DESIGN OF WIRELESS POWER TRANSFER FOR HEART ASSIST DEVICES

APPENDICES

Microcontroller Code

```
#include <SoftwareSerial.h>
SoftwareSerial esp8266Module(6, 7);
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
String network = "XXXXXX";
String password = "XXXXXXXX";
#define IP "184.106.153.149"
String GET = "GET /update?api_key=XXXXXXXXXX";
#define pul A0   //---------------------------------For Flow Sensor
byte sensorInterrupt = 0;  // 0 = digital pin 2
byte sensorPin       = 2;
float calibrationFactor = 4.5;
volatile byte pulseCount;

float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
unsigned long oldTime;  //---------------------------------For Pulse Sensor
double alpha = 0.9;
```

```
double change = 0.0;
double val;
int sec1;
void setup() {
  // put your setup code here, to run once:
  lcd.begin(16, 2 );
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
  totalMilliLitres  = 0;
  oldTime           = 0;
  esp8266Module.begin(115200);
  setupEsp8266();
}
void loop() {
  sec1++;
  static double oldValue = 0;
  static double oldChange = 0;
  int rawValue = analogRead (pul);                    // Reading the sensors
values
  val = alpha * oldValue + (1 - alpha) * rawValue;
```

```
val = (val / 3) * 2;
if ((millis() - oldTime) > 1000)   // Only process counters once per second
{
  detachInterrupt(sensorInterrupt);
  flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor;
  oldTime = millis();
  flowMilliLitres = (flowRate / 60) * 1000;
  totalMilliLitres += flowMilliLitres;
  unsigned int frac;
  // Serial.print("Flow rate: ");
  Serial.print(int(flowRate));
  // Serial.print(".");
  frac = (flowRate - int(flowRate)) * 10;
  Serial.print(frac, DEC) ;
  Serial.print("L/min  ");
  Serial.print("||");
  Serial.print("  Current Blood Flowing: ");
  Serial.print(flowMilliLitres);
  Serial.print("mL/Sec  ");
  Serial.print("||");
  Serial.print("  Pulse: ");
  Serial.println(val);
  /*   // Print the cumulative total of litres flowed since starting
  Serial.print("  Output Liquid Quantity: ");            // Output separator
  Serial.print(totalMilliLitres);
```

```
    Serial.println("mL");*/
    pulseCount = 0;
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  }
  oldValue = val;
  if (sec1 > 180)
  {
    lcd.clear(); lcd.setCursor(0, 0); lcd.print("------IoT------");
    lcd.setCursor(0, 1);
    Serial.println("IoT Data Sending...");
    lcd.print("--Data_sending--"); send2(); lcd.clear(); sec1 = 0;
  }
}
void send2() {
  updateTemp(String(flowMilliLitres), String(val));
}
void pulseCounter()
{
  pulseCount++;
}
void setupEsp8266()
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("----Waiting-----");
```

```
  lcd.setCursor(0, 1);
  lcd.print("---For a while---");
  // esp8266Module.flush();
  esp8266Module.println(F("AT+RST"));
  delay(7000);
  if (esp8266Module.find("OK"))
  {
    Serial.println("Found OK");
    Serial.println("Changing espmode");
    // esp8266Module.flush();
    changingMode();
    delay(5000);
    //  esp8266Module.flush();
    connectToWiFi(); delay(2000);
    //  mux(); delay(2000);
    // ser();
  }
  else
  {
    Serial.println("OK not found");
  }
}
//-----------------------------------------------------------------
// Following function sets esp8266 to station mode
//-----------------------------------------------------------------
```

```
bool changingMode()
{
  esp8266Module.println(F("AT+CWMODE=1"));
  if (esp8266Module.find("OK"))
  {
    Serial.println("Mode changed");
    return true;
  }
  else if (esp8266Module.find("NO CHANGE")) {
    Serial.println("Already in mode 1");
    return true;
  }
  else
  {
    Serial.println("Error while changing mode");
    return false;
  }
}
//-----------------------------------------------------------------
// Following function connects esp8266 to wifi access point
//-----------------------------------------------------------------
bool connectToWiFi()
{
  Serial.println("inside connectToWiFi");
  String cmd = F("AT+CWJAP=\"");
```

```
cmd += network;
cmd += F("\",\"");
cmd += password;
cmd += F("\"");
esp8266Module.println(cmd);
delay(15000);
if (esp8266Module.find("OK"))
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("-WiFi-Connected-");
  lcd.setCursor(0, 1);
  lcd.print("Network:");
  lcd.setCursor(8, 1);
  lcd.print(network);
  delay(2000);
  lcd.clear();
  Serial.println("Connected to Access Point");
  return true;
}
else
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("-WiFi-Connected-");
```

```
    lcd.setCursor(0, 1);
    lcd.print("Network:");
    lcd.setCursor(8, 1);
    lcd.print(network);
    delay(2000);
    lcd.clear();
    Serial.println("Could not connect to Access Point");
    return false;
  }
}
void updateTemp(String voltage1, String voltage2)
{
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += IP;
  cmd += "\",80";
  esp8266Module.println(cmd);
  delay(5000);
  if (esp8266Module.find("Error")) {
    Serial.println("ERROR while SENDING");
    return;
  }
  cmd = GET + "&field1=" + voltage1 + "&field2=" + voltage2 + "\r\n";
  esp8266Module.print("AT+CIPSEND=");
  esp8266Module.println(cmd.length());
  delay(15000);
```

```
  if (esp8266Module.find(">"))
  {
    esp8266Module.print(cmd);
    Serial.println("Data
sent**********************************************************************
**********************************************************************");
  } else
  {
    esp8266Module.println("AT+CIPCLOSE");
    Serial.println("Connection closed");
  }}
```