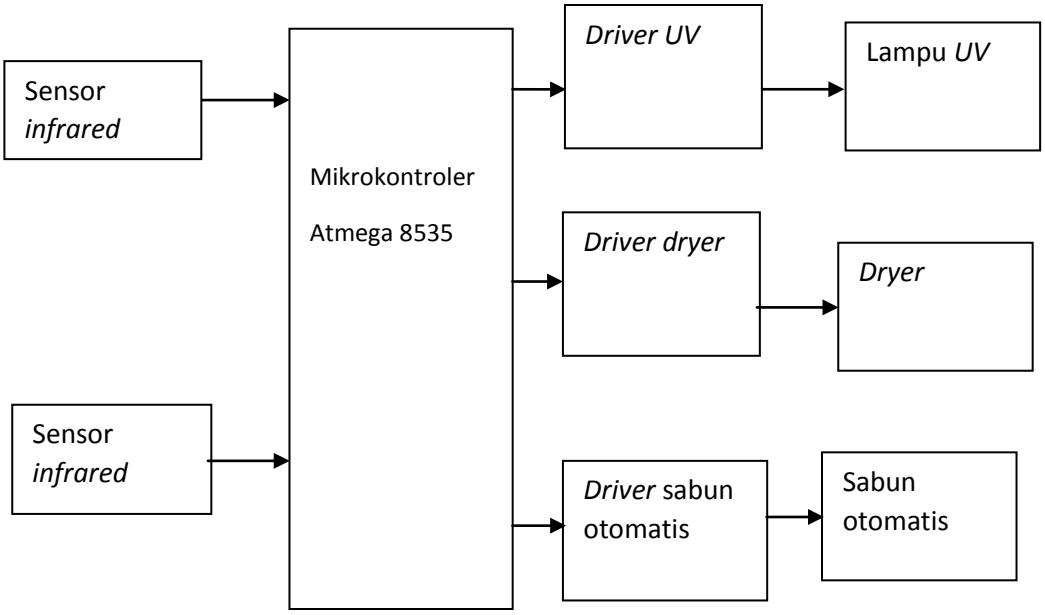
BAB III

METODOLOGI PENELITIAN

3.1. Diagram Blok

Untuk blok diagram dapat dilihat pada gambar 3.1. di bawah ini:



Gambar 3.1. Blok Diagram

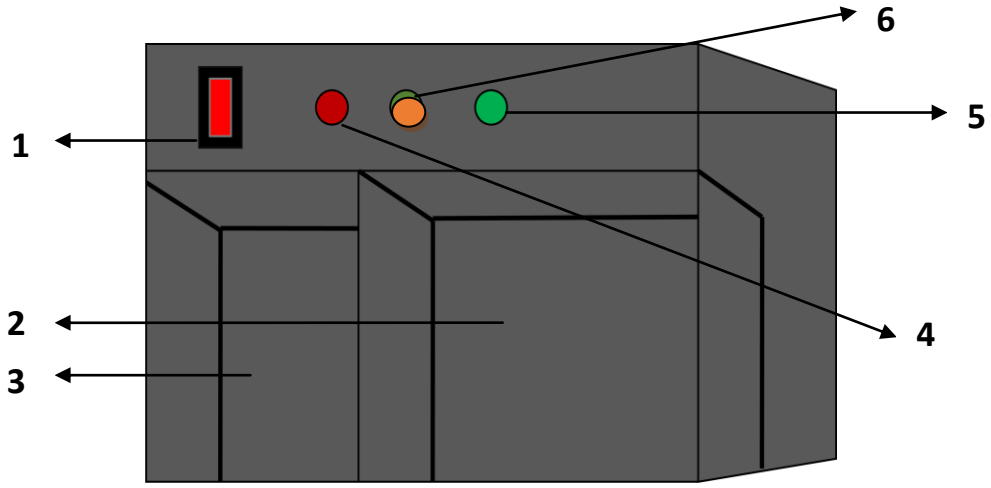
Cara kerja diagram blok

Pada saat sensor untuk sabun otomatis mendeteksi adanya objek maka tegangan *output* dari sensor akan masuk ke *ADC* mikrokontroler, kemudian mikrokontroler akan memproses untuk mengaktifkan *driver* motor *wash* untuk mengeluarkan sabun dari *box.*

Begitu juga sensor mendeteksi adanya objek (tangan) maka *output* tegangan dari sensor akan masuk ke *ADC* mikrokontroler untuk di proses, kemudian mirkokontroler akan mengaktifkan *driver heat dryer* dan *driver* lampu *UV* selama 20 detik dan setelah detik ke 20 maka *dryer* dan lampu *UV* akan mati secara bersama-sama.

