University of Bahrain
College of Engineering
Internet of Things (IoT)
EEM601

Prepared By: Najeeba Mohd
University Id: 20112695
Supervised By: Dr. Mohab Mangoud

Final Project Report: Remote Heart Rate Monitoring System for HIoT Based Application

1. Introduction:

Healthcare Internet of Things (HIoT) is a technology that is used to manage and monitor the state of a patient body health or the patient surrounding area remotely, in order to reduce the risk and save more lives outside the hospital. It is a combination of several fields altogether, microelectronics, medical, computer science and others [1]. HIoT is also used to improve and assist the medical organizations to take actions for emergency situations in case of any abnormalities detected, with the help of sensor networks, internet network and data analysis methods [2]. In this project, a continuous heart rate monitoring is going to be applied through an IoT based system.

HIoT is needed for many situations, some of the cases develops suddenly for people, especially to patients with a chronic disease. Also, medical parties are unaware about the adherence of the patients for taking their medicines regularly, or if they are frequently measuring their health status, or could be the situation is that the patient is not following hospital appointments, then the doctors are not notified about cases that needs their intervention. Although, there are no medical remote monitoring services, and no enough awareness for the patients and their family about what is their current health state, or what instructions they should follow in case of emergency.

A review: there are multiple solutions that HIoT can provide, such as:

i. Medicine tracking: There are applications available for the patient like MyTherapy, Medisafe or Pill Reminder, to track the record of medicine intake and set reminders for appointments and refills, some of them also generates reports based on patient condition to be sent to the doctors [3], but this solution is not suitable for all type of patients.
ii. Smart NFC tag system: This system gives the ability to since the pills to track when it has been taken from their box, and log the time for that event, for more accuracy [4].
iii. Remote checkups and monitoring: This project interest is mainly for this type of HIoT, it allows the patient to update the medical records without the need to make rounds to the hospital, especially for those who needs daily examination or who needs continuous monitoring, because sometimes their disease status could change without being noticed. Remote monitoring and checkups can be done through the connected body sensors that communicates the collected data to the medical institute, by measuring the physical state of the patient or the surrounding area [2].
iv. Increase awareness: It could affect medical institute to be aware about the patient condition and location in case of any emergency needed. Also increases the awareness of the patient about his own health state and receive feedbacks or instructions directly from the doctors [2].

The outline of the remaining of this report is organized as follows. Section 2 presents a brief about what is needed and understand what are we targeting for our measurements. Section 3 discusses the used tools for the project application and describe the setup connecting them together. While Section 4 shows the code steps and results of the applied system. Section 5 discusses the possible improvements. Ending with Section 6 for concluding the work and have a short summary.

2. Heart Rate Measurement:

The heart rate is one of the most important things to be monitored for a patient kept under observation, then, we need to know what are the normal measurements for a heart rate. It is measured while in rest, and can be manually done by placing two fingers between the bone and the tendon over the radial artery located on the thumb side of the wrist. A normal adult has a heart rate range between 70 to 100 Beat Per Minute (BPM) [5]. This BPM normal reading varies depending on different factors, Table 1 shows the normal BPM reading based on age [6]. The activity level is another factor, as for a well-trained athlete, the BPM might be closer to 40, indicating a good health condition when measures in normal resting state. Other factors affecting the heart rate like air temperature, emotions, body size, smoking… etc [5].

