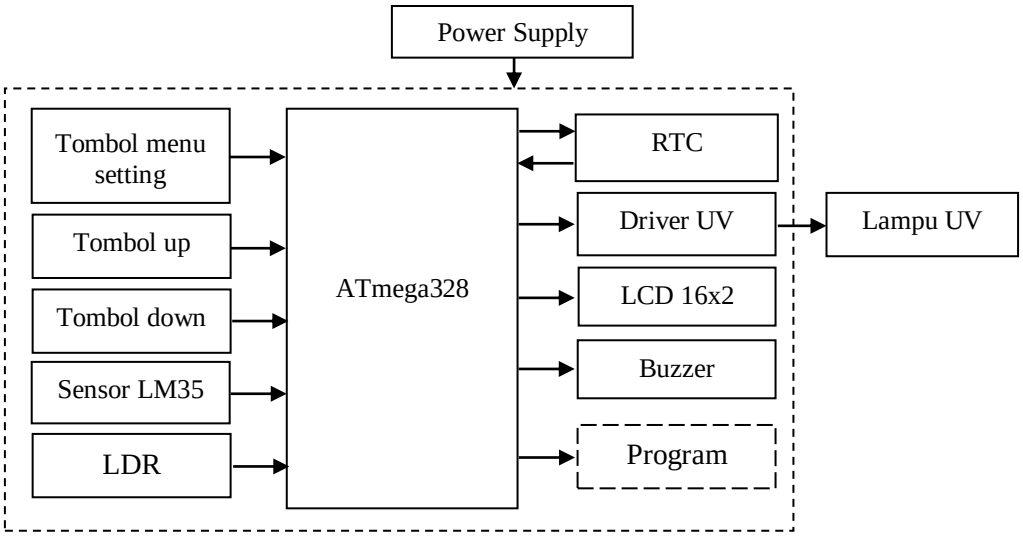
BAB III

METODOLOGI PENELITIAN

3.1 Diagram Blok Sistem

Pembuatan sistem dapat dijelaskan dengan lebih baik melalui diagram, dapat dilihat pada Gambar 3.1 berikut.



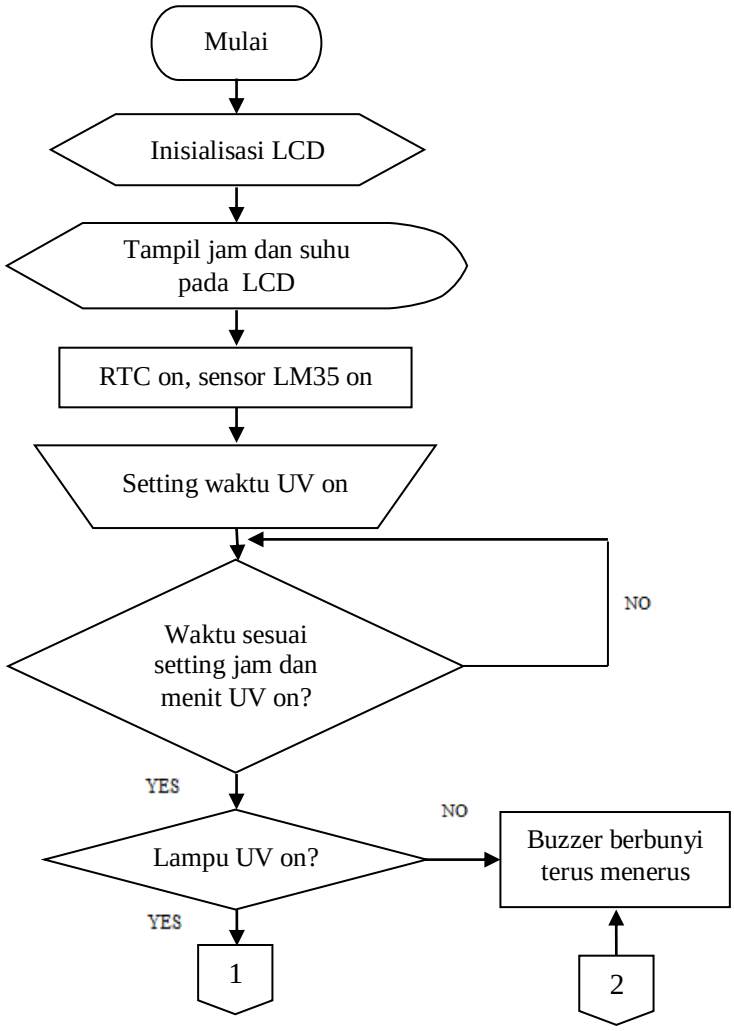
Gambar 3.1 Diagram Blok Sistem

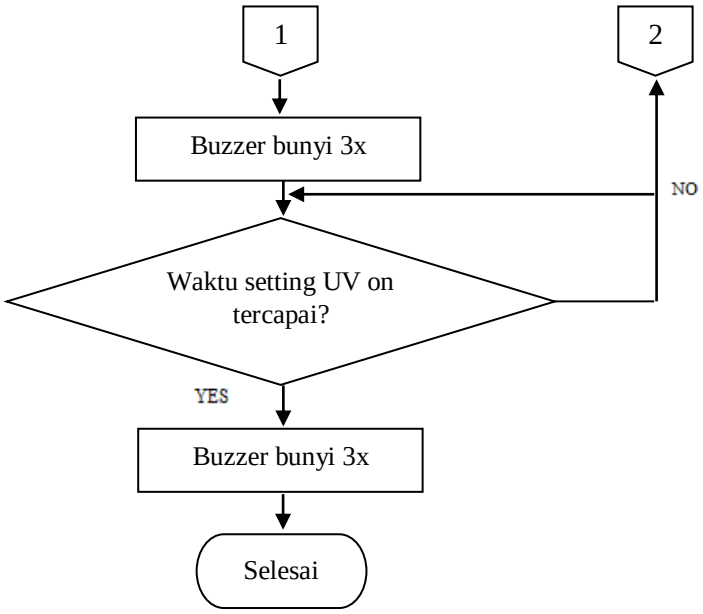
Ketika alat dihidupkan maka *power supply* akan menyuplai tegangan ke seluruh rangkaian. Tekan tombol menu untuk mengatur menu pilihan yang akan di*setting*, kemudian untuk mengatur angkanya dapat menekan tombol *up* atau *down*. Adapun pilihan pada tombol menu yaitu *setting* jam, menit, tanggal, bulan, tahun, jam *UV on*, menit *UV on*, interval jam, dan interval menit. Pada menu jam, menit, tanggal, bulan, dan tahun di*setting* sesuai waktu penggunaan alat. Pada menu jam dan menit *UV on* fungsinya untuk mengatur jam dan menit ke berapa lampu *UV* akan menyala. Kemudian pada menu interval jam dan menit fungsinya untuk mengatur waktu lamanya lampu *UV* menyala. *RTC* berfungsi sebagai

pewaktu secara *real time*. *LDR* berfungsi sebagai sensor cahaya lampu *UV*. *Buzzer* berfungsi sebagai indikator untuk mengetahui lampu *UV* menyala atau mati, dan sensor LM35 berfungsi sebagai sensor suhu di dalam *box*.

