LAMPIRAN PROGRAM

```
/* 1- PZEM-017 DC Energy Meter */
#include <ModbusMaster.h>
#define MAX485_DE     3
#define MAX485_RE     4
static uint8_t pzemSlaveAddr = 0x01;
static uint16_t NewshuntAddr = 0x0001;
ModbusMaster node;                         /* activate modbus master codes*/
float PZEMVoltage = 0;                     /* Declare value for DC voltage */
float PZEMCurrent = 0;                     /* Declare value for DC current*/
float PZEMPower = 0;                       /* Declare value for DC Power */
float PZEMEnergy = 0;                      /* Declare value for DC Energy */
unsigned long startMillisPZEM;
unsigned long currentMillisPZEM;
const unsigned long periodPZEM = 1000;     // Default 1000 = 1 second
unsigned long startMillisLCD;
unsigned long currentMillisLCD;
const unsigned long periodLCD = 1000;      //Default 1000 = 1 second */
int ResetEnergy = 0;                    /* reset energy function */
unsigned long startMillisEnergy;
unsigned long currentMillisEnergy;
const unsigned long periodEnergy = 1000;   //Default 1000 = 1 second
float PZEMEnergynew, PZEMOffset;
/* 2 - Sensor Kecepatan  */
const byte PulsesPerRevolution = 2;
const unsigned long ZeroTimeout = 100000;
const byte numReadings = 2;
volatile unsigned long LastTimeWeMeasured;
volatile unsigned long PeriodBetweenPulses = ZeroTimeout + 1000;
volatile unsigned long PeriodAverage = ZeroTimeout + 1000;
unsigned long FrequencyRaw;
unsigned long FrequencyReal;
unsigned long RPM;
unsigned int PulseCounter = 1;
unsigned long PeriodSum;
unsigned long LastTimeCycleMeasure = LastTimeWeMeasured;
unsigned long CurrentMicros = micros();
unsigned int AmountOfReadings = 1;
unsigned int ZeroDebouncingExtra;
unsigned long readings[numReadings];
unsigned long readIndex;
unsigned long total;
unsigned long average;
const int relayPin1 = 6;  // Pin yang terhubung ke relay 1
const int relayPin2 = 7;  // Pin yang terhubung ke relay 2
const int relayPin3 = 8;  // Pin yang terhubung ke relay 3
const int relayPin4 = 9;  // Pin yang terhubung ke relay 4
const int relayPin5 = 10;  // Pin yang terhubung ke relay 5
const int relayPin6 = 11;  // Pin yang terhubung ke relay 6
const int relayPin7 = 12;  // Pin yang terhubung ke relay 7
unsigned long waktu_mulai;
unsigned long durasi;
void setup(){
  /*0 General*/
  Serial.begin(9600);
  waktu_mulai = millis(); // menyimpan waktu mulai
  pinMode(relayPin1, OUTPUT); // Mengatur pin relay 1 sebagai output
  pinMode(relayPin2, OUTPUT); // Mengatur pin relay 2 sebagai output
  pinMode(relayPin3, OUTPUT); // Mengatur pin relay 3 sebagai output
  pinMode(relayPin4, OUTPUT); // Mengatur pin relay 4 sebagai output
  pinMode(relayPin5, OUTPUT); // Mengatur pin relay 5 sebagai output
  pinMode(relayPin6, OUTPUT); // Mengatur pin relay 6 sebagai output
  pinMode(relayPin7, OUTPUT); // Mengatur pin relay 7 sebagai output
  digitalWrite(relayPin1, HIGH);
  digitalWrite(relayPin2, HIGH);
  digitalWrite(relayPin3, HIGH);
  digitalWrite(relayPin4, HIGH);
  digitalWrite(relayPin5, HIGH);
  digitalWrite(relayPin6, HIGH);
  digitalWrite(relayPin7, HIGH);
  attachInterrupt(digitalPinToInterrupt(2), Pulse_Event, RISING);
  pinMode(5, INPUT_PULLUP);
  /* 1- PZEM-017 DC Energy Meter */
  setShunt(pzemSlaveAddr);
  startMillisPZEM = millis();
```

```
 Serial2.begin(9600, SERIAL_8N2);
 node.begin(pzemSlaveAddr, Serial2);
 pinMode(MAX485_RE, OUTPUT);
 pinMode(MAX485_DE, OUTPUT);
 digitalWrite(MAX485_RE, 0);
 digitalWrite(MAX485_DE, 0);
 node.preTransmission(preTransmission);
 node.postTransmission(postTransmission);
 changeAddress(0XF8, pzemSlaveAddr);
 startMillisLCD = millis();    }
 long int millisButton;
 void loop(){
  if (millis() - millisButton > 500) {
   bool button = digitalRead (5);
   if (button == 0) {
     ResetEnergy += 1;}
   millisButton = millis(); }
 currentMillisPZEM = millis();
 if (millis() - startMillisPZEM >= periodPZEM)  {
   uint8_t result;
   result = node.readInputRegisters(0x0000, 6);
   if (result == 0x00)       {