3.2. Diagram Mekanis

Untuk diagram mekanis dapat dilihat pada gambar 3.2. di bawah ini:

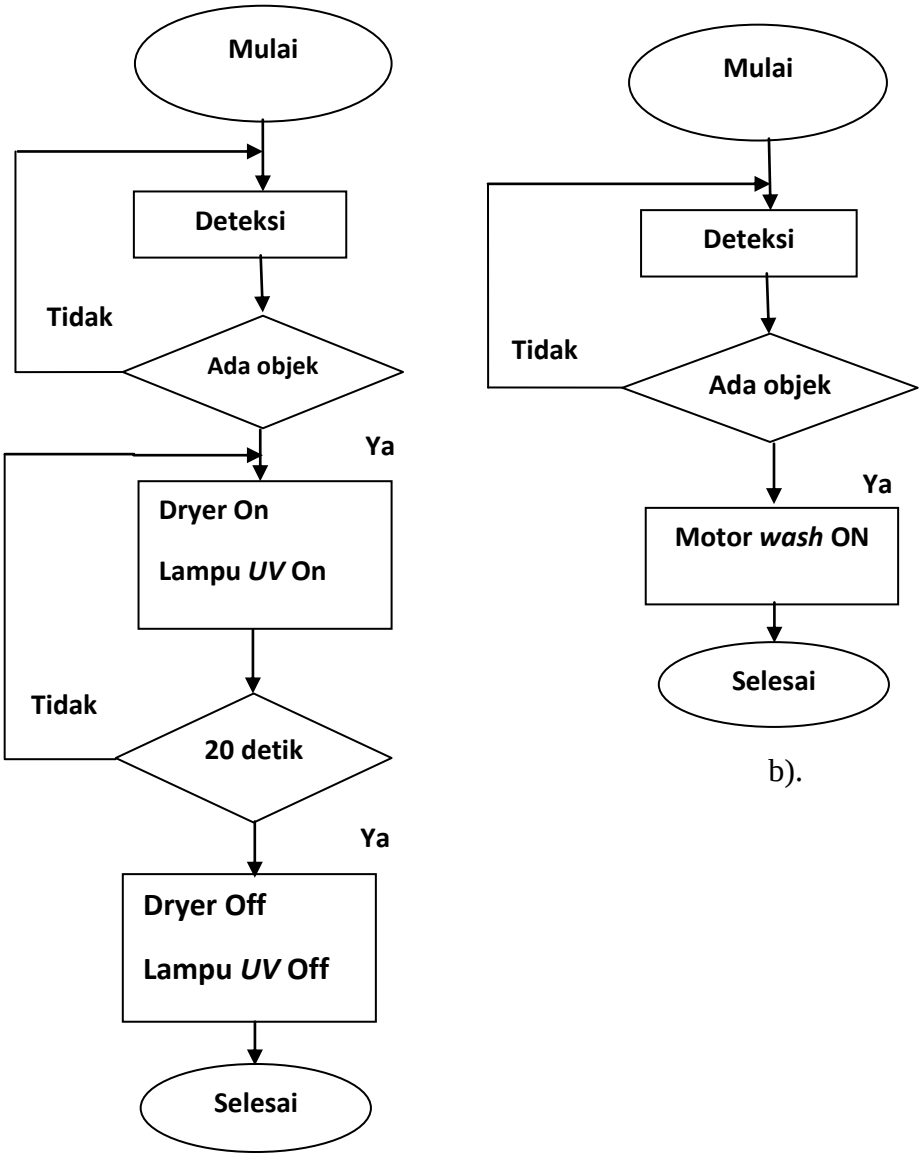


Gambar 3.2. Diagram Mekanis

1. Tombol *power*
2. Blok bagian pengering dan lampu *UV*
3. Blok bagian sabun otomatis
4. Lampu indikator *dryer*
5. Lampu indikator lampu *UV*
6. Lampu indikator sabun otomatis

3.3. Diagram Alir

Untuk gambar diagram alir dapat dilihat pada gambar 3.3. di bawwah ini :



