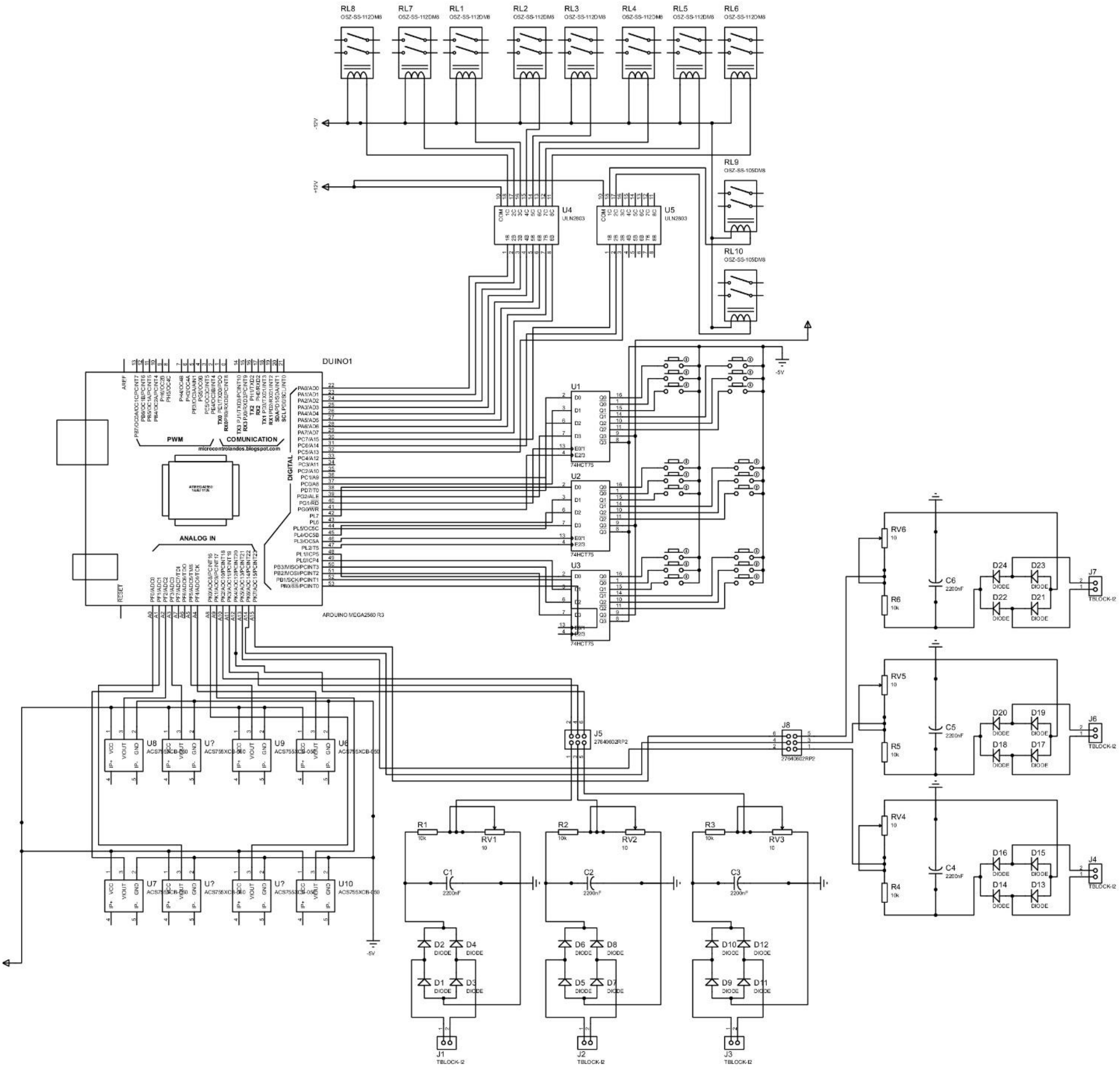
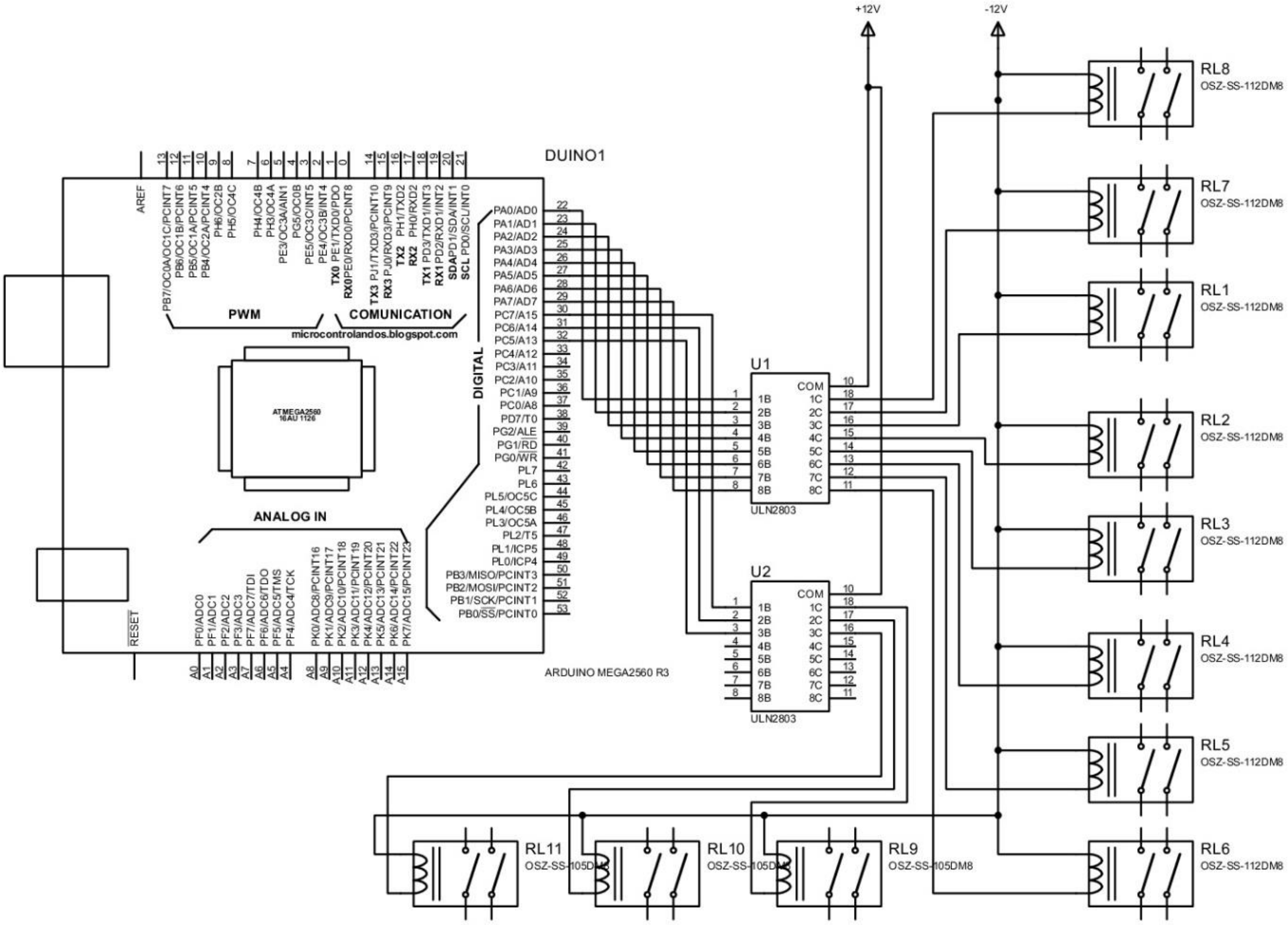
LAMPIRAN 1



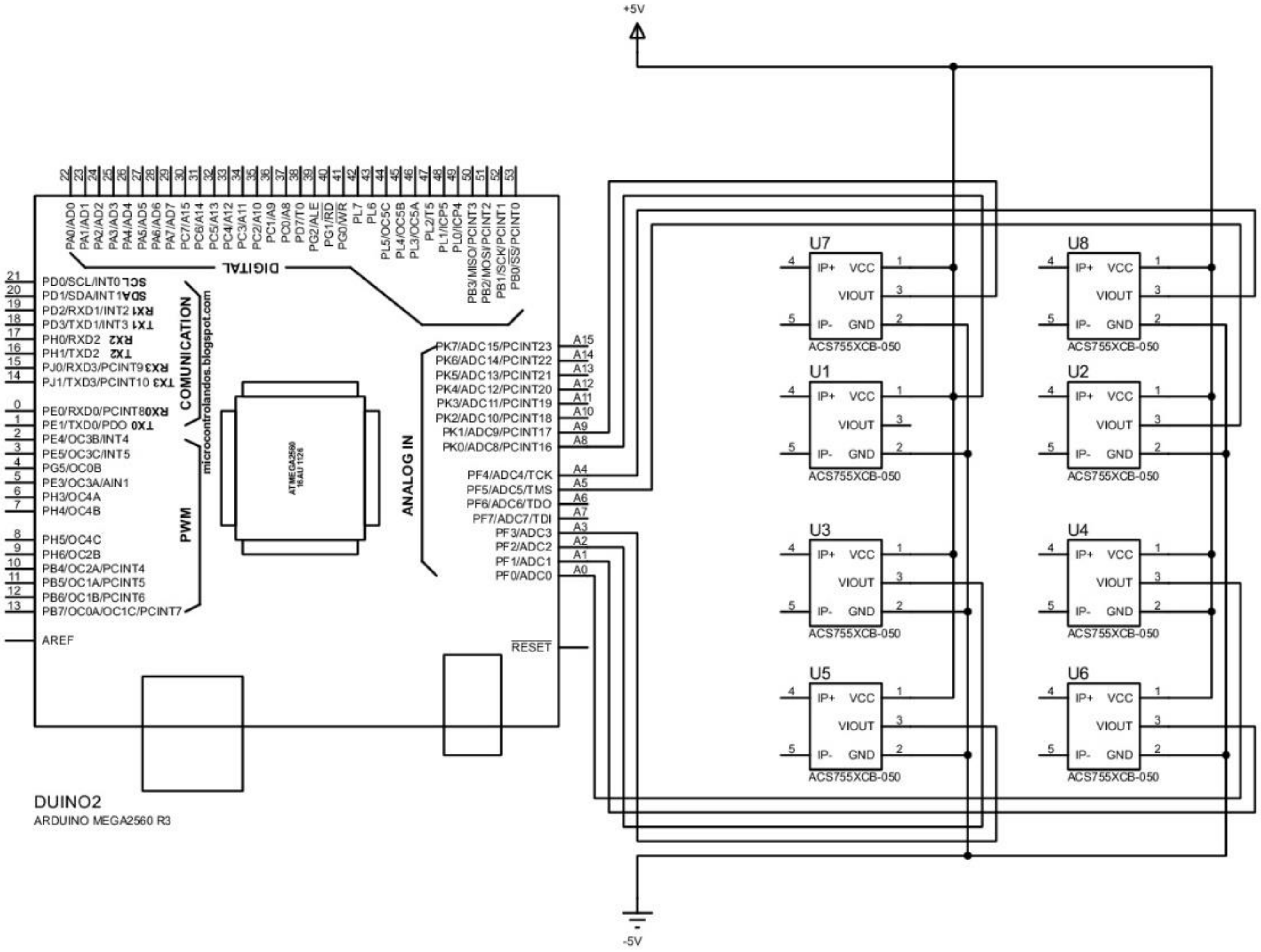
Gambar 3.11 Rangkaian Keseluruhan
(Sumber: *Proteus Schematic Project*, Modifikasi diambil pada 6 Juni 2018)

LAMPIRAN 2



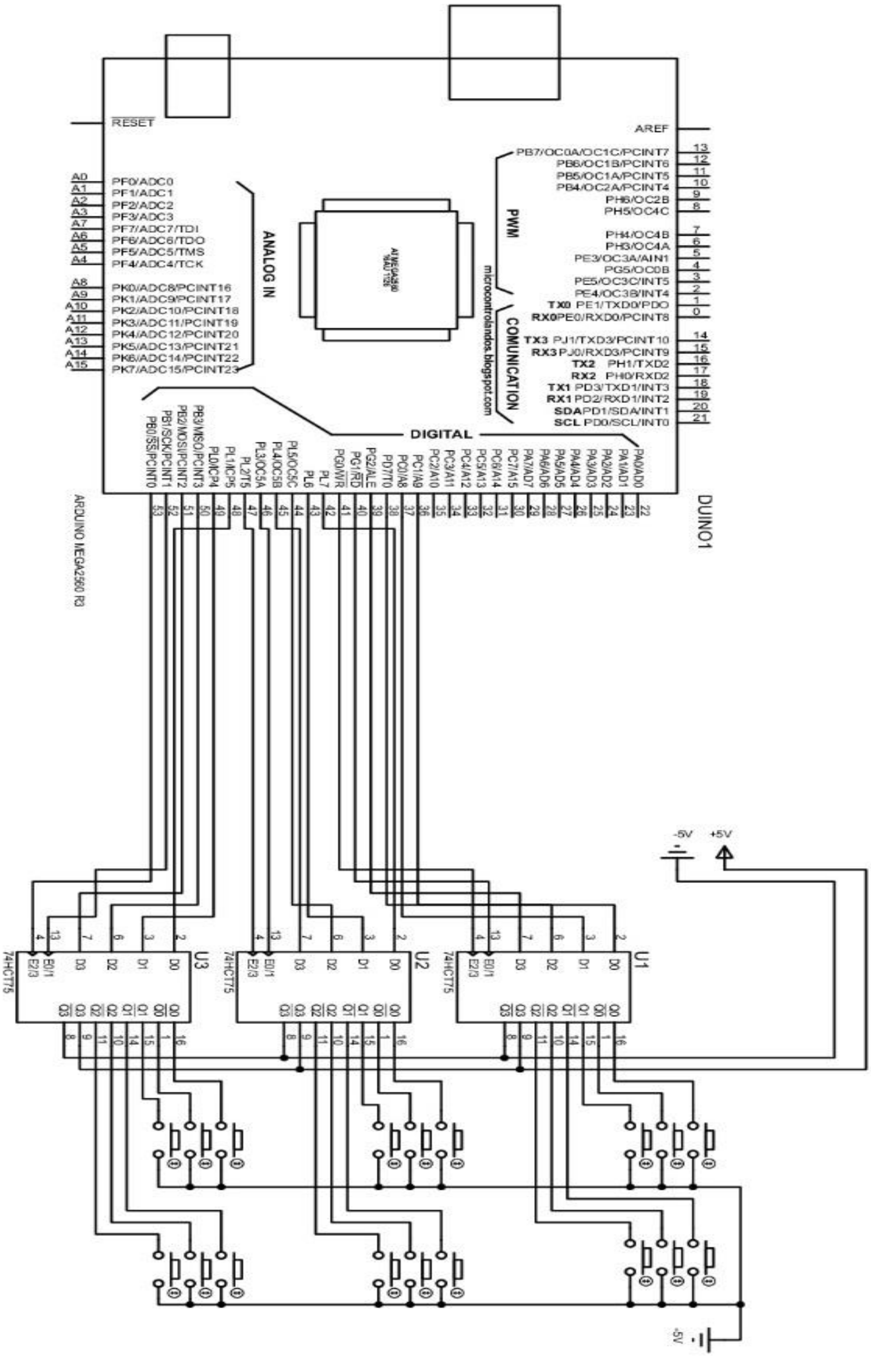
Gambar 3.4 Rangkaian *Driver Relay* dan Rangkaian *Relay*
(Sumber: *Proteus Schematic Project*, Modifikasi diambil pada 5 Juni 2018)

LAMPIRAN 3



Gambar 3.6 Sensor arus ACS712 dengan Arduino Mega 2560
(Sumber: Proteus Schematic Project, Modifikasi diambil pada 6 Juni 2018)

LAMPIRAN 4



Gambar 3.7 Rangkaian *Debouncer*

(Sumber: Proteus Schematic Project, Modifikasi diambil pada 6 Juni 2018)

LAMPIRAN 5

LISTING PROGRAM ARDUINO

```
#include <Mudbus.h>

#include <SPI.h>

#include <Ethernet.h>

Mudbus Mb;

unsigned long delay_now1 = 0;

unsigned long delay_last1 = 0;

unsigned long delay_limit1 = 500;

unsigned long delay_now2 = 0;

unsigned long delay_last2 = 0;

unsigned long delay_limit2 = 500;

unsigned long delay_now3 = 0;

unsigned long delay_last3 = 0;

unsigned long delay_limit3 = 500;

int penghitung_rec = 0;

int penghitung =0;

int penghitungR =0;
```

```
const int tombolpmt1open = 36;
const int tombolpmt1close = 37;
const int tombolrec1open = 38;
const int tombolrec1close = 39;
const int tombollbs1open = 40;
const int tombollbs1close = 41;
const int tombolrec2open = 44;
const int tombolrec2close = 43;
