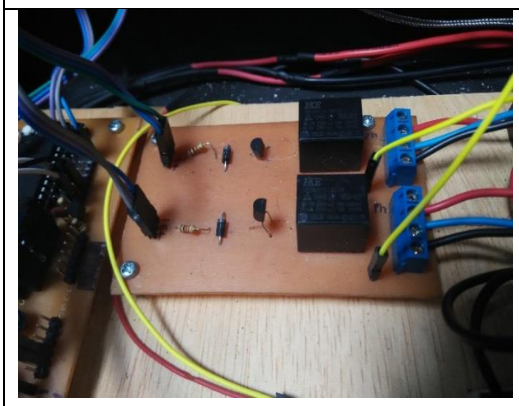
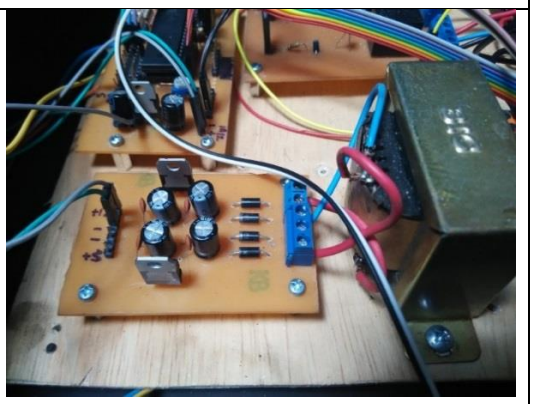
Lampiran 1