Table 1 Normal resting heart rate readings based on age [6]

Age	Normal resting heart rate
Newborn-12 months	100-160 BPM
1-3 years	90-150
3-5 years	80-140
5-12 years	70-120
12-18 years	60-100
Above 18	70-100

3. Tools and Setup:

The tools used to complete this project is as below:

- Hardware:
  - ESP 8266 (NodeMCU): ESP8266 board **can connect objects together and** has internal antenna to **transfer data using the Wi-Fi protocol**. It is easy to use and programmable with Arduino IDE language [7].
  - Heart rate sensor (comimark): It senses the heartbeat by measuring the change in the transmitted light amount as a result from the expansion of the capillary blood vessels, a pulse occurrence means less light reflection, and the time between two pulses means more light reflection (sensor can be placed on the fingertip).
  - OLED screen (128*64 display): It is used to display the measured value through the microcontroller, to confirm the reading and synchronize with the cloud.
  - Breadboard: used to place all parts together.
  - Connecting wires: used to connect from a pin to another, over the breadboard.
  - Arduino Nano: used only for testing. It was not used for the project because it has no wireless technology, which is needed for this application.

- Software:

- Arduino IDE 1.8.16: used to prepare the code and verify it with the compatibility of the board used, then upload it to that board.
- Ubidots.com cloud (student access): used as the cloud end, to track the measured heart rate in a remote web (real-time) that can be opened anywhere and far from the patient, through the internet.

Figure 1 shows more details about the heart rate sensor. It has 3 pins, ground pin (GND), 3V/5V pin (VCC) and analog pin (A0). It also has an LED to help detecting the heartbeat, and another part is the noise elimination to keep the noise away to not affect the reading [8].

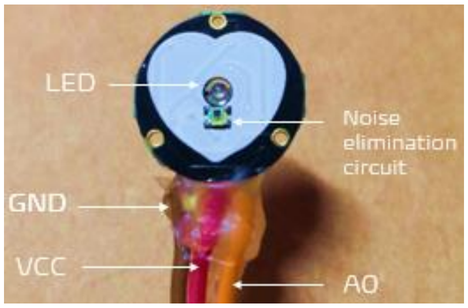


Figure 1 Heart rate sensor details

Figure 2 shows the hardware connection for getting ready. Connected the GND and VCC pins from both sensor and OLED to the ESP8266 same pins, the sensor A0 pin connected to ESP8266 A0 pin (analog), and OLED SDA, SCK pins connected to D2, D1 pins in ESP8266 to read the date for the display.

This setup and idea are taken from [9], with many code changes, the details will be discussed later.