const int tombolpmt2open = 42;
const int tombolpmt2close = 45;
const int tombollbs2open = 46;
const int tombollbs2close = 47;
const int tombolssoopen = 7;
const int tombolssoclose = 8;
const int tombolrec3open = 5;
const int tombolrec3close = 6;
const int tombolpmt3open = 48;
const int tombolpmt3close = 49;
const int pmt1 = 22;

const int rec1 = 23;

const int lbs1 = 24;

const int rec2 = 25;

const int pmt2 = 26;

const int lbs2 = 27;

const int pmt3 = 28;

const int rec3 = 29;

const int sso = 30;

const int lbs3y1 = 31;

const int lbs3y2 = 32;

const int rec21 =A13;

const int rec31 =A14;

const int sso21=A15;

int statetombolpmt1open;

int statetombolpmt1close;

int statetombolrec1open;
int statetombolrec1close;

int statetombollbs1open;

int statetombollbs1close;

int statetombolrec2open;

int statetombolrec2close;

int statetombolpmt2open;

int statetombolpmt2close;

int statetombollbs2open;

int statetombollbs2close;

int statetombolpmt3open;

int statetombolpmt3close;

int statetombolrec3open;

int statetombolrec3close;

int statetombolssoopen;

int statetombolssoclose;

//variables will change;

int baca_sensorarus1;

int nilaiarus1;
int baca_sensorarus2;
int nilaiarus2;
int baca_sensorarus3;
int nilaiarus3;
int baca_sensorarus4;
int nilaiarus4;
int baca_sensorarus5;
int nilaiarus5;
int baca_sensorarus6;
int nilaiarus6;
int baca_sensorarus7;
int nilaiarus7;
int baca_sensorarus8;
int nilaiarus8;
int arusgangguan;
int baca_sensorteg1;