3.2 Diagram Alir

Diagram alir adalah gambar yang mewakili alur atau proses yang menampilkan langkah-langkah dalam bentuk simbol grafis dan urutannya dihubungkan menggunakan panah. Adapun gambar dari diagram alir proses berjalannya sistem kerja alat ditunjukkan oleh Gambar 3.2 berikut.



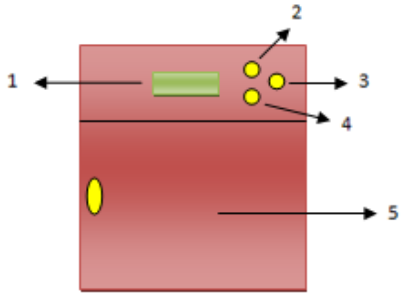


Gambar 3.2 Diagram Alir

Ketika alat dihidupkan maka *LCD* akan menginisialisasi. *RTC* bekerja sebagai pewaktu secara *real time* dan sensor LM35 bekerja untuk membaca atau *monitoring* suhu di dalam *box* yang ditampilkan pada *LCD*. Kemudian *setting* waktu (jam, menit, tanggal, bulan, tahun, jam *UV on*, menit *UV on*, interval jam dan interval menit) pada menu pilihan, untuk mengaturnya dapat menggunakan tombol *up* atau *down*. Ketika *LCD* menampilkan waktu sesuai *setting* interval jam dan menit, maka lampu *UV* menyala dan sensor *LDR* akan mendeteksi kondisi terang dari lampu sehingga *buzzer* akan berbunyi sebanyak 3x. Lampu *UV* menyala selama selang waktu yang telah ditentukan. Setelah waktu habis maka lampu *UV* mati dan sensor *LDR* akan mendeteksi kondisi gelap dari lampu kemudian *buzzer* akan berbunyi lagi sebanyak 3x. Namun, apabila lampu *UV* mati pada waktu yang seharusnya menyala, maka *buzzer* akan berbunyi terus-menerus sampai waktu tercapai.

3.3 Diagram Mekanis

Diagram mekanis Simulasi Alat Penyimpanan *Dialyzer Reuse* dilengkapi Lampu *UV* dapat dilihat pada Gambar 3.3 berikut.



Gambar 3.3 Diagram Mekanis Alat

Keterangan dari Gambar 3.3 di atas adalah sebagai berikut.

1. *LCD* : sebagai penampil
2. Tombol *up* : untuk menambahkan angka *setting*an.
3. Tombol *down* : untuk mengurangi angka *setting*an.
4. Tombol menu : menu yang ingin diatur.
5. Pintu alat : sebagai akses keluar masuknya alat/instrumen yang akan disimpan.

3.4 Alat dan Bahan

3.4.1 Alat

Pada penelitian ini digunakan beberapa peralatan, dapat dilihat pada Tabel 3.1 berikut.

Tabel 3.1 Alat

No.	Komponen	Jumlah
1	*Toolset*	1
2	Bor	1
3	Mata bor	1
4	Spidol permanen	1

3.4.2 Bahan

Pada penelitian ini digunakan beberapa bahan elektronika, dapat dilihat pada Tabel 3.2 berikut.

Tabel 3.2 Bahan

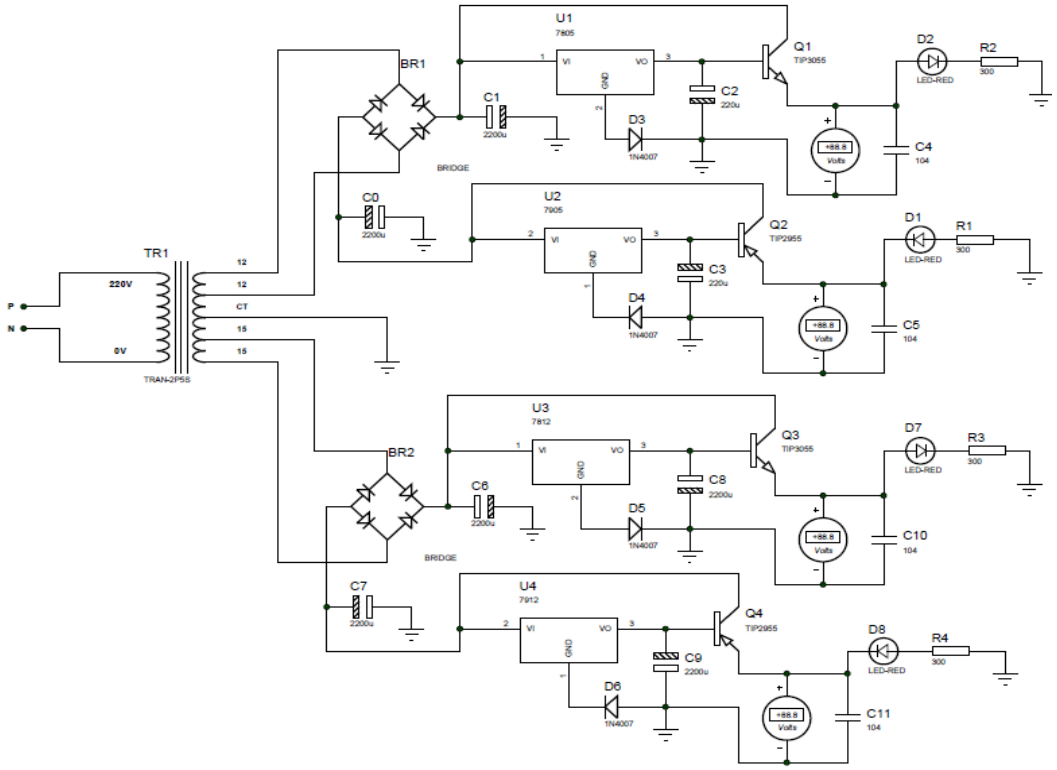
No.	Komponen	Jumlah
1	Papan *PCB*	3
2	Kabel *power*	1
3	Saklar	1
4	*Limit switch*	1
5	*Transformator* 2 A	1
6	Lampu *UV*	1
7	*Fitting* lampu	1
8	*LCD* 16x2	1
9	Sensor LM35	1
10	Sensor *LDR*	1
11	Modul *RTC* DS1307	1
12	*Push button*	3
13	Dioda *bridge*	2
14	Kapasitor elektrolit	7
15	Kapasitor keramik	4
16	*LED*	3
17	Resistor	11
18	*IC* regulator	2
19	Transistor NPN	4
20	Dioda	3
21	*Relay* 5 VDC	1
22	ATmega328	1
23	Kristal 16 MHz	1
24	*Buzzer*	1
25	*T-Block*	6
26	Pin sisir	Secukupnya
27	Kabel *jumper*	Secukupnya