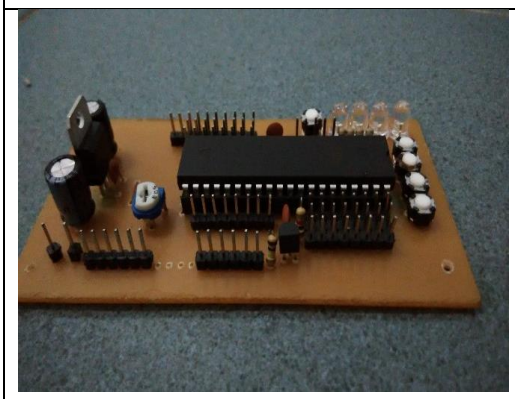
Proses pembuatan rangkaian alat



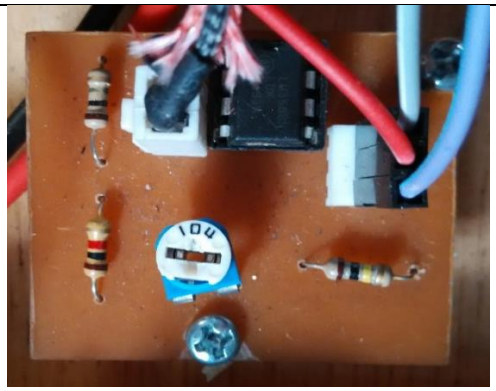
Rangkaian driver uv & heater



Rangkaian Power Supply



Rangkaian minimum sistem



Rangkaian driver termokopel

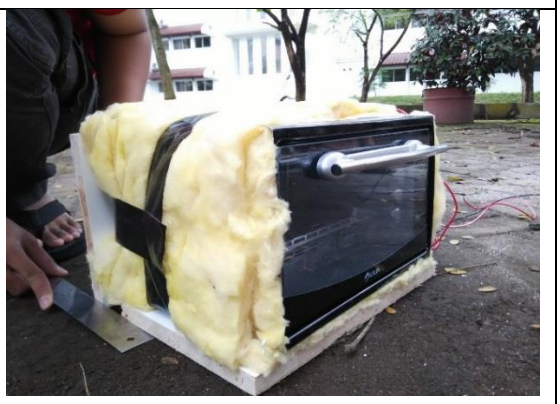
Gambar rangkaian keseluruhan

Proses pembuatan box modul

Pemotongan plat besi

Perakitan rangka box

Pengecatan box modul



Perakitan komponen

Box modul jadi

Uji fungsi kerja modul

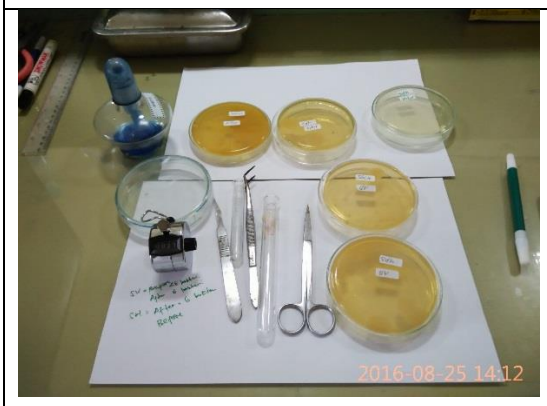
Uji laboratorium

Pemberian bakteri

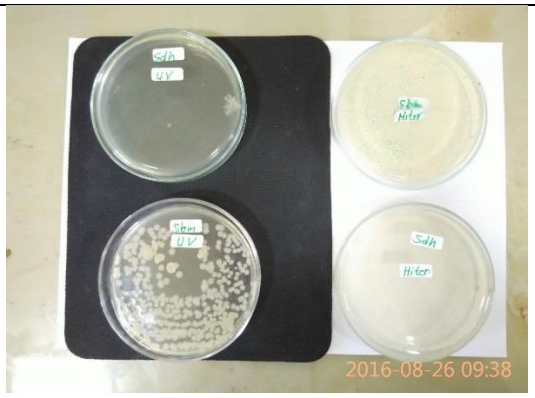
Proses sterilisasi

Persiapan sterilisasi

Penelitian



Proses sebelum inkubasi bakteri



Hasil preoses sterilisasi

Lampiran 2

CODINGAN PROGRAM YANG DI BUAT :

```
/*****************************************************
This program was created by the
CodeWizardAVR V3.12 Advanced
Automatic Program Generator
© Copyright 1998-2014 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 16/08/2016
Author  :
Company :
Comments:

Chip type               : ATmega16
Program type            : Application
AVR Core Clock frequency: 12,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega16.h>
#include <stdio.h>
#include <delay.h>

// Alphanumeric LCD functions
#include <alcd.h>
```

```
// Declare your global variables here
unsigned int set_m=1,set_s,remain_s,remain_m,tim_s,start;
char buff[33];
int temp;
float vin;
// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Reinitialize Timer1 value
TCNT1H=0xD23A >> 8;
TCNT1L=0xD23A & 0xff;
// Place your code here
tim_s++;
//PORTD=!PORTD;
//lcd_clear();

}

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
```

```
return ADCW;
}

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

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) |
(0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) |
(0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) |
(0<<DDB2) | (0<<DDB1) | (0<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) |
(0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out
Bit1=Out Bit0=Out
DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) |
(1<<DDC2) | (1<<DDC1) | (1<<DDC0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
```

```
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) |
(0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In
Bit0=In
DDRD=(1<<DDD7) | (1<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) |
(0<<DDD2) | (0<<DDD1) | (0<<DDD0);
// State: Bit7=0 Bit6=0 Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTD=(0<<PORTD7) | (0<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) |
(1<<PORTD3) | (1<<PORTD2) | (1<<PORTD1) | (1<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) |
(0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 11,719 kHz
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer Period: 0,99994 s
```

```
// Timer1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0)
| (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) |
(1<<CS12) | (0<<CS11) | (1<<CS10);
TCNT1H=0xD2;
TCNT1L=0x3A;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) |
(0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) |
(0<<OCIE1B) | (1<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);
```

```
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) |
(0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) |