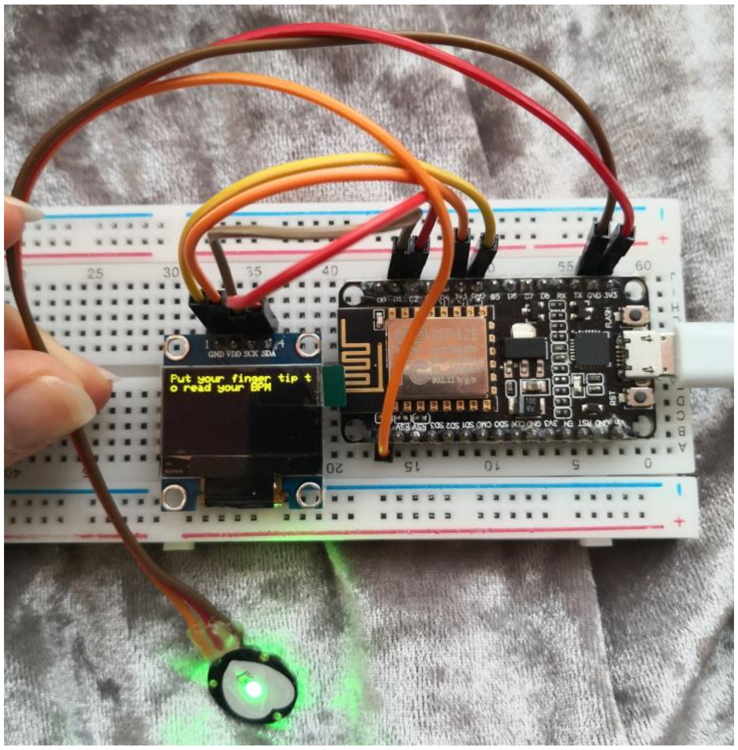
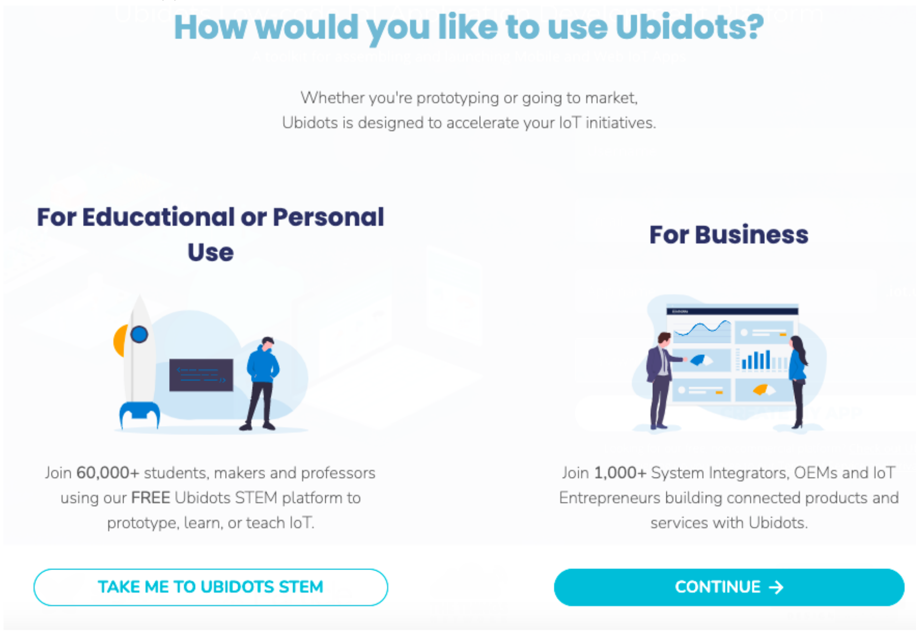


Figure 2 Hardware connection for the remote heart rate monitoring system

4. Steps, code and Results:

a. First, I have created an educational account in Ubidots.com that is free with limited features, for example limited number points (number of times to push data) and a number of connected devices to the account, another type of access is available for businesses with more features.



How would you like to use Ubidots?

Whether you're prototyping or going to market,
Ubidots is designed to accelerate your IoT initiatives.

For Educational or Personal Use

Join 60,000+ students, makers and professors using our FREE Ubidots STEM platform to prototype, learn, or teach IoT.

For Business

Join 1,000+ System Integrators, OEMs and IoT Entrepreneurs building connected products and services with Ubidots.

Figure 3 Ubodots create account

b. Downloaded Arduino IDE [10]

c. From [11]and [12], I have learnt how to install ESP on my IDE and had a guidance to establish a serial connection with my laptop.

d. Learnt from [13] to download and install the USB driver (to read the port of ESP8266 through my MacBook, as I had difficulties to figure this out in the end).

e. Downloaded the needed libraries from github.com.

f. I have tested using the same setup and code in [9], but things were not working, so I went for separate collection and separate testing for each part of the circuit.

g. Tested the OLED screen with the help of the ready examples in Arduino IDE from the installed library for the OLED (File> examples > Adafruit SSD1306 > ssd1306_128x64i2c) , removed all the not needed parts for me, and modified the code to display the connected wifi name, then a small message "put your finger tip to read your BPM", then start displaying the BPM value.

h. From Ubidots help page, I have learnt to download their library and connect my wifi to ESP, then to connect the ESP to the cloud (using Token key), and send the variables to my created dashboard [14] (the wifi code from [9] was also not working, and I was not getting any variables or devices on my dashboard, getting guidance from ubidots was the best).

i. I can access my dashboard through https://stem.ubidots.com/app/dashboards/61c61bf46c9667000a6606ed and copy my token key to use it in my code.