double nilaiteg1;
int baca_sensorteg2;
```

```
double nilaiteg2;
int baca_sensorteg3;
double nilaiteg3;
int baca_sensorteg4;
float nilaiteg4;
int baca_sensorteg5;
float nilaiteg5;
int baca_sensorteg6;
float nilaiteg6;
const int a_pmt1 = A0;
const int a_rec1 = A1;
const int a_rec2 = A2;
const int a_pmt2 = A3;
const int a_pmt3 = A4;
const int a_rec3 = A5;
const int a_sso = A8;
const int a_3y = A9;
const int t_rec2 = A13;
const int t_rec3 = A14;

const int t_sso = A15;

const int PB_LOCAL_REMOTE = 35;

int stats_PB_LOCAL_REMOTE = 0;

float sensorarus1()

{

   int ACS_sensorarus1;    //value read from the sensor;

   int max_ACS_sensorarus1 = 0;

   int min_ACS_sensorarus1 = 1024;

   uint32_t start_time1 = millis();

   while (millis()-start_time1 <= 30) //sample for 500ms;

   {

      ACS_sensorarus1 = analogRead(a_pmt1);

      if (ACS_sensorarus1 > max_ACS_sensorarus1)

      {

         //record the maximum sensor value;

         max_ACS_sensorarus1 = ACS_sensorarus1;

      }
```

```
  }
  float baca_sensorarus1 = (float) abs((max_ACS_sensorarus1 - 520)*5.00/1023/0.185); // /1.414*0.925
  return baca_sensorarus1;
}
float sensorarus2()
{
  int ACS_sensorarus2;    //value read from the sensor;
  int max_ACS_sensorarus2 = 0;
  int min_ACS_sensorarus2 = 1024;
  uint32_t start_time2 = millis();
  while (millis()-start_time2 <= 30) //sample for 500ms;
  {
    ACS_sensorarus2 = analogRead(a_rec1);
