BAB III

METODALOGI PENELITIAN

3.1. Alat Dan Bahan

3.1.1. Alat

1. Solder listrik
2. *Soldering pump*
3. *Tool set*
4. Bor PCB
5. Timah
6. Multimeter

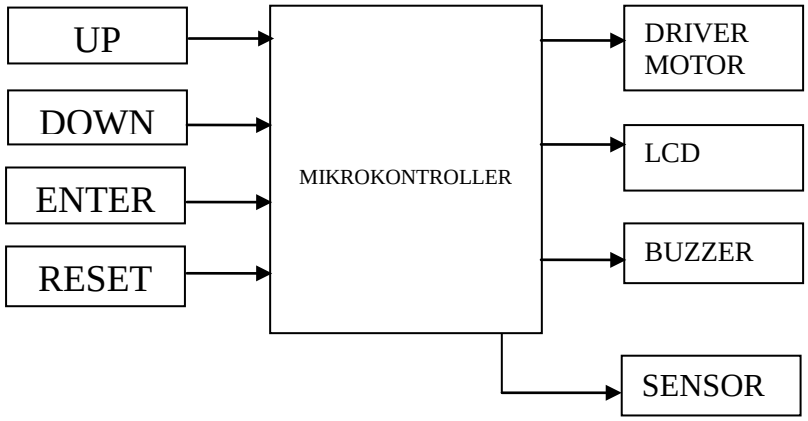
3.1.2. Bahan

1. *LCD*
2. *SSR*
3. Motor *AC*
4. Lampu *LED*
5. Capasitor
6. Transistor
7. Diode
8. IC *ATMega8*
9. Potensio
10. *Push button*
11. Socket *IC ATMega8*

12. Sakelar *On/Off*

3.2. Diagram Blok Sistem

Setting timer berfungsi sebagai pengatur waktu berapa lama untuk memutar sampel, setelah mengatur semua tekan *enter/ok*. Sampel yang digunakan pada *centrifuge* yaitu urine dan darah. Fungsi dari motor untuk memutar sampel sesuai dengan berapa kecepatan yang di*setting* hingga terjadi gaya *centrifugal*. Kecepatan pada motor dikendalikan oleh *driver* motor. Sensor Rpm digunakan untuk melihat atau membaca kecepatan motor dan hasil *output* dari sensor berupa tegangan tersebut akan masuk pada pin *adc* 0 mikrokontroler, pada mikrokontroler data akan diolah dan selanjutnya akan ditampilkan di *display*.

