IoT Based Heart Monitoring System Using ECG

Karingala Suraj Raj - 180102026

November 2020

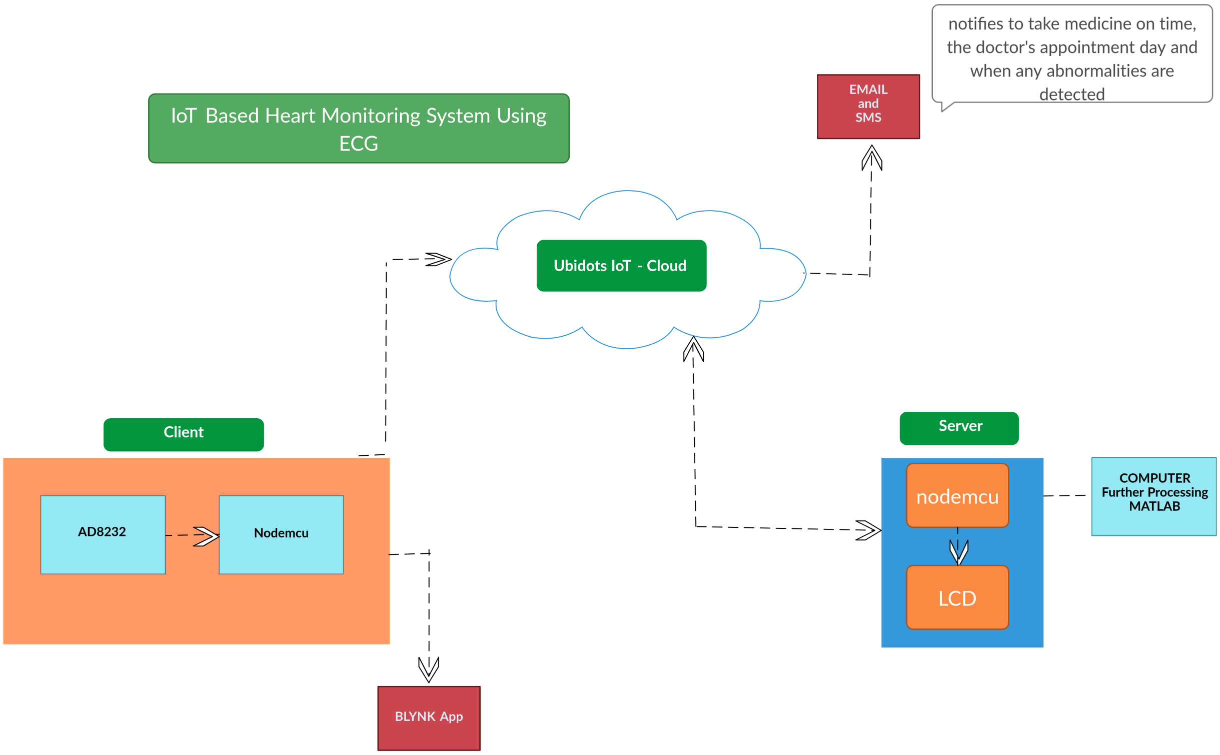
1 Main Objective

In the present world where people have become so busy with their work and with increased pollution, irregular eating habits most of the people are neglecting their health and have no time for health checkups.This could lead to serious health issues.So, the main objective of this project is to continuously monitor the heart of the people especially old people who either live in remote areas or have no one to look after them. The traditional 12 electrode ECG test is expensive and not portable, this project aims at providing cheap portable and continuous monitoring of heart without compromising much on the results. The data collected via sensors is sent to the cloud so that the healthcare personnel can access and analyse the data and can call the person for a checkup if they find any abnormalities.