(0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);

// ADC initialization
// ADC Clock frequency: 750,000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: Timer1 Overflow
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (1<<ADATE) | (0<<ADIF) |
(0<<ADIE) | (1<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);
SFIOR=(1<<ADTS2) | (1<<ADTS1) | (0<<ADTS0);
```

```
// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) |
(0<<CPHA) | (0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) |
(0<<TWIE);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 0
// RD - PORTC Bit 1
// EN - PORTC Bit 2
// D4 - PORTC Bit 3
// D5 - PORTC Bit 4
// D6 - PORTC Bit 5
// D7 - PORTC Bit 6
// Characters/line: 16
lcd_init(16);

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

while (1)
      {

        if(PIND.5==0)
       {
```

```
 if(PIND.1==0)
{
delay_ms(80);
set_m--;
}

if(PIND.0==0)
{
delay_ms(80);
set_m++;
}

set_s=set_m*60;
sprintf(buff,"SetTime %dmin ",set_m);
lcd_gotoxy(2,1);
lcd_puts(buff);
if(PIND.2==0){
tim_s=0;
start=1;
 while(start){

    temp=read_adc(1);
    vin=(float)temp*0.489-85.4;
    lcd_gotoxy(0,0);
    sprintf(buff,"Suhu : %0.0001f C",vin);
    lcd_puts(buff);
    if(temp>=498)
    {
     PORTD.6=0;
    }
    else{PORTD.6=1;}
```

```
    remain_s=set_s-tim_s;
    remain_m=remain_s/60;
    if(remain_s>60)sprintf(buff,"Remain %dmin ",remain_m);
    else sprintf(buff,"Remain %ds ",remain_s);
    lcd_gotoxy(0,1);
    lcd_puts(buff);
    if(remain_s==0)start=0;

    if(remain_m>=45)
    {
     PORTD.7=1;
    }
    if(remain_m<=44)
    {
     PORTD.7=0;
     PORTC.7=0;
     //PORTD.6=1;
    }



  }
   lcd_clear();
}


}
if(PIND.3==0)
{
set_s=set_m*60;
sprintf(buff,"SetTime %dmin ",set_m);
```

```
lcd_gotoxy(2,1);
lcd_puts(buff);
lcd_gotoxy(2,0);
lcd_puts(" MODE Heater");




 if(PIND.1==0)
{
delay_ms(80);
set_m--;
}

if(PIND.0==0)
{
delay_ms(80);
set_m++;
}


if(PIND.2==0){
tim_s=0;
start=1;
 while(start){

    temp=read_adc(1);
    vin=(float)temp*0.489-85.4;
    lcd_gotoxy(0,0);
    sprintf(buff,"Suhu : %0.0001f C",vin);
    lcd_puts(buff);
```

```
    if(temp>=498)
    {
     PORTD.6=0;
    }
    else{PORTD.6=1;PORTD.7=0;PORTC.7=0;}
    remain_s=set_s-tim_s;
    remain_m=remain_s/60;
    if(remain_s>60)sprintf(buff,"Remain %dmin ",remain_m);
    else sprintf(buff,"Remain %ds ",remain_s);
    lcd_gotoxy(0,1);
    lcd_puts(buff);

    if(remain_s==0)start=0;
  }

  lcd_clear();
 }
 else PORTC.7=1;
}

if (PIND.4==0)
{
 set_s=set_m*60;
 sprintf(buff,"SetTime %dmin ",set_m);
 lcd_gotoxy(2,1);
 lcd_puts(buff);
 lcd_gotoxy(4,0);
 lcd_puts(" MODE UV");
 PORTC.7=0;
```

```
 if(PIND.1==0)
{
delay_ms(80);
set_m--;
}

if(PIND.0==0)
{
delay_ms(80);
set_m++;
}


if(PIND.2==0){
tim_s=0;
start=1;
while(start){
   lcd_clear();
   lcd_gotoxy(2,0);
   lcd_puts("UV Working");
   PORTD.7=1;

   remain_s=set_s-tim_s;
   remain_m=remain_s/60;
   if(remain_s>60)sprintf(buff,"Remain %dmin ",remain_m);
   else sprintf(buff,"Remain %ds ",remain_s);
   lcd_gotoxy(2,1);
   lcd_puts(buff);
   if(remain_s==0)start=0;
```

```
        }

        lcd_clear();
       }
       }

       else
       {
       PORTD.7=0;
       PORTD.6=0;
       PORTC.7=1;
       //delay_ms(80);
       PORTC.7=0;
       //delay_ms(80);
       }

      }
}
```

Lampiran 3