DAFTAR PUSTAKA

Amazon.com (no date) *PZEM-014 / 016 AC communication module*. Available at: https://images-na.ssl-images-amazon.com/images/I/81GtkIOyZaL.pdf (Accessed: 1 March 2021).

Anhar, A. S., Sara, I. D. and Siregar, R. H. (2017) 'Desain Prototype Sel Surya Terkonsentrasi Menggunakan Lensa Fresnel', *Jurnal Karya Ilmiah Teknik Elektro*, 2(3), pp. 1–7.

JakartaHardware (2021) *Digital Multimeter Sanwa CD771 dan Analog Multimeter Sanwa YX360TRF*. Available at: http://www.jakartahardware.com/products/digital-multimeter-sanwa-cd771-6368.aspx#.YVXNjpoza00 (Accessed: 20 September 2021).

Jurnal, R. T. (2019) 'KAJIAN SISTEM KINERJA PLTS OFF-GRID 1 kWp DI STT-PLN', *Energi & Kelistrikan*, 10(1), pp. 38–44. doi: 10.33322/energi.v10i1.322.

Latasya, Z. *et al.* (2019) 'Analisis Rancangan Pembangkit Listrik Tenaga Surya (Plts) Off-Grid Terpusat Dusun Ketubong Tunong Kecamatan Seunagan Timur Kabupaten Nagan Raya', *Kitektro*, 4(2), pp. 1–14.

MANURUNG, F. (2020) 'RANCANG BANGUN ALAT DETEKSI BANJIR MENGGUNAKAN IoT (BLYNK) BERBASIS ARDUINO UNO', p. 16.

Muliadi, Imran, A. and Rasul, M. (2020) 'Pengembangan tempat sampah pintar menggunakan esp32', *Media Elektrik*, 17(2), pp. 1907–1728.

Munarso and Suryono (2014) 'SISTEM TELEMETRI PEMANTAUAN SUHU LINGKUNGAN MENGGUNAKAN MIKROKONTROLER DAN JARINGAN WIFI', *Youngster Physic Journal*, 3(3), pp. 249–256.

Munggaran, Z. R. (2017) 'RANCANG BANGUN SISTEM TELEMETERING GARDU DISTRIBUSI PT.PLN BERBASIS ANDROID', (May 2016). doi: 10.13140/RG.2.2.32501.88801.

Muttaqin, R. (2017) 'Analisa Performansi dan Monitoring Pembangkit Listrik Tenaga Surya di Departemen Teknik Fisika FTI-ITS', p. 120. Available at: http://repository.its.ac.id/47444/.

Prastyo, E. A. (2019) *Arsitektur dan Fitur ESP32 (Module ESP32) IoT*. Available at: https://www.edukasielektronika.com/2019/07/arsitektur-dan-fitur-esp32-module-esp32.html (Accessed: 1 March 2021).

Satria, H. and Syafii, S. (2018) 'Sistem Monitoring Online dan Analisa Performansi

PLTS Rooftop Terhubung ke Grid PLN', *Jurnal Rekayasa Elektrika*, 14(2). doi: 10.17529/jre.v14i2.11141.

Setiadi, D. and Abdul Muhaemin, M. N. (2018) 'PENERAPAN INTERNET OF THINGS (IoT) PADA SISTEM MONITORING IRIGASI (SMART IRIGASI)', *Infotronik : Jurnal Teknologi Informasi dan Elektronika*, 3(2), p. 95. doi: 10.32897/infotronik.2018.3.2.108.

Sianipar, R. (2014) 'Dasar Perencanaan Pembangkit Listrik Tenaga Surya', 11(2), pp. 61–78.

Solar-thailand (no date) *PZEM-003 / 017 DC communication module*. Available at: https://www.solar-thailand.com/pdf/PZEM-003-Manual.pdf (Accessed: 1 March 2021).

Solarduino (2020a) *PZEM-016 AC Energy Meter Online Monitoring with Blynk App*. Available at: https://solarduino.com/pzem-016-ac-energy-meter-online-monitoring-with-blynk-app/ (Accessed: 1 March 2021).