3.5 Perancangan Perangkat Keras

3.5.1 Perakitan Rangkaian *Power Supply*

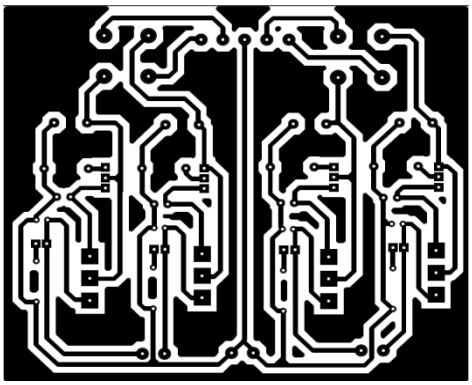
Adapun langkah-langkah perakitan perangkat keras *power supply* sebagai berikut.

1. Membuat skematik rangkaian *power supply* dengan menggunakan aplikasi *proteus* yang sudah diinstal di laptop. Rangkaian skematik dapat dilihat pada Gambar 3.4 berikut.



Gambar 3.4 Rangkaian Skematik *Power Supply*

2. Setelah skematik rangkaian selesai, selanjutnya membuat *layout power supply* dan dicetak ke papan *PCB*. *Layout* dapat dilihat pada Gambar 3.5 berikut.



Gambar 3.5 *Layout Power Supply*

3. Rakit komponen yang dibutuhkan dengan menggunakan solder. *Hardware power supply* dapat dilihat pada Gambar 3.6 berikut.

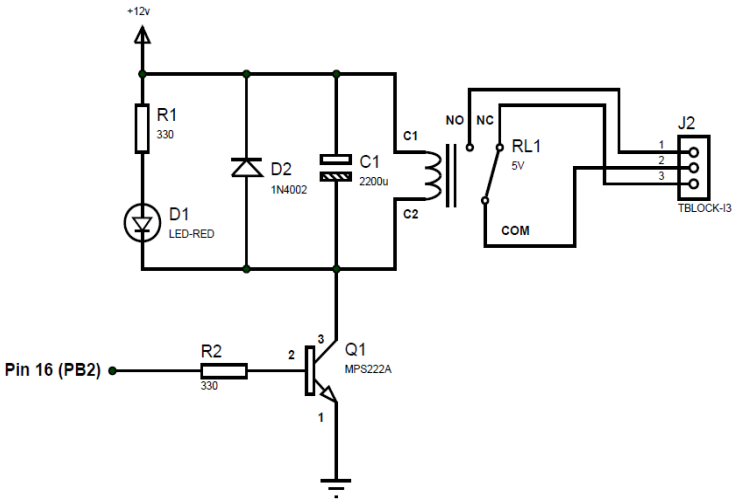
Gambar 3.6 *Power Supply*

Rangkaian *power supply* berfungsi sebagai penyuplai tegangan ke seluruh rangkaian pada modul tugas akhir yang menggunakan tegangan *DC*. Prinsip kerjanya adalah mengubah tegangan *AC* dari sumber listrik menjadi tegangan *DC* yang masuk ke rangkaian. Pada rangkaian ini menggunakan *transformator step down* yang berfungsi sebagai penurun tegangan. *Input* dari *transformator* adalah 220 VAC yang terhubung langsung dari sumber listrik (PLN). *Output transformator* yang digunakan adalah 9 VAC dan 15 VAC. Kemudian masuk ke *diode bridge* maka tegangan menjadi 9 VDC dan 15 VDC karena dioda berfungsi sebagai penyearah arus bolak balik (*AC*) yang diubah menjadi arus searah (*DC*). Kemudian untuk mendapatkan *output* 5 VDC dan 12 VDC sesuai yang dibutuhkan, pada rangkaian *power supply* ini ditambahkan *IC* regulator 7805 dan 7812 yang berfungsi sebagai pembatas tegangan. Rangkaian yang membutuhkan suplai tegangan 5 VDC adalah rangkaian minimum sistem dan *RTC*, sedangkan tegangan 12 VDC adalah rangkaian *driver*.

3.5.2 Perakitan Rangkaian *Driver UV*

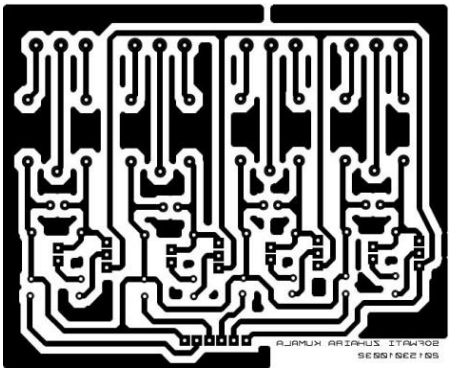
Rangkaian *driver* pada modul tugas akhir ini digunakan untuk mengontrol atau sebagai saklar otomatis untuk lampu *UV*. Adapun langkah-langkah perakitan perangkat keras *driver* sebegai berikut.

1. Membuat skematik rangkaian *driver* dengan menggunakan aplikasi *proteus* yang sudah diinstal di laptop. Rangkaian skematik *driver* dapat dilihat pada Gambar 3.7 berikut.