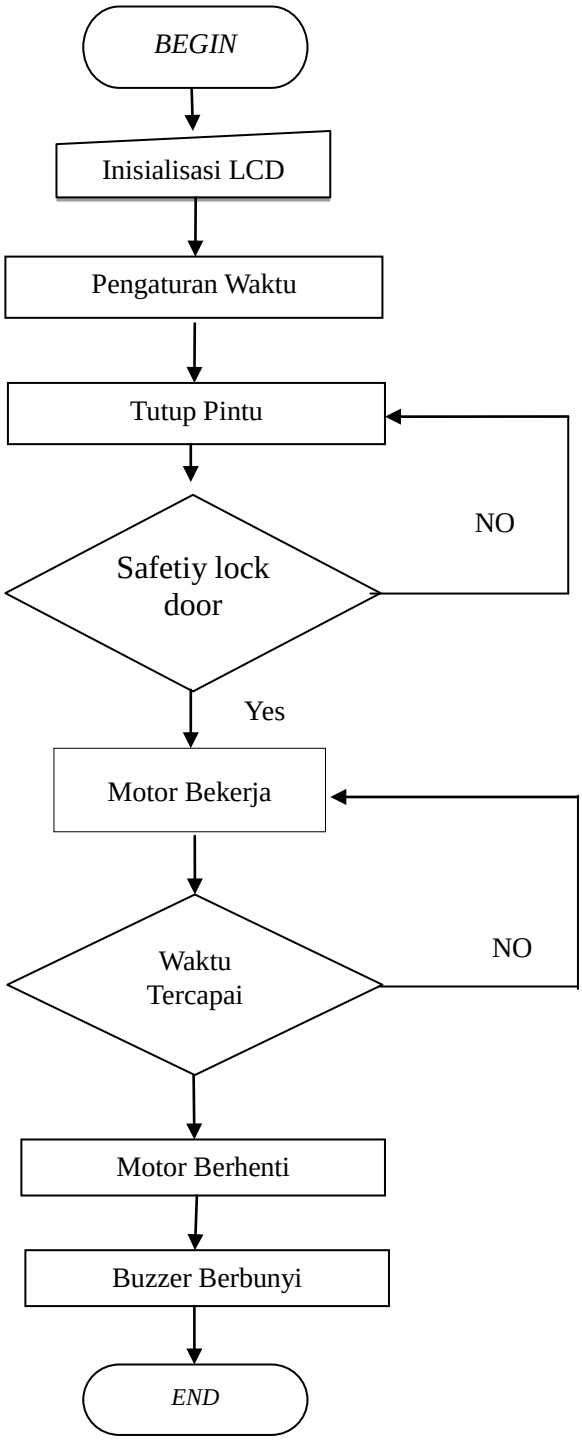


Gambar 2.2. Diagram Blok

3.3. Diagram Alir Program

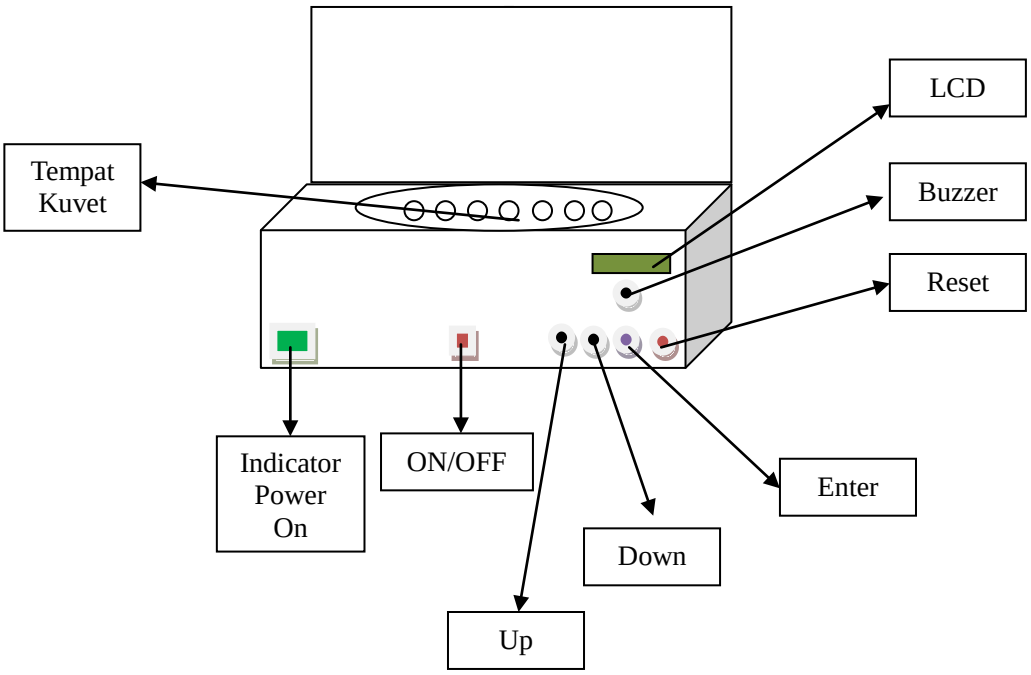
Proses akan dimulai saat alat on maka ditampilan *LCD setting timer*, setelah itu menutup pintu jika pintu belum tertutup maka motor tidak bekerja, setelah itu setting *timer* lalu tekan tombol *enter* yang akan

mengaktifkan timer dan motor bekerja, setelah waktu tercapai maka motor akan berhenti dan *buzzer* akan berbunyi.



