC)	Operating current (mA)	Clock frequency (MHz)	Data rate (Mbps)	SRAM Memory (KB)
4 MB	Two Low-Power Xtensa 32-bit LX6 Microprocessors	3.0 V-3.6V	80	80 to 240	150	520

ESP WROOM 32 is a powerful, generic WiFi-BT-BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming, and MP3 decoding.

At the core of this module is the ESP32S chip, which is designed to be scalable and adaptive. There are 2 CPU cores that can be individually controlled or powered. In our project, the ESP32 is in full control of the receival of RF transmission from the CGM, the android application and the insulin pump operations.

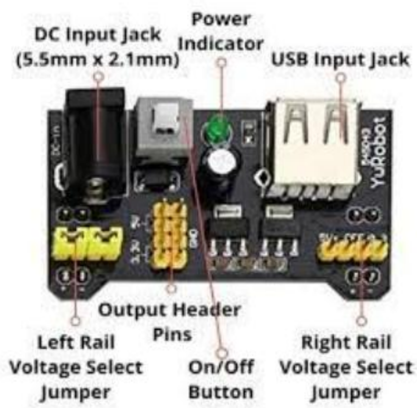
8. **5V Relay Module**



Trigger Voltage	**Trigger Current**	**Maximum Switching Voltage**
5 VDC	20 mA	250 VAC @ 10A 30 VDC @ 10 A

A **1-channel 5V control Single-Pole Double-Throw (SPDT) relay board** can be controlled directly via a microcontroller and switch up to 10A at 250 VAC. The inputs are isolated to protect any delicate control circuitry. This might be replaced with MOSFET in the final design.

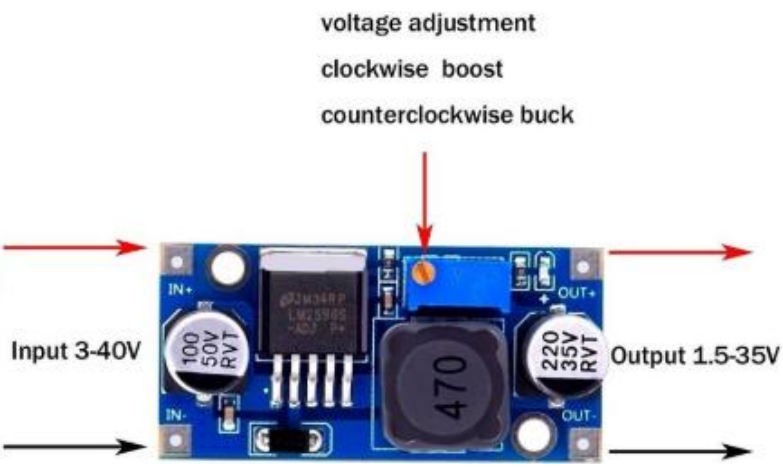
9. **3.3V/5V MB102 Breadboard Power Supply Module:**



Input Voltage	**Output Voltage**	**Maximum output current**
6.5-12 V	3.3 V/ 5V	<700 mA

It is a **3.3V/5V MB102 Breadboard Power Supply Module** which provides a dual 5V and 3.3V power rails and has a multi-purpose female USB socket. The module can also output 5V on USB connector or input through the USB connector. The idea behind the usage of this module is that, it can directly power the ESP32, as well as the 6V DC pump (at a lower 5V, hence lower speed) without having to use multiple buck converters.

10. **LM2596 DC-DC Buck Converter** (Alternative)



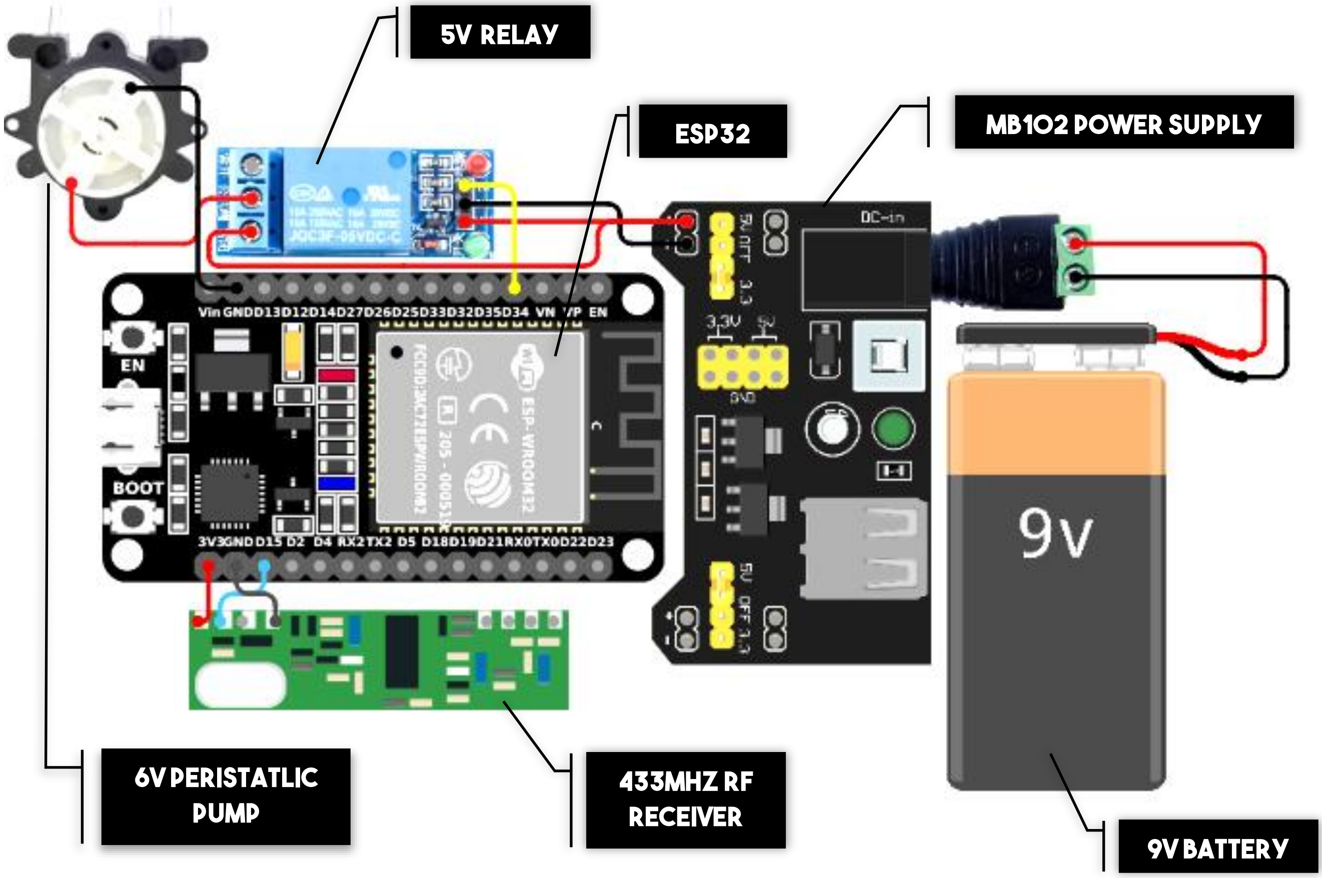
Input voltage	Output voltage	Output current	Switching Frequency	Operating temperature	Conversion efficiency
3-40V	1.5-35V	Rated current is 2A, maximum 3A	150 KHz	Industrial grade (-40 to +85)	92% (highest)

It is a **step-down (buck) switching regulator**, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5V, 12V, and an adjustable output version.