Gambar 3.3. a). Diagram Alir Bagian Dryer dan UV,

b). Diagram Alir Bagian Sabun

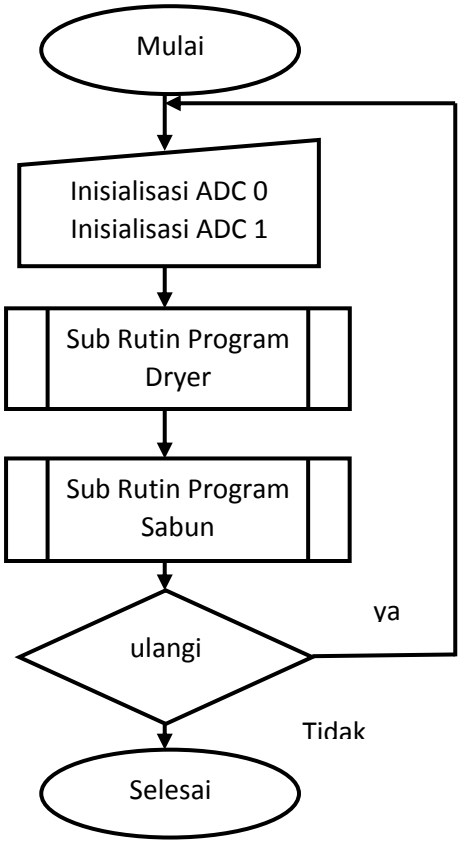
Cara kerja diagram alir

Ketika *start* terjadi proses pembacaan oleh sensor *infrared*, apabila tidak terdeteksi objek maka sensor akan terus membaca. Dan apabila terdapat objek (tangan) yang terdeteksi maka motor *wash* akan bekerja memompa sabun.

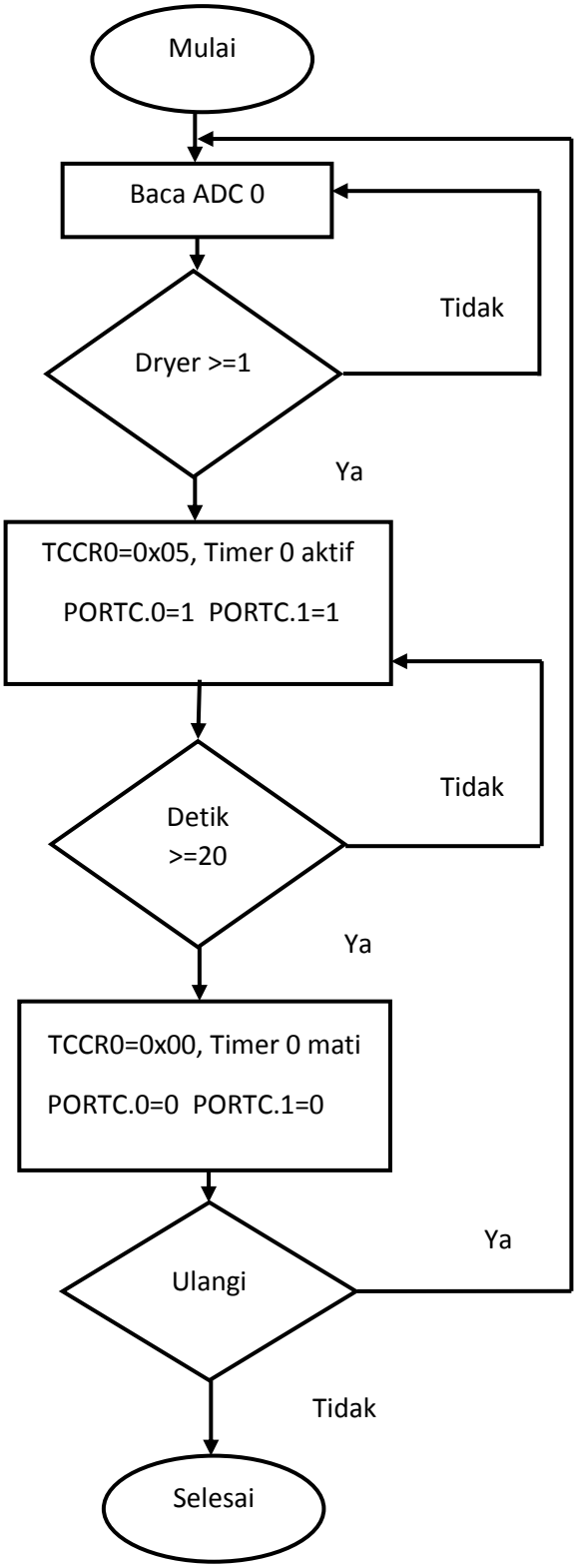
Setelah selesai mencuci tangan sensor mendeteksi objek (tangan) untuk menghidupkan *heat dryer* dan lampu *UV* selama 20 detik. Setelah waktu tercapai maka lampu *UV* dan *heat dryer* akan mati secara bersamaan.

3.4. Flowchart Program

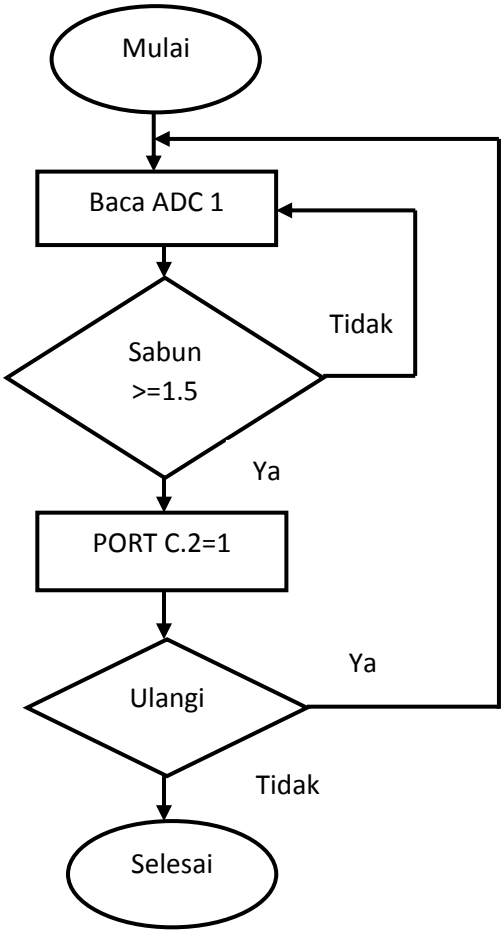
Untuk gambar flowchart program dapat dilihat pada gambar 3.4. di bawah ini :