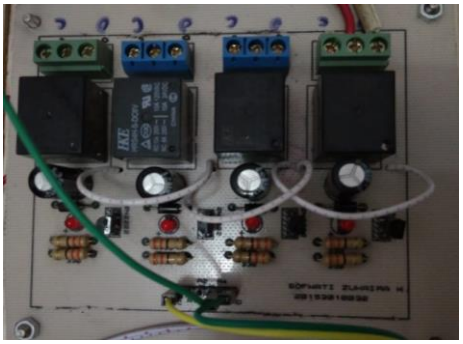
Gambar 3.7 Rangkaian Skematik *Driver*

2. Setelah skematik rangkaian selesai, selanjutnya membuat *layout driver* dan dicetak ke papan *PCB*. *Layout* dapat dilihat pada Gambar 3.8 berikut.



Gambar 3.8 Layout *Driver*

3. Rakit komponen yang dibutuhkan dengan menggunakan solder. *Hardware driver* dapat dilihat pada Gambar 3.9 berikut.



Gambar 3.9 *Driver*

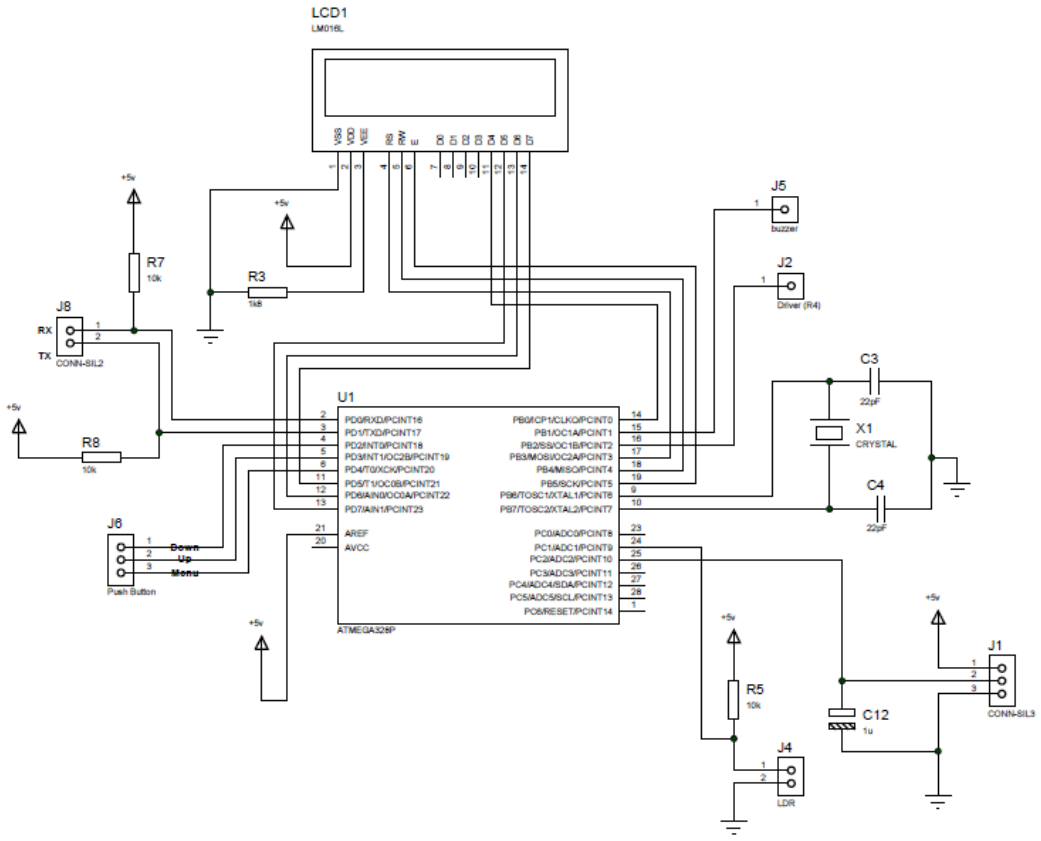
Relay merupakan komponen elektronika yang bersifat induktif. Pada suatu rangkaian elektronika yang menggunakan beban induktif akan menimbulkan lonjakan tegangan yang terjadi pada saat beban induktor diberikan tegangan dan saat diputuskan tegangan, sehingga pada rangkaian *driver* diberikan rangkaian *snubber* untuk memproteksi dari lonjakan tegangan. Pada rangkaian proteksi ini komponen dioda di paralel dengan *coil relay* yang berfungsi sebagai pengaman komponen elektronika yang mudah rusak seperti transistor akibat lonjakan tegangan. Ketika *relay* dalam kondisi *off* (*NC*) maka dioda pada rangkaian tersebut mendapat bias maju sehingga arus yang mengalir ke *coil* selanjutnya akan mengalir ke dioda secara *looping*. Pada prinsipnya, *driver* akan *on* pada saat mikrokontroler diberikan logika *high* untuk memberikan tegangan 5 Volt agar transistor saturasi dan *coil relay* mendapatkan *ground* melalui transistor, sehingga mengakibatkan posisi *NC* akan berpindah ke *NO*. Kemudian *driver* akan kembali *off* apabila mikrokontroler diberikan logika *low* untuk memberikan tegangan 0 Volt agar transistor tidak bekerja dan *coil relay* tidak mendapatkan *ground*,

sehingga posisi *NO* akan berpindah kembali ke *NC*. Komponen kapasitor yang di paralel dengan *coil relay* berfungsi untuk menyimpan muatan sehingga dapat memperlambat gerakan perpindahan posisi *NC* ke *NO* atau sebaliknya.

3.5.3 Perakitan Rangkaian Sistem Minimum

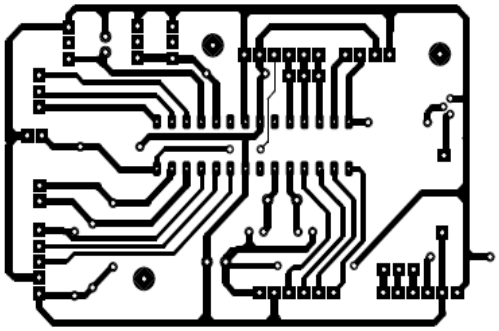
Rangkaian sistem minimum merupakan rangkaian utama yang dibutuhkan sebagai pengendali sistem. Adapun langkah-langkah perakitan perangkat keras sistem minimum sebegai berikut.

