ARDUINO code for gathering Temperature, Heart rate and oxygen saturation:

#include <OneWire.h>

#include <DallasTemperature.h>

#include <Wire.h>

#include "MAX30100\_PulseOximeter.h"

#include <SoftwareSerial.h>

#define REPORTING\_PERIOD\_MS     10000

// Initialize Arduino to NodeMCU (5=Rx & 6=Tx)

SoftwareSerial SUART(5, 6);

// Create a PulseOximeter object

PulseOximeter pox;

// Time at which the last beat occurred

uint32\_t tsLastReport = 0;

uint32\_t lastTime = 0;

// Data wire is plugged into port 2 on the Arduino

#define ONE\_WIRE\_BUS 2

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)

OneWire oneWire(ONE\_WIRE\_BUS);

// Pass our oneWire reference to Dallas Temperature.

DallasTemperature sensors(&oneWire);

DeviceAddress tempDeviceAddress;

int resolution = 12;

unsigned long lastTempRequest = 0;

int delayInMillis = 0;

float temperature = 0.0;

int idle = 0;

void onBeatDetected() {

  Serial.println("Beat!");

}

void configureMax30100() {

  pox.setIRLedCurrent(MAX30100\_LED\_CURR\_7\_6MA);

  pox.setOnBeatDetectedCallback(onBeatDetected);

}

void setup(void) {

  Serial.begin(9600);

  SUART.begin(9600);

  Serial.println("Dallas Temperature Control Library - Async Demo");

  Serial.print("Library Version: ");

  Serial.println(DALLASTEMPLIBVERSION);

  Serial.println("\n");

  sensors.begin();

  sensors.getAddress(tempDeviceAddress, 0);

  sensors.setResolution(tempDeviceAddress, resolution);

  sensors.setWaitForConversion(false);

  sensors.requestTemperatures();

  delayInMillis = 750 / (1 << (12 - resolution));

  lastTempRequest = millis();

  pinMode(13, OUTPUT);

  Serial.print("Initializing pulse oximeter..");

  Wire.setClock(400000UL);

  if (!pox.begin()) {

    Serial.println("FAILED");

    while (1);

  } else {

    Serial.println("SUCCESS");

  }

  configureMax30100();

  lastTime = millis();

}

void loop(void) {

  if (millis() - lastTempRequest >= 5000) {

    temperature = sensors.getTempCByIndex(0);

    idle = 0;

    resolution++;

    if (resolution > 12) resolution = 9;

    sensors.setResolution(tempDeviceAddress, resolution);

    sensors.requestTemperatures();

    delayInMillis = 750 / (1 << (12 - resolution));

    lastTempRequest = millis();

    Serial.print(" Temperature: ");

    temperature = sensors.getTempCByIndex(0);

    Serial.println(temperature, resolution - 8);

  }

  delay(10); // Increased delay to slow down the loop

  idle++;

  pox.update();

  if (millis() - tsLastReport > REPORTING\_PERIOD\_MS) {

    Serial.print("Heart rate:");

    Serial.print(pox.getHeartRate());

    Serial.print("bpm / SpO2:");

    Serial.print(pox.getSpO2());

    Serial.println("%");

    SUART.print("Temperature=");

    SUART.print(temperature);

    SUART.print("\n");

    SUART.print("HeartRate=");

    SUART.print(pox.getHeartRate());

    SUART.print("\n");

    SUART.print("SpO2=");

    SUART.print(pox.getSpO2());

    SUART.print("\n");

  }

  if (millis() - lastTime > REPORTING\_PERIOD\_MS) {

    lastTime = millis();

  }

}

NodeMCU ESP8266 code for receiving data from ARDUINO and sending it to ThingSpeak:

#include <ESP8266WiFi.h>

#include "ThingSpeak.h"

#include <SoftwareSerial.h>

char ssid[] = "AndroidAP3b82"; // Your network SSID (name)

char pass[] = "jppb8534"; // Your network password

int keyIndex = 0;

WiFiClient client;

unsigned long myChannelNumber = 2302542;

const char \*myWriteAPIKey = "3TTQUVNCBU057UMH";

SoftwareSerial SUART(D2, D1); // SRX = D2, STX = D1

float temperature = 0.0;

int heartRate = 0;

int spo2 = 0;

void parseData(String receivedData) {

  if (receivedData.startsWith("Temperature=")) {

    temperature = receivedData.substring(12).toFloat();

  } else if (receivedData.startsWith("HeartRate=")) {

    heartRate = receivedData.substring(10).toInt();

  } else if (receivedData.startsWith("SpO2=")) {

    spo2 = receivedData.substring(5).toInt();

  }

}

void setup() {

  Serial.begin(9600);

  SUART.begin(9600);

  WiFi.mode(WIFI\_STA);

  WiFi.begin(ssid, pass);

  ThingSpeak.begin(client);

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

    delay(1000);

    Serial.println("Connecting to WiFi...");

  }

  Serial.println("Connected to WiFi");

}

void loop() {

  if (SUART.available()) {

    String receivedData = SUART.readStringUntil('\n'); // Read a line of data

    Serial.println("Received data: " + receivedData);

    parseData(receivedData);

    // Print the parsed values for debugging

    Serial.print("Parsed Data - Temperature: ");

    Serial.print(temperature);

    Serial.print(", Heart Rate: ");

    Serial.print(heartRate);

    Serial.print(", SpO2: ");

    Serial.println(spo2);

    // Send parsed data to ThingSpeak

    ThingSpeak.setField(1, temperature);

    ThingSpeak.setField(2, heartRate);

    ThingSpeak.setField(3, spo2);

    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

    // Print the URL for debugging

    String url = "http://api.thingspeak.com/update?api\_key=" + String(myWriteAPIKey) +

                 "&field1=" + String(temperature) +

                 "&field2=" + String(heartRate) +

                 "&field3=" + String(spo2);

    Serial.println("Sending data to: " + url);

    if (x == 200) {

      Serial.println("Channel update successful.");

    } else {

      Serial.print("Problem updating channel. HTTP error code ");

      Serial.println(x);

    }

  }

  delay(15000); // Wait 15 seconds to update the channel again

}