DESIGN AND DEVELOPMENT OF CONTINUOUS POSITIVE AIRWAY PRESSURE USING SMART-PHONE

A PROJECT REPORT

Submitted by

ANANDHI RAGHURAMAN

ASMITHA A

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND INSTRUMENTATION ENGINEERING

EASWARI ENGINEERING COLLEGE

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**DESIGN AND DEVELOPMENT OF CONTINUOUS POSITIVE AIRWAY PRESSURE USING SMART-PHONE**" is the bonafide work of " **ANANDHI RAGHURAMAN** and **ASMITHA A** " who carried out the project work under my supervision.

SIGNATURE

DR. S. NAGARAJAN

HEAD OF THE DEPARTMENT

Department of Electronics and

Instrumentation Engineering,

Easwari Engineering College,

Ramapuram, Chennai-89 .

SIGNATURE

MR. T. SURENDRAN

SUPERVISOR

Assistant Professor,

Department of Electronics and

Instrumentation Engineering,

Easwari Engineering College,

Ramapuram, Chennai-89.

Appendices :

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include "DHT.h"



LiquidCrystal lcd( 12, 11, 10, 9, 8, 7);
SoftwareSerial blue_tx_rx (2, 3);
define ard_node blue_tx_rx
#define splash splash1
#define DHTTYPE DHT11

#define resp A0
#define DHTPIN A1
#define ky1 4
#define ky2 5
#define compres 6

DHT dht(DHTPIN, DHTTYPE);

int a, b;
int H_B, SPO_2;
int hum;
float temp;
String IncomingData = "";


String outB = "";

String modefn = "";




int res_r, resfn;
int res_r_t = 700;
float res_tim = 0.0;
```

```
int C_Status;
int C_tim = 10;
float cc_tim = C_tim;

int  k1_r, k2_r;
int C_auto;


int DispDelay;



void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  blue_tx_rx.begin(9600);
  dht.begin();
  pinMode(resp, INPUT);
  pinMode(ky1, INPUT_PULLUP);
  pinMode(ky2, INPUT_PULLUP);
  pinMode(compres, OUTPUT);
  digitalWrite(compres, LOW);
  LcDSet();
  lcd.clear();

}


void LcDSet() {


  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.clear();

  splash(0, "CPAP");
  splash(1, "MACHINE");
  delay(1000);

}
```

```
void loop() {

  setDisp();
  getArd();
  getDht();
  k1_r = digitalRead(ky1);
  k2_r = digitalRead(ky2);
  if (k1_r == 0 and C_auto == 0) {
    splash(1, "Auto Mode OFF");
    delay(100);
    C_auto = 1;

  }

  else if (k1_r == 0 and C_auto == 1) {
    splash(1, "Auto Mode ON");
    resfn = 0;
    res_tim = 0;
    delay(100);
    C_auto = 0;

  }

  if (C_auto == 0) {
    getres();
    outB = "T : ";
    outB += String(temp);
    outB += "  H : ";
    outB += String(hum);
      outB += "% HB : ";
    outB += String(H_B);
    outB += "  SPO2 : ";
    outB += String(SPO_2);


    outB += "%  R : ";
    outB += String(resfn);
```

```
  outB += "  ||  ";
  outB += " Mode : ";
  outB += "Auto ";
  outB += "  ||  ";
  outB += "Compressor Status = ";
  if (C_Status == 1) {
    digitalWrite(compres, HIGH);
    outB += "ON";

  }
  Else

{
    digitalWrite(compres, LOW);
    outB += "OFF";
  }

 }
 else {

  if (k2_r == 0 and C_Status == 0) {
    //      digitalWrite(led, HIGH);

    splash(1, "C ON");
    delay(100);
    C_Status = 1;
  }

  else if (k2_r == 0 and C_Status == 1) {


    splash(1, "C OFF");
    //      digitalWrite(led, LOW);
    lcd.setCursor(0, 0);
    lcd.print("                          ");
    delay(100);
    C_Status = 0;
  }

  outB = "T : ";
```

```
  outB += String(temp);
  outB += "  H : ";
  outB += String(hum);

  outB += "% HB : ";
  outB += String(H_B);
  outB += "  SPO2 : ";
  outB += String(SPO_2);


  outB += "%  R : ";
  outB += String(resfn);
  outB += "  ||  ";
  outB += " Mode : ";
  outB += "Manual ";
  outB += "  ||  ";
  outB += "Compressor Status = ";
  if (C_Status == 1) {
    outB += "ON";
  }
  else
{
    outB += "OFF";
  }
 }

 if (C_Status == 1) {
  digitalWrite(compres, HIGH);

 }
 Else
 {
  digitalWrite(compres, LOW);

