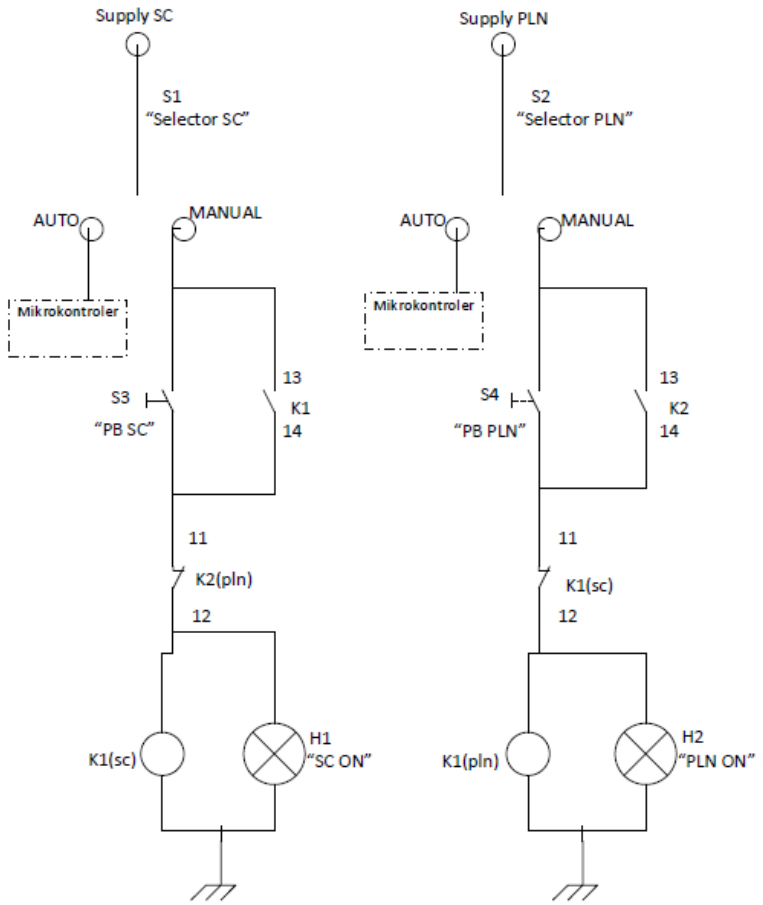
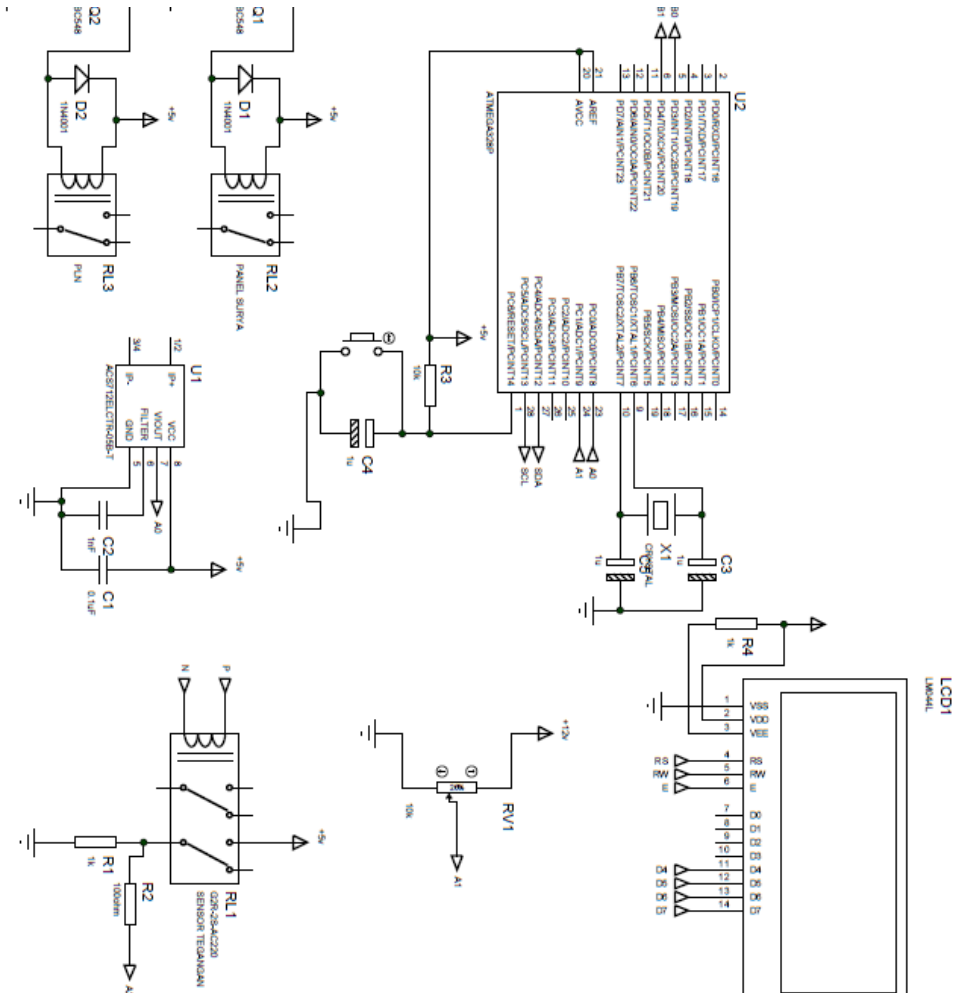
Lampiran 1. Rangkaian Kontrol



Lampiran 2. Rangkaian Kontrol Versi Proteus



Lampiran 3. Coding

```
#include <LiquidCrystal_PCF8574.h>
#include<Wire.h>
LiquidCrystal_PCF8574 lcd(0x27);
unsigned char DataAmount = 500;
const int OutputBaterai = 3;
const int OutputPLN = 4;
int outputFuzzy[9] = {1,0,0,
                      1,1,1,
                      1,1,1}; //0 = Solar Panel, 1 = PLN
float derajat_Keanggotaan_Daya[3]; // [0] = Rendah, [1] = Sedang, [2] = Tinggi
float derajat_Keanggotaan_Baterai[3]; // [0] = Drop, [1] = Normal, [2] = Bagus
float scale_Baterai = 204.8;
//arus = A0
//tegangan baterai = A1
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Wire.begin();
  lcd.setBacklight(255);
  lcd.home();
  lcd.clear();
  lcd.noBlink();
```

```
  lcd.noCursor();
  lcd.setCursor(0, 0);
  lcd.print("=Sistem Kontrol ATS=");
  lcd.setCursor(0, 1);
  lcd.print("Vbat=24V Arus=0.4A");
  lcd.setCursor(0, 2);
  lcd.print("Daya=100Watt");
  lcd.setCursor(0, 3);
  lcd.print("Supply=SolarCell");
  pinMode(OutputBaterai,OUTPUT);
  pinMode(OutputPLN,OUTPUT);
}
void loop() {
 unsigned int nilaiAdc = ReadCurrents();
  float skala = 2.1;
  float Daya = nilaiAdc*skala;
  Serial.print("Daya = ");
  Serial.print(Daya);
  Serial.print(" W    ");

  unsigned int DataADCTegangan=analogRead(A0);
  float TeganganBaterai=(float)(((DataADCTegangan / scale_Baterai)-3.0) / 0.2) +
20.0;
```

```
Serial.print("Vbatt = ");
Serial.print(TeganganBaterai);
Serial.println(" V");
fuzzyBaterai(TeganganBaterai);
fuzzyDaya(Daya);
int outputRule = defuzzifikasi();
if (outputFuzzy[outputRule] == 1) Serial.println("Sumber PLN");
else if (outputFuzzy[outputRule] == 0) Serial.println("Sumber Solar Cell");
Serial.print("rule yg aktif = ");
Serial.println(outputRule);
Serial.print("derajat_Keanggotaan_Daya = ");
Serial.print(derajat_Keanggotaan_Daya[0]);
Serial.print(" , ");
Serial.print(derajat_Keanggotaan_Daya[1]);
Serial.print(" , ");
Serial.println(derajat_Keanggotaan_Daya[2]);
////////////// Tampilan LCD ////////////////////////////