     uint32_t tempdouble = 0x00000000;
     PZEMVoltage = node.getResponseBuffer(0x0000) / 100.0 ;
     PZEMCurrent =( node.getResponseBuffer(0x0001) / 600.0) ;
     PZEMPower = PZEMVoltage * PZEMCurrent ;
     tempdouble =  (node.getResponseBuffer(0x0005) << 16) +
node.getResponseBuffer(0x0004);
     PZEMEnergy = tempdouble ;
     durasi = (millis() - waktu_mulai)/1000;
     PZEMEnergynew =  (durasi * PZEMPower /3600) ;
     /* 2 - Sensor Kecepatan  */
     LastTimeCycleMeasure = LastTimeWeMeasured;
     CurrentMicros = micros(){
       LastTimeCycleMeasure = CurrentMicros;    }
     FrequencyRaw = 10000000000 / PeriodAverage;
     if (PeriodBetweenPulses > ZeroTimeout - ZeroDebouncingExtra ||
CurrentMicros - LastTimeCycleMeasure > ZeroTimeout -
ZeroDebouncingExtra) {
       FrequencyRaw = 0;  // Set frequency as 0.
       ZeroDebouncingExtra = 2000;
     } else {
       ZeroDebouncingExtra = 0; }
     FrequencyReal = FrequencyRaw / 10000;
     RPM = FrequencyRaw / PulsesPerRevolution * 60;
     RPM = RPM / 10000;
     total = total - readings[readIndex];
     readings[readIndex] = RPM;
     total = total + readings[readIndex];
     readIndex = readIndex + 1;
     if (readIndex >= numReadings) {
       readIndex = 0;  }
     average = total / numReadings;
     String kirim_1 = String(PZEMVoltage, 3);
     String kirim_2 = String(PZEMCurrent, 3);
     String kirim_3 = String(PZEMPower, 3);
     String kirim_4 = String(PZEMEnergynew, 3);
     kirim_1.replace('.', ',');
     kirim_2.replace('.', ',');
     kirim_3.replace('.', ',');
     kirim_4.replace('.', ',');
     Serial.print(kirim_1);
     Serial.print(".");
     Serial.print(kirim_2);
     Serial.print(".");
     Serial.print(kirim_3);
     Serial.print(".");
     Serial.print(kirim_4);
     Serial.print(".");
     Serial.print(RPM);
     Serial.print(".");   }
   else  {
     Serial.print("Failed to read modbus ");
     Serial.println(result);  }
   startMillisPZEM = currentMillisPZEM ;        }
 resetEnergy();
  readrelay();          }
 /* 2 - Sensor Kecepatan  */
```

```
 void Pulse_Event() {
 PeriodBetweenPulses = micros() - LastTimeWeMeasured;
 LastTimeWeMeasured = micros();
 if (PulseCounter >= AmountOfReadings)  {
   PeriodAverage = PeriodSum / AmountOfReadings;
   PulseCounter = 1;
   PeriodSum = PeriodBetweenPulses;
   int RemapedAmountOfReadings = map(PeriodBetweenPulses, 40000,
5000, 1, 10);
   RemapedAmountOfReadings =
constrain(RemapedAmountOfReadings, 1, 10);
   AmountOfReadings = RemapedAmountOfReadings;
 } else {
   PulseCounter++;
   PeriodSum = PeriodSum + PeriodBetweenPulses;    }  }
 /* 1- PZEM-017 DC Energy Meter */
 void preTransmission()  {
 digitalWrite(MAX485_RE, 1);
 digitalWrite(MAX485_DE, 1);
 delay(1);     }
 void postTransmission()     {
 delay(3);
 digitalWrite(MAX485_RE, 0);
 digitalWrite(MAX485_DE, 0);     }
 void setShunt(uint8_t slaveAddr)      {
 static uint8_t SlaveParameter = 0x06;
 static uint16_t registerAddress = 0x0003;
 uint16_t u16CRC = 0xFFFF;
 u16CRC = crc16_update(u16CRC, slaveAddr);
 u16CRC = crc16_update(u16CRC, SlaveParameter);
 u16CRC = crc16_update(u16CRC, highByte(registerAddress));
 u16CRC = crc16_update(u16CRC, lowByte(registerAddress));
 u16CRC = crc16_update(u16CRC, highByte(NewshuntAddr));
 u16CRC = crc16_update(u16CRC, lowByte(NewshuntAddr));
 Serial.println("Change shunt address");
 preTransmission();
 Serial2.write(slaveAddr);
 Serial2.write(SlaveParameter);
 Serial2.write(highByte(registerAddress));
```

```
Serial2.write(lowByte(registerAddress));
Serial2.write(highByte(NewshuntAddr));
Serial2.write(lowByte(NewshuntAddr));
Serial2.write(lowByte(u16CRC));
Serial2.write(highByte(u16CRC));
delay(10);
postTransmission();
delay(100);
while (Serial2.available())      {
  Serial.print(char(Serial2.read()), HEX);
  Serial.print(" ");        }   }
void changeAddress(uint8_t OldslaveAddr, uint8_t NewslaveAddr)     {
static uint8_t SlaveParameter = 0x06;
static uint16_t registerAddress = 0x0002;