Solarduino (2020b) *PZEM-017 DC Energy Meter with Arduino*. Available at: https://solarduino.com/pzem-017-dc-energy-meter-with-arduino/ (Accessed: 1 March 2021).

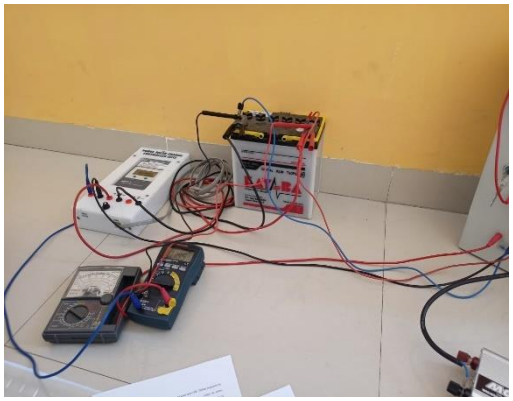
Stevens, L. (2021) *Modul Antarmuka MAX485 TTL ke RS-485*. Available at: https://protosupplies.com/product/max485-ttl-to-rs-485-interface-module/ (Accessed: 1 March 2021).

Syahwil, M. and Kadir, N. (2021) 'Rancang Bangun Modul Pembangkit Listrik Tenaga Surya (PLTS) Sistem Off-grid Sebagai Alat Penunjang Praktikum Di Laboratorium', 3(1), pp. 26–35.

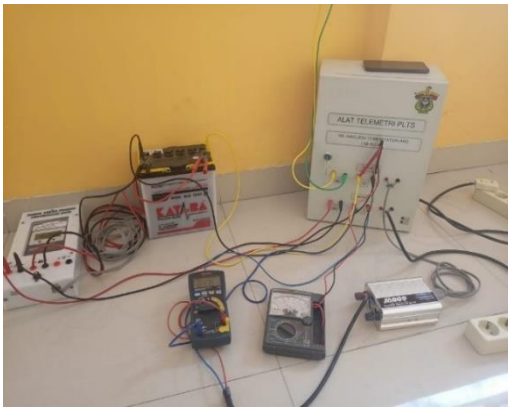
Ulfah Tian, S. (2017) 'ROTOTIPE SISTEM MONITORING PARAMETER PEMBANGKIT LISTRIK TENAGA SURYA BERBASIS INTERNET OF THINGS', *Skripsi*, p. 2. doi: 10.1088/1751-8113/44/8/085201.

LAMPIRAN

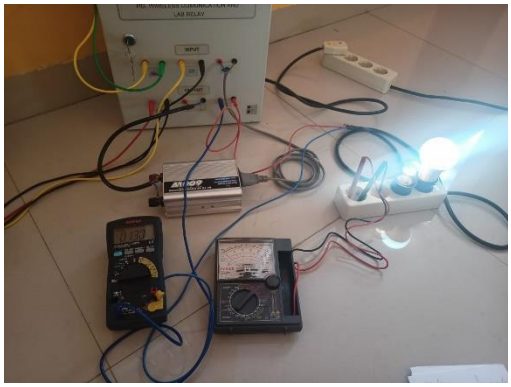
Lampiran 1 Proses uji coba alat telemetri



(a) Proses pengkuran parameter komponen *photovoltaic* dan baterai saat pengisian.



