DAFTAR PUSTAKA

Asghar, M. S. (2011). Power Electronics. New Delhi: PHI Learning .

Atmel. (2016). 8-Bit AVR Microcontrollers : ATMega328P Datasheet Complete. California: Atmel Press.

Chapman, S. J. (2012). *Electric Machinery Fundamentals.* New York: The McGraw-Hill Companies.

Dewi, K. (2016). Implementasi Zero Crosing pada Kontrol Unit untuk Pengaturan Iluminasi Lampu Pijar dan Kipas Angin Berbasis Nuvoton. *Prosiding Seminar Teknik Elektro & Informatika.* PNUP, Makassar.

Gowda, Y., Hema, G., Prasad, S., & Vilasitha, V. (2016). Perfomance and Speed Control Cycloconverter Fed Split Phase Induction Motor. *Internasional Researh Journal of Engineering and Technology (IRJET), 3*(9), 1142-1146.

Gupta, A., Thakur, R., & Murarka, S. (2013). An Efficient Approach to Zero Crossing Detection Based On Opto-Coupler. *Int. Journal of Engineering Research and Applications, 3*(5), 834-838.

Kavitha, R. Premalatha, & K., R. (2019). Harmonic Research in Direct AC-AC Variable Frequency Power Converter. *International Journal of Recent Technology and Engineering (IJRTE), 8*(2S8), 1686-1690.

Lazim, M. T. (2019). *Power Electronics and Drives.* Jordan: Philadelphia University.

Lou, F. L., & Ye, H. (2018). *Power Electronics Advanced Conversion Technologies.* Florida : CRC Press.

NXP Semiconductors. (2009). Product Datasheet : BT151-650R. Eindhoven.

Nyein, A. (2018). Analysis of Cycloconverter Fed Induction Motor Drive. *Internasional Jurnal of Trend in Scientific Research and Development (IJTSRD), 3*(1), 569-574.

Permana, H., Azhar, & Finawan, A. (2021). Rancang Bangun Cycloconverter 1,5kW untuk Pengendalian Kecepatan Putar Motor Induksi 1 Phasa 500-1000 RPM pada Mesin Pengupas Pinang. *Jurnal TEKTRO, 5*(1), 8-14.

Rashid, M. H. (2011). *Microelectronic Circuits Analysis and Design Second Edition.* Florida: Cengage Learning.

Sen, P. C. (2014). *Principles of Electric Machines and Power Electronics.* Ontario: John Wiley and Sons, Inc.

Setiyono, B. D. (2021). Performance Comparison Modelling Between Single-Phase Cycloconverters and Three-Phase Cycloconverters Using Matlab Simulink Tools. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI), 7*(2), 217-229.

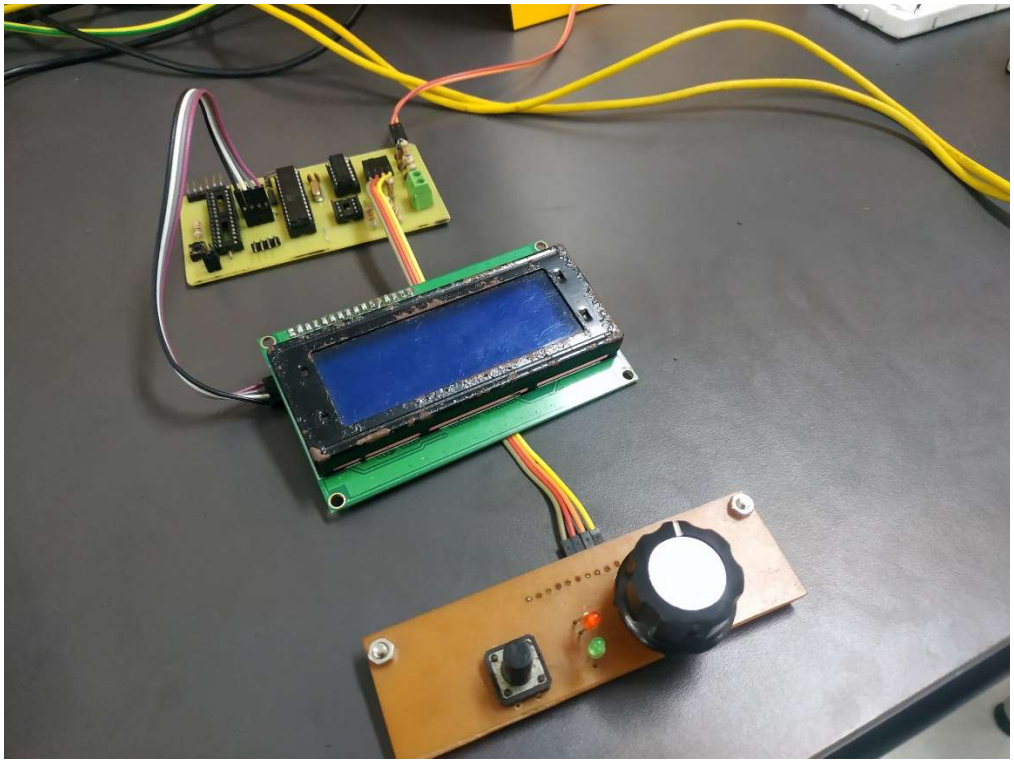
Sharma, S. S. (2008). *Power Electronics.* New Delhi: Laxmi Publications.

