LAMPIRAN A

a. Pembuatan Rangkaian PLTS *Portable*

b. Pengukuran Tegangan dan Arus Panel Surya, dan Tegangan Baterai

1. Hari Jum'at 30 Juni 2023

2. Hari Sabtu 1 Juli 2023

1. Hari Minggu 2 Juli 2023

c. Hasil Perbandingan Pengukuran Antara Sensor Dengan Multitester

LAMPIRAN B

```
#define BLYNK_TEMPLATE_ID "TMPL6pfiwHOpu"
#define BLYNK_TEMPLATE_NAME "MONITORING EMERGENCY
PLTS"
#define BLYNK_AUTH_TOKEN
"IAmJSeORk_QjKOuyY7bVeCpDx4eu_Cz0"

      /* Virtual Serial Port */
      #include <SoftwareSerial.h>                    /* include virtual
Serial Port coding */
      SoftwareSerial PZEMSerial;                     // Move the PZEM
DC Energy Meter communication pins from Rx to pin D1 = GPIO 5 &
TX to pin D2 = GPIO 4
      /* 0- Blynk Server and Wifi Connection */
      #include <ESP8266WiFi.h>                       // Enable the use of
wifi module. Make sure you downloaded and installed the ESP8266
library
      #include <BlynkSimpleEsp8266.h>
      char auth[] = "IAmJSeORk_QjKOuyY7bVeCpDx4eu_Cz0";     //
Put in the Auth Token for the project from Blynk. You should receive it
in your email.
      char ssid[] = "GALAXY";                // Key in your wifi name. You
can check with your smart phone for your wifi name
      char pass[] = "12345678";                      // Key in your wifi
password.

      /* 1- PZEM-017 DC Energy Meter */

      #include <ModbusMaster.h>                      // Load the
(modified) library for modbus communication command codes. Kindly
install at our website.
      #define MAX485_DE  16                          // Define DE Pin to
Arduino pin. Connect DE Pin of Max485 converter module to Pin D0
(GPIO 16) Node MCU board
```

```
#define MAX485_RE  5                    // Define RE Pin to Arduino pin. Connect RE Pin of Max485 converter module to Pin D1 (GIPO 5) Node MCU board
                                        // These DE anr RE pins can be any other Digital Pins to be activated during transmission and reception process.
static uint8_t pzemSlaveAddr = 0x01;    // Declare the address of device (meter 1) in term of 8 bits.
static uint16_t NewshuntAddr = 0x0001;  // Declare your external shunt value for DC Meter. Default 0x0000 is 100A, replace to "0x0001" if using 50A shunt, 0x0002 is for 200A, 0x0003 is for 300A

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;          /* start counting time for LCD Display */
unsigned long currentMillisPZEM;        /* current counting time for LCD Display */
const unsigned long periodPZEM = 1000;  // refresh every X seconds (in seconds) in LED Display. Default 1000 = 1 second

/* 2 - Data submission to Blynk Server  */

unsigned long startMillisReadData;      /* start counting time for data collection */
```

```
      unsigned long currentMillisReadData;              /* current
counting time for data collection */
      const unsigned long periodReadData = 1000;        /* refresh
every X seconds (in seconds) in LED Display. Default 1000 = 1 second
*/
      int ResetEnergy = 0;                              /* reset energy function
*/
      int a = 1;
      unsigned long startMillis1;                       // to count time during
initial start up (PZEM Software got some error so need to have initial
pending time)
void setup()
{
      /*0 General*/

  startMillis1 = millis();
      Serial.begin(9600);                               /* To assign
communication port to communicate with meter. with 2 stop bits (refer
to manual)*/
      PZEMSerial.begin(9600,SWSERIAL_8N2,4,0);          // 4 =
Rx/R0/ GPIO 4 (D2) & 0 = Tx/DI/ GPIO 0 (D3) on NodeMCU
      Blynk.begin(auth, ssid, pass, "blynk.cloud",8080);

      /* 1- PZEM-017 DC Energy Meter */

      startMillisPZEM = millis();                       /* Start counting time
for run code */
      pinMode(MAX485_RE, OUTPUT);                       /* Define RE
Pin as Signal Output for RS485 converter. Output pin means Arduino
command the pin signal to go high or low so that signal is received by
the converter*/
      pinMode(MAX485_DE, OUTPUT);                       /* Define DE
Pin as Signal Output for RS485 converter. Output pin means Arduino
command the pin signal to go high or low so that signal is received by