(b) Proses pengukuran parameter komponen baterai saat inverter dan beban aktif



(c) Proses pengukuran parameter beban saat inverter dan beban aktif

Lampiran 2 Kode program ESP 32 alat telemetri PLTS

```
 #define BLYNK_PRINT Serial
#include <SoftwareSerial.h>
SoftwareSerial PZEMSerial;
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char auth[] = "Kb6560RAzhAX_HCDsyV1DPNQZYR7Pjo-";
char ssid[] = "HOME ALONE lt.1";
char pass[] = "Warkopik2021";
#include <ModbusMaster.h>
#define MAX485_DE  5
#define MAX485_RE  4

static uint8_t pzemSlaveAddr = 0x01;
static uint8_t pzemSlaveAddr2 = 0x02;
static uint8_t pzemSlaveAddr3 = 0x03;
static uint16_t NewshuntAddr = 0x0001;
static uint16_t NewshuntAddr2 = 0x0001;
ModbusMaster node;
ModbusMaster node2;
ModbusMaster node3;
```

```
float PZEMVoltage =0;
float PZEMCurrent =0;
float PZEMPower =0;
float PZEMEnergy=0;

float PZEMVoltage2 =0;
float PZEMCurrent2 =0;
float PZEMPower2 =0;
float PZEMEnergy2 =0;

float PZEMVoltage3 =0;
float PZEMCurrent3 =0;
float PZEMPower3 =0;
float PZEMEnergy3=0;
float PZEMHz3 =0;
float PZEMPf3=0;

unsigned long startMillisPZEM;
unsigned long currentMillisPZEM;
unsigned long startMilliSetShunt;
const unsigned long periodPZEM = 1000;
```

```
        unsigned long startMillisReadData;
        unsigned long currentMillisReadData;
        const unsigned long periodReadData = 1000;
        int ResetEnergy = 0;
        int ResetEnergy2 = 0;
        int ResetEnergy3 = 0;
        int a = 0;
        unsigned long startMillis1;
void setup()
{   startMillis1 =millis();
        Serial.begin(9600);
        PZEMSerial.begin(9600,SWSERIAL_8N2,13,15);
        Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8080);
        startMilliSetShunt = millis();
        startMillisPZEM = millis();
        pinMode(MAX485_RE, OUTPUT);
        pinMode(MAX485_DE, OUTPUT);
        digitalWrite(MAX485_RE, 0);
        digitalWrite(MAX485_DE, 0);
        node.preTransmission(preTransmission);
```

```
        node.postTransmission(postTransmission);
        node2.preTransmission(preTransmission);
        node2.postTransmission(postTransmission);
        node3.preTransmission(preTransmission);
        node3.postTransmission(postTransmission);
        delay(1000);
        startMillisReadData = millis();
}
void loop()
{
        Blynk.run();
        currentMillisPZEM = millis();
        if (millis()- startMilliSetShunt == 10000)
        {setShunt(0x01);}
        if (millis()-startMilliSetShunt == 15000)
        {setShunt2(0x02);}
        if(a == 0)
        {
          node.begin(pzemSlaveAddr, PZEMSerial);
          if (currentMillisPZEM - startMillisPZEM >= periodPZEM)
          {
          uint8_t result;
          result = node.readInputRegisters(0x0000, 6);
```

```
if (result == node.ku8MBSuccess)
  {
    uint32_t tempdouble = 0x00000000;
    PZEMVoltage = node.getResponseBuffer(0x0000) / 100.0;
    PZEMCurrent = node.getResponseBuffer(0x0001) / 100.0;
    tempdouble =  (node.getResponseBuffer(0x0003) << 16) +
node.getResponseBuffer(0x0002);
    PZEMPower = tempdouble / 10.0;
    tempdouble =  (node.getResponseBuffer(0x0005) << 16) +
node.getResponseBuffer(0x0004);
    PZEMEnergy = tempdouble;        }
if (pzemSlaveAddr==5)
    {}
else
    {}
    a = 1;
    startMillisPZEM = currentMillisPZEM ;     } }
if(a ==1)
{
node2.begin(pzemSlaveAddr2, PZEMSerial);
if (currentMillisPZEM - startMillisPZEM >= periodPZEM)
  {
   uint8_t result2;
result2 = node2.readInputRegisters(0x0000, 6);