2 Implemented Attributes

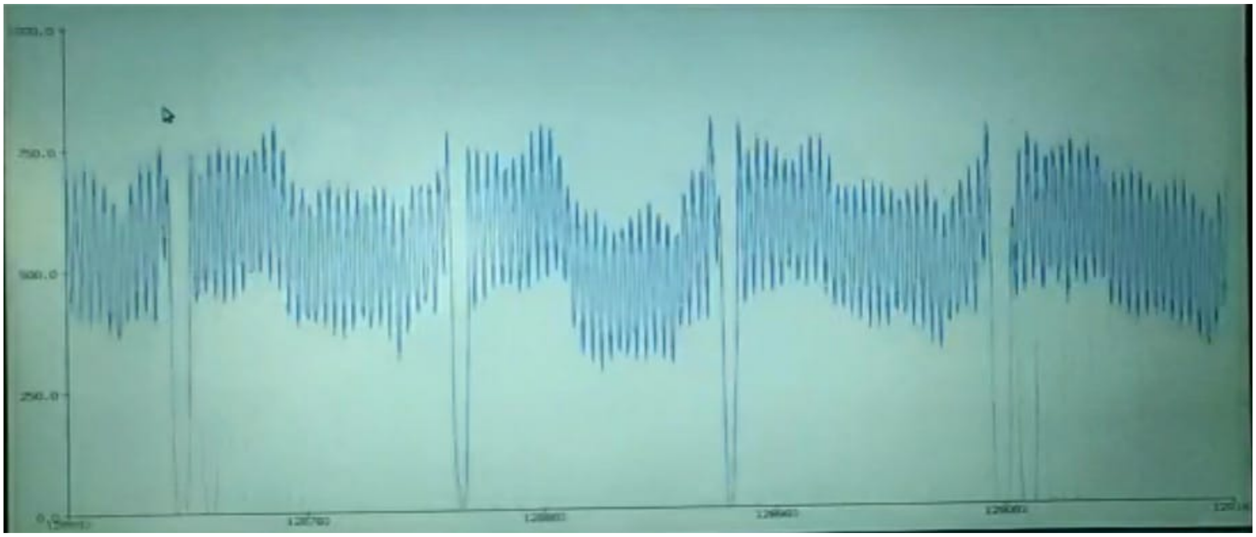
- AD8232 ECG Sensor is used to get the ecg data from the patient via 3 electrodes and to use it for analysis.
- Used the Blynk mobile application to display the ecg signal and BPM of the patient.
- Used Ubidots, a IOT platform as a cloud server to collect data from nodemcu over internet and display the information on its dashboard and also provide data for further analysis through MQTT protocol.
- An alert email and sms is sent to the patient and doctor from the ubidots cloud whenever their is any abnormality found or to remind the doctor's appointment or to remind the patient to take medicine on time.
- An LCD display displays the patients heart status regularly at server end.

3 Configuration Diagram


IoT Based Heart Monitoring System Using ECG
notifies to take medicine on time, the doctor's appointment day and when any abnormalities are detected
EMAIL and SMS
Ubidots IoT - Cloud
Client
AD8232
Nodemcu
BLYNK App
Server
nodemcu
LCD
COMPUTER Further Processing MATLAB

4 Sample Outputs

- The real-time raw data BPM will be calculated continuously and displayed.
- On pressing the Detailed analysis signal, few samples will get collected analysed and resulting BPM would be displayed in the app.



Sample output of the ECG Signal plotted on Serial Plotter.

5 Codes

The codes are attached with the document.

