APPENDIX A

SOURCE CODE

Code for 2nd Arduino Uno to Control Speed

```
#include  <TimerOne.h>
volatile int i = 0;
volatile boolean zero_cross = 0;
int AC_pin = 3;
int ZERO_CROSS_PIN = 2;
volatile long dim = 0;
int inc = 1;
int freqStep = 75;

void setup() {
  pinMode(AC_pin, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(ZERO_CROSS_PIN),
zero_cross_detect, RISING);
  Timer1.initialize(freqStep);
  Timer1.attachInterrupt(dim_check, freqStep);
  Serial.begin(115200);
}

void zero_cross_detect() {
  zero_cross = true;
  i = 0;
  digitalWrite(AC_pin, LOW);
}

void dim_check() {
  if (zero_cross == true) {
    if (i >= dim) {
      digitalWrite(AC_pin, HIGH);
      i = 0;
      zero_cross = false;
    }
    else {
      i++;
    }
  }
}

void loop() {
  if (Serial.available()) {
    String msg = Serial.readStringUntil('@');
```

```
    if (msg.toInt() > 0) {
      dim = msg.toInt();
    }
    Serial.println(dim);
  }
}
```

Code for 1st Arduino Uno to Process Sensor Values

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include <math.h>
SoftwareSerial SoftSerial(6, 7);
const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
const int CURRENT_IN_PIN = 1;
const int VOLTAGE_IN_PIN = A0;
const int RELAY_IN_PIN = 5;
const int SPEED_IN_PIN = 2; //Analog pin 2
const int SENSITIVITY = 66;
const int OFFSET_VOLTAGE = 503;
double Vpeak = 0;
double Vrms = 0;
double Irms = 0;
double Ipeak = 0;
int rpm = 0;
double pf = 0;
int rawMaxCurrent = 0;
int rawAvgVoltage = 0;
int rawAvgSpeed = 0;
long displayRefreshTime = 0;
int displayMenuNumber = 0;
bool overCurrent = false;
bool overVoltage = false;

void setup()
{
  pinMode(RELAY_IN_PIN, OUTPUT);
  digitalWrite(RELAY_IN_PIN, LOW);

  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Monitor and control");
  lcd.setCursor(0, 1);
  lcd.print(" of Induction Motor");
  delay(500);
  Serial.begin(115200);
```

```
  SoftSerial.begin(115200);
}

void loop()
{
  readRawVoltageAndCurrentAndSpeed();
  calcVoltage();
  calcCurrent();
  calcSpeed();
  calcPf();
  checkOverCurrentAndVoltage();
  lcdDisp();
  updateStatus();
}

void readRawVoltageAndCurrentAndSpeed() {
  long startTime = millis();
  int rawCurrent = 0;
  long voltageCounter = 0;
  long rawVoltageSum = 0;
  int lastSpeedValue = 0;
  int newSpeedValue = 0;
  int speedCounter = 0;
  rawMaxCurrent = 0;
  while (millis() - startTime < 500) {
    rawCurrent = analogRead(CURRENT_IN_PIN);
    if (rawCurrent > rawMaxCurrent) {
      rawMaxCurrent = rawCurrent;
    }
    rawVoltageSum += analogRead(VOLTAGE_IN_PIN);
    voltageCounter++;
    lastSpeedValue = newSpeedValue;
    newSpeedValue = analogRead(SPEED_IN_PIN);
    if (newSpeedValue < 50 && lastSpeedValue > 50) {
      speedCounter++;
    }
  }
  rawAvgVoltage = rawVoltageSum / voltageCounter;
  rawAvgSpeed = speedCounter;
}

void calcSpeed() {
  rpm = rawAvgSpeed * 120;
  Serial.print("Speed=");
  Serial.println(rpm);
  String rpmStr = String(rpm);
```