1. Membuat skematik rangkaian sistem minimum dengan menggunakan aplikasi *proteus* yang sudah diinstal di laptop. Rangkaian skematik sistem minimum dapat dilihat pada Gambar 3.10 berikut.



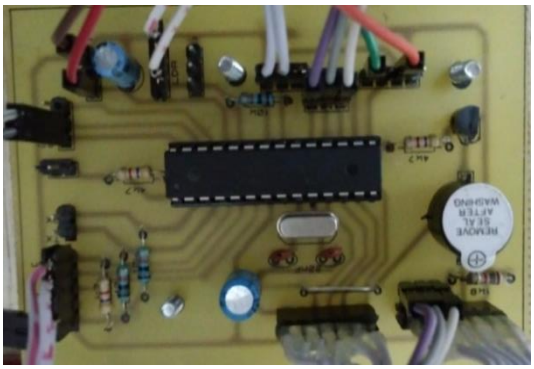
Gambar 3.10 Rangkaian Skematik Sistem Minimum

2. Setelah skematik rangkaian selesai, selanjutnya membuat *layout* sistem minimum dan dicetak ke papan *PCB*. *Layout* dapat dilihat pada Gambar 3.11 berikut.



Gambar 3.11 *Layout* Sistem Minimum

3. Rakit komponen yang dibutuhkan dengan menggunakan solder. *Hardware* sistem minimum dapat dilihat pada Gambar 3.12 berikut.

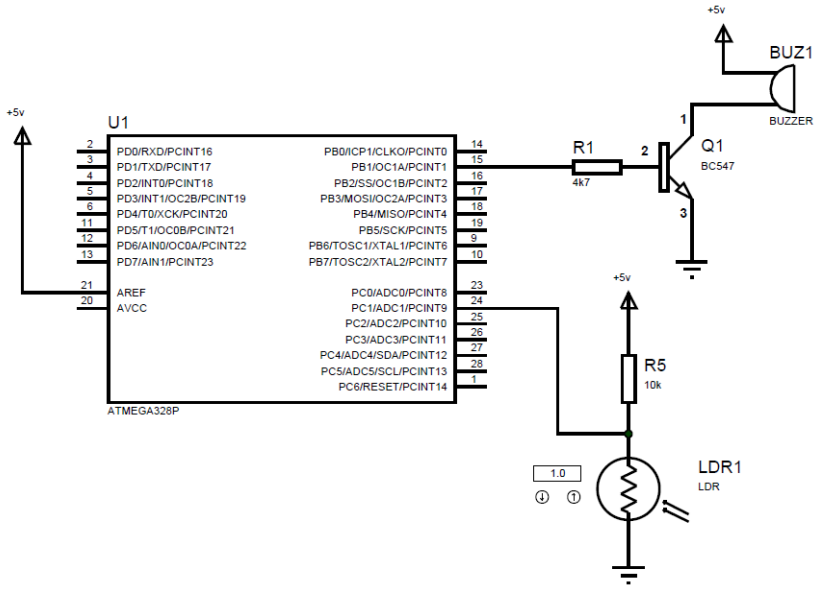


Gambar 3.12 Sistem Minimum

Mikrokontroler pada sistem minimum ini menggunakan ATmega328 yang memiliki 28 pin. Pin-pin yang digunakan untuk *input* tombol-tombol adalah PD2, PD3, dan PD4. Pin PC1/ADC1 digunakan untuk *input* sensor *LDR* dan PC2/ADC2 digunakan untuk *input* sensor LM35. Kemudian pin PB1 adalah *output buzzer* dan pin PB2 adalah *driver*.

3.5.4 Rangkaian Sensor *LDR* dan *Buzzer*

Sensor *LDR* adalah salah satu jenis resistor yang nilai resistansinya (hambatan) dipengaruhi oleh cahaya yang diterimanya. Semakin banyak cahaya yang diterima sensor maka nilai resistansinya semakin kecil, sebaliknya semakin sedikit cahaya yang diterima sensor maka nilai resistansinya semakin besar. Dapat dilihat pada Gambar 3.13 yang merupakan rangkaian sensor *LDR* dan rangkaian *buzzer*.



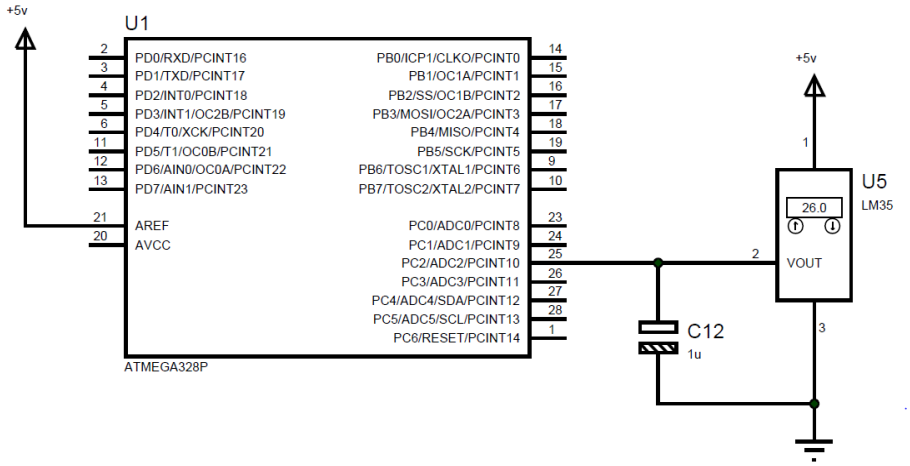
Gambar 3.13 Rangkaian Sensor *LDR* dan *Buzzer*

Sensor *LDR* difungsikan untuk mendeteksi cahaya dari *UV*. Kemudian *buzzer* difungsikan sebagai indikator dalam 3 kondisi. Apabila sensor *LDR* mendeteksi terang pada saat *setting* waktu lampu menyala, maka *buzzer* akan berbunyi sebanyak 3x. Kemudian apabila sensor mendeteksi gelap pada saat *setting* waktu telah tercapai, *buzzer* akan berbunyi lagi sebanyak 3x. Namun, apabila sensor mendeteksi gelap ketika *setting* waktu lampu menyala, maka *buzzer* akan berbunyi terus-menerus sampai *setting* waktu tercapai.