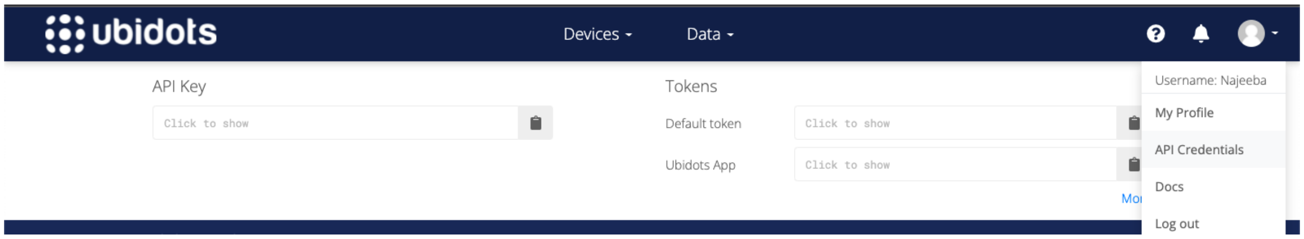


Figure 4 Ubidots start dashboard, copy the token

j. After getting connection to my dashboard, I can see my variables available to be added in widget, and use the suitable type (I set a random function to generate random integer for testing the connection).

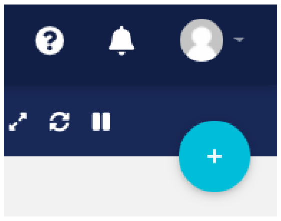


Figure 5 add widget button in Ubidots dashboard

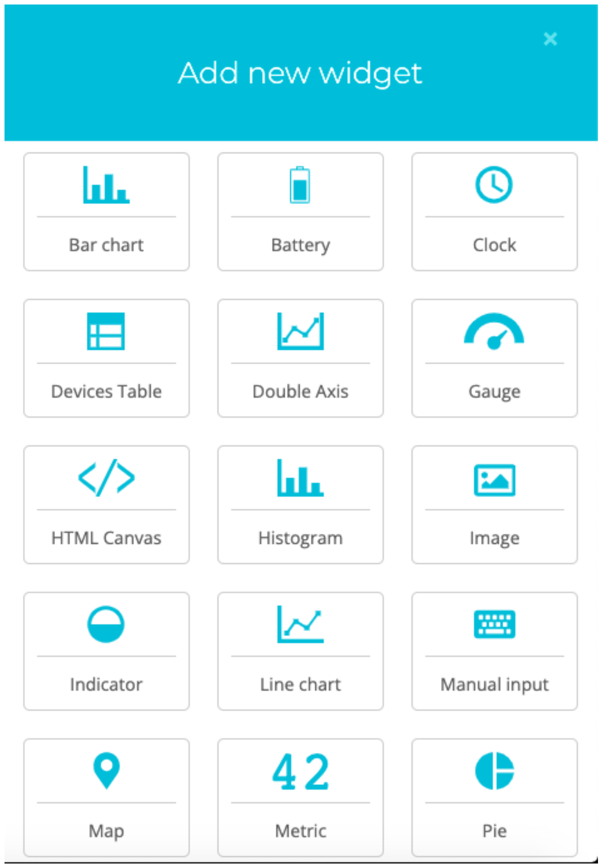


Figure 6 Type of widgets can be chosen for variable displaying

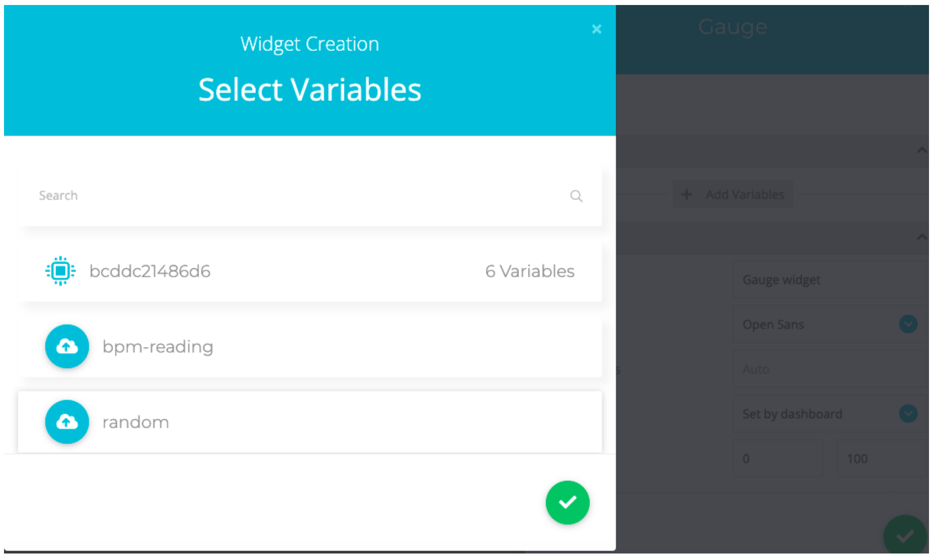


Figure 7 variables selection for the widget creation

k. From [15] and [16], I had guidance to learn how to test my pulse sensor through the ready examples in Arduino IDE from the installed library of the pulse sensor, and correct the values to read from my ESP8266, and find the threshold from the serial plotter generated by that example, in order to use that threshold in changing the value from analog to digital (counting a pulse once reaching this threshold).

l. The ESP pin was reading the values but not accurately, it gives very low reading between 6-24 even when I try changing calculations or variables or using other codes from other sources and use other libraries.

m. I have tested the same code on Arduino, it is reading correctly and I could view the reading on the serial monitor (as shown in the presentation first video).

n. In my final code for the ESP8266, I kept the random function, OLED code and the Ubidots connection and pushing my variable to the cloud, in web and mobile app.