Gambar 3.4. Flowchart Program Modul



a).



b).

Gambar 3.5. a). Flowchart Subrutin Dryer

b). Flowchart Subrutin sabun

Cara kerja program

Pada program modul, *ADC* 0 dan *ADC* 1 mulai melakukan pembacaan. Ketika tegangan pada *ADC* 0 lebih dari sama dengan 1 maka *port* C.0=1, *port* C.1=1, TCCR0=0x05 dan timer 0 akan aktif. Apabila tagangan pada *ADC* 0 kurang dari sama dengan 1 maka *ADC* 0 akan terus melakukan pembacaan. Waktu pada timer 0 diatur selama 20 detik, apabila waktu pada timer habis maka *port* C.0=0, *port* C.1=0, TCCR0=0x00 dan timer akan mati.

Begitu juga pada *ADC* 1 ketika tegangan yang masuk lebih dari sama dengan 1.5 maka *port* C.2=1. Apabila tegangan pada *ADC* 1 kurang dari sama dengan 1.5 maka *ADC* 1 akan terus melakukan pembacaan.

3.5. Perakitan Rangkaian *Driver*

3.5.1. Alat

1. Papan *pcb*
2. Solder
3. Timah
4. penyedot timah

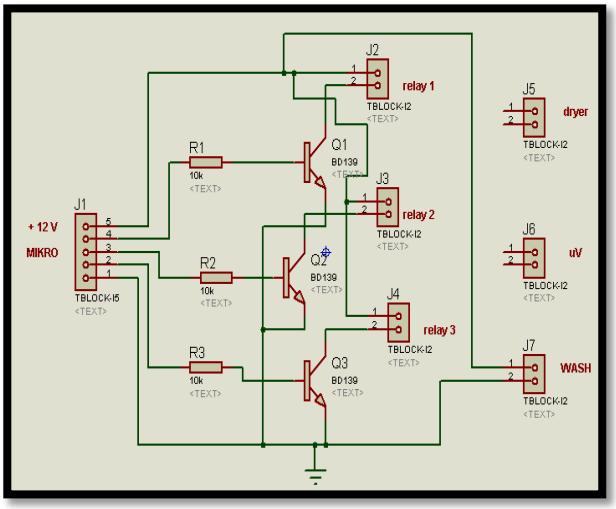
3.5.2. komponen

1. Transistor BD139
2. Resistor 1k ohm
3. *T-blok*
4. Relay 8 pin 12 VDC

3.5.3. Langkah Perakitan

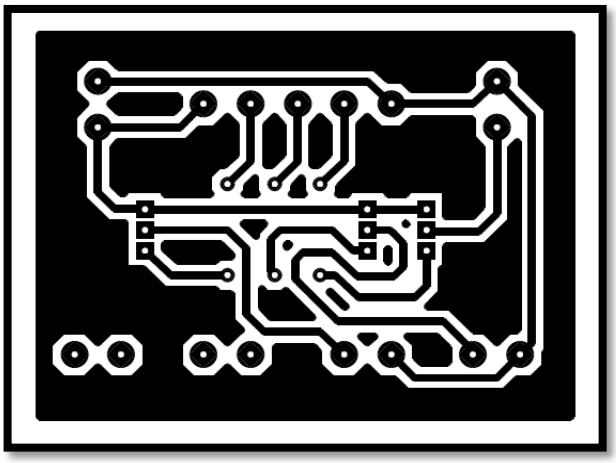
1. Rangkai sistematik rangkaian *driver* dengan mengunakan aplikasi pada laptop, aplikasi yang digunakan pada pembuatan modul ini adalah *proteus.*

   Untuk gambar sistematik rangkaian *driver* pada aplikasi dapat dilihat pada gambar 3.6. di bawah ini:



Gambar 3.6. Sistematik Rangkaian Driver

2. Setelah sistematik rangkaian jadi, tahap selanjutnya membuat *lay out* nya dan disablon ke papan *pcb*. Untuk gambar *lay out driver* pada papan *pcb* dapat dilihat pada gambar 3.7. di bawah ini:

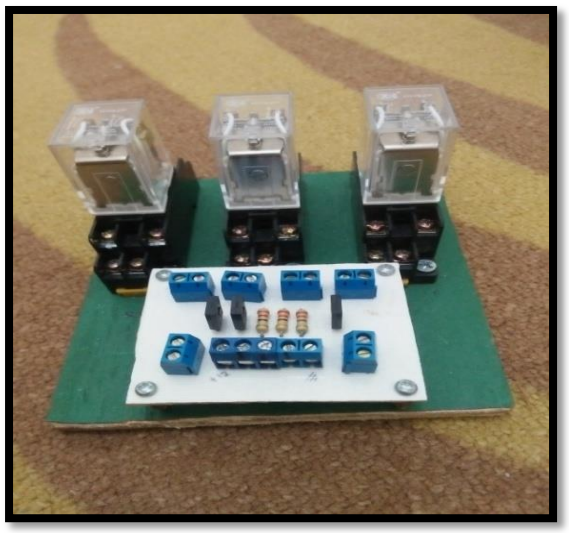


Gambar 3.7. Lay Out Driver

3. Rakit komponen yang dibutuhkan dengan menggunakan timah dan solder.

3.5.4. Gambar Rangkain *Driver*

Gambar rangkaian *driver* dapat dilihat pada gambar 3.8. di bawah ini :



Gambar 3.8. Rangkaian Driver

Rangkaian *driver* pada modul ini berfungsi sebagai kontak dari tegangan DC ke tegangan AC. Prinsip kerjanya dengan memanfaatkan fungsi kerja transistor BD139 yaitu, ketika kaki *basis* mendapat tegangan lebih dari 0,7 maka akan saturasi sehingga dapat menghidupkan *relay* 12 VDC dengan kontak AC. Dan ketika *basis* mendapat tegangan kurang dari 0,7 maka akan *cut off* sehingga *relay* akan mati karena terputus dengan *ground.*

3.6. Perakitan Rangkaian Minimum Sistem

3.6.1. Alat

1. Papan *pcb*
2. Solder
3. Timah
4. Penyedot timah

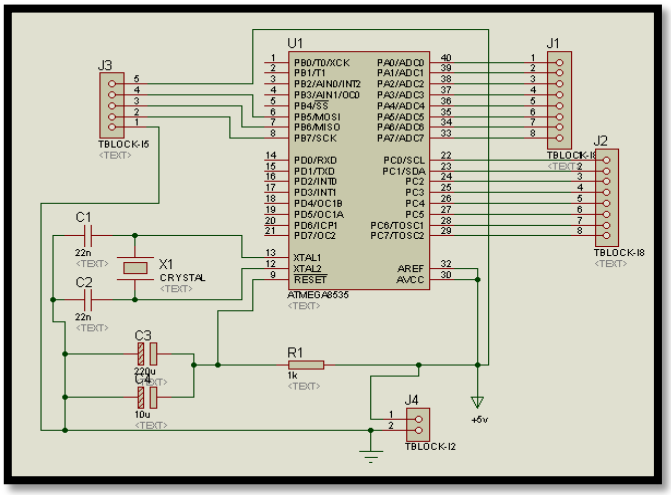
3.6.2. Komponen

1. Atmega 8535
2. Kapasitor 10 µf 25 v
3. Kapasitor *non* polar
4. *Crystal*

3.6.3. Langkah perakitan

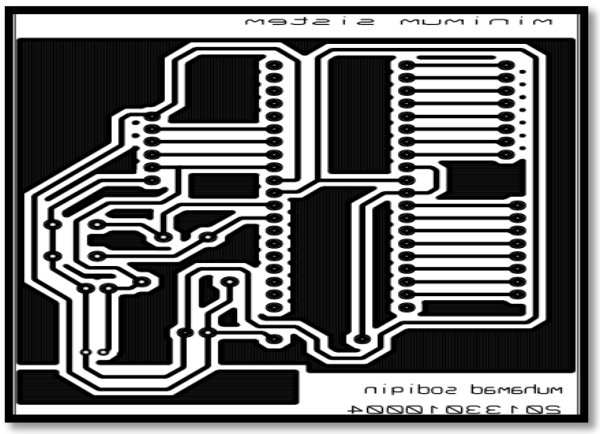
1. Rangkai sistematik rangkaian minimum sistem dengan mengunakan aplikasi pada laptop, aplikasi yang digunakan pada pembuatan modul ini adalah *proteus.*

   Untuk gambar sistematik rangkaian minimum sistem pada aplikasi dapat dilihat pada gambar 3.9. di bawah ini:



Gambar 3.9. Sistematik Minimum Sistem

2. Setelah sistematik rangkaian jadi, tahap selanjutnya membuat *lay out* nya dan disablon ke papan *pcb*. Untuk gambar *lay out* minimum sistem pada papan *pcb* dapat dilihat pada gambar 3.10. di bawah ini:



Gambar 3.10. Lay Out Rangkaian Minimum Sistem

3. Rakit komponen yang dibutuhkan dengan menggunakan solder.

3.6.4. Gambar Minimum Sistem

Untuk gambar minimum sistem dapat dilihat pada gambar 3.9. di bawah ini:

Gambar 3.11. Minimum Sistem

Rangkaian minimum sistem pada modul ini berfungsi sebgai kontrol kerja modul secara keseluruhan. Cara kerja rangkaian minimum sistem ini dengan memanfaatkan kapasitas penyimpanan yang dimiliki oleh *IC* Atmega 8535. Pada *IC* Atmega 8535 ini diberi program yang akan mengontrol sistem kerja modul secara keseluruhan. Adapun program yang digunakan pada modul ini adalah *ADC* sebagai pembaca tegangan dari sensor *infrared* dan program *timer* sebagai pengendali waktu pada modul.

3.7. Perakitan Rangkaian *Power supply*

3.7.1. Alat

1. Papan *pcb*
2. Solder
3. Timah
4. Penyedot timah

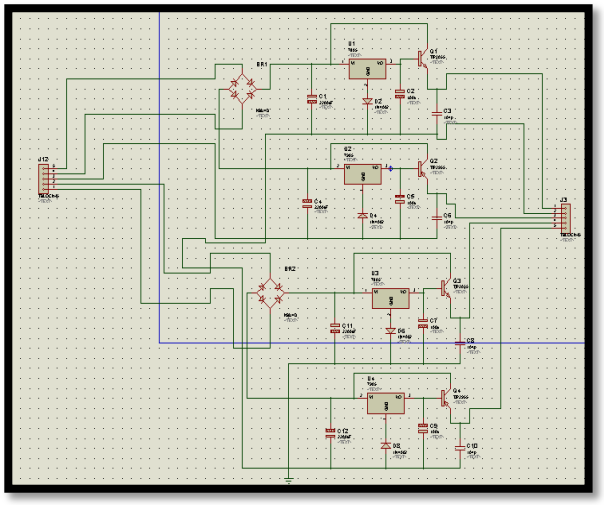
3.7.2. Bahan

1. Dioda bridge 2 A
2. Travo 2 A
3. Kapasitor 2200 µf (4)
4. Kapasitor non polar 104 (4)
5. *IC* regulator 7805, 7905, 7812, dan 7912
6. Dioda
7. Transistor TIP 2955 dan 3055
8. *T-blok*

3.7.3. Langkah Perakitan

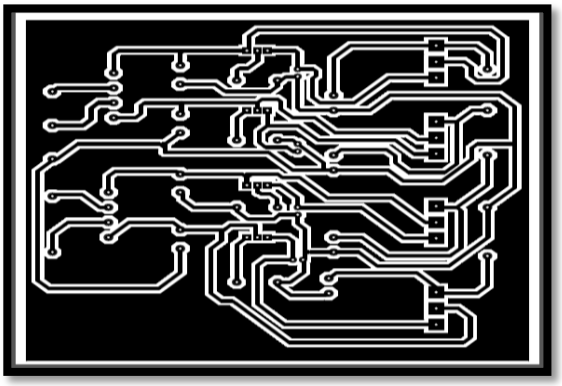
1. Rangkai sistematik rangkaian *power supply* dengan mengunakan aplikasi pada laptop, aplikasi yang digunakan pada pembuatan modul ini adalah *proteus.*

   Untuk gambar sistematik rangkaian *power supply* pada aplikasi dapat dilihat pada gambar 3.12. di bawah ini:



Gambar 3.12. Sistematik Power Supply

2. Setelah sistematik rangkaian jadi, tahap selanjutnya membuat *lay out* nya dan disablon ke papan *pcb*. Untuk gambar *lay out power supply* pada papan *pcb* dapat dilihat pada gambar 3.13. di bawah ini:

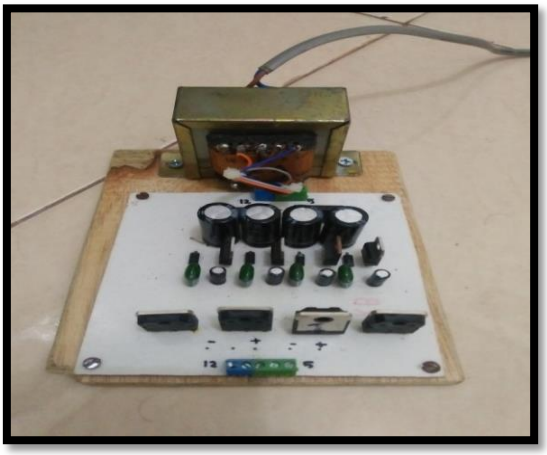


Gambar 3.13. Lay Out Power supply

3. Rakit komponen yang dibutuhkan dengan menggunakan solder.

3.7.4. Gambar *Power supply*

Untuk gambar *power supply* dapat dilihat pada gambar 3.12. di bawah ini:

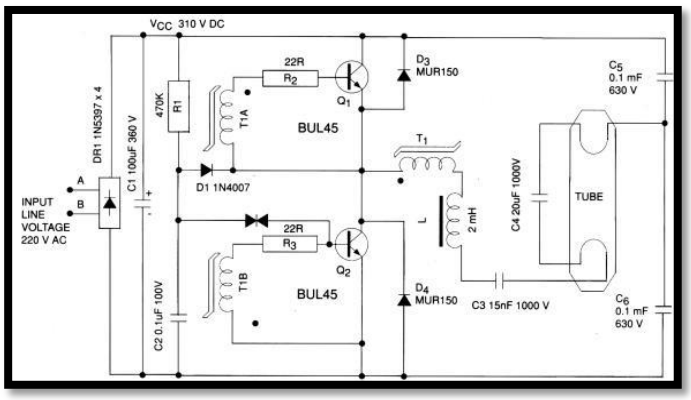


Gambar 3.14. Power Supply

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 *IC regulator* 7805 dan 7809. Adapun tegangan 5 VDC digunakan untuk rangkaian minimum sistem sedangkan tegangan 12 VDC digunakan untuk relay 12 VDC dan motor *wash*.

3.8. Sistemtik Rangkaian Lampu *UV*

Untuk gambar sistematik rangkaian *Ballast* lampu *UV* dapat dilihat pada gambar 3.13. di bawah ini:



Gambar 3.15. Sistematik Rangkaian *Ballast* Lampu ***UV***

Sistematik rangkaian *Ballast* elektronik untuk lampu *UV* pada gambar di atas adalah sistematik rangkaian *Ballast* elektronik jenis *voltage source resonant*. Pada dasarnya sistematik rangkaian *Ballast* elektronika di atas terbagi dalam bagian-bagian sebagai berikut.

Rectifier, berperan untuk menyearahkan tegangan AC 220V menjadi tegangan DC 220V. Sisi *Rectifier* tersebut terbagi dalam dioda bridge & kapasitor/kondensator elektrolit.

Converter DC *to* AC, sisi *Converter* DC *to* AC berperan untuk merubah tegangan DC220V menjadi tegangan tinggi AC antara 738 volt dengan frekuensi antara 20 KHz sampai 60 KHz.

Sistematik rangkaian *Converter* DC *to* AC di *ballast* elektronik di atas di bangun memakai 2 buah transistor dengan tipe *power* BUL45 & sistematik rangkaian *resonator*. Rangkaian *resonator* menggunakan R 470K Ohm & kapasitor/kondensator 100nF. Frekuensinya ditetapkan oleh nilai resistor & kapasitor itu. Dengan memakai sistematik rangkaian *ballast* elektronik maka untuk menyalakan lampu *UV* tidak butuh memakai starter lampu *UV*

3.9. Pembuatan Program kontrol *Driver*

Untuk pembuatan program pada modul ini menggunakan aplikasi AVR dengan bahasa C. Program yang digunakan ialah program *ADC* sebgai pengendali *driver* dan *timer* sebagai pengontrol waktunya.

Tabel 3.1. Program kontrol Driver

D:\KULIAH\SEMESTER 6\TUGAS AKHIR\coba karo kang tiar\a.c

Notes | a.c

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 6/5/2016
Author  : Muhamad Sodiqin
Company : TEM_YK
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>
#include <stdlib.h>
#include <delay.h>
unsigned char mikrodetik,detik;
float data_adc,dryer;
float data_adc1,sabun;
bit a=0, b=0, c=0;

interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x9E;
```

Notes a.c

```
34  // Reinitialize Timer 0 value
35  TCNT0=0x9E;
36  if(a==0);
37  {
38  mikrodetik++;
39  if(mikrodetik==10)
40  {detik++;
41  mikrodetik=0;
42  }
43  }
44
45  }
46
47
48  #define ADC_VREF_TYPE 0x40
49
50  // Read the AD conversion result
51  unsigned int read_adc(unsigned char adc_input)
52  {
53  ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
54  // Delay needed for the stabilization of the ADC input voltage
55  delay_us(10);
56  // Start the AD conversion
57  ADCSRA|=0x40;
58  // Wait for the AD conversion to complete
59  while ((ADCSRA & 0x10)==0);
60  ADCSRA|=0x10;
61  return ADCW;
62  }
63
64  void dryer_aktif()
65  {
66   if (dryer>=1)
67   {
68    b=1;
69   }
70   if(b==1)
```

Notes a.c