    if (ACS_sensorarus2 > max_ACS_sensorarus2)
    {
      //record the maximum sensor value;
      max_ACS_sensorarus2 = ACS_sensorarus2;
    }
  }
  float baca_sensorarus2 = (float) abs(((max_ACS_sensorarus2 -
530)*5.00/1023/0.185)); // /1.414*0.925
  return baca_sensorarus2;
}
float sensorarus3()
{
  int ACS_sensorarus3;    //value read from the sensor;
  int max_ACS_sensorarus3 = 0;
  int min_ACS_sensorarus3 = 1024;
  uint32_t start_time3 = millis();
  while (millis()-start_time3 <= 30) //sample for 500ms;
  {
    ACS_sensorarus3 = analogRead(a_rec2);
    if (ACS_sensorarus3 > max_ACS_sensorarus3)
    {
      //record the maximum sensor value;
```

```
         max_ACS_sensorarus3 = ACS_sensorarus3;
      }
   }
   float baca_sensorarus3 = (float) abs((max_ACS_sensorarus3 -
525)*5.00/1023/0.185); // /1.414*0.925
   return baca_sensorarus3;
}
float sensorarus4()
{
   int ACS_sensorarus4;    //value read from the sensor;
   int max_ACS_sensorarus4 = 0;
   int min_ACS_sensorarus4 = 1024;
   uint32_t start_time4 = millis();
   while (millis()-start_time4 <= 30) //sample for 500ms;
   {
      ACS_sensorarus4 = analogRead(a_pmt2);
      if (ACS_sensorarus4 > max_ACS_sensorarus4)
      {
            //record the maximum sensor value;
            max_ACS_sensorarus4 = ACS_sensorarus4;
        }
    }
    float baca_sensorarus4 = (float) abs((max_ACS_sensorarus4 - 518)*5.00/1023/0.185); // /1.414*0.925
    return baca_sensorarus4;
}
void setup(){