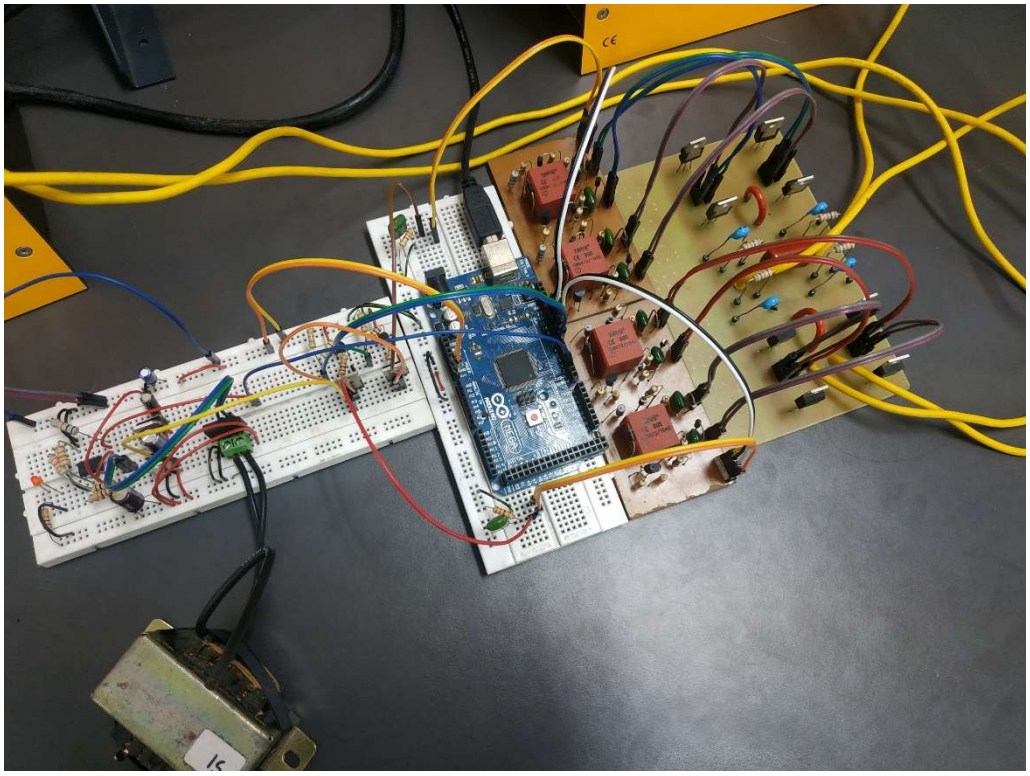
Shepherd, W., & Zhang, L. (2004). *Power Converter Circuits.* New York: Marcel Dekker, Inc.

