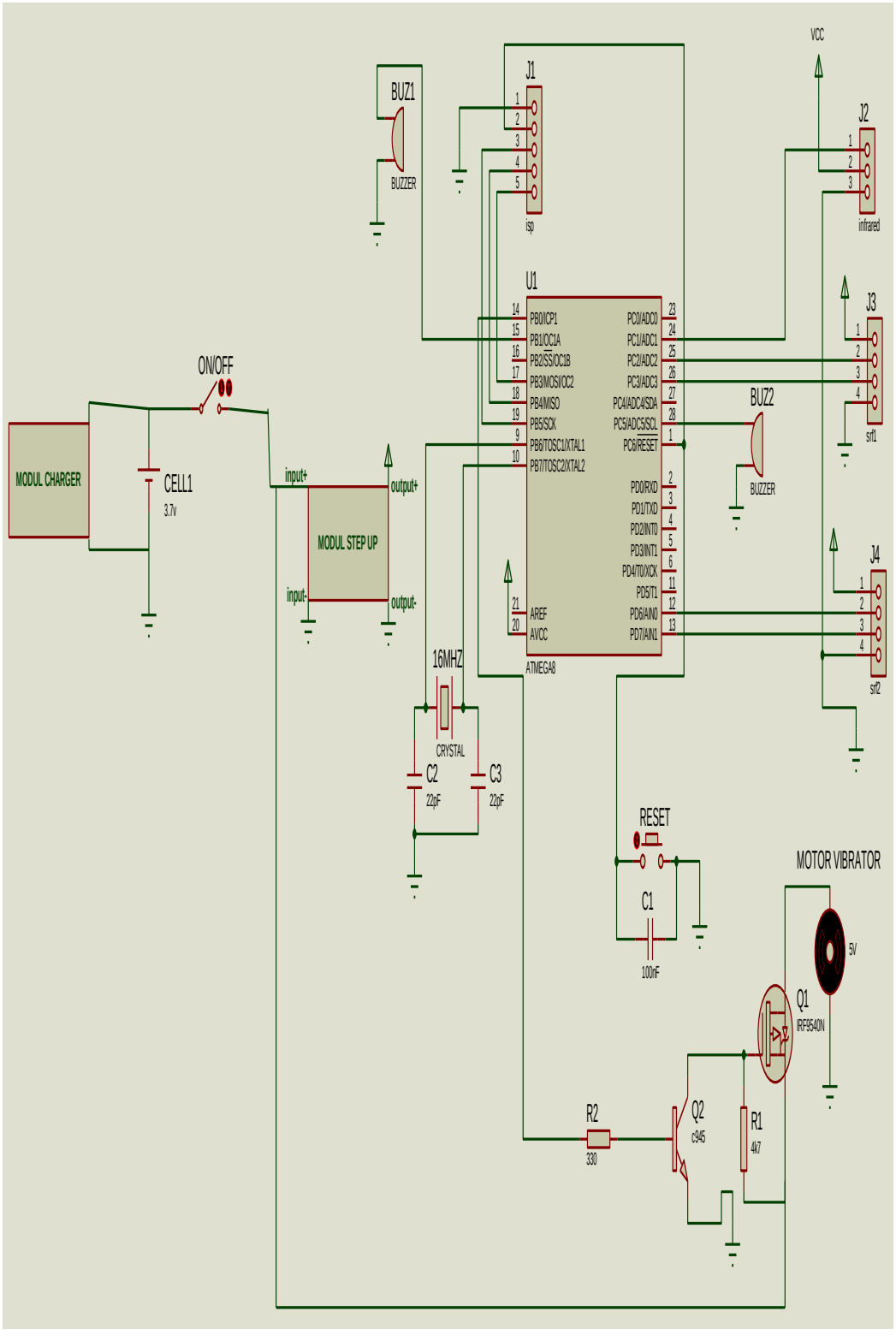
LAMPIRAN

1. Rangkaian Keseluruhan Alat



2. Tampilan Program

```
/*****************************************************
Chip type               : ATmega8
Program type            : Application
AVR Core Clock frequency: 16,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/
#include <mega8.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>
#define batery_sensor 0

#define ir          PINC.1
#define triger1     PORTC.2
#define echo1       PINC.3
#define triger2     PORTD.6
#define echo2       PIND.7
#define motor       PORTB.0
#define buzzer      PORTB.1

//char buffer[33];
#define ADC_VREF_TYPE 0x40

// Read the AD conversion result
unsigned int 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;
}
float read_volt(){
float volt,voltout;
```

```
float voltmax=3.5,involtmax=20.0;
 int data=read_adc(batery_sensor);

 // rumus sensor tegangan
 // r1= 4,7k r2= 1k
 // involt max= 20.0 volt (sesuai kebutuhan yang peting hasil perhitungan voltmax
tidak lebih dari 5v)
 // mencari voltmax ( max 5V), voltmax=r2/(r1+r2)*involtmax
 // voltmax=1k/(4,7+1)*20.0 h
 // voltmax= 3.50 volt

 volt=data*((float)5/1023);  // mengubah nilai adc ke tegangan
 voltout=volt*((float)involtmax/voltmax); // mengubah nilai tegangan kecil ke
tegangan sensor
return voltout;
}
// fungsi baca sensor ultrasonik
float read_ultra1(){
unsigned int pulsