**HEALTH MONITOR CODES**

**ARDUINO CODE WITH ALL FUNCTIONALITIES (CHANGES AND TESTING NEEDED):**

#include <Wire.h>

#include <Adafruit\_MLX90614.h>

#include <Adafruit\_MPU6050.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_MAX30105.h>

#include <TinyGPS++.h>

#include <WiFi.h>

#include <PubSubClient.h>

// Wi-Fi credentials

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

// MQTT broker details

const char\* mqtt\_server = "broker\_address"; // Replace with your broker address

WiFiClient espClient;

PubSubClient client(espClient);

// GPS

TinyGPSPlus gps;

HardwareSerial gpsSerial(1); // Use UART1 for GPS communication

// Sensors

Adafruit\_MAX30105 max30102;

Adafruit\_MPU6050 mpu;

Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();

int steps = 0; // Step counter

float heartRate = 0;

float spO2 = 0;

float bodyTemp = 0;

String location = "No Fix";

// MPU6050 step counting variables

float accX, accY, accZ;

float lastAccZ = 0;

void setup() {

Serial.begin(115200);

// Setup Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println("Wi-Fi connected");

// Setup MQTT

client.setServer(mqtt\_server, 1883);

// Setup GPS (NEO-6M)

gpsSerial.begin(9600, SERIAL\_8N1, 16, 17); // RX = 16, TX = 17

// Initialize sensors

if (!max30102.begin()) {

Serial.println("MAX30102 not found!");

}

if (!mpu.begin()) {

Serial.println("MPU6050 not found!");

}

if (!mlx.begin()) {

Serial.println("MLX90614 not found!");

}

// Set sensor ranges for MPU6050

mpu.setAccelerometerRange(MPU6050\_RANGE\_2\_G);

mpu.setGyroRange(MPU6050\_RANGE\_250\_DEG);

// Initialize the MAX30102

max30102.setup();

max30102.setPulseAmplitudeRed(0x0A); // Turn Red LED to low to indicate a heartbeat

max30102.setPulseAmplitudeGreen(0); // Turn off Green LED

}

void reconnect() {

while (!client.connected()) {

if (client.connect("ESP32HealthMonitor")) {

Serial.println("Connected to MQTT");

} else {

delay(5000);

}

}

}

void readHeartRateAndSpO2() {

long irValue = max30102.getIR();

if (irValue > 50000) { // Only report heart rate if the signal is above 50,000

heartRate = random(60, 100); // Simulate data for heart rate (replace with actual processing)

spO2 = random(90, 100); // Simulate SpO2 data

} else {

heartRate = 0;

spO2 = 0;

}

}

void readTemperature() {

bodyTemp = mlx.readObjectTempC(); // Read body temperature in Celsius

}

void readGPS() {

while (gpsSerial.available()) {

gps.encode(gpsSerial.read());

if (gps.location.isUpdated()) {

location = String(gps.location.lat(), 6) + ", " + String(gps.location.lng(), 6);

}

}

}

void countSteps() {

sensors\_event\_t accelEvent;

mpu.getEvent(&accelEvent, NULL, NULL);

accX = accelEvent.acceleration.x;

accY = accelEvent.acceleration.y;

accZ = accelEvent.acceleration.z;

if (abs(accZ - lastAccZ) > 1.5) { // Detect significant motion in Z-axis

steps++;

}

lastAccZ = accZ;

}

void publishData() {

if (!client.connected()) {

reconnect();

}

client.loop();

String payload = "{";

payload += "\"heartRate\":" + String(heartRate) + ",";

payload += "\"spO2\":" + String(spO2) + ",";

payload += "\"steps\":" + String(steps) + ",";

payload += "\"temperature\":" + String(bodyTemp) + ",";

payload += "\"location\":\"" + location + "\"";

payload += "}";

client.publish("health\_monitor/data", payload.c\_str());

Serial.println("Published Data: " + payload);

}

void loop() {

readHeartRateAndSpO2(); // Read heart rate and SpO2 from MAX30102

readTemperature(); // Read temperature from MLX90614

readGPS(); // Read GPS data from U-BLOX NEO 6M

countSteps(); // Count steps using MPU6050

publishData(); // Send data over MQTT

delay(5000); // Adjust the delay based on how frequently you want to publish

}

**ARDUINO CODE USING CUSTOM BROKER (WORKING):**

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <PubSubClient.h>

// Wi-Fi credentials

const char\* ssid = "";

const char\* wifi\_password = "";

// MQTT broker details

const char\* mqtt\_server = "";

const int mqtt\_port = 8883;

const char\* mqtt\_user = "";

const char\* mqtt\_password = "";

const char\* topic = "";

// Create a WiFiClientSecure instance

WiFiClientSecure espClient;

PubSubClient client(espClient);

void setup\_wifi() {

  Serial.print("Connecting to Wi-Fi...");

  WiFi.begin(ssid, wifi\_password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("Connected!");

}

void reconnect() {

  espClient.setInsecure();

  while (!client.connected()) {

    Serial.print("Connecting to MQTT broker...");

    if (client.connect("ESP8266Client", mqtt\_user, mqtt\_password)) {

      Serial.println("Connected!");

      client.publish(topic, "HELLO WORLD");

    } else {

      Serial.print("Failed. Error code: ");

      Serial.print(client.state());

      Serial.println(" Retrying in 5 seconds...");

      delay(5000);

    }

  }

}

void setup() {

  Serial.begin(115200);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

}

void loop() {

  if (!client.connected()) {

    reconnect();

  }

  client.loop();

}

**ARDUINO CODE USING PUBLIC BROKER (NEED TO BE TESTED):**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

// WiFi credentials

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

// MQTT broker details

const char\* mqtt\_server = "broker.hivemq.com"; // Use your broker

const int mqtt\_port = 1883;

const char\* topic = "esp8266/hello";

// Create WiFi and MQTT client instances

WiFiClient espClient;

PubSubClient client(espClient);

void setup\_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

if (client.connect("ESP8266Client")) {

Serial.println("connected");

// Once connected, publish "HELLO WORLD" to the topic

client.publish(topic, "HELLO WORLD");

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

delay(5000);

}

}

}

void setup() {

Serial.begin(115200);

setup\_wifi();

client.setServer(mqtt\_server, mqtt\_port);

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

}

**HTML CODE USING CUSTOM BROKER (WORKING):**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>ESP8266 MQTT Webpage</title>

    <script src="https://unpkg.com/mqtt/dist/mqtt.min.js"></script>

</head>

<body>

    <h1 id="message">Waiting for message...</h1>

    <script>

        // MQTT broker details

        const broker = {

            hostname: '',

            port: 8884,

            username: '',

            password: ''

        };

        const options = {

            keepalive: 60,

            clean: true,

            reconnectPeriod: 1000,

            connectTimeout: 30 \* 1000,

            username: broker.username,

            password: broker.password,

            clientId: 'webclient\_' + Math.random().toString(16).substr(2, 8)

        };

        const topic = '';

        const client = mqtt.connect(`wss://${broker.hostname}:${broker.port}/mqtt`, options);

        client.on('connect', function () {

            console.log('Connected to MQTT broker');

            client.subscribe(topic, function (err) {

                if (err) {

                    console.error('Subscription error:', err);

                } else {

                    console.log('Subscribed to topic:', topic);

                }

            });

        });

        client.on('message', function (topic, message) {

            console.log('Received message:', message.toString());

            document.getElementById('message').innerText = message.toString();

        });

        client.on('error', function (err) {

            console.error('Connection error:', err);

        });

    </script>

</body>

</html>

**HTML CODE USING PUBLIC BROKER (NEED TO BE TESTED):**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>ESP8266 MQTT</title>

<script src="https://cdnjs.cloudflare.com/ajax/libs/mqtt/4.2.6/mqtt.min.js"></script>

</head>

<body>

<h1 id="message">Waiting for message...</h1>

<script>

// Connect to MQTT broker

const client = mqtt.connect('wss://broker.hivemq.com:8000/mqtt');

const topic = 'esp8266/hello';

client.on('connect', () => {

console.log('Connected to MQTT broker');

client.subscribe(topic, (err) => {

if (!err) {

console.log('Subscribed to topic');

}

});

});

client.on('message', (topic, message) => {

document.getElementById('message').innerText = message.toString();

});

</script>

</body>

</html>

FULL ARDUINO? :

#include <Wire.h>

#include <Adafruit\_MLX90614.h>

#include <Adafruit\_MPU6050.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_MAX30105.h>

#include <TinyGPS++.h>

#include <WiFi.h>

#include <PubSubClient.h>

// Wi-Fi credentials

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

// HiveMQ public MQTT broker details

const char\* mqtt\_server = "broker.hivemq.com";

WiFiClient espClient;

PubSubClient client(espClient);

// GPS

TinyGPSPlus gps;

HardwareSerial gpsSerial(1);  // Use UART1 for GPS communication

// Sensors

Adafruit\_MAX30105 max30102;

Adafruit\_MPU6050 mpu;

Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();

int steps = 0;      // Step counter

float heartRate = 0;

float spO2 = 0;

float bodyTemp = 0;

String location = "No Fix";

// MPU6050 step counting variables

float accX, accY, accZ;

float lastAccZ = 0;

void setup() {

  Serial.begin(115200);

  // Setup Wi-Fi

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("Wi-Fi connected");

  // Setup MQTT

  client.setServer(mqtt\_server, 1883);

  // Setup GPS (NEO-6M)

  gpsSerial.begin(9600, SERIAL\_8N1, 16, 17);  // RX = 16, TX = 17

  // Initialize sensors

  if (!max30102.begin()) {

    Serial.println("MAX30102 not found!");

  }

  if (!mpu.begin()) {

    Serial.println("MPU6050 not found!");

  }

  if (!mlx.begin()) {

    Serial.println("MLX90614 not found!");

  }

  // Set sensor ranges for MPU6050

  mpu.setAccelerometerRange(MPU6050\_RANGE\_2\_G);

  mpu.setGyroRange(MPU6050\_RANGE\_250\_DEG);

  // Initialize the MAX30102

  max30102.setup();

  max30102.setPulseAmplitudeRed(0x0A); // Turn Red LED to low to indicate a heartbeat

  max30102.setPulseAmplitudeGreen(0);  // Turn off Green LED

}

void reconnect() {

  while (!client.connected()) {

    if (client.connect("ESP32HealthMonitor")) {

      Serial.println("Connected to HiveMQ MQTT");

    } else {

      delay(5000);

    }

  }

}

void readHeartRateAndSpO2() {

  long irValue = max30102.getIR();

  if (irValue > 50000) {  // Only report heart rate if the signal is above 50,000

    heartRate = random(60, 100);  // Simulate data for heart rate (replace with actual processing)

    spO2 = random(90, 100);       // Simulate SpO2 data

  } else {

    heartRate = 0;

    spO2 = 0;

  }

}

void readTemperature() {

  bodyTemp = mlx.readObjectTempC();  // Read body temperature in Celsius

}

void readGPS() {

  while (gpsSerial.available()) {

    gps.encode(gpsSerial.read());

    if (gps.location.isUpdated()) {

      location = String(gps.location.lat(), 6) + ", " + String(gps.location.lng(), 6);

    }

  }

}

void countSteps() {

  sensors\_event\_t accelEvent;

  mpu.getEvent(&accelEvent, NULL, NULL);

  accX = accelEvent.acceleration.x;

  accY = accelEvent.acceleration.y;

  accZ = accelEvent.acceleration.z;

  if (abs(accZ - lastAccZ) > 1.5) {  // Detect significant motion in Z-axis

    steps++;

  }

  lastAccZ = accZ;

}

void publishData() {

  if (!client.connected()) {

    reconnect();

  }

  client.loop();

  // Publish each sensor data to its respective topic

  client.publish("health\_monitor/heartRate", String(heartRate).c\_str());

  client.publish("health\_monitor/spO2", String(spO2).c\_str());

  client.publish("health\_monitor/steps", String(steps).c\_str());

  client.publish("health\_monitor/temperature", String(bodyTemp).c\_str());

  client.publish("health\_monitor/location", location.c\_str());

  // Print published data

  Serial.println("Published Data:");

  Serial.println("Heart Rate: " + String(heartRate));

  Serial.println("SpO2: " + String(spO2));

  Serial.println("Steps: " + String(steps));

  Serial.println("Temperature: " + String(bodyTemp));

  Serial.println("Location: " + location);

}

void loop() {

  readHeartRateAndSpO2();  // Read heart rate and SpO2 from MAX30102

  readTemperature();       // Read temperature from MLX90614

  readGPS();               // Read GPS data from U-BLOX NEO 6M

  countSteps();            // Count steps using MPU6050

  publishData();           // Send data over MQTT

  delay(5000);             // Adjust the delay based on how frequently you want to publish

}