```

```
  if (result2 == node2.ku8MBSuccess)
    {


      uint32_t tempdouble2 = 0x00000000;
      PZEMVoltage2 = node2.getResponseBuffer(0x0000) / 100.0;
      PZEMCurrent2 = node2.getResponseBuffer(0x0001) / 100.0;
      tempdouble2 =  (node2.getResponseBuffer(0x0003) << 16) +
node2.getResponseBuffer(0x0002);
      PZEMPower2 = tempdouble2 / 10.0;
      tempdouble2 =  (node2.getResponseBuffer(0x0005) << 16) +
node2.getResponseBuffer(0x0004);
      PZEMEnergy2 = tempdouble2;
     }
      if (pzemSlaveAddr==5)
      {}
      else
      {}
      a=2;
      startMillisPZEM = currentMillisPZEM ; }}
  if(a ==2)
 {
  node3.begin(pzemSlaveAddr3, PZEMSerial);
 if (currentMillisPZEM - startMillisPZEM >= periodPZEM)
 {
```

```
uint8_t result3;

result3 = node3.readInputRegisters(0x0000, 9);

if (result3 == node3.ku8MBSuccess)

  {

    uint32_t tempdouble3 = 0x00000000;

    PZEMVoltage3 = node3.getResponseBuffer(0x0000) / 10.0;


    tempdouble3 =  (node3.getResponseBuffer(0x0002) << 16) +
node3.getResponseBuffer(0x0001);

    PZEMCurrent3 = tempdouble3 / 2000.00;


    tempdouble3 =  (node3.getResponseBuffer(0x0004) << 16) +
node3.getResponseBuffer(0x0003);

    PZEMPower3 = tempdouble3 / 10.0;


    tempdouble3 =  (node3.getResponseBuffer(0x0006) << 16) +
node3.getResponseBuffer(0x0005);

    PZEMEnergy3 = tempdouble3;


    PZEMHz3 = node3.getResponseBuffer(0x0007) / 10.0;

    PZEMPf3 = node3.getResponseBuffer(0x0008) / 100.00;


    if (pzemSlaveAddr==5)

      {  }}
```

```
    else
      { }
      a=0;
    startMillisPZEM = currentMillisPZEM ;      }  }
currentMillisReadData = millis();
if (currentMillisReadData - startMillisReadData >= periodReadData)
  {
    Serial.println("Energy Meter 1 : ");
    Serial.print("Vdc : "); Serial.print(PZEMVoltage); Serial.println(" V ");
    Serial.print("Idc : "); Serial.print(PZEMCurrent); Serial.println(" A ");
    Serial.print("Power : "); Serial.print(PZEMPower); Serial.println(" W ");
    Serial.print("Energy : "); Serial.print(PZEMEnergy); Serial.println(" Wh ");

    Serial.println("Energy Meter 2 : ");
    Serial.print("Vdc : "); Serial.print(PZEMVoltage2); Serial.println(" V ");
    Serial.print("Idc : "); Serial.print(PZEMCurrent2); Serial.println(" A ");
    Serial.print("Power : "); Serial.print(PZEMPower2); Serial.println(" W ");
    Serial.print("Energy : "); Serial.print(PZEMEnergy2); Serial.println(" Wh ");

    Serial.println("Energy Meter 3 : ");
    Serial.print("Vac : "); Serial.print(PZEMVoltage3); Serial.println(" V ");
    Serial.print("Iac : "); Serial.print(PZEMCurrent3); Serial.println(" A ");
    Serial.print("Power : "); Serial.print(PZEMPower3); Serial.println(" W ");
```

```
        Serial.print("Energy : "); Serial.print(PZEMEnergy3); Serial.println(" Wh ");
        Serial.print("Power Factor : "); Serial.print(PZEMPf3); Serial.println(" pF ");
        Serial.print("Frequency : "); Serial.print(PZEMHz3); Serial.println(" Hz ");

        Blynk.virtualWrite(V0,PZEMVoltage);
        Blynk.virtualWrite(V1,PZEMCurrent);
        Blynk.virtualWrite(V2,PZEMPower);
        Blynk.virtualWrite(V3,PZEMEnergy);

        Blynk.virtualWrite(V5,PZEMVoltage2);
        Blynk.virtualWrite(V6,PZEMCurrent2);
        Blynk.virtualWrite(V7,PZEMPower2);
        Blynk.virtualWrite(V8,PZEMEnergy2);

        Blynk.virtualWrite(V10,PZEMVoltage3);
        Blynk.virtualWrite(V11,PZEMCurrent3);
        Blynk.virtualWrite(V12,PZEMPower3);