Singh, M. D., & Khanchandani, K. B. (2007). *Power Electronics Second Edition.* New Delhi: Tata McGraw-Hill Publishing Company.

LAMPIRAN

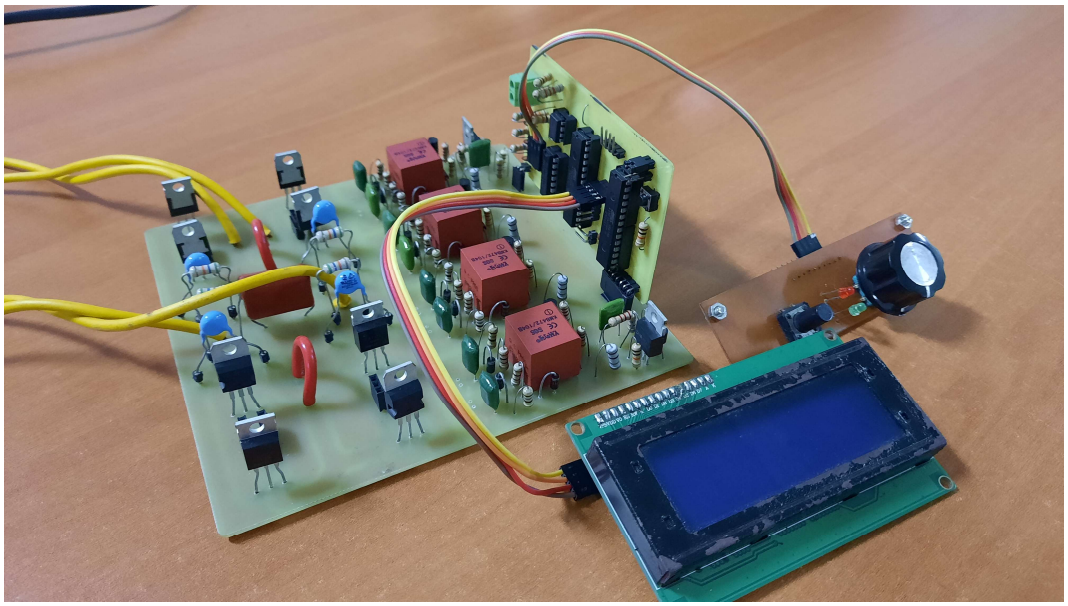
Lampiran I. Dokumentasi Pengujian Alat






DE LORENZO
TRUE RMS
PRAKTIKUM


JURNAL LENGKAP
ELEKTRONIKA DAYA
SEMESTER AKHIR 2018/2019
NAMA
NIM
Mem



Lampiran II. Listing kode pemrograman mikrokontroler ATMega328P

- ATMega328P I :

```
#include <LiquidCrystal_I2C.h>

#define on_off 9
#define sw_Pin 5
#define clk_Pin 3
#define dt_Pin 4

LiquidCrystal_I2C lcd(0x27, 20, 4);

volatile int step_pos = 0;
volatile int pos;
static int pos_lama = 0;
volatile byte sw = 0;

volatile int counter = 0;
int counter_lama = 1;

int pengulang;
int sisa;
int batas_ss;

bool set_timer;
bool set_off = false;
bool flag_timer = false;
bool off_state = false;
bool latch_menu = false;

byte menu = 0;
static int menu_lama = 1;
byte batas_menu;

int n_mode = 0;
byte n_pngtrn = 0;
int n_pilihan = 0;
byte z;
```

```
volatile byte freq;
byte freq_lama = 0;
byte freq_set = 5;

byte durasi;
float durasi_ss;
byte durasi_set = 15;

byte alpha = 135;
byte alpha_set = 90;
String s_alpha;
byte n_char_alpha;
byte sym_derajat = B11011111;

byte fr;
byte fr_set;
byte fr_lama = 0;
int delta_freq;

uint8_t dt_Freq;
uint8_t dt_Fr;
uint8_t dt_Alpha[3];
byte n_Alpha[3];

String mode[]     = {"Manual", "Softstart/stop"};
String kond_mtr[]  = {"Stop", "Running"};
String op_mtr[]    = {"Hidupkan Motor", "Matikan Motor"};
String pngtrn[][5] = {{"", "Ubah Frekuensi", "Ubah Sudut Picu", "Ubah Mode", "Matikan Alat"},
                      {"", "Ubah D S.startstop","Ubah Sudut Picu", "Ubah Mode", "Matikan Alat"}
};

void setup()
{
  Serial.begin(9600);

  lcd.init();
  lcd.clear();
  lcd.backlight();
```

```
  pinMode(on_off, OUTPUT);
  pinMode(clk_Pin, INPUT);
  pinMode(dt_Pin, INPUT);
  pinMode(sw_Pin, INPUT_PULLUP);

  digitalWrite(on_off, LOW);


  noInterrupts();
  PCICR   |= (1 << PCIE2);
  PCMSK2  |= (1 << PCINT21);

  TCCR1A  = 0;
  TCCR1B  = 0;
  TCNT1 = 0;
  OCR1A = 15625;
  TCCR1B |= (1 << CS12) | (1 << CS10); //prescaler 1024
  interrupts();

  attachInterrupt(digitalPinToInterrupt(clk_Pin), r_encoder, CHANGE);
}

ISR (PCINT2_vect)
{
  if(!digitalRead(sw_Pin) && !latch_menu)
  {
    sw++;
    step_pos = 0;
    pos = 0;

    if (menu == 4 && sw == 5)
    {
      switch (n_pngtrn)
      {
        case 0: // matikan/hidupkan motor
          fr_set = (fr_set + 1) % 2;