Gambar 3.3. Diagram Alir

3.4. Diagram Mekanis

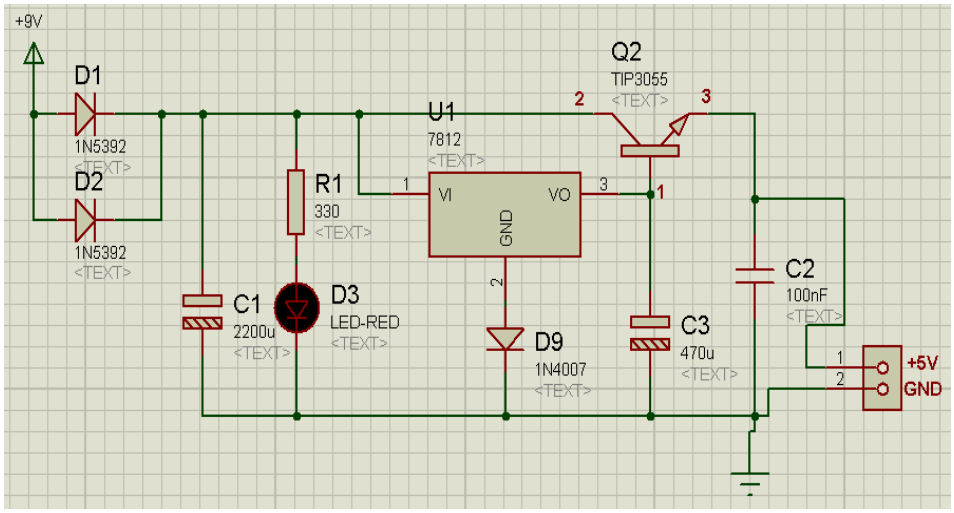


Gambar 3.3. DiagramMekanis

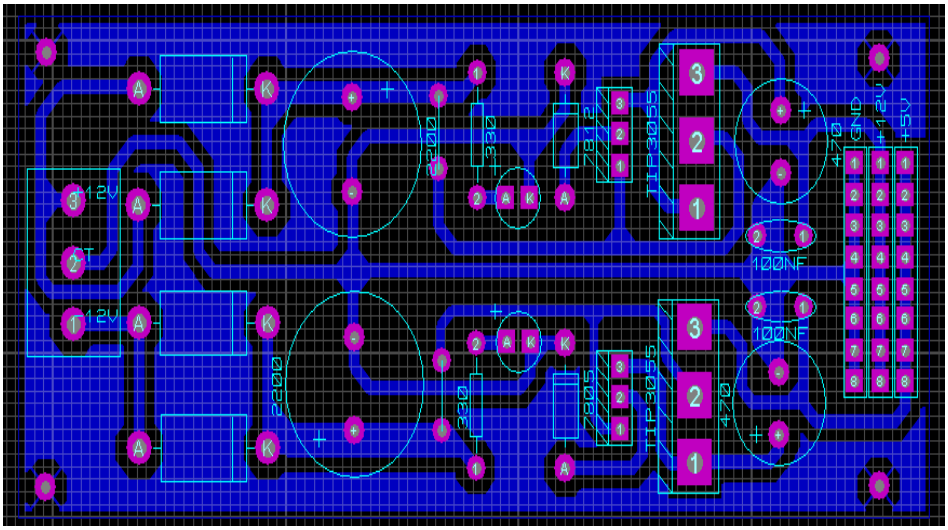
1. Pengaman *(Fuse)*
2. *Buzzer*
3. Tombol *power*
4. *LCD display*
5. Tombol *UP*
6. Tombol *ENTER*
7. Tombol *DOWN*
8. Tombol *RESET*
9. Motor

3.5. Rangkaian *Power Supply*

Rangkaian power supply pada modul ini brfungsi sebagai *supply* tegangan ke semua rangkaian yang menggunakan tegangan DC. Prinsip kerja *power supply* adalah merubah tegangan AC menjadi tegangan DC. Skematik *Power supply* dapat dilihat di Gambar 3.4. dan *layout power supply* dapat dilihat di Gambar 3.5.