Code for Client (NodeMCU) :

```
//CODE FOR CLIENT NODEMCU

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define BLYNK_PRINT Serial    // Comment this out to disable prints and save space
#include <SPI.h>
#include <BlynkSimpleEsp8266.h>

#define WIFISSID "Karingala"                               // Put your WifiSSID here
#define PASSWORD "karingala@3579"                                // Put your wifi password here
#define TOKEN "BBFF-kfFt6GXIEVhTrDQeK3yqdzxJJUSqOX"          // Put your Ubidots' TOKEN
#define MQTT_CLIENT_NAME "myecgsensor"                         // MQTT client Name, please enter your own 8-12 alphanumeric character ASCII string;
                                                           //it should be a random and unique ascii string and different from all other devices
char auth[] = "I2q95uptbB_wj5dXURbpagVw6yWmRSzd"; //Enter the Auth code which was send by Blink

#define VARIABLE_LABEL "ECG_graph" // Assigning the variable label, this will automatically create a variable in UBIDOTS Website
#define VARIABLE_LABEL_1 "BPM_graph" // Assigning the variable label, this will automatically create a variable in UBIDOTS Website
#define DEVICE_LABEL "ECG_Monitor" // Assigning the device label as created in UBIDOTS Website


#define SENSOR A0 // Set the A0 as SENSOR

char mqttBroker[]  = "industrial.api.ubidots.com";
char payload[100];
char topic[150];   //payload and topic are for ECG_graph payload1 and topic1 are for BPM_graph
char payload1[100];
char topic1[150];
// Space to store values to send
char str_sensor[10];
char str_sensor1[10];

int bpm=0;
unsigned long oldMillis=0;
unsigned long newMillis=0;
long total=0;
long count=0;
long BPM=0;
float myecg=0.0;

WiFiClient ubidots;
PubSubClient client(ubidots);

void callback(char* topic, byte* payload, unsigned int length) {
  char p[length + 1];
  memcpy(p, payload, length);
  p[length] = NULL;
  Serial.write(payload, length);
  Serial.println(topic);
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");

    // Attemp to connect
    if (client.connect(MQTT_CLIENT_NAME, TOKEN, "")) {
      Serial.println("Connected");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 2 seconds");
      // Wait 2 seconds before retrying
      delay(2000);
    }
  }
}

BlynkTimer timer;
void sendSensor()
{


  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V5, myecg);  //V5 is a virtual pin for ecg assigned in BLYNK app
  Blynk.virtualWrite(V0, BPM);    //V0 is a virtual pin for BPM assigned in BLYNK app
}


void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFISSID, PASSWORD);
  // Assign the pin as INPUT
  pinMode(SENSOR, INPUT);

  Serial.println();
  Serial.print("Waiting for WiFi...");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println("");
  Serial.println("WiFi Connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqttBroker, 1883);
  client.setCallback(callback);
  newMillis=millis();
  Blynk.begin(auth, WIFISSID, PASSWORD);



  // Setup a function to be called every 10 milliseconds
  timer.setInterval(10L, sendSensor);

}

void loop() {
  if (!client.connected()) {
    reconnect();
  }

  sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
  sprintf(topic1, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
  sprintf(payload, "%s", ""); // Cleans the payload
  sprintf(payload, "{\"%s\":", VARIABLE_LABEL); // Adds the variable label
  sprintf(payload1, "%s", ""); // Cleans the payload
  sprintf(payload1, "{\"%s\":", VARIABLE_LABEL_1); // Adds the variable label


  myecg = analogRead(SENSOR);
  Serial.println(myecg);

  //code to calculate BPM
  if(myecg>500.0){
    oldMillis=newMillis;
    newMillis=millis();
    long diff=(newMillis-oldMillis);
    bpm=60000/diff;
    total+=bpm;
  }

  count++;
  if(count==500){
  BPM=total/500;
  Serial.println(BPM);
  count=0;
  total=0;
  }

  /* 4 is mininum width, 2 is precision; float value is copied onto str_sensor*/
  dtostrf(myecg, 4, 2, str_sensor);
  dtostrf(BPM, 2, 1, str_sensor1);
  sprintf(payload, "%s {\"value\": %s}}", payload, str_sensor); // Adds the value
  sprintf(payload1, "%s {\"value\": %s}}", payload1, str_sensor1); // Adds the value
  Serial.println("Publishing data to Ubidots Cloud");
  client.publish(topic, payload);
  client.publish(topic1, payload1);
  client.loop();
  delay(10);
  Blynk.run(); // Initiates Blynk
  timer.run(); // Initiates SimpleTimer
}
```