          menu = 3;
          break;
```

```
      case 1: // ubah freq/durasi ss
        menu = 1;
        break;

      case 2: // ubah sudut picu
        menu = 2;
        break;

      case 3: // pilih mode
        freq_set = freq;
        menu = 0;
        break;

      case 4: // matikan sistem
        fr_set = 0;
        set_off = true;
        menu = 5;
        break;
    }
    sw = menu;
    n_pngtrn = 0;
    n_pilihan = 0;
    z = 0;
   }
   else
   {
    menu = sw;
   }
  }
}

void r_encoder()
{
  if (digitalRead(clk_Pin) != digitalRead(dt_Pin))
  {
   step_pos++;
  }
  else
  {
```

```
   step_pos--;
  }

  if (step_pos >= 2)
  {
   pos++;
   step_pos = 0;
  }
  if (step_pos <= -2)
  {
   pos--;
   step_pos = 0;
  }
}

ISR (TIMER1_COMPA_vect) //ISR untuk softstart/stop
{
  if (OCR1B == 0 && counter == pengulang && !flag_timer)
  {
   freq  = freq + (delta_freq);
   counter = 0;
  }
  else
  {
   counter++;
   TCNT1 = 0;
  }
}

ISR (TIMER1_COMPB_vect)
{
  if (counter == pengulang && !flag_timer)
  {
   freq  = freq + (delta_freq);
   counter = 0;
   TCNT1 = 0;
  }
}
void loop()
```

```
{

  if (menu_lama != menu || (pos_lama != pos && menu != 3) || freq != freq_lama || (counter != counter_lama
&& flag_timer))
  {
   if(menu_lama != menu)
   {
    lcd.clear();
   }

   switch (menu)
   {
    case 0:
     fr_set = 1;
     pilih_Mode();
     break;

    case 1:
     if (n_mode == 0)
     {
      atur_Freq();;
     }
     else
     {
      atur_Time_ss();
     }
     break;

    case 2:
     atur_Alpha();
     break;

    case 3:
     operate_Motor();
     kirim_Data();
     monitoring();
     break;

    case 4:
     pengaturan();
```

```
      break;

    case 5:
      matikan_Motor();
      kirim_Data();
      matikan_Sys();
      break;
   }
   menu_lama = menu;
   pos_lama = pos;
   freq_lama = freq;
   counter_lama = counter;
  }
}

int encoder(int var, int b_bawah, int b_atas)
{
 noInterrupts();
 var = var + pos;
 pos = 0;
 interrupts();
 var = constrain(var, b_bawah, b_atas);
 return var;
}

void operate_Motor()
{
 alpha = alpha_set;

 if (n_mode == 0)
 {
  freq = freq_set;
  fr_lama = fr_set;
  fr = fr_set;
 }

 else if (n_mode == 1)
 {
  durasi = durasi_set;
```

```
    durasi_ss = durasi_set;
    if (fr_set != fr_lama)
    {
     latch_menu = true;

     if (fr_set == 1)
     {
      fr = 1;
      freq = 5;
      delta_freq = -1;
      batas_ss = 1;
     }
     else if (fr_set == 0)