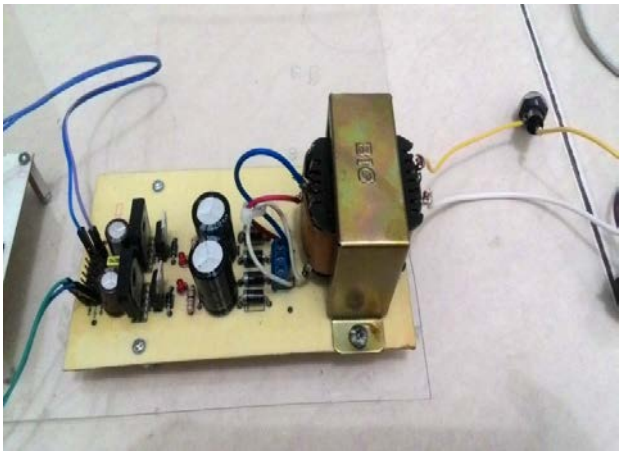
Gambar 3.4. Skematik *Power Supply.*



Gambar 3.5. *Layout Power Supply.*

3.6. Gambar *Power Supply*

Untuk gambar *power supply* dapat dilihat pada gambar di bawah ini:



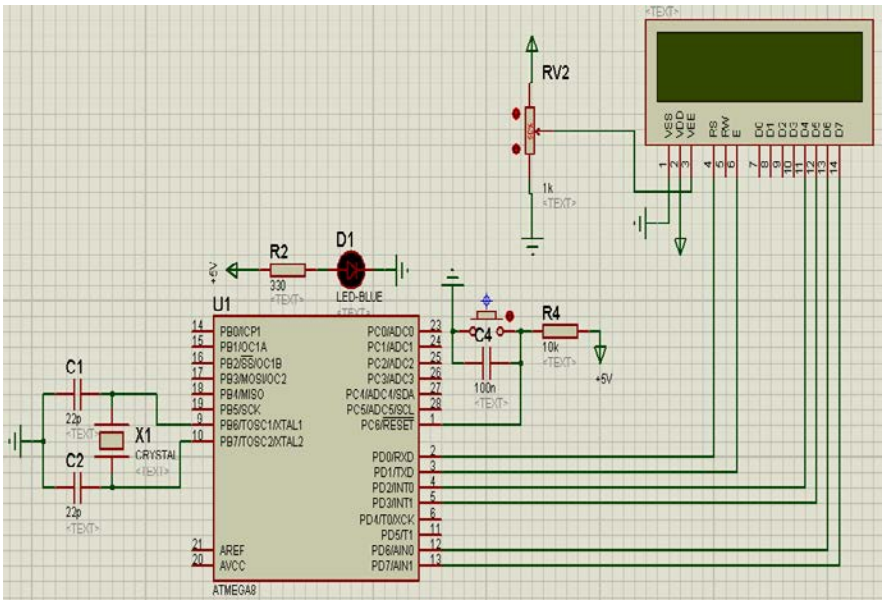
Gambar 3.6. *Power Supply*

Rangkaian *power supply* pada modul ini berfungsi sebagai *supply* tegangan ke semua rangkain yang menggunakan tegangan *DC*. Prinsip kerja *power supply* adalah mengubah tegangan *AC* menjadi tegangan *DC* dengan menggunakan *transformator* sebagai penurun tegangan dan dioda sebagai komponen yang berfungsi sebagai penyearah tegangan. Pada modul ini *power supply* akan mengubah tagangan *AC* menjadi DC sebesar 5 *VDC* dan 12 *VDC* dengan mengunakan *ICregulator* 7805 dan 7809. Adapun tegangan 5 *VDC* digunakan untuk rangkaian minimum sistem sedangkan tegangan 12VDC bisa digunakan sewaktu-waktu ketika diperlukan oleh SSR.