Code for Server (NodeMCU) :

```
//CODE FOR SERVER SIDE NODEMCU

#include "UbidotsESPMQTT.h"
#define BLYNK_PRINT Serial
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <LinkedList.h>
// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char auth[] = "I2q95uptbB_wj5dXURbpagVw6yWmRSzd"; //Enter the Auth code which was send by Blynk to your registered email id
int bval=0;

#define TOKEN "BBFF-i0MP7CIRKgFSC1vRMnp7g2cyK5DHsF" // Your Ubidots TOKEN
#define WIFINAME "Karingala" //Your SSID
#define WIFIPASS "karingala@3579" // Your Wifi Pass
#define VARIABLE_LABEL "ecg_graph" // Assinging the variable label
#define DEVICE_LABEL "ECG_Monitor" // Assinging the device label

Ubidots client(TOKEN);


int bpm=0;
unsigned long oldMillis=0;
unsigned long newMillis=0;
unsigned long startMillis=0;
long total=0;
long BPM=0;
String s="";
int a;
```

```

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i=0;i<length;i++) {
    s+=(char)payload[i];
    Serial.print((char)payload[i]);


  }
  Serial.println();
   a=s.toInt();
   s="";
  //Serial.println(a);
}

BLYNK_WRITE(V1){
  int pinvalue = param.asInt();
  Serial.println(pinvalue); //pinvalue is the instruction that we are getting from a button in BLYNK app
  bval=pinvalue;
}

BlynkTimer timer;
void sendSensor()
{


  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V2, BPM); //V2 is a virtual pin assigned in BLYNK app to show BPM after detailed analysis in a gauge variable
}


void setup() {
    // initialize the LCD
  lcd.begin();

  // Turn on the blacklight and print a message.
  lcd.backlight();
  lcd.clear();
  // Printing welcome message
  lcd.print("Bonjoure!!");


  // put your setup code here, to run once:
  client.ubidotsSetBroker("industrial.api.ubidots.com"); // Sets the broker properly for the Industrial account
  client.setDebug(false); // Pass a true or false bool value to activate debug messages
  Serial.begin(115200);
  client.wifiConnection(WIFINAME, WIFIPASS);
  client.begin(callback);
  client.ubidotsSubscribe(DEVICE_LABEL, VARIABLE_LABEL); //Insert the dataSource and Variable's Labels
  Blynk.begin(auth, WIFINAME, WIFIPASS);
  lcd.clear();
  newMillis=millis();
  startMillis=millis();
  // Setup a function to be called every 2 seconds
  timer.setInterval(2000L, sendSensor);
  }

int count=0;
int check=0;
int result=0;
int flag=0;
int i=0;
float mylist[200]; // A list to store ecg values that could be sent for further detailed analysis


unsigned long DateMillis=0; //give the date of the arrival of doctor here in millis
unsigned long TimeMillis=0; //give the time when patient has to take medicine here in millis
int doc=0;
int med=0;

//code for doctor arrival
void  doctorarrival(){
  unsigned long cur=millis();
  if(DateMillis-cur==0){
    doc=1;
    return;
  }
}

//code for medicine time
void medicinetime(){
  unsigned long cur=millis();
  if(TimeMillis-cur==0){
    med=1;
    return;
  }
}
void loop() {
  // put your main code here, to run repeatedly:
  doctorarrival();
  medicinetime();
  if(!client.connected()){
      client.reconnect();
      if(bval==1){
      client.ubidotsSubscribe(DEVICE_LABEL, VARIABLE_LABEL); //Insert the dataSource and Variable's Labels
      mylist[i]=a;
      i++;
      if(i>=200){
        bval=0;
        Serial.println(bval);
        check=1; //A flag to tell the compiler to send the list for detailed analysis
      }
      }
      }
  client.loop();
  Blynk.run(); // Initiates Blynk
  //code to display the patient's on lcd display when the results are being computed.
  if(bval==1 || check==1){
      lcd.setCursor(0,0);
      lcd.print("Monitoring......");
      delay(4000);
      result=1;
      bval=0;
      check=0;
    }
  // For simplicity i am applyong the same formula as used for calculating real time BPM for raw data on the list of collected ecg values
  //we can send this list to MATLAB for further processing and come back to nodemcu with the processed results.
  if(check==1){
    for(int j=0;i<200;i++){
      if(mylist[i]>500.0){
        oldMillis=newMillis;
        newMillis=millis();
        long diff=(newMillis-oldMillis);
        bpm=60000/diff;
        total+=bpm;
        count++;

      }
     }
      BPM=total/count;
      Serial.println(BPM);
      count=0;
      total=0;
      check=0;
      result=1;
   }

//code to display the final results of patient's heart condition on lcd display
 if(result==1 && flag==0){
  lcd.clear();
  lcd.print("BPM : ");
  lcd.print(BPM);
  //There could be many conditions applied afrom analysis but for simplicity im just considering BPM less than and greater then 100
  if(BPM<=100){
    lcd.setCursor(0,1);
    lcd.print("Status : OK");
  }
  else{
    lcd.setCursor(0,1);
    lcd.print("Status : CONSULT");
  }
  flag++;
 }
  timer.run();
  }
```

6 User Manual

- Connect nodemcu to the LCD display at the server side and Connect nodemcu to the AD8232 sensor at the client side.
- Place the electrodes in a triangular formation and make sure you wear some footwear while performing the experiment.
- Make sure that you are not charging your laptop while performing the experiment as it could cause 50-60Hz AC noise. If possible try to power the client side setup using a battery.
- Adjust the brightness of the LCD display by rotating the potentiometer attached to the I2C chip with a screwdriver.
- Create an account on Ubidots Website and you will be mailed the Ubidots token to your email id which you need to give in the code.
- Install Blynk app from playstore and add gauges,buttons and lineplots as mentioned in the pic(Click Here).
- Finally upload the corresponding codes to the corresponding Nodemcus and install the required libraries if they are not installed on your PC.
- To get sms and email notifications add events in the data section of ubidots website.

7 Video Demo and other links

- Drive link to the video demo - Click Here
- Code for Server Nodemcu - Click Here
- Code for Client Nodemcu - Click Here