     {
      freq += 1;
      delta_freq = 1;
      batas_ss = 5 + 1;
     }

     freq_set = freq;
     pengulang = durasi_ss / abs(batas_ss - freq_set);
     OCR1B = ((durasi_ss / (abs(batas_ss - freq_set))) - pengulang) * 15625 ;
     set_timer = 1;
     fr_lama = fr_set;
    }

    if (freq == batas_ss)
    {
     TIMSK1 &= (0 << OCIE1A) | (0 << OCIE1B);
     latch_menu = false;
     if (freq > 5)
     {
      freq = 5;
      fr = 0;
     }
    }
   }
}
```

```
void matikan_Motor()
{
 while (set_off)
 {
  if (fr_lama == 1 && n_mode == 1)
  {
   freq += 1;
   delta_freq = 1;
   batas_ss = 5 + 1;
   freq_set = freq;
   pengulang = durasi_ss / abs(batas_ss - freq_set);
   OCR1B = ((durasi_ss / (abs(batas_ss - freq_set))) - pengulang) * 15625 ;
  }
  else
  {
   if (fr_lama == 1 && n_mode == 0)
   {
    freq = 5;
   }
   pengulang = 3;
   OCR1B = 0;
   batas_ss = 4;
   flag_timer = true;
  }
  set_timer = 1;
  set_off = false;
 }

 if (freq == batas_ss && !flag_timer)
 {
  TIMSK1 &= (0 << OCIE1A) | (0 << OCIE1B);
  if (freq > 5)
  {
   freq = 5;
   fr = 0;
   off_state = true;
  }
 }
 else if (counter == batas_ss && flag_timer)
```

```
  {
   TIMSK1 &= (0 << OCIE1A) | (0 << OCIE1B);
   fr = 0;
   off_state = true;
  }
}

void kirim_Data()
{
  dt_Freq = freq + '0';
  dt_Fr = fr + '0';
  n_Alpha[2] = alpha;

  Serial.write('#');
  Serial.write(dt_Freq);
  Serial.write('#');
  Serial.write(dt_Fr);
  Serial.write('#');
  for (int n = 2; n > -1; n--)
  {
   byte bagi = pow(10,n);
   dt_Alpha[n] = ((n_Alpha[n]/bagi)) + '0';

   if (n > 0)
   {
    n_Alpha[n-1] = n_Alpha[n]%bagi;
   }

   Serial.write(dt_Alpha[n]);
  }
  Serial.write('@');
}

void pilih_Mode()
{
  n_mode = encoder(n_mode, 0, 1);

  lcd.setCursor(0, 0);
  lcd.print("Pilih Mode :");
```

```
  lcd.setCursor(2, 1);
  lcd.print(mode[0]);
  lcd.setCursor(2, 2);
  lcd.print(mode[1]);

  lcd.setCursor(0, 1);
  lcd.print(" "); //clear kolom

  lcd.setCursor(0, 2);
  lcd.print(" ");
  lcd.setCursor(0, (n_mode + 1));
  lcd.print(">");
}

void atur_Freq()
{
  freq_set = encoder(freq_set, 1, 5);

  lcd.setCursor(0, 0);
  lcd.print("Atur Frekuensi");
  lcd.setCursor(0, 1);
  lcd.print("Kerja :");
  lcd.setCursor(9, 3);
  lcd.print("F/");

  lcd.setCursor(11, 3);
  lcd.print(" "); //clear kolom

  lcd.setCursor(11, 3);
  lcd.print(freq_set);
}

void atur_Time_ss()
{
  durasi_set = encoder(durasi_set, 10, 20);

  lcd.setCursor(0, 0);