Mikrokontroler tidak bisa membaca nilai resistansi *LDR* sehingga pada sensor *LDR* digunakan rangkaian pembagi tegangan untuk mendapatkan nilai *output* berupa tegangan, namun *output* tegangan sensor masih berupa analog sehingga dihubungkan ke pin ADC mikrokontroler untuk dikonversi ke dalam bentuk digital yang kemudian dapat diolah datanya oleh mikrokontroler.

3.5.5 Rangkaian Sensor LM35

Sensor LM35 merupakan sensor suhu yang pada alat ini difungsikan hanya untuk *monitoring* suhu di dalam *box*. Dapat dilihat pada Gambar 3.14 yang merupakan rangkaian sensor LM35.

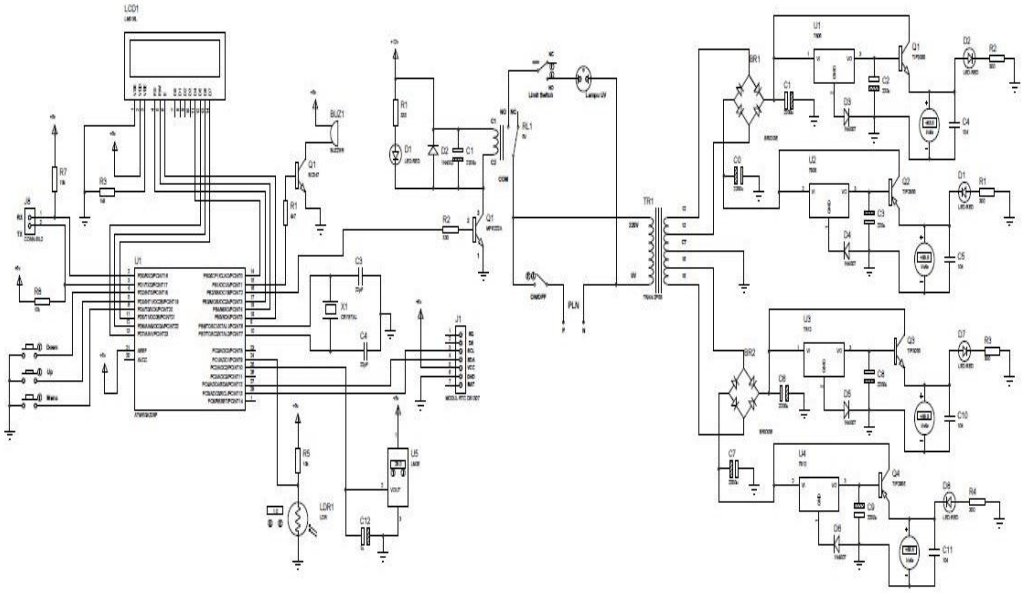


Gambar 3.14 Rangkaian Sensor LM35

Output dari sensor suhu LM35 ini sudah berupa tegangan sehingga tidak memerlukan rangkaian tambahan untuk menghasilkan *output* tegangan. Namun terdapat kapasitor pada rangkaian tersebut yang difungsikan sebagai penstabil tegangan.

3.5.6 Rangkaian Keseluruhan

Rangkaian keseluruhan dari simulasi alat penyimpanan *dialyzer reuse* dilengkapi lampu *UV* ini dapat dilihat pada Gambar 3.15 berikut.



Gambar 3.15 Rangkaian Keseluruhan

Rangkaian *power supply* pada alat ini berfungsi sebagai penyuplai tegangan ke seluruh rangkaian. *Output* yang dihasilkan oleh rangkaian ini adalah 5 VDC dan 12 VDC. Tegangan 5 VDC dihubungkan ke rangkaian sistem minumum dan modul *RTC*, sedangkan tegangan 12 VDC dihubungkan ke rangkaian *driver*. Rangkaian sistem minimum adalah pengendali semua aktifitas kerja alat karena pada mikrokontroler yang dipasang pada rangkaian inilah tempat program di*input*. Apabila diberikan logika *high* (1), maka rangkaian *output* akan bekerja. Sebaliknya, apabila diberikan logika *low* (0), maka rangkaian *output* tidak bekerja. Rangkaian *input* yang dikendalikan oleh sistem minimum ini adalah *push button* (tombol-tombol), sensor LM35, sensor *LDR*, dan *RTC*. Sedangkan rangkaian *output* yang dikendalikan adalah *LCD*, *RTC*, *buzzer*, dan *driver*.

3.6 Langkah-langkah Pengujian Alat

Setelah membuat alat, langkah berikutnya adalah melakukan pengujian dan pengukuran yang bertujuan untuk mengetahui kinerja alat dan memastikan bahwa alat dapat berfungsi sesuai dengan yang telah direncanakan. Langkah-langkah pengukuran dan pengujian alat dapat diuraikan dalam beberapa tahap berikut.

1. Menyiapkan alat yang dibutuhkan, baik alat ukur maupun alat pembanding.
2. Menyiapkan tabel data untuk mencatat hasil pengukuran.
3. Melakukan pengukuran suhu dengan cara memantau suhu yang terbaca oleh sensor pada layar *display* kemudian membandingkannya menggunakan *temperature meter*, hasil pengukuran dicatat pada tabel yang telah disediakan.
4. Melakukan perhitungan angka kuman pada objek, sebelum dan sesudah menggunakan alat.
5. Setelah pendataan, selanjutnya melakukan perhitungan untuk mengetahui besar simpangan dan *error* yang dihasilkan.

3.7 Rumus Statistik

3.7.1 Rata-rata

Rata – rata adalah nilai atau hasil pembagian dari jumlah data yang diambil atau diukur dengan banyaknya pengambilan data atau banyaknya pengukuran.

$$\text{Rata – Rata } (\bar{x}) = \frac{\sum Xi}{n} \qquad (3\text{-}1)$$

Dimana:

$\bar{x}$ = rata – rata

ΣXi = Jumlah nilai data

n = Banyak data (1,2,3,..,n)

3.7.2 Simpangan

Simpangan adalah selisih dari rata–rata nilai harga yang dikehendaki dengan nilai yang diukur. Berikut rumus dari simpangan.