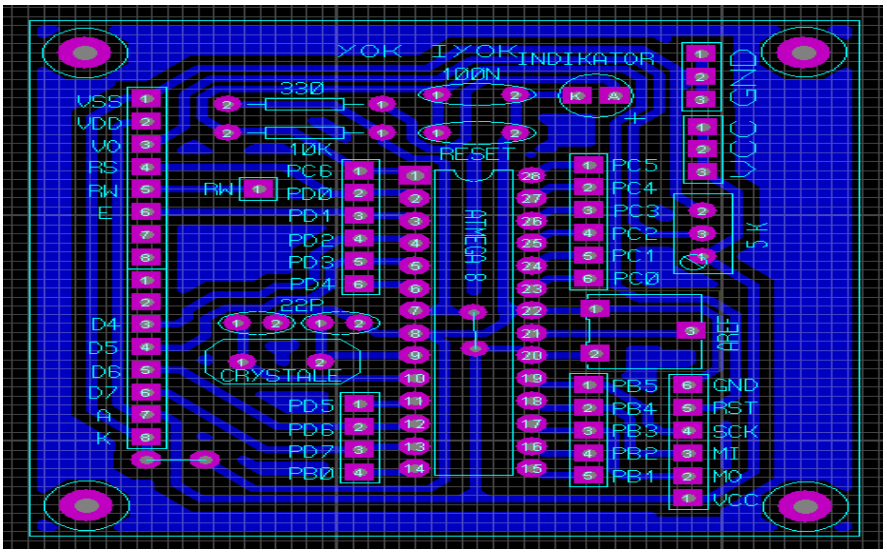
3.7. Rangkaian Minimum Sistem

Minimum sistem berfungsi sebagai kontrol kerja atau otak dari alat secara keseluruhan. Cara kerja rangkaian minimum sistem ini dengan

memanfaatkan kapasitas penyimpanan yang dimiliki oleh IC ATMega8. Skematik minimum sistem dapat dilihat di Gambar 3.7. dan*layout* minimum sistem dapat dilihat di Gambar.