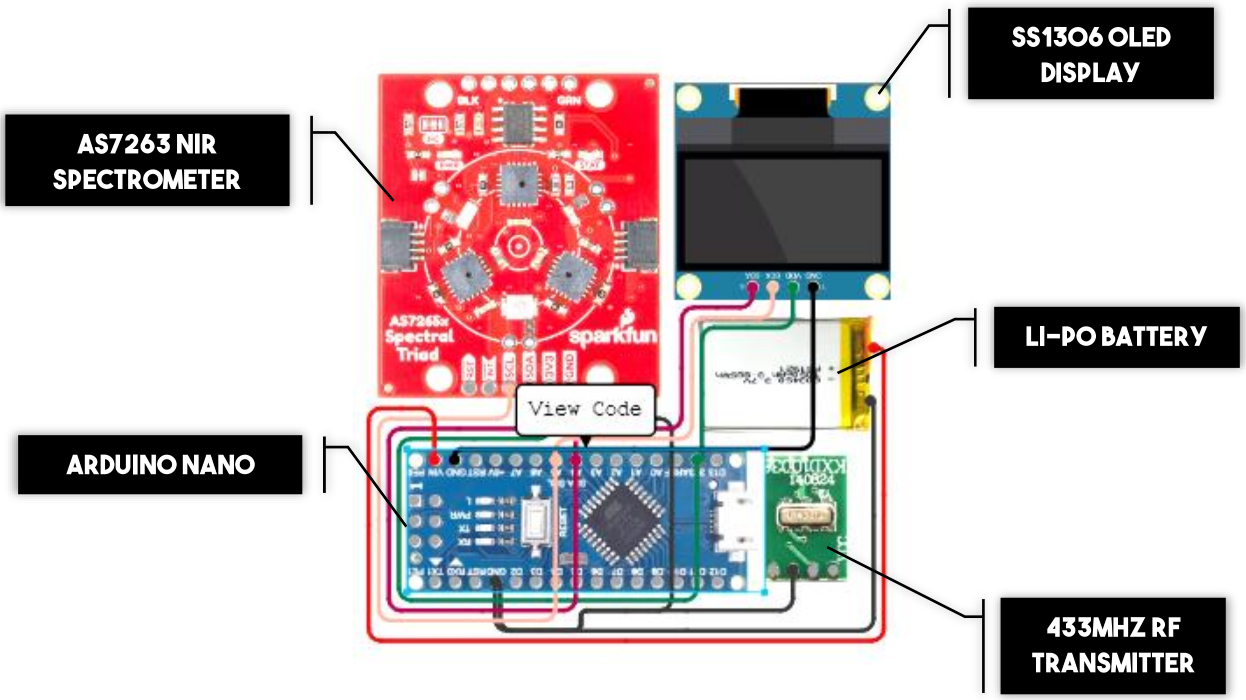
To allow more precise control over the regulation of the 9V coming from our battery, this buck converter could be used. Since the MB102 Power supply module only outputs a fixed voltage of either 5V or 3.3V, the LM2596 can be used to run the peristaltic pump at its maximum potential (6V) with precise control over the voltage.

Circuit Diagram and Pin Connections

1. **Insulin Pump,** controlled by ESP32-WROOM-32.



2. **CGM watch module,** controlled by Arduino Nano.



CODE: ESP32 - 433 MHz Receiver

```
#include <RH_ASK.h>
#include <SPI.h>

RH_ASK rf_driver(2000, 34, 22);

void setup() {
  Serial.begin(115200);
  delay(4000);
  Serial.println("ESP32 433MHz receiver");
  if (RH_PLATFORM == RH_PLATFORM_ESP32)
    Serial.println("RH_PLATFORM_ESP32");
  delay(5000);
  Serial.println("Receiver: rf_driver initialising");
  if (!rf_driver.init()) {
    Serial.println("init failed");
    while (1) delay(1000);
  }
  Serial.println("Receiver: rf_driver initialised");
}

void loop() {
  uint8_t buf[20]={0};
  uint8_t buflen = sizeof(buf);
  // Check if received packet is correct size
  if (rf_driver.recv(buf, &buflen)) {
    // Message received with valid checksum
    Serial.print("Message Received: ");
    Serial.println((char*)buf);
  }
}
```