  lcd.print("Atur Durasi");
  lcd.setCursor(0, 1);
```

```
  lcd.print("Softstart/stop :");

  lcd.setCursor(6, 3);
  lcd.print("  "); //clear kolom
  lcd.setCursor(6, 3);
  lcd.print(durasi_set);
  lcd.setCursor(9, 3);
  lcd.print("detik");
}

void atur_Alpha()
{
  alpha_set = encoder(alpha_set, 10, 170);

  s_alpha = String (alpha_set);
  n_char_alpha = s_alpha.length();

  lcd.setCursor(0, 0);
  lcd.print("Atur Sudut Picu :");

  lcd.setCursor(8,3);
  lcd.print("     ");
  lcd.setCursor((11-n_char_alpha), 3);
  lcd.print(alpha_set);
  lcd.setCursor(11, 3);
  lcd.write(sym_derajat);
}

void monitoring()
{
  lcd.setCursor(0, 0);
  lcd.print("-----MONITORING-----");

  lcd.setCursor(0, 1);
  lcd.print("MODE :");
  lcd.setCursor(6, 1);
  lcd.print(mode[n_mode]);

  lcd.setCursor(0, 2);
```

```
  lcd.print("FREQ :");
  lcd.setCursor(6, 2);
  lcd.print("F/");
  lcd.setCursor(8, 2);
  lcd.print(" "); //clear kolom
  lcd.setCursor(8, 2);
  lcd.print(freq);

  lcd.setCursor((14-n_char_alpha), 2);
  lcd.print(alpha);
  lcd.setCursor(14, 2);
  lcd.write(sym_derajat);

  lcd.setCursor(0, 3);
  lcd.print("MOTOR:");
  lcd.setCursor(6, 3);
  lcd.print("       ");
  lcd.setCursor(6, 3);
  lcd.print(kond_mtr[fr]);

  if (n_mode == 1)
  {
    lcd.setCursor((17), 2);
    lcd.print(durasi);
    lcd.setCursor(19, 2);
    lcd.print("s");

    if (set_timer)
    {
      set_timer = 0;
      noInterrupts();
      TIMSK1 |= (1 << OCIE1A) | (1 << OCIE1B);
      counter  = -1;
      interrupts();
    }
  }
}

void pengaturan()
```

```
{
 for (int  x = 0; x < 2; x++)
 {
  pngtrn[x][0] = op_mtr[fr];
 }

 n_pilihan = encoder(n_pilihan, -1, 4);

 if (n_pilihan > 3)
 {
  if (z == 0)
  {
   lcd.clear();
  }
  z = 1;
  n_pilihan = 3;
 }
 else if (n_pilihan < 0 )
 {
  if (z == 1)
  {
   lcd.clear();
  }
  z = 0;
  n_pilihan = 0;
 }

 for (int y = 0; y <= 3; y++)
 {
  lcd.setCursor(2, y);
  lcd.print(pngtrn[n_mode][(y+z)]);
  lcd.setCursor(0, y);
  lcd.print(" "); //clear kolom
 }
 lcd.setCursor(0, n_pilihan);
 lcd.print(">");

 n_pngtrn = n_pilihan + z;
 lcd.setCursor(19, 3);
```

```
  lcd.print(n_pngtrn);
}

void matikan_Sys()
{
  lcd.setCursor(6, 1);
  lcd.print("Mematikan");
  lcd.setCursor(7, 2);
  lcd.print("Sistem");

  if (set_timer)
  {
    set_timer = 0;
    noInterrupts();
    TIMSK1 |= (1 << OCIE1A) | (1 << OCIE1B);
    interrupts();
    counter = -1;
  }

  if (off_state)
  {
    digitalWrite(on_off, HIGH);
  }
}
```