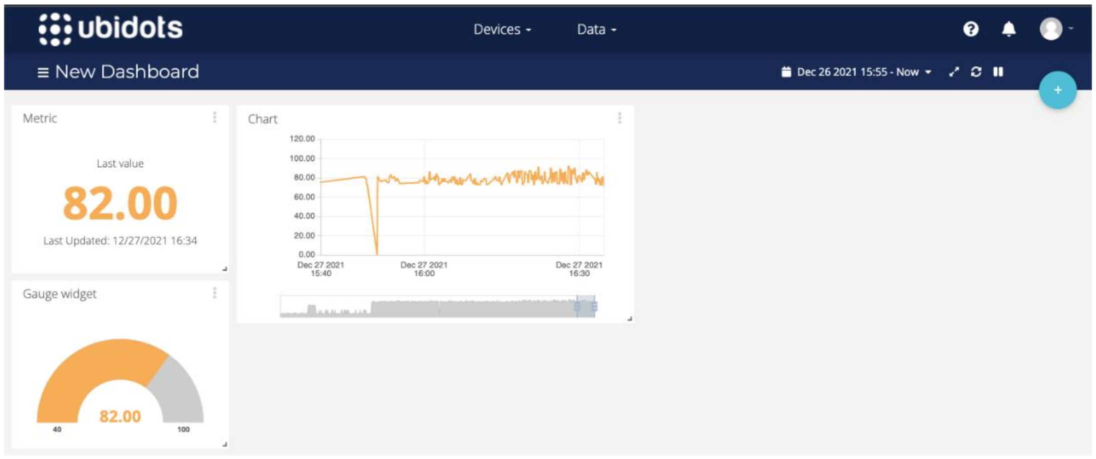


Figure 8 variables in dashboard, live reading from the ESP8266

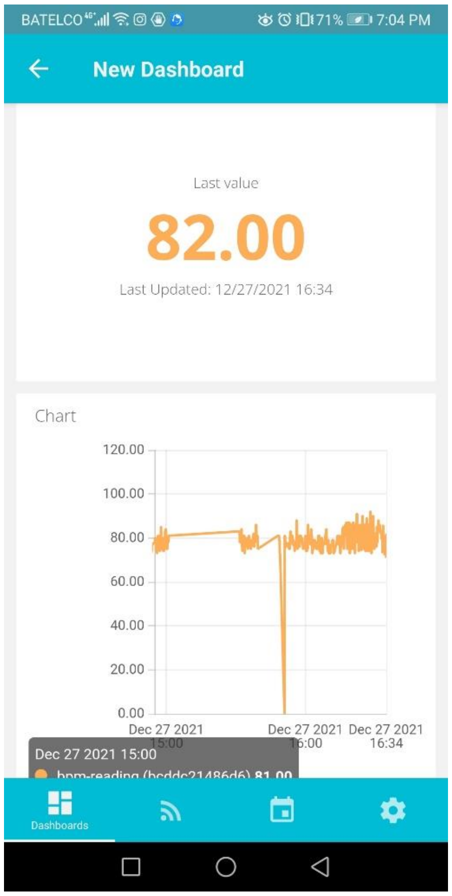


Figure 9 Ubidots on the mobile app

Below will be both codes for Arduino (pulse reading).

```
/*  Getting_BPM_to_Monitor prints the BPM to the Serial Monitor, using the least lines of code and PulseSensor Library.
 *  Tutorial Webpage: https://pulsesensor.com/pages/getting-advanced
 *
--------Use This Sketch To------------------------------------------
1) Displays user's live and changing BPM, Beats Per Minute, in Arduino's native Serial Monitor.
2) Print: "♥  A HeartBeat Happened !" when a beat is detected, live.
2) Learn about using a PulseSensor Library "Object".
4) Blinks LED on PIN 13 with user's Heartbeat.
--------------------------------------------------------------------*/

#define USE_ARDUINO_INTERRUPTS true    // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h>     // Includes the PulseSensorPlayground Library.

//  Variables
const int PulseWire = 0;       // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
const int LED13 = 13;          // The on-board Arduino LED, close to PIN 13.
int Threshold = 550;           // Determine which Signal to "count as a beat" and which to ignore.
                               // Use the "Gettting Started Project" to fine-tune Threshold Value beyond default setting.
                               // Otherwise leave the default "550" value.

PulseSensorPlayground pulseSensor;  // Creates an instance of the PulseSensorPlayground object called "pulseSensor"


void setup() {

  Serial.begin(9600);          // For Serial Monitor
  // Configure the PulseSensor object, by assigning our variables to it.
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13);       //auto-magically blink Arduino's LED with heartbeat.
  pulseSensor.setThreshold(Threshold);

  // Double-check the "pulseSensor" object was created and "began" seeing a signal.
   if (pulseSensor.begin()) {
    Serial.println("We created a pulseSensor Object !");  //This prints one time at Arduino power-up,  or on Arduino reset.
  }
}

void loop() {

 int myBPM = pulseSensor.getBeatsPerMinute();  // Calls function on our pulseSensor object that returns BPM as an "int".
                                               // "myBPM" hold this BPM value now.

if (pulseSensor.sawStartOfBeat()) {            // Constantly test to see if "a beat happened".
 Serial.println("♥  A HeartBeat Happened ! "); // If test is "true", print a message "a heartbeat happened".
 Serial.print("BPM: ");                        // Print phrase "BPM: "
 Serial.println(myBPM);                        // Print the value inside of myBPM.
}

  delay(20);                    // considered best practice in a simple sketch.

}
```

Below is ESP8266 code (push the random integer between 70-90 to Ubidots cloud, and display the same on the OLED screen).