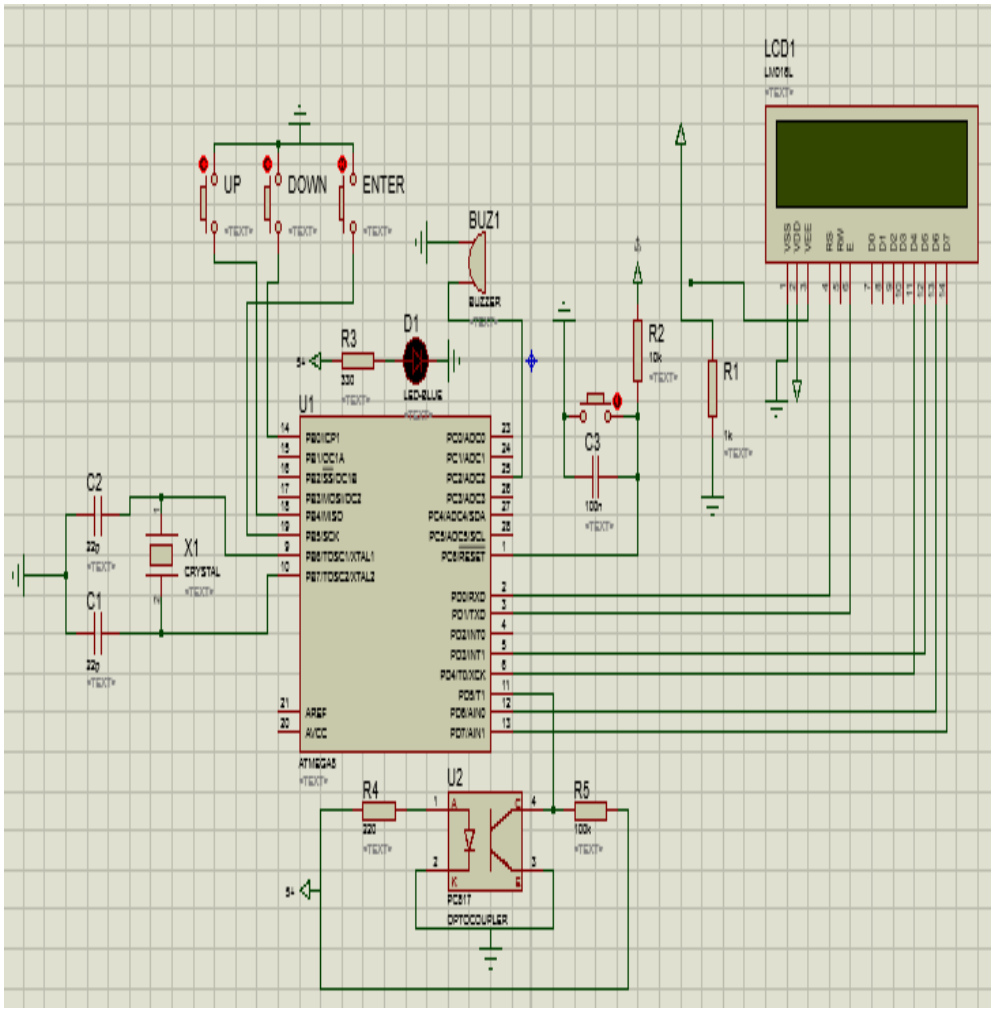


Gambar 3.7. *Skematik Minimum System.*



Gambar 3.8. *LayoutMinimum System.*

3.8. Rangkaian Keseluruhan



Gambar 3.10. Rangkaian Keseluruhan

3.9. PROGRAM

Pembuatan program pada modul menggunakan codevisionAVR seperti dibawah ini.

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 24/08/2016
Author  :
Company :
Comments:


Chip type               : ATmega8
Program type            : Application
AVR Core Clock frequency: 1,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega8.h>
#include <stdlib.h>
#include <delay.h>
#include <alcd.h>
unsigned int mikrodetik, data, data1=0, detik;
unsigned char i=0;
float pulsa,frekuensi=0;
int menit;
bit a=0;

unsigned char temp[2],temp2[2],temp3[4];
// Timer2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here

}
```

```
// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x9E;
// Place your code here
mikrodetik++;
if(mikrodetik==10)
{
 pulsa=frekuensi*60;
 TCNT1=0;
 if (detik==0)
 {menit--;detik=59;}else{detik--;}
 mikrodetik=0;
}
}

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input
voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

void set_timer()
{
 if(PINB.4==0)
 {menit=menit+10;if(menit>60){menit=0;}delay_ms(200);}
 else if(PINB.0==0)
 {menit=menit-10;if(menit<0){menit=0;}delay_ms(200);}
}
void tampilkan_timer()
{
 lcd_gotoxy(0,0);
 lcd_putsf("TIMER:");
```

```
if(menit<10)
 {
 lcd_gotoxy(6,0);
 lcd_putsf("0");
 lcd_gotoxy(7,0);
 itoa(menit,temp);
 lcd_puts(temp);}
 else
 {
 lcd_gotoxy(6,0);
 itoa(menit,temp);
 lcd_puts(temp);}

 if(detik<10)
 {
 lcd_gotoxy(9,0);
 lcd_putsf("0");
 lcd_gotoxy(10,0);
 itoa(detik,temp2);
 lcd_puts(temp2);}
 else
 {
 lcd_gotoxy(9,0);
 itoa(detik,temp);
 lcd_puts(temp);}

 lcd_gotoxy(8,0);
 lcd_putsf(":");

}

void start_stop()
{
 if(PINB.5==0)
 {a=1;}
 if (a==1&&menit>0)
 {
  TCCR0=0x05;
 }
 if(menit==0&&detik==0&&a==1)
 {
    TCCR0=0x00;
    a=0;
    lcd_clear();
    while(1)
    {
    PORTC.2=1;
    PORTD.2=0;
    TCCR0=0x00;
    TCCR1A=0x00;
    TCCR1B=0x00;
    TCCR2=0x00;
```

```
lcd_gotoxy(5,0);
    lcd_putsf("SELESAI");

    lcd_gotoxy(0,1);
    lcd_putsf("BY.AGRIANSYAH");
    }}}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=In
Func1=In Func0=In
// State7=T State6=T State5=P State4=P State3=0 State2=T
State1=T State0=P
PORTB=0x31;
DDRB=0x08;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In
Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T
State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 0,977 kHz
TCCR0=0x05;
TCNT0=0x9E;

// Timer/Counter 1 initialization
// Clock source: T1 pin Falling Edge
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
```

```
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 1000,000 kHz
// Mode: Fast PWM top=0xFF
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x69;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x41;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x83;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
```