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("=Sistem Kontrol ATS=");
lcd.setCursor(0, 1);
lcd.print("Vbat=");
lcd.print(int(TeganganBaterai));
```

```
 lcd.print("V Arus=");
 lcd.print(Daya/220);
 lcd.print("A");
 lcd.setCursor(0, 2);
 lcd.print("Daya=");
 lcd.print(Daya);
 lcd.print(" Watt");
 if (outputFuzzy[outputRule] == 1) {
   Serial.println("Sumber PLN");
   lcd.setCursor(0, 3);
   lcd.print("Supply=PLN");
   digitalWrite(OutputBaterai,LOW);
    digitalWrite(OutputPLN,HIGH);
 }
 else if (outputFuzzy[outputRule] == 0) {
   Serial.println("Sumber Solar Cell");
   lcd.setCursor(0, 3);
   lcd.print("Supply=SolarCell");
    digitalWrite(OutputBaterai,HIGH);
     digitalWrite(OutputPLN,LOW);
 }
 //unsigned int TeganganPLN=analogRead(A2);
 delay(1100);
```

```
}

int defuzzifikasi(){

  float rule[9];

  rule[0] = derajat_Keanggotaan_Baterai[0] + derajat_Keanggotaan_Daya[0]; // If Vbatt low and power low

  rule[1] = derajat_Keanggotaan_Baterai[1] + derajat_Keanggotaan_Daya[0]; // If Vbatt normal and power low

  rule[2] = derajat_Keanggotaan_Baterai[2] + derajat_Keanggotaan_Daya[0]; // If Vbatt good and power low

  rule[3] = derajat_Keanggotaan_Baterai[0] + derajat_Keanggotaan_Daya[1]; // If Vbatt low and power mid

  rule[4] = derajat_Keanggotaan_Baterai[1] + derajat_Keanggotaan_Daya[1]; // If Vbatt normal and power mid

  rule[5] = derajat_Keanggotaan_Baterai[2] + derajat_Keanggotaan_Daya[1];

  rule[6] = derajat_Keanggotaan_Baterai[0] + derajat_Keanggotaan_Daya[2];

  rule[7] = derajat_Keanggotaan_Baterai[1] + derajat_Keanggotaan_Daya[2];

  rule[8] = derajat_Keanggotaan_Baterai[2] + derajat_Keanggotaan_Daya[2];

  int ruleTerpilih;

  float bufferRule=0;

  for(int i=0; i<9; i++){

    if(bufferRule<rule[i]){

      bufferRule = rule[i];

      ruleTerpilih=i;

    }

  }
```

```
  return ruleTerpilih;
}
void fuzzyBaterai(float Tegangan){
  if(Tegangan < 23.0) {
    derajat_Keanggotaan_Baterai[0] = 1.0;
    derajat_Keanggotaan_Baterai[1] = 0;
    derajat_Keanggotaan_Baterai[2] = 0;
  }
  else if (Tegangan >= 23.0 && Tegangan < 24.0){