```
  rpmStr = "Speed" + rpmStr + "rpm!";
  SoftSerial.print (rpmStr);
}

void calcVoltage() {
  Vrms = rawAvgVoltage / 3.69 ;
  Vpeak = Vrms * sqrt(2);
  Serial.print("Vrms=");
  Serial.println(Vrms);
  String VrmsStr = String(Vrms);
  VrmsStr = "Vrms" + VrmsStr + "V!";
  SoftSerial.print (VrmsStr);
  String VpeakStr = String(Vpeak);
  VpeakStr = "Vpeak" + VpeakStr + "V!";
  SoftSerial.print (VpeakStr);
}

void calcCurrent() {
  float equivalentVoltage = (rawMaxCurrent / 1023.0) * 5000; //
Gets you mV
  Ipeak = ((equivalentVoltage - OFFSET_VOLTAGE) / SENSITIVITY) ;
  Irms = Ipeak /  sqrt(2);
  Serial.print("Irms= ");
  Serial.println(Irms);
  String IrmsStr = String(Irms);
  IrmsStr = "Irms" + IrmsStr + "A!";
  SoftSerial.println (IrmsStr);
  String IpeakStr = String(Ipeak);
  IpeakStr = "Ipeak" + IpeakStr + "A!";
  SoftSerial.println (IpeakStr);
}

void checkOverCurrentAndVoltage() {
  if (Vrms > 250) {
    overVoltage = true;
    SoftSerial.print ("Over Voltage!");
  } else {
    overVoltage = false;
  }
  if (Irms > 5) {
    overCurrent = true;
    SoftSerial.print ("Over Current!");
  } else {
    overCurrent = false;
  }
}
```

```
void calcPf() {
  float Ireactive = 2.10;
  Serial.println("Ireactive==");
  Serial.println(Ireactive);
  float I = (Ireactive / Irms);
  Serial.println("Ireactive/Irms==");
  Serial.println(I);
  float Isquare = pow(I, 2);
  Serial.println("Isquare==");
  Serial.println(Isquare);
  pf = sqrt(1 - Isquare);
  Serial.println("pF==");
  Serial.println(pf);
  String pfStr = String(pf);
  pfStr = "PF" + pfStr + "!";
  SoftSerial.print (pfStr);
}

void updateStatus() {
  if (SoftSerial.available()) {
    String msg = SoftSerial.readStringUntil('@');
    Serial.print("message is ");
    Serial.println(msg);
    if (msg.indexOf("OFF") >= 0) {
      digitalWrite(RELAY_IN_PIN, HIGH);
    } else if (msg.indexOf("ON") >= 0) {
      digitalWrite(RELAY_IN_PIN, LOW);
    }
  }
}
void lcdDisp()
{
  if (millis() - displayRefreshTime > 1000) {
    displayRefreshTime = millis();
    lcd.clear();
    if (overVoltage) {
      //Show Over Voltage
      lcd.setCursor(0, 0);
      lcd.print("Over Voltage");
      lcd.setCursor(0, 1);
      lcd.print("Switch off");
      digitalWrite(RELAY_IN_PIN, HIGH);
    }
    else if (overCurrent) {
      //Show Over Current
```