Simpangan = Xn – $\bar{X}$ (3-2)

Dimana:

Xn = rata-rata alat

$\bar{X}$ = Rata-rata pembanding

3.7.3 *Error (%)*

Error (kesalahan) adalah selisih antara *mean* terhadap masing-masing data. Rumus *error* adalah:

$$Error\ \% = \frac{Rerata\ pembanding - Modul}{Rerata\ Pembanding}\ X\ 100\ \% \quad (3\text{-}3)$$

3.8 *Listing Program*

Program yang digunakan dalam pembuatan alat Simulasi Penyimpanan *Dialyzer Reuse* ini adalah program pada aplikasi arduino.

```
#include <TimerOne.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(11, 13, 8, 7, 6, 5);
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;
#include <EEPROM.h>
```

Listing 3.1 Kode *File Header*

Dapat dilihat pada Listing 3.1 yang merupakan kode *file header* dari program. Kode *file header* merupakan *library* program yang berfungsi untuk

menyimpan daftar fungsi yang akan digunakan dalam suatu program. #include <TimerOne.h> merupakan *library timer* yang berfungsi sebagai *stopwatch* pada *setting* waktu lamanya lampu *UV* menyala. #include <LiquidCrystal.h> merupakan *library LCD* dan LiquidCrystal lcd(11, 13, 8, 7, 6, 5) merupakan pemetaan koneksi pin *LCD* dan pin arduino. #include <Wire.h> merupakan *library* penggunaan fungsi I2C. #include "RTClib.h" merupakan *library RTC* dan RTC_DS1307 merupakan pemberitahuan bahwa *RTC* yang digunakan adalah *RTC* DS1307. #include <EEPROM.h> merupakan *library* EEPROM internal.

```
const int RW = 12,BUZZ=9, RLY_PIN=10;
int thn=0, bln=0, hri=0, jam=0, mnt=0, dtk=0,suhu=0;
int val=0,inten_adc=0;
const int set=4, up=3, down=2;
int set_mode,set_up,set_down,indeks=0;
int buttonState=LOW, buttonUp=LOW, buttonDown=LOW;
int lastButtonState=LOW, lastStateUp=LOW,lastStateDown=LOW;

float mv;
float cel=0,cel_temp=0;
long tot_jam=0,tot_run=0,tot_off=0;
int addrs_jam=0,addrs_mnt=1,addrs_intjam=2,addrs_intmnt=3;
int jam_comp=0,mnt_comp=0,int_jam=0,int_menit;
int msec=0,msec_buz=0,sec=0,buz_indeks=0;
int indeks_jam=0,jam_time=0,time_off=0;
```

Listing 3.2 Program Deklarasi Variabel dan Tipe Data

Dapat dilihat pada Listing 3.2 yang merupakan program deklarasi variabel yang digunakan beserta tipe datanya. Variabel dalam program digunakan untuk menyimpan suatu nilai tertentu di mana nilai tersebut dapat berubah-ubah. Kemudian tipe data merupakan istilah pemberian nilai untuk suatu variabel. *Const int* berfungsi untuk memberi nilai yang konstan atau tetap pada suatu variabel. *Int* atau *integer* merupakan tipe data bilangan bulat yang hanya mengenal bilangan

bernilai positif dan negatif. *Float* merupakan tipe data yang mengasilkan bilangan desimal. *Long* merupakan tipe data untuk data yang besar atau banyak.

```
lcd.begin(16, 2);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("  PENYIMPANAN   ");
lcd.setCursor(0,1);
lcd.print(" DIALYZER REUSE ");
delay(5000);

void tampil_disp()
{
 if(indeks==0){
 DateTime now = RTC.now();
 thn = now.year();bln = now.month(); hri = now.day();
 jam = now.hour(); mnt = now.minute(); dtk = now.second();

lcd.setCursor(0,0);
lcd.print("Jam: ");lcd.print(jam); lcd.print(":");
lcd.print(mnt); lcd.print(":"); lcd.print(dtk);lcd.print(" ");
lcd.setCursor(0,1);
lcd.print("Suhu: ");lcd.print(cel);lcd.print("C ");
}}
```

Listing 3.3 Program Inisialisasi Awal

Dapat dilihat pada Listing 3.3 yang merupakan inisialisasi awal program. Program ini difungsikan untuk tampilan awal pada *LCD* setelah alat dihidupkan dengan *delay* 5 sekon, kemudian akan menampilkan jam dan suhu.

```
if(dat_filt<=200)
{
  if(val<analogRead(A2)){val = analogRead(A2);}
}
if(++dat_filt>200)
{cel_temp = (float)(cel_temp + val)/2;val=0;
 dat_filt=0;
}
mv = ( cel_temp/1024.0)*5000;
cel = (mv/10)+2;
inten_adc = (inten_adc + analogRead(A1))/2;
tampil_disp();
tombol();
}
```

Listing 3.4 Program Sensor *LDR* dan Sensor LM35

Dapat dilihat pada Listing 3.4 yang merupakan program pembacaan sensor *LDR* dan sensor LM35. Program untuk sensor LM35 ini difungsikan untuk membaca data yang diperoleh sensor berupa data analog kemudian diubah ke digital dan hasil pengolahan datanya ditampilkan pada *LCD* berupa suhu.

```
if(indeks==1)
{lcd.setCursor(0,0);
lcd.print("Set Jam");;
lcd.setCursor(0,1);
lcd.print(jam); lcd.print(":"); lcd.print(mnt);
lcd.print(":"); lcd.print(dtk);lcd.print(" ");}

if(indeks==2)
{lcd.setCursor(0,0);
lcd.print("Set Menit");;
lcd.setCursor(0,1);
lcd.print(jam); lcd.print(":"); lcd.print(mnt);
lcd.print(":"); lcd.print(dtk);lcd.print(" ");}

if(indeks==3)
{lcd.setCursor(0,0);
lcd.print("Tanggal");;
lcd.setCursor(0,1);
lcd.print(hri);lcd.print("/");lcd.print(bln);lcd.print("/");lc
d.print(thn);lcd.print(" ");}

if(indeks==4)
{lcd.setCursor(0,0);
lcd.print("Bulan");;
lcd.setCursor(0,1);
lcd.print(hri);lcd.print("/");lcd.print(bln);lcd.print("/");lc
d.print(thn);lcd.print(" ");}

if(indeks==5)
{lcd.setCursor(0,0);
lcd.print("Tahun");;
lcd.setCursor(0,1);
lcd.print(hri);lcd.print("/");lcd.print(bln);lcd.print("/");lc
d.print(thn);lcd.print(" ");}

if(indeks==6)
{lcd.setCursor(0,0);
lcd.print("Jam UV On");;
lcd.setCursor(0,1);
lcd.print(jam_comp);lcd.print(":");lcd.print(mnt_comp);lcd.pri
nt("  ");}

if(indeks==7)
{lcd.setCursor(0,0);
```