    derajat_Keanggotaan_Baterai[0] = 24.0 - Tegangan;
    derajat_Keanggotaan_Baterai[1] = Tegangan - 23.0;
    derajat_Keanggotaan_Baterai[2] = 0;
  }
  else if (Tegangan >= 24.0 && Tegangan < 25.0){
    derajat_Keanggotaan_Baterai[0] = 0;
    derajat_Keanggotaan_Baterai[1] = 25.0 - Tegangan;
    derajat_Keanggotaan_Baterai[2] = Tegangan - 24.0;
  }
  else if (Tegangan >= 25.0){
    derajat_Keanggotaan_Baterai[0] = 0;
    derajat_Keanggotaan_Baterai[1] = 0;
    derajat_Keanggotaan_Baterai[2] = 1.0;
  }
```

```
}
void fuzzyDaya(float Daya){
  if(Daya < 100.0) {
   derajat_Keanggotaan_Daya[0] = 1.0;
   derajat_Keanggotaan_Daya[1] = 0;
   derajat_Keanggotaan_Daya[2] = 0;
  }
  else if (Daya >= 100.0 && Daya < 300.0){
   derajat_Keanggotaan_Daya[0] = (300.0 - Daya)/200.0;
   derajat_Keanggotaan_Daya[1] = (Daya - 300.0)/200.0;
   derajat_Keanggotaan_Daya[2] = 0;
  }
  else if (Daya >= 300.0 && Daya < 500.0){
   derajat_Keanggotaan_Daya[0] = 0;
   derajat_Keanggotaan_Daya[1] = 1.0;
   derajat_Keanggotaan_Daya[2] = 0;
  }
  else if (Daya >= 500.0 && Daya < 700.0){
   derajat_Keanggotaan_Daya[0] = 0;
   derajat_Keanggotaan_Daya[1] = (700.0 - Daya)/200.0;
   derajat_Keanggotaan_Daya[2] = (Daya - 700.0)/200.0;
  }
  else if (Daya >= 700.0){
```

```
    derajat_Keanggotaan_Daya[0] = 0;
    derajat_Keanggotaan_Daya[1] = 0;
    derajat_Keanggotaan_Daya[2] = 1.0;
  }
}
unsigned int ReadCurrents(){
  unsigned int data_arus[DataAmount];
  for(int i=0; i<DataAmount; i++){
    data_arus[i] = analogRead(A2);
  }
  for(int i=0; i<DataAmount; i++){
    //Serial.println(data_arus[i]);
  }
  unsigned int maxValue=0;
  unsigned int minValue=0xFFFF;
  //Filter Moving Average
  /*
  for(int i=0; i<DataAmount-10; i++){
    for(int j=0; j<9; j++){
      data_arus[i] += data_arus[j+1];
    }
    data_arus[i] /= 10;
  }
```

```
  for(int i=0; i<DataAmount-10; i++){
    Serial.println(data_arus[i]);
  }*/
  for(int i=0; i<DataAmount-10; i++){
    if(maxValue<data_arus[i]) maxValue = data_arus[i];
    if(minValue>data_arus[i]) minValue = data_arus[i];
  }
  unsigned int ipp =  maxValue-minValue;
  return ipp;
}
```