Arduino Nano - CGM Sensor with MAX30100

```
#include <Wire.h>
```

```
#include <EEPROM.h>
#include "MAX30100_PulseOximeter.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <RH_ASK.h>
#include <SPI.h>

#define ENABLE_MAX30100 1
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#if ENABLE_MAX30100
#define REPORTING_PERIOD_MS 5000

PulseOximeter pox;
RH_ASK driver;
#endif

uint32_t tsLastReport = 0;
uint32_t tsLastSave = 0;

float glucose_records[3] = {0.0, 0.0, 0.0}; // Array to store the last three
records

void onBeatDetected()
{
    Serial.println("Beat!");
}

void setup()
{
    Serial.begin(115200);
```

```
    Serial.println("SSD1306 128x32 OLED TEST");

    if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;);
    }

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(20, 18);
    display.print("Pulse OxiMeter");
    int temp1 = 0;
    int temp2 = 40;
    int temp3 = 80;
    display.display();
    delay(2000);
    display.cp437(true);
    display.clearDisplay();

    Serial.print("Initializing pulse oximeter..");
#if ENABLE_MAX30100
    if (!pox.begin()) {
        Serial.println("FAILED");
        for (;;);
    } else {
        Serial.println("SUCCESS");
    }

    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
    pox.setOnBeatDetectedCallback(onBeatDetected);
#endif

    load_glucose_records();
    print_glucose_records(); // Print loaded records to Serial
}
```

```
void loop()
{
#if ENABLE_MAX30100
    pox.update();
    int bpm = 0;
    int spo2 = 0;
    float glucose_level = 0.0;

    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        bpm = pox.getHeartRate();
        spo2 = pox.getSpO2();

        if (bpm > 0 && spo2 > 0) {
            glucose_level = 16714.61 + 0.47 * bpm - 351.045 * spo2 + 1.85 * (spo2
* spo2);
        }

        Serial.print("Heart rate: ");
        Serial.println(bpm);
        Serial.print("SpO2: ");
        Serial.println(spo2);
        Serial.print("Glucose Level: ");
        Serial.println(glucose_level);

        tsLastReport = millis();
        display_data(bpm, spo2, glucose_level);
    }

    // Save the glucose level if it is smaller than 500 every 10 seconds
    if (glucose_level < 500.0 && glucose_level > 0 && millis() - tsLastSave >
10000) {
        save_glucose_level(glucose_level);
        tsLastSave = millis();
    }
#endif
}

void display_data(int bpm, int spo2, float glucose_level)
```

```
{
    display.clearDisplay();

    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 0);
    display.print("BPM:");
    display.println(bpm);

    display.setCursor(0, 10);
    display.print("SpO2:");
    display.println(spo2);

    display.setCursor(0, 20);
    if (glucose_level > 500.0 || glucose_level < 0) {
        display.print("Glucose:MEAS");
    } else {
        display.print("Glucose:");
        display.println(glucose_level);
    }

    // Display last three glucose records
    display.drawLine(88, 0, 88, 32, WHITE);
    for (int i = 0; i < 3; i++) {
        display.setCursor(92, 0 + (i * 10));
        display.print(glucose_records[i]);
    }

    display.display();
}

void save_glucose_level(float glucose_level)
{
    // Shift old records
    glucose_records[2] = glucose_records[1];
    glucose_records[1] = glucose_records[0];
    glucose_records[0] = glucose_level;
```

```
    // Save records to EEPROM
    for (int i = 0; i < 3; i++) {
        EEPROM.put(i * sizeof(float), glucose_records[i]);
    }

    print_glucose_records(); // Print saved records to Serial for debugging
}

void load_glucose_records()
{
    for (int i = 0; i < 3; i++) {
        EEPROM.get(i * sizeof(float), glucose_records[i]);
    }
}

void print_glucose_records()
{
    Serial.println("Glucose Records:");
    char msg[10];
    for (int i = 0; i < 3; i++) {
        Serial.print(i);
        Serial.print(": ");
        Serial.println(glucose_records[i]);
        dtostrf(glucose_records[i], 6, 2, msg);
        driver.send((uint8_t *)msg, strlen(msg));
        driver.waitPacketSent();
        delay(1000);
    }

}
```