uint16_t u16CRC = 0xFFFF;
u16CRC = crc16_update(u16CRC, OldslaveAddr);
u16CRC = crc16_update(u16CRC, SlaveParameter);
u16CRC = crc16_update(u16CRC, highByte(registerAddress));
u16CRC = crc16_update(u16CRC, lowByte(registerAddress));
u16CRC = crc16_update(u16CRC, highByte(NewslaveAddr));
u16CRC = crc16_update(u16CRC, lowByte(NewslaveAddr));
Serial.println("Change Slave Address");
preTransmission();
Serial2.write(OldslaveAddr);
Serial2.write(SlaveParameter);
Serial2.write(highByte(registerAddress));
Serial2.write(lowByte(registerAddress));
Serial2.write(highByte(NewslaveAddr));
Serial2.write(lowByte(NewslaveAddr));
Serial2.write(lowByte(u16CRC));
Serial2.write(highByte(u16CRC));
delay(10);
postTransmission();
delay(100);
while (Serial2.available())       {
  Serial.print(char(Serial2.read()), HEX);
  Serial.print(" ");      }   }
void resetEnergy() {
if (ResetEnergy == 0)startMillisEnergy = millis();
```

```
  if (ResetEnergy == 1) {
    if (( millis() - startMillisEnergy <= 5000))      {
      PZEMOffset = PZEMEnergy;
      uint16_t u16CRC = 0xFFFF;
      static uint8_t resetCommand = 0x42;
      uint8_t slaveAddr = 0X01;
      u16CRC = crc16_update(u16CRC, slaveAddr);
      u16CRC = crc16_update(u16CRC, resetCommand);
      Serial.println("Resetting Energy");
      preTransmission();
      Serial.println("Resetting Energy1");
      Serial2.write(slaveAddr);
      Serial.println("Resetting Energy2");
  Serial2.write(resetCommand);    Serial.println("Resetting Energy3");
  Serial2.write(lowByte(u16CRC));  Serial.println("Resetting Energy4");
   Serial2.write(highByte(u16CRC));
Serial.println("Resetting Energy5");
      delay(10);
      postTransmission();   Serial.println("Resetting Energy5");
      delay(100);
      Serial.println("Resetting Energyla");
      while (Serial2.available())        {
        Serial.print(char(Serial2.read()), HEX);
        Serial.print(" ");        }
      ResetEnergy = 0;        }  }  }
  void readrelay() {
  if (Serial.available() > 0) {
    char command = Serial.read(); // Membaca data yang masuk
    switch (command) {
      case '1':
        digitalWrite(relayPin1, LOW); // Menyalakan relay 1
        break;
      case '2':
        digitalWrite(relayPin1, HIGH); // Mematikan relay 1
        break;
      case '3':
        digitalWrite(relayPin2, LOW); // Menyalakan relay 2
        break;
      case '4':
```

```
    digitalWrite(relayPin2, HIGH); // Mematikan relay 2
    break;
  case '5':
    digitalWrite(relayPin3, LOW); // Menyalakan relay 3
    break;
  case '6':
    digitalWrite(relayPin3, HIGH); // Mematikan relay 3
    break;
  case '7':
    digitalWrite(relayPin4, LOW); // Menyalakan relay 4
    break;
  case '8':
    digitalWrite(relayPin4, HIGH); // Mematikan relay 4
    break;
  case '9':
    digitalWrite(relayPin5, LOW); // Menyalakan relay 5
    break;
  case '0':
    digitalWrite(relayPin5, HIGH); // Mematikan relay 5
    break;
  case 'a':
    digitalWrite(relayPin6, LOW); // Menyalakan relay 6
    break;
  case 'b':
    digitalWrite(relayPin6, HIGH); // Mematikan relay 6
    break;
    case 'c':
    digitalWrite(relayPin7, LOW); // Menyalakan relay 7
    break;
  case 'd':
    digitalWrite(relayPin7, HIGH); // Mematikan relay 7
    break;
}   }   }
```

BIODATA PENULIS

Nama	: Rasyid Fadjrianto
Tempat/Tanggal Lahir	: Cilacap, 17 Juni 2001
Alamat	: Dusun Manggisari, Desa Karangtawang RT01/ RW03, Nusawungu, Cilacap
E-Mail	: rasyidfadjrianto@gmail.com
Telepon/HP	: +628232013980
Hobi	: Billyard
Motto	: Kerjakan sesuatu jangan menunda-nunda.

Riwayat Pendidikan

- SD Negeri 02 Karangtawang Tahun 2007-2015
- SMP Islam Al-Irsyad Cilacap Tahun 2015-2018
- SMK Negeri Nusawungu Tahun 2018-2021
  Jurusan Instalasi Teknik Listrik
- Politeknik Negeri Cilacap Tahun 2021-2024
  Prodi D3 Teknik Elektronika

Penulis telah mengikuti seminar hasil tugas akhir pada tanggal 19 Agustus 2024. Sebagai salahsatu persyaratan untuk memperoleh gelar Ahli Madya (A.Md.)