the converter*/
```

```
        digitalWrite(MAX485_RE, 0);                 /* Arduino create
output signal for pin RE as LOW (no output)*/
        digitalWrite(MAX485_DE, 0);                 /* Arduino create
output signal for pin DE as LOW (no output)*/
                                                    // both pins no output means the
converter is in communication signal receiving mode
        node.preTransmission(preTransmission);      // Callbacks
allow us to configure the RS485 transceiver correctly
        node.postTransmission(postTransmission);
        node.begin(pzemSlaveAddr,PZEMSerial);
        delay(1000);                                /* after everything done,
wait for 1 second */
        /* 2 - Data submission to Blynk Server  */
        startMillisReadData = millis();             /* Start counting time
for data submission to Blynk Server*/
}

void loop()
{
        Blynk.run();

        if ((millis()- startMillis1 >= 10000) && (a ==1))
        {
          setShunt(pzemSlaveAddr);                  // Delete the
"//" to set shunt rating (0x01) is the meter address by default. If you
changed the meter address, you must change here as well.
          changeAddress(0XF8, pzemSlaveAddr);       // By
delete the double slash symbol, the meter address will be set as 0x01.
          a = 0;
        }                                           // By default I allow this code to
run every program startup. Will not have effect if you only have 1 meter

        /* 1- PZEM-017 DC Energy Meter */

        currentMillisPZEM = millis();
```

```
        if (currentMillisPZEM - startMillisPZEM >= periodPZEM)
/* for every x seconds, run the codes below*/
        {
          uint8_t result;                                                    /*
Declare variable "result" as 8 bits */
          result = node.readInputRegisters(0x0000, 6);
/* read the 9 registers (information) of the PZEM-014 / 016 starting
0x0000 (voltage information) kindly refer to manual)*/
          if (result == node.ku8MBSuccess)
/* If there is a response */
            {
              uint32_t tempdouble = 0x00000000;
/* Declare variable "tempdouble" as 32 bits with initial value is 0 */
              PZEMVoltage = node.getResponseBuffer(0x0000) / 100.0;
/* get the 16bit value for the voltage value, divide it by 100 (as per
manual) */
                                                                          //
0x0000 to 0x0008 are the register address of the measurement value
              PZEMCurrent = node.getResponseBuffer(0x0001) / 100.0;
/* get the 16bit value for the current value, divide it by 100 (as per
manual) */

              tempdouble =  (node.getResponseBuffer(0x0003) << 16) +
node.getResponseBuffer(0x0002);      /* get the power value. Power
value is consists of 2 parts (2 digits of 16 bits in front and 2 digits of 16
bits at the back) and combine them to an unsigned 32bit */
              PZEMPower = tempdouble / 10.0;
/* Divide the value by 10 to get actual power value (as per manual) */

              tempdouble =  (node.getResponseBuffer(0x0005) << 16) +
node.getResponseBuffer(0x0004);      /* get the energy value. Energy
value is consists of 2 parts (2 digits of 16 bits in front and 2 digits of 16
bits at the back) and combine them to an unsigned 32bit */
              PZEMEnergy = tempdouble;
```

```
            if (pzemSlaveAddr==2)
/* just for checking purpose to see whether can read modbus*/
              {
              }
          }
            else
              {
              }
           startMillisPZEM = currentMillisPZEM ;
/* Set the starting point again for next counting time */
      }
                                                                    /*
count time for program run every second (by default)*/
      /* 2 - Data submission to Blynk Server  */

      currentMillisReadData = millis();
/* Set counting time for data submission to server*/
      if (currentMillisReadData - startMillisReadData >=
periodReadData)                    /* for every x seconds, run the
codes below*/
        {
         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 ");