- ATMega328P II :

```
#define ZCD_Pin 2

const int scr_Pin[][2] = { {12, 11}, {10, 9} };
const int conv_Pin[2] = {8, 7};

volatile byte freq = 5; //frekuensi output f/5 - f/1
volatile bool fr = false; //set untuk firing atau tidak
volatile uint16_t alpha = 1500; //range 0 - 2500 (0 - 180)
uint16_t cal = 20; // kalibrasi utk ZCD yang leading

byte ZCD_Cycle; // siklus dari P/N dari sinyal ZCD
byte freq_Cycle = 0; // jumlah siklus P/N dari sinyal keluaran
bool conv_Type = 0; // jenis converter yang aktif (Conv P / N)
```

```
String data;
char dt_Char;
int dt_Freq; // data-data yang diterima dari arduino #1
bool dt_Fr;
int dt_Alpha;
int arr_dt_Alpha[3];

void setup()
{
 Serial.begin(9600);

 for (int i = 0; i < 2; i++)
 {
  for (int j = 0; j < 2; j++)
  {
   pinMode(scr_Pin[i][j], OUTPUT);
   digitalWrite(scr_Pin[i][j], LOW);
  }
 }

 for (int i = 0; i < 2; i++)
 {
  pinMode(conv_Pin[i], OUTPUT);
  digitalWrite(conv_Pin[i], LOW);
 }

 dt_Freq = freq;
 dt_Fr = fr;
 dt_Alpha = alpha;

 noInterrupts();
 TCCR1A = 0;
 TCCR1B = 0;
 TCCR1B |= (1 << CS11) | (1 << CS10); // atur prescaler ke-64
 interrupts();

 attachInterrupt(digitalPinToInterrupt(ZCD_Pin), ZCD, CHANGE);
}
```

```
void ZCD()
{
 if (fr)
 {
  if (digitalRead(ZCD_Pin))
  {
   ZCD_Cycle = 0; //ZCD berada pada siklus positif
  }
  else
  {
   ZCD_Cycle = 1; // ZCD berada pada siklus negatif
  }

  OCR1A = alpha;
  TCNT1 = 0; // mengatur timer dimulai dari 0
  TIMSK1 |= (1 << OCIE1A); // aktifkan timer A

  digitalWrite(conv_Pin[conv_Type], HIGH);
 }

 else
 {
  digitalWrite(conv_Pin[conv_Type], LOW);
  freq_Cycle = 0;
 }
}

ISR (TIMER1_COMPA_vect)
{
 TIMSK1 &= (0 << OCIE1A); // non-aktifkan timer A

 digitalWrite(scr_Pin[conv_Type][ZCD_Cycle], HIGH);
 delayMicroseconds(50);
 digitalWrite(scr_Pin[conv_Type][ZCD_Cycle], LOW);

 freq_Cycle = freq_Cycle + 1;

 if(freq_Cycle >= freq)
 {
```

```
    digitalWrite(conv_Pin[conv_Type], LOW);
    conv_Type = !conv_Type;
    freq_Cycle = 0;
   }
}

void loop()
{
  while (Serial.available())
  {
    dt_Char = Serial.read();
    data += dt_Char;

    if(dt_Char == '@')
    {
      if(data.length() == 9 && data[0] == '#')
      {
        dt_Freq = data[1] - '0';
        dt_Fr = data[3] - '0';
        dt_Alpha = 0;
        for (int n = 0; n < 3; n++)
        {
          arr_dt_Alpha[n] = data[n+5] - '0';
          dt_Alpha = dt_Alpha + arr_dt_Alpha[n] * pow(10,(2-n));
        }
        dt_Alpha = map(dt_Alpha, 0, 180, 0, 2500);
        data = "";
      }
      else
      {
        data = "";
      }
    }
  }

  if(dt_Freq != freq || dt_Fr != fr || dt_Alpha != alpha)
  {
    noInterrupts();
    freq = dt_Freq;
```

```
    fr = dt_Fr;
    alpha = dt_Alpha;
    interrupts();
  }
}
```