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include "Ubidots.h"

//////////////-------- libraries done

unsigned long previousMillisGetHR = 0; //--> will store the last time Millis (to get Heartbeat) was updated.
unsigned long previousMillisHR = 0; //--> will store the last time Millis (to get BPM) was updated.

const long intervalGetHR = 10; //--> Interval for reading heart rate (Heartbeat) = 10ms.
const long intervalHR = 10000; //--> Interval for obtaining the BPM value based on the sample is 10 seconds.

const int PulseSensorHRWire = A0; //--> PulseSensor connected to ANALOG PIN 0 (A0 / ADC 0).
const int LED_D1 = D1; //--> LED to detect when the heart is beating. The LED is connected to PIN D1 (GPIO5) on the NodeMCU ESP12E.
int Threshold = 500; //--> Determine which Signal to "count as a beat" and which to ignore.

int cntHB = 0; //--> Variable for counting the number of heartbeats.
boolean ThresholdStat = true; //--> Variable for triggers in calculating heartbeats.
int BPMval = 0; //--> Variable to hold the result of heartbeats calculation.

//////////////-------- needed variables created

const char* UBIDOTS_TOKEN = "BBFF-MFMDcIAYhtzExe60yX8KXFJdltoYUr";  // Put here your Ubidots TOKEN
const char* WIFI_SSID = "xxxxxx"; // Put here your Wi-Fi SSID
const char* WIFI_PASS = "xxxxxx"; // Put here your Wi-Fi password

Ubidots ubidots(UBIDOTS_TOKEN, UBI_HTTP); // push the details to ubidots to establish connection with the cloud

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET     LED_BUILTIN //4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); // defining values to be used by the OLED screen


void setup() // the setup function to run once
{
  Serial.begin(115200); // WIFI Serial
  ubidots.wifiConnect(WIFI_SSID, WIFI_PASS);

   Serial.begin(9600); // display and sensor serial

  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS))
  {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
 // delay(2000); // Pause for 2 seconds

  // Clear the buffer
  display.clearDisplay();

  WIFI3();      // function to print WIFI name for test
  BPM();      // function to print "BPM" for test

  // Invert and restore display, pausing in-between
  display.invertDisplay(true);
  delay(500);
  display.invertDisplay(false);
  delay(500);

////////////-------- for serial monitor
  Serial.println();
  Serial.println("Please wait few seconds to get the BPM Value");
////////////--------

}


void printOLED(){ // created for test only

  display.clearDisplay(); // Clear the display buffer
  display.setCursor(10, 0);
  display.setTextSize(2);
  display.println(F("BPM <3 = "));
  display.println(); ///////
  display.println(BPMval); ///////
  display.display(); // Show the display buffer on the screen
//  delay(2000);

}


void loop() {

/////------
//GetHeartRate(); // no need to call this function if we use random function
//////////--------
BPMval = 60 + random(0, 9) + random(1, 15);/////////// the random variable
//printOLED();

//////////------------

 float value1 = BPMval; //= random(0, 9) * 10;
  ubidots.add("BPM reading", value1);  // Change for your variable name /// this function reads the variable to be sent to ubidots


    display.clearDisplay(); // Clear the display buffer
//  Serial.println(F("BPM Serial ")); ///////
//  Serial.println(BPMval); /////// BPM value
  display.setCursor(10, 0);
  display.setTextSize(2);
  display.println(F("BPM <3 = "));
  display.println(); ///////
  display.println(BPMval); ///////
  display.display(); // Show the display buffer on the screen


  bool bufferSent = false;
  bufferSent = ubidots.send();  // Will send data to a device label that matches the device Id > to ubidots

  if (bufferSent) {
    // Do something if values were sent properly
    Serial.println("Values sent by the device");
  }
//////