  uint8_t mac[]     = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // dirubah sesuai kesepakatan
  uint8_t ip[]      = { 192, 168, 0, 123 };   // dirubah sesuai kesepakatan
  uint8_t gateway[] = { 192, 168, 0, 1 };
  uint8_t subnet[]  = { 255, 255, 255, 0 };
  Ethernet.begin(mac, ip, gateway, subnet);
  //Avoid pins 4,10,11,12,13 when using ethernet shield//
      Serial.begin(9600);
 pinMode(tombolpmt1open, INPUT); //
 pinMode(tombolpmt1close, INPUT); //
 pinMode(tombolrec1open, INPUT); //
 pinMode(tombolrec1close, INPUT); //
 pinMode(tombollbs1open, INPUT); //
 pinMode(tombollbs1close, INPUT); //
 pinMode(tombolpmt2open, INPUT);
 pinMode(tombolpmt2close, INPUT);
 pinMode(tombolrec2open, INPUT);
 pinMode(tombolrec2close, INPUT);
 pinMode(tombollbs2open, INPUT);
 pinMode(tombollbs2close, INPUT);
 pinMode(tombolpmt3open, INPUT);
 pinMode(tombolpmt3close, INPUT);
 pinMode(tombolrec3open, INPUT);
 pinMode(tombolrec3close, INPUT);
 pinMode(tombolssoopen, INPUT);
 pinMode(tombolssoclose, INPUT);
pinMode (t_rec3, INPUT);
pinMode (t_rec2, INPUT);
pinMode (PB_LOCAL_REMOTE, INPUT);

 pinMode(pmt1, OUTPUT);

 pinMode(rec1, OUTPUT);

 pinMode(lbs1, OUTPUT);

 pinMode(rec2, OUTPUT);

 pinMode(pmt2, OUTPUT);

 pinMode(lbs2, OUTPUT);

 pinMode(pmt3, OUTPUT);

 pinMode(rec3, OUTPUT);

 pinMode(sso, OUTPUT);

 pinMode(lbs3y1, OUTPUT);

 pinMode(lbs3y2, OUTPUT);

}

void loop () {

  Mb.Run();

  Mb.R[0]=(sensorarus1()*100);