Lanjut

Lanjut

```
lcd.print("Menit UV On");;
lcd.setCursor(0,1);
lcd.print(jam_comp);lcd.print(":");lcd.print(mnt_comp);lcd.pri
nt("  ");}

if(indeks==8)
{lcd.setCursor(0,0);
lcd.print("Interval Jam");;
lcd.setCursor(0,1);
lcd.print(int_jam);lcd.print(":");lcd.print(int_menit);lcd.pri
nt("  ");}

if(indeks==9)
{lcd.setCursor(0,0);
lcd.print("Interval Menit");;
lcd.setCursor(0,1);
lcd.print(int_jam);lcd.print(":");lcd.print(int_menit);lcd.pri
nt("  ");}}
```

Listing 3.5 Program Tombol Menu

Dapat dilihat pada Listing 3.5 yang merupakan program tombol menu yang difungsikan untuk menampilkan menu pilihan yang akan di*setting*. Indeks==1 artinya tombol menu ditekan pertama kali akan memunculkan menu "Set Jam". Indeks==2 artinya tombol ditekan yang kedua kali akan memunculkan menu "Set Menit". Indeks==3 artinya tombol ditekan yang ketiga kali akan memunculkan menu "Tanggal". Indeks==4 artinya tombol ditekan yang keempat kali akan memunculkan menu "Bulan". Indeks==5 artinya tombol ditekan yang kelima kali akan memunculkan menu "Tahun". Indeks==6 artinya tombol ditekan yang keenam kali akan memunculkan menu "Jam UV On". Indeks==7 artinya tombol ditekan yang ketujuh kali akan memunculkan menu "Menit UV On". Indeks==8 artinya tombol ditekan yang kedelapan kali akan memunculkan menu "Interval Jam". Indeks==9 artinya tombol ditekan yang kedelapan kali akan memunculkan menu "Interval Menit".

```
void tombol()
{
if(indeks!=0){
set_up = digitalRead(up);
if(set_up != lastStateUp){lastDebounUPdown = millis();}
if((millis() - lastDebounUPdown) > debounceDelay)
 {if(set_up != buttonUp)
  {buttonUp = set_up;
   if(buttonUp == LOW)
   {
    if(indeks==1){if(++jam>23){jam=0;}indeks_jam=1;}
    if(indeks==2){if(++mnt>59){mnt=0;}indeks_jam=1;}
    if(indeks==3){if(++hri>31){hri=1;}indeks_jam=1;}
    if(indeks==4){if(++bln>12){bln=1;}indeks_jam=1;}
    if(indeks==5){if(++thn>3000){thn=2015;}indeks_jam=1;}
    if(indeks==6){if(++jam_comp>23){jam_comp=0;}jam_time=1;}
    if(indeks==7){if(++mnt_comp>59){mnt_comp=0;}jam_time=1;}
    if(indeks==8){if(++int_jam>23){int_jam=0;}time_off=1;}
    if(indeks==9){if(++int_menit>59){int_menit=0;}time_off=1;}
    lcd.clear();
   }}}
lastStateUp = set_up;
```

Listing 3.6 Program Tombol *Up*

Dapat dilihat pada Listing 3.6 yang merupakan program tombol *up* yang difungsikan untuk penambahan 1 angka setiap menekan tombolnya.

```
set_down = digitalRead(down);
if(set_down != lastStateDown){lastDebounUPdown = millis();}
if((millis() - lastDebounUPdown) > debounceDelay)
 {if(set_down != buttonDown)
  {buttonDown = set_down;
   if(buttonDown == LOW)
   {
    if(indeks==1){if(--jam<0){jam=23;}indeks_jam=1;}
    if(indeks==2){if(--mnt<0){mnt=59;}indeks_jam=1;}
    if(indeks==3){if(--hri<1){hri=31;}indeks_jam=1;}
    if(indeks==4){if(--bln<1){bln=12;}indeks_jam=1;}
    if(indeks==5){if(--thn<2015){thn=3000;}indeks_jam=1;}
    if(indeks==6){if(--jam_comp<0){jam_comp=23;}jam_time=1;}
    if(indeks==7){if(--mnt_comp<0){mnt_comp=59;}jam_time=1;}
    if(indeks==8){if(--int_jam<0){int_jam=23;}time_off=1;}
    if(indeks==9){if(--int_menit<0){int_menit=59;}time_off=1;}
    lcd.clear();
   }}}
lastStateDown = set_down;
}}
```

Listing 3.7 Program Tombol *Down*

Dapat dilihat pada Listing 3.7 yang merupakan program tombol *down* yang difungsikan untuk pengurangan 1 angka setiap menekan tombolnya.

```
void timerIsr()
{
 if(buz_indeks==1 || buz_indeks==3 || buz_indeks==2)
 {
  if(++msec>msec_buz)
  {
   if(buz_indeks==1)
   {if(++sec>6){buz_indeks=2;sec=0;}
    digitalWrite(BUZZ, sec%2);
   }
   if(buz_indeks==3)
   {if(++sec>6){buz_indeks=0;sec=0;}
    digitalWrite(BUZZ, sec%2);
   }
   if(buz_indeks==2 && buz_warn==1)
   {if(++sec>5){sec=0;}
    digitalWrite(BUZZ, sec%2);
   }
   msec=0;
  }}
else
{if(buz_warn==0){digitalWrite(BUZZ, LOW);}
}}
```

Listing 3.8 Program *Buzzer*

Dapat dilihat pada Listing 3.8 yang merupakan program yang difungsikan sebagai *output* suara *buzzer*. Buz_indeks==1 merupakan kondisi apabila lampu menyala saat *setting* waktu *UV on*, maka *buzzer* akan berbunyi sebanyak 3x. Buz_indeks==3 merupakan kondisi apabila lampu mati setelah waktu tercapai, maka *buzzer* berbunyi lagi sebanyak 3x. Buz_indeks==2 merupakan kondisi apabila lampu tidak menyala ketika di*setting* menyala, maka *buzzer* akan berbunyi terus-menerus sampai waktu *setting* tercapai.