 }
 Serial.println(outB);
 blue_tx_rx.println(outB);
 delay(300);
}
```

```
void getres()
{

 res_tim += 0.40;


 res_r = analogRead(resp);
 Serial.println(res_r);
 if (C_Status == 0) {
  if (res_r > res_r_t) {
   resfn++;
   splash(0, "Snoring");
   splash(1, "Detected");
  }
 }

 if (res_tim > 8 and resfn > 2 and C_Status == 0) {
  Serial.println("Comp ON");
  C_Status = 1;
  splash(1, "C ON");
  resfn = 0;
  res_tim = 0;
 }

 else if (res_tim > C_tim and C_Status == 1) {
  C_Status = 0;
  splash(1, "C OFF");
  lcd.setCursor(0, 0);
  lcd.print("                ");
  Serial.println("Comp OFF");
  resfn = 0;
  res_tim = 0;
 }

 else if (res_tim > 10) {
  C_Status = 0;

  Serial.println("Normal");
  resfn = 0;
  res_tim = 0;
```

```
  }

}


void getArd() {
  String SS = ""; String SSs = "";
  while (ard_node.available() > 0) {

    char S = ard_node.read();
    SS += S;

    delay(10);

  }
  if (SS.length() > 0) {
    Serial.println(SS);
    String sys = getSplitValue(SS , ',', 0);
    String dia =  getSplitValue(SS , ',', 1);

    H_B = sys.toInt();
    SPO_2 = dia.toInt();


    ard_node.flush();

  }
}



void getDht() {

  int humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  /* Serial.print(humidity, 1);
    Serial.print("\t\t");
    Serial.println(temperature, 1);*/
  temp = temperature;
  hum = humidity;
```

```
}


void setDisp() {
  DispDelay++;
  if (DispDelay >= 0 and DispDelay < 15) {
    display_ard(H_B, SPO_2);

  }
  if (DispDelay >= 15 and DispDelay < 30) {
    display_T(temp, hum);

  }
  else  if (DispDelay >= 30 ) {
    DispDelay = 0;
  }

  if (C_Status == 0) {

    lcd.setCursor(0, 1);
    lcd.print("S :               ");
    lcd.setCursor(3, 1);
    lcd.print(resfn);
    lcd.setCursor(12, 1);
    lcd.print(res_tim);
    cc_tim = C_tim;
  }
  else if (C_Status == 1 and C_auto == 1 ) {
    lcd.setCursor(0, 0);
    lcd.print("Air Flow ON              ");

  }
  else
{
    lcd.setCursor(0, 0);
    lcd.print("Air Flow ON              ");
    lcd.setCursor(0, 1);
    lcd.print("                         ");
    lcd.setCursor(12, 1);
    lcd.print(cc_tim -= 0.38);
```

```
  }

}
void display_ard(int val1 , int val2) {

  lcd.setCursor(0, 0);
  lcd.print("HB:               ");
  lcd.setCursor(3, 0);
  lcd.print(val1);
  lcd.setCursor(7, 0);
  lcd.print("SPo2:               ");
  lcd.setCursor(12, 0);
  lcd.print(val2);
  lcd.setCursor(15, 0);
  lcd.print("% ");

}
void display_T(float val1 , int val2) {


  lcd.setCursor(0, 0);
  lcd.print("T:               ");
  lcd.setCursor(3, 0);
  lcd.print(val1, 1);
  lcd.setCursor(8, 0);
  lcd.print("H:               ");
  lcd.setCursor(11, 0);
  lcd.print(val2);
  lcd.setCursor(14, 0);
  lcd.print("% ");
}
```

Arduino Code

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <SoftwareSerial.h>
SoftwareSerial ard_node(2,3);


#define REPORTING_PERIOD_MS     1000

PulseOximeter pox;

uint32_t tsLastReport = 0;

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


void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  ard_node.begin(9600);
  Serial.print("Initializing pulse oximeter..");


  // Initialize the PulseOximeter instance
  // Failures are generally due to an improper I2C wiring, missing power supply
  // or wrong target chip
  if (!pox.begin()) {
    Serial.println("FAILED");
    for (;;);
  } else {
    Serial.println("SUCCESS");
  }

  // The default current for the IR LED is 50mA and it could be changed
  //   by uncommenting the following line. Check MAX30100_Registers.h for all
the
```

```
  //   available options.
//   pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

  // Register a callback for the beat detection
  pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop() {
  // put your main code here, to run repeatedly:
  pox.update();

  // Asynchronously dump heart rate and oxidation levels to the serial
  // For both, a value of 0 means "invalid"
  if (millis() - tsLastReport > REPORTING_PERIOD_MS) {

    Serial.print("Heart rate:");
    Serial.print(pox.getHeartRate());
    Serial.print("bpm / SpO2:");
    Serial.print(pox.getSpO2());
    Serial.println ("%");



    ard_node.print(pox.getHeartRate());
    ard_node.print(",");
    ard_node.print(pox.getSpO2());
    ard_node.print(",");



    tsLastReport = millis();
  }
}
```