        Blynk.virtualWrite(V13,PZEMEnergy3);
        Blynk.virtualWrite(V14,PZEMPf3);
        Blynk.virtualWrite(V15,PZEMHz3);
        startMillisReadData = millis();
      } }
void preTransmission()
```

```
{
        if(millis() - startMillis1 > 5000)
        {
          digitalWrite(MAX485_RE, 1);
          digitalWrite(MAX485_DE, 1);
          delay(1);
    }}
void postTransmission()
{
        if(millis() - startMillis1 > 5000)
        {
          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))
    preTransmission();
    PZEMSerial.write(slaveAddr);
    PZEMSerial.write(SlaveParameter);
    PZEMSerial.write(highByte(registerAddress));
    PZEMSerial.write(lowByte(registerAddress));
    PZEMSerial.write(highByte(NewshuntAddr));
    PZEMSerial.write(lowByte(NewshuntAddr));
    PZEMSerial.write(lowByte(u16CRC));
    PZEMSerial.write(highByte(u16CRC));
    delay(10);
    postTransmission();
    delay(100);
    while (PZEMSerial.available())
      {
      }
}
void setShunt2(uint8_t slaveAddr2)
{
```

```
static uint8_t SlaveParameter2 = 0x06;
static uint16_t registerAddress2 = 0x0003;
uint16_t u16CRC2 = 0xFFFF;
u16CRC2 = crc16_update(u16CRC2, slaveAddr2);
u16CRC2 = crc16_update(u16CRC2, SlaveParameter2);
u16CRC2 = crc16_update(u16CRC2, highByte(registerAddress2));
u16CRC2 = crc16_update(u16CRC2, lowByte(registerAddress2));
u16CRC2 = crc16_update(u16CRC2, highByte(NewshuntAddr2));
u16CRC2 = crc16_update(u16CRC2, lowByte(NewshuntAddr2));
preTransmission();
PZEMSerial.write(slaveAddr2);
PZEMSerial.write(SlaveParameter2);
PZEMSerial.write(highByte(registerAddress2));
PZEMSerial.write(lowByte(registerAddress2));
PZEMSerial.write(highByte(NewshuntAddr2));
PZEMSerial.write(lowByte(NewshuntAddr2));
PZEMSerial.write(lowByte(u16CRC2));
PZEMSerial.write(highByte(u16CRC2));
delay(10);
postTransmission();
delay(100);
while (PZEMSerial.available())
  {
```

```
        }
}
BLYNK_WRITE(V4)
{
      if(param.asInt()==1)
        {
          uint16_t u16CRC = 0xFFFF;
          static uint8_t resetCommand = 0x42;
          uint8_t slaveAddr =0X01;
          u16CRC = crc16_update(u16CRC, slaveAddr);
          u16CRC = crc16_update(u16CRC, resetCommand);
          preTransmission();
          PZEMSerial.write(slaveAddr);
          PZEMSerial.write(resetCommand);
          PZEMSerial.write(lowByte(u16CRC));
          PZEMSerial.write(highByte(u16CRC));
          delay(10);
          postTransmission();
          delay(100);
        }
}
```

```
BLYNK_WRITE(V9)
{
    if(param.asInt()==1)
     {
       uint16_t u16CRC = 0xFFFF;
       static uint8_t resetCommand = 0x42;
       uint8_t slaveAddr =0X02;
       u16CRC = crc16_update(u16CRC, slaveAddr);
       u16CRC = crc16_update(u16CRC, resetCommand);
       preTransmission();
       PZEMSerial.write(slaveAddr);
       PZEMSerial.write(resetCommand);
       PZEMSerial.write(lowByte(u16CRC));
       PZEMSerial.write(highByte(u16CRC));
       delay(10);
       postTransmission();
       delay(100);
     }
}
BLYNK_WRITE(V16)
    {
     if(param.asInt()==1)
     {
```

```
    uint16_t u16CRC = 0xFFFF;
    static uint8_t resetCommand = 0x42;
    uint8_t slaveAddr =pzemSlaveAddr3;
    u16CRC = crc16_update(u16CRC, slaveAddr);
    u16CRC = crc16_update(u16CRC, resetCommand);
    preTransmission();
    PZEMSerial.write(slaveAddr);
    PZEMSerial.write(resetCommand);
    PZEMSerial.write(lowByte(u16CRC));
    PZEMSerial.write(highByte(u16CRC));
    delay(10);
    postTransmission();
    delay(100);
  }
}
```