         Blynk.virtualWrite(V0,PZEMVoltage);
// Send data to Blynk Server. Voltage value as virtual pin V0
          Blynk.virtualWrite(V1,PZEMCurrent);
          Blynk.virtualWrite(V2,PZEMPower);
          Blynk.virtualWrite(V3,PZEMEnergy);
          startMillisReadData = millis();
        }
```

```
}

void preTransmission()
/* transmission program when triggered*/
{
      /* 1- PZEM-017 DC Energy Meter */
      if(millis() - startMillis1 > 5000)
// Wait for 5 seconds as ESP Serial cause start up code crash
      {
       digitalWrite(MAX485_RE, 1);
/* put RE Pin to high*/
      digitalWrite(MAX485_DE, 1);
/* put DE Pin to high*/
      delay(1);                                                    //
When both RE and DE Pin are high, converter is allow to transmit
communication
      }
}
void postTransmission()
/* Reception program when triggered*/
}
      /* 1- PZEM-017 DC Energy Meter */
      if(millis() - startMillis1 > 5000)
// Wait for 5 seconds as ESP Serial cause start up code crash
      {
       delay(3);                                                   //
When both RE and DE Pin are low, converter is allow to receive
communication
      digitalWrite(MAX485_RE, 0);
/* put RE Pin to low*/
      digitalWrite(MAX485_DE, 0);
/* put DE Pin to low*/
      }
}
```

```
void setShunt(uint8_t slaveAddr)
//Change the slave address of a node
{
      /* 1- PZEM-017 DC Energy Meter */
      static uint8_t SlaveParameter = 0x06;
/* Write command code to PZEM */
      static uint16_t registerAddress = 0x0003;
/* change shunt register address command code */

      uint16_t u16CRC = 0xFFFF;
/* declare CRC check 16 bits*/
      u16CRC = crc16_update(u16CRC, slaveAddr);
// Calculate the crc16 over the 6bytes to be send
      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();
/* trigger reception mode*/
      delay(100);
}

BLYNK_WRITE(V4)                                   // Virtual push button
to reset energy for Meter 1
{
```

```
        if(param.asInt()==1)
          {
            uint16_t u16CRC = 0xFFFF;                    /* declare CRC
check 16 bits*/
            static uint8_t resetCommand = 0x42;           /* reset command
code*/
            uint8_t slaveAddr = pzemSlaveAddr;            // if you set
different address, make sure this slaveAddr must change also
            u16CRC = crc16_update(u16CRC, slaveAddr);
            u16CRC = crc16_update(u16CRC, resetCommand);
            preTransmission();                          /* trigger transmission
mode*/
            PZEMSerial.write(slaveAddr);                /* send device
address in 8 bit*/
            PZEMSerial.write(resetCommand);             /* send reset
command */
            PZEMSerial.write(lowByte(u16CRC));          /* send CRC
check code low byte  (1st part) */
            PZEMSerial.write(highByte(u16CRC));         /* send CRC
check code high byte (2nd part) */
            delay(10);
            postTransmission();                         /* trigger reception
mode*/
            delay(100);
          }
}

void changeAddress(uint8_t OldslaveAddr, uint8_t NewslaveAddr)
//Change the slave address of a node
{
        /* 1- PZEM-017 DC Energy Meter */

        static uint8_t SlaveParameter = 0x06;
/* Write command code to PZEM */
```

```
        static uint16_t registerAddress = 0x0002;
/* Modbus RTU device address command code */
        uint16_t u16CRC = 0xFFFF;
/* declare CRC check 16 bits*/
        u16CRC = crc16_update(u16CRC, OldslaveAddr);
// Calculate the crc16 over the 6bytes to be send
        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));
        preTransmission();
/* trigger transmission mode*/
        PZEMSerial.write(OldslaveAddr);
/* these whole process code sequence refer to manual*/
        PZEMSerial.write(SlaveParameter);
        PZEMSerial.write(highByte(registerAddress));
        PZEMSerial.write(lowByte(registerAddress));
        PZEMSerial.write(highByte(NewslaveAddr));
        PZEMSerial.write(lowByte(NewslaveAddr));
        PZEMSerial.write(lowByte(u16CRC));
        PZEMSerial.write(highByte(u16CRC));
        delay(10);
        postTransmission();
/* trigger reception mode*/
        delay(100);
}
```