//  delay(500);
  }


  void WIFI3(void) /////////// ok
  {
  display.clearDisplay();
  display.setTextSize(1); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(10, 0);
  display.println("WiFi name is: ");
  display.println(WIFI_SSID);
  display.display();      // Show initial text
  delay(1000);
}

void BPM(void) { /////////// ok
  display.clearDisplay();

 display.setTextSize(1.5); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("Put your finger tip to read your BPM "));
  display.display();      // Show initial text
  delay(2000);
}


// This subroutine is for reading the heart rate and calculating it to get the BPM value.
// not the same code used in Arduino, as I was testing several codes
// To get a BPM value based on a heart rate reading for 10 seconds.
void GetHeartRate() {
  //----------------------------------------Process of reading heart rate.
  unsigned long currentMillisGetHR = millis();

  if (currentMillisGetHR - previousMillisGetHR >= intervalGetHR) {
    previousMillisGetHR = currentMillisGetHR;

    int PulseSensorHRVal = analogRead(PulseSensorHRWire);

    if (PulseSensorHRVal > Threshold && ThresholdStat == true) {
      cntHB++;
      ThresholdStat = false;
      digitalWrite(LED_D1,HIGH);
    }

    if (PulseSensorHRVal < Threshold) {
      ThresholdStat = true;
      digitalWrite(LED_D1,LOW);
    }
  }

  //----------------------------------------The process for getting the BPM value.
  unsigned long currentMillisHR = millis();

  if (currentMillisHR - previousMillisHR >= intervalHR) {
    previousMillisHR = currentMillisHR;
    BPMval = cntHB * 6; //--> The taken heart rate is for 10 seconds. So to get the BPM value, the total heart rate in 10 seconds x 6.
    Serial.print("BPM : ");
    Serial.println(BPMval);

    cntHB = 0;
  }
  //----------------------------------------
}
```

In the end, we could see live and in the demonstration through the presentation, that the value has been sent to the OLED and Ubidots at the same time, with a 3-4 seconds delay for confirmation that the cloud has received the value.

Conclusion:

In this project, I have designed and implemented a remote real-time heart rate monitoring system based on HIoT. This project can be improved by using the code to be connected through Arduino with embedded WIFI, or find solution for ESP8266 with analog sensor. Also, could be improved by implementing other sensors for patient measurements, and send to the cloud. Overall, I have gained a good knowledge about programming microcontrollers for IoT applications, which can be done for many different applications other than healthcare sector.

References:

[1] Dhanvijay, M.M. and Patil, S.C. (2019). Internet of Things: A survey of enabling technologies in healthcare and its applications. In: Computer Networks, 153, pp.113-131.
[2] Habibzadeh, H., Dinesh, K., Shishvan, O.R., Boggio-Dandry, A., Sharma, G. and Soyata, T. (2020).A Survey of Healthcare Internet-of-Things (HIoT): A Clinical Perspective.In:IEEE Internet of Things Journal, 7(1), pp.53–71.
[3] 10 Best Medication Tracker Apps (asbestos.com)
[4] The Anatomy of Smart Pharma: IoT Connected NFC Tags for Patient Engagement & Protection | NXP Semiconductors
[5] Heart rate: What's normal? - Mayo Clinic
[6] What Your Child’s Heart Rate and Other Vital Signs Tell You (webmd.com)
[7] Getting Started w/ NodeMCU ESP8266 on Arduino IDE - Arduino Project Hub
[8] Pulse Sensor and Arduino - Interfacing (circuitstoday.com)
[9] https://how2electronics.com/iot-mqtt-based-heart-rate-monitor-using-esp8266-arduino/
[10] https://www.arduino.cc/en/donate/
[11] https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/
[12] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html
[13] https://cityos-air.readme.io/docs/1-mac-os-usb-drivers-for-nodemcu
[14] https://help.ubidots.com/en/articles/513321-connect-an-esp8266-stand-alone-to-ubidots-over-http
[15] Getting (Calculating) BPM: – World Famous Electronics llc. (pulsesensor.com)
[16] The "Getting BPM To Serial Monitor" Sketch - YouTube