```
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD
menu:
// RS - PORTD Bit 0
// RD - PORTB Bit 7
// EN - PORTD Bit 1
// D4 - PORTD Bit 4
// D5 - PORTD Bit 3
// D6 - PORTD Bit 6
// D7 - PORTD Bit 7
// Characters/line: 16
lcd_init(16);

// Global enable interrupts
#asm("sei")

DDRD.5=0;
PORTD.5=0;
DDRD.2=1;
PORTD.2=0;
DDRB.4=0;
PORTB.4=1;
DDRB.5=0;
PORTB.5=1;
DDRC.2=1;
PORTC.2=0;
lcd_clear();
lcd_gotoxy(6,0);
lcd_putsf("");
TCCR0=0x00;
while (1)
      {
      data1=0;
      frekuensi=TCNT1;
      data=read_adc(0);
      if(a==1)
      {
       OCR2=data;
      }else{OCR2=0;}

      lcd_gotoxy(0,1);
      lcd_putsf("RPM:");
      lcd_gotoxy(4,1);
      itoa(pulsa,temp3);
      lcd_puts(temp3);
      start_stop();
      set_timer();
      tampilkan_timer();
      delay_ms(100);
      }
}
```

3.6. Perancangan Pengujian

3.6.1. Jenis Pengujian

1. Mengukur waktu *centrifuge* dengan menggunakan *stopwatch*.
2. Uji alat dengan praktek ke *sampel.*

3.6.2. Pengolahan Data

Jenis penelitian ini menggunakan metode *Pre Eksperimental* dengan jenis *"One group Post Test Design"* yaitu alat *centrifuge* ini bekerja dengan timer yang di atur kemudian motor akan berhenti apabila waktu telah tercapai kemudian proses selesai. Sehingga penulis Hanya Melihat Hasil Tanpa Mengukur Keadaan Sebelumnya.

3.7 . Variabel Penelitian

3.7.1. Variabel Bebas

Sebagai variabel bebas yaitu kecepatan Rpm motor.

1.7.2. Variabel Tergantung

Sebagai variabel tergantung yaitu pengontrol *timer.*

3.7.3. Variabel Terkendali

Variabel terkendali terdiri dari tampilan waktu yang dikendalikan oleh *Mikrokontroler ATMega8.*

3.8 . Sistematika Pengukuran

1. Rata-rata Pengukuran

Adalah nilai atau hasil pembagian dari jumlah data yang diambil atau diukur dengan banyaknya pengambilan data atau

banyaknya pengukuran dirumuskan sebagai berikut :

Rata-rata

$$\overline{x} = \frac{\sum x_n}{n} \qquad \text{.....(3.1)}$$

dengan :

$\bar{x}$ = Rata – rata

$\sum x_n$ = Jumlah $x$sebanyak n

n = Banyak data

2. Simpangan *(Error)*

Adalah selisih dari rata-rata nilai dari harga yang dikehendaki dengan nilai yang diukur dirumuskan sebagai berikut :

$$simpangan = x_n - \bar{x} \qquad \text{.....(3.2)}$$

Simpangan = Nilai error yang dihasilkan

x_n = Rata – rata data DPM

$\bar{x}$ = Rata – rata data modul

3. Persentase Error

Adalah nilai persen dari simpangan (*Error*) terhadap nilai yang dikehendaki dirumuskan sebagai berikut :

$$Persentase\ Error = \frac{simpangan}{x_n}\ x\ 100\% \qquad \text{.....(3.3)}$$

dengan :

$$Persentase\ error \quad = Besarnya\ nilai\ simpangan\ atau\ error\ dalam\ \%$$

$$x_n \quad = rata - rata\ data\ kalibrator$$

4. Standard Deviasi (SD)

Adalah suatu nilai yang menunjukkan tingkat v(derajat) variasi kelompok data atau ukuran standard penyimpanan dari rata-ratanya. Jika *standard* deviasi semakin kecil maka data tersebut semakin presesi dirumuskan sebagai berikut :

$$SD = \frac{\sqrt{\sum(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_n - \bar{x})^2}}{n - 1} \quad \text{……..(3.4)}$$

dengan :

SD = Standar deviasi

x = Data x

$\bar{x}$ = Rata-rata

n =Banyak data