  Mb.R[1]=(sensorarus2()*100);

  Mb.R[2]=(sensorarus3()*100);
  Mb.R[3]=(sensorarus4()*100);

  if(sensorarus2() >=1.00) {

  delay (500);

  digitalWrite (rec1, HIGH);

     Mb.C[11] = 1;

  delay (2000);

  digitalWrite (rec1, LOW);

     Mb.C[11] = 0;

    if (statetombolrec1close == HIGH) {

      digitalWrite (rec1, LOW);

         Mb.C[11] = 0;

          if (statetombolrec1open == HIGH) {

      digitalWrite (rec1,HIGH);

         Mb.C[11] = 1;

    }

  }

   }

   if (sensorarus3() >=1.90) {
```

```
delay (500);

digitalWrite (rec2, HIGH);

  Mb.C[12] = 1;

delay (1500);

digitalWrite (rec2, LOW);

Mb.C[12] = 0;

  }

   if((sensorarus3() >=1.90) &&(penghitungR<=2)) {

    delay(1500);

    digitalWrite(rec2, HIGH);

    Mb.C[12] = 1;

    penghitung_rec++;

    penghitungR++;

   }

    if(sensorarus4() >=3.45) {

delay (500);

digitalWrite (pmt2, HIGH);

Mb.C[13] = 1;
```

```
  delay (2000);
  digitalWrite (pmt2, LOW);
  Mb.C[13] = 0;
  if (sensorarus4() <=0.30) {
    delay(1000);
    digitalWrite (pmt2, LOW);
     Mb.C[13] = 0;
    if (statetombolpmt2close == HIGH) {
      digitalWrite (pmt2, LOW);
       Mb.C[13] = 0;
    }
  }
     }
if  (sensorarus1() >= 2.10) {
  delay (50);
digitalWrite (pmt1, HIGH);
 Mb.C[10] = 1;
}
```

```
if (statetombolpmt1close == HIGH ) {
  digitalWrite (pmt1, LOW);
   Mb.C[10] = 0;
  }
 statetombolpmt1open   = digitalRead(tombolpmt1open) ;
 statetombolpmt1close  = digitalRead(tombolpmt1close);