```
70    if(b==1)
71    {
72     PORTC.0=1;
73     PORTC.1=1;
74    }
75   }
76
77   void dryer_mati()
78   {
79    if(b==0)
80    {
81     PORTC.0=0;
82     PORTC.1=0;
83
84    }
85   }
86
87   void timer_aktif()
88   {
89    if(b==1)
90    {TCCR0=0x05;}else{TCCR0=0x00;}
91   }
92   void timer_mati()
93   {
94    if(detik>=20)
95    {
96     b=0;
97     detik=0;
98    }
99   }
100
101  void sabun_aktif()
102  {
103   if (sabun>=1.5)
104   {
105    PORTC.2=1;
106   }
```

Notes a.c

```
103   if (sabun>=1.5)
104   {
105    PORTC.2=1;
106   }
107   else
108   {
109    PORTC.2=0;
110   }
111  }
112
113  void main(void)
114  {
115  // Declare your local variables here
116
117  // Input/Output Ports initialization
118  // Port A initialization
119  // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
120  // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
121  PORTA=0x00;
122  DDRA=0x00;
123
124  // Port B initialization
125  // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
126  // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
127  PORTB=0x00;
128  DDRB=0x00;
129
130  // Port C initialization
131  // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
132  // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
133  PORTC=0x00;
134  DDRC=0xFF;
135
136  // Port D initialization
137  // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
138  // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

Notes a.c

```
139     PORTD=0x00;
140     DDRD=0x00;
141
142     // Timer/Counter 0 initialization
143     // Clock source: System Clock
144     // Clock value: 0.977 kHz
145     // Mode: Normal top=0xFF
146     // OC0 output: Disconnected
147     TCCR0=0x05;
148     TCNT0=0x9E;
149     OCR0=0x00;
150
151     // Timer/Counter 1 initialization
152     // Clock source: System Clock
153     // Clock value: Timer1 Stopped
154     // Mode: Normal top=0xFFFF
155     // OC1A output: Discon.
156     // OC1B output: Discon.
157     // Noise Canceler: Off
158     // Input Capture on Falling Edge
159     // Timer1 Overflow Interrupt: Off
160     // Input Capture Interrupt: Off
161     // Compare A Match Interrupt: Off
162     // Compare B Match Interrupt: Off
163     TCCR1A=0x00;
164     TCCR1B=0x00;
165     TCNT1H=0x00;
166     TCNT1L=0x00;
167     ICR1H=0x00;
168     ICR1L=0x00;
169     OCR1AH=0x00;
170     OCR1AL=0x00;
171     OCR1BH=0x00;
172     OCR1BL=0x00;
173
174     // Timer/Counter 2 initialization
```

Notes a.c

```
175  // Clock source: System Clock
176  // Clock value: 0.977 kHz
177  // Mode: Normal top=0xFF
178  // OC2 output: Disconnected
179  ASSR=0x00;
180  TCCR2=0x00;
181  TCNT2=0x00;
182  OCR2=0x00;
183
184  // External Interrupt(s) initialization
185  // INT0: Off
186  // INT1: Off
187  // INT2: Off
188  MCUCR=0x00;
189  MCUCSR=0x00;
190
191  // Timer(s)/Counter(s) Interrupt(s) initialization
192  TIMSK=0x41;
193
194  // USART initialization
195  // USART disabled
196  UCSRB=0x00;
197
198  // Analog Comparator initialization
199  // Analog Comparator: Off
200  // Analog Comparator Input Capture by Timer/Counter 1: Off
201  ACSR=0x80;
202  SFIOR=0x00;
203
204  // ADC initialization
205  // ADC Clock frequency: 7.813 kHz
206  // ADC Voltage Reference: AVCC pin
207  // ADC High Speed Mode: Off
208  // ADC Auto Trigger Source: ADC Stopped
209  ADMUX=ADC_VREF_TYPE & 0xff;
210  ADCSRA=0x87;
```

Notes a.c

```
202   SFIOR=0x00;
203
204   // ADC initialization
205   // ADC Clock frequency: 7.813 kHz
206   // ADC Voltage Reference: AVCC pin
207   // ADC High Speed Mode: Off
208   // ADC Auto Trigger Source: ADC Stopped
209   ADMUX=ADC_VREF_TYPE & 0xff;
210   ADCSRA=0x87;
211   SFIOR&=0xEF;
212
213   // SPI initialization
214   // SPI disabled
215   SPCR=0x00;
216
217   // TWI initialization
218   // TWI disabled
219   TWCR=0x00;
220
221   // Global enable interrupts
222   #asm("sei")
223   a=0;b=0;c=0;
224   while (1)
225         {
226            timer_mati();
227            timer_aktif();
228            data_adc=read_adc(0);
229            dryer=(float)data_adc*5/1024;
230            data_adc1=read_adc(1);
231            sabun=(float)data_adc1*5/1024;
232            dryer_aktif();
233            dryer_mati();
234            sabun_aktif();
235         }
236   }
237
```