```
      lcd.setCursor(0, 0);
      lcd.print("Over Current");
      lcd.setCursor(0, 1);
      lcd.print("Switch off");
      digitalWrite(RELAY_IN_PIN, HIGH);
    }
    else if (displayMenuNumber == 0) {
      lcd.setCursor(0, 0);
      lcd.print("Ipeak="); lcd.print(Ipeak); lcd.print(" ");
lcd.print("A");
      lcd.setCursor(0, 1);
      lcd.print("Irms="); lcd.print(Irms); lcd.print(" ");
lcd.print("A");
    } else if (displayMenuNumber == 1) {
      lcd.setCursor(0, 0);
      lcd.print("Vpeak="); lcd.print(Vpeak); lcd.print(" ");
lcd.print("V");
      lcd.setCursor(0, 1);
      lcd.print("Vrms="); lcd.print(Vrms); lcd.print(" ");
lcd.print("V");
    } else if (displayMenuNumber == 2) {
      lcd.setCursor(0, 0);
      lcd.print("Speed="); lcd.print(rpm); lcd.print(" ");
lcd.print("rpm");
      lcd.setCursor(0, 1);
      lcd.print("PF="); lcd.print(pf);
    }
    displayMenuNumber++;
    if (displayMenuNumber > 2) {
      displayMenuNumber = 0;
    }
  }
}
```

Code for NodeMcu for Remote Monitor and Control

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WebSocketsServer.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <Hash.h>
#define USE_SERIAL Serial
long oldtime = 0;
ESP8266WiFiMulti WiFiMulti;
ESP8266WebServer server(80);
```

```
WebSocketsServer webSocket = WebSocketsServer(81);

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload,
size_t length) {
    switch(type) {
        case WStype_DISCONNECTED:
            USE_SERIAL.printf("[%u] Disconnected!\n", num);
            break;
        case WStype_CONNECTED: {
            IPAddress ip = webSocket.remoteIP(num);
        }
            break;
        case WStype_TEXT:
            String strPayload = (char*)payload;
            if (strPayload.indexOf("#0") >=0 ){
              Serial.print("OFF@");
            }else if(strPayload.indexOf("#1") >=0 ){
              Serial.print("ON@");
            }else if(strPayload.indexOf("*100") >=0 ){
              Serial.print("1@");
            }else if(strPayload.indexOf("*75") >=0 ){
              Serial.print("40@");
            }else if(strPayload.indexOf("*50") >=0 ){
              Serial.print("68@");
            }else if(strPayload.indexOf("*25") >=0 ){
              Serial.print("98@");
            }else if(strPayload.indexOf("*1") >=0 ){
              Serial.print("127@");
            }
    }
}

void setup() {
    USE_SERIAL.begin(115200);
    USE_SERIAL.println();
    USE_SERIAL.println();
    USE_SERIAL.println();
    for(uint8_t t = 4; t > 0; t--) {
      USE_SERIAL.printf("[SETUP] BOOT WAIT %d...\n", t);
      USE_SERIAL.flush();
      delay(1000);
    }
    pinMode(13 , OUTPUT);
    digitalWrite(13, 1);
    WiFiMulti.addAP("smps", "432FACC99A");
    while(WiFiMulti.run() != WL_CONNECTED) {
```

```
    delay(100);
  }
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);
  if(MDNS.begin("esp8266")) {
    USE_SERIAL.println("MDNS responder started");
    USE_SERIAL.println(WiFi.localIP());
  }
  server.on("/", []() {
    server.send(200, "text/html", " <html> <head> <script> var
connection = new WebSocket('ws://' + location.hostname + ':81/',
['arduino']); connection.onopen = function()
{connection.send('Connect ' + new Date()); }; connection.onerror =
function(error) {console.log('WebSocket Error ', error); };
connection.onmessage = function(e) {console.log('Server: ',
e.data); if (e.data.indexOf('Irms') >= 0) {var ret =
e.data.replace('Irms',''); console.log('Irms:',ret);
document.getElementById('Irms').innerHTML=ret; } else
if(e.data.indexOf('Ipeak') >= 0){var ret =
e.data.replace('Ipeak',''); console.log('Ipeak:',ret);
document.getElementById('Ipeak').innerHTML=ret; } else
if(e.data.indexOf('Vrms') >= 0){var ret =
e.data.replace('Vrms',''); console.log('Vrms:',ret);
document.getElementById('Vrms').innerHTML=ret; } else
if(e.data.indexOf('Vpeak') >= 0){var ret =
e.data.replace('Vpeak',''); console.log('Vpeak:',ret);
document.getElementById('Vpeak').innerHTML=ret; }else
if(e.data.indexOf('Speed') >= 0){var ret =
e.data.replace('Speed',''); console.log('Speed:',ret);
document.getElementById('Speed').innerHTML=ret; }else
if(e.data.indexOf('PF') >= 0){var ret = e.data.replace('PF','');
console.log('PF:',ret);
document.getElementById('powerFactor').innerHTML=ret; } };
function on(){console.log('on'); connection.send('#1'); } function
off(){console.log('off:'); connection.send('#0'); } function
sendSpeed(spd){console.log('speed percentage:' + spd); spd = '*' +
spd; connection.send(spd); } </script> </head> <title>Group
8</title> <body> <!--header start --> <h2>MONITORING AND
CONTROLLING <br /> SINGLE PHASE INDUCTION MOTOR </h2> <h3>Motor
Operation</h3> <button id=\"send\" onclick=\"on();\">ON</button>
<button id=\"send\" onclick=\"off();\">OFF</button> <h3>Data
Readings</h3> <table width=\"283\" border=\"0\" > <tr> <td
width=\"55\">Vrms : </td> <td width=\"73\" id=\"Vrms\"> </td>
<td width=\"10\"> </td> <td width=\"55\">Vpeak : </td> <td
width=\"73\" id=\"Vpeak\"> </td> </tr> <tr> <td>Irms : </td>
<td id=\"Irms\"> </td> <td> </td> <td>Ipeak : </td> <td
```