 statetombolrec1open   = digitalRead(tombolrec1open);
 statetombolrec1close  = digitalRead(tombolrec1close);
 statetombollbs1open   = digitalRead(tombollbs1open);
 statetombollbs1close  = digitalRead(tombollbs1close);
 statetombolrec2open   = digitalRead(tombolrec2open);
 statetombolrec2close  = digitalRead(tombolrec2close);
 statetombolpmt2open   = digitalRead(tombolpmt2open);
 statetombolpmt2close  = digitalRead(tombolpmt2close);
 statetombollbs2open   = digitalRead(tombollbs2open);
 statetombollbs2close  = digitalRead(tombollbs2close);
 statetombolpmt3open   = digitalRead(tombolpmt3open);
 statetombolpmt3close  = digitalRead(tombolpmt3close);
 statetombolrec3open   = digitalRead(tombolrec3open);

 statetombolrec3close  = digitalRead(tombolrec3close);

 statetombolssoopen    = digitalRead(tombolssoopen);

 statetombolssoclose   = digitalRead(tombolssoclose);

 stats_PB_LOCAL_REMOTE   = digitalRead(PB_LOCAL_REMOTE);




 digitalWrite(22, Mb.C[10]); //pmt 1

 digitalWrite(23, Mb.C[11]); //rec 1

 digitalWrite(25, Mb.C[12]); // rec 2

 digitalWrite(26, Mb.C[13]); // pmt 2

 //LOCAL

 if  ((statetombolpmt1open  ==  HIGH)&&(stats_PB_LOCAL_REMOTE  ==
LOW))  {

   digitalWrite(pmt1, HIGH);

   Mb.C[10] = 1;

   }

   else if ((statetombolpmt1close == HIGH)&&(stats_PB_LOCAL_REMOTE
== LOW))  {
   digitalWrite(pmt1, LOW);
 Mb.C[10] = 0;
   }

 if ((statetombolrec1open == HIGH)&&(stats_PB_LOCAL_REMOTE ==
LOW)) {
   digitalWrite(rec1, HIGH);
   Mb.C[11] = 1;
   }
   else if ((statetombolrec1close == HIGH)&&(stats_PB_LOCAL_REMOTE
== LOW)) {
   digitalWrite(rec1, LOW);
   Mb.C[11] = 0;
   }
 if ((statetombolrec2open == HIGH)&&(stats_PB_LOCAL_REMOTE ==
LOW)) {
   digitalWrite(rec2, HIGH);
   Mb.C[12] = 1;
   }
```

```
   else if ((statetombolrec2close == HIGH)&&(stats_PB_LOCAL_REMOTE
== LOW))  {
   digitalWrite(rec2, LOW);
   Mb.C[12] = 0;
   }
 if  ((statetombolpmt2open  ==  HIGH)&&(stats_PB_LOCAL_REMOTE  ==
LOW))  {
   digitalWrite(pmt2, HIGH);
   Mb.C[13] = 1;
   }
   else if ((statetombolpmt2close == HIGH)&&(stats_PB_LOCAL_REMOTE
== LOW))  {
   digitalWrite(pmt2, LOW);
   Mb.C[13] = 0;
   }
  //REMOTE
 if (stats_PB_LOCAL_REMOTE == HIGH) {
   digitalWrite(22, Mb.C[10]);
 }
 if (stats_PB_LOCAL_REMOTE == HIGH) {

    digitalWrite(23, Mb.C[11]);

 }

 if (stats_PB_LOCAL_REMOTE == HIGH) {

    digitalWrite(25, Mb.C[12]);

 }

 if (stats_PB_LOCAL_REMOTE == HIGH) {

    digitalWrite(26, Mb.C[13]);

 }

digitalWrite (pmt3, HIGH);

  nilaiarus1 = digitalRead(baca_sensorarus1);

  nilaiarus2 = digitalRead(baca_sensorarus2);

  nilaiarus3 = digitalRead(baca_sensorarus3);

  nilaiarus4 = digitalRead(baca_sensorarus4);

Serial.print ("PMT1 =");

Serial.println (sensorarus1());

Serial.print ("rec1 =");

Serial.println (sensorarus2());
Serial.print ("rec2 =");

Serial.println (sensorarus3());

Serial.print ("pmt2 =");

Serial.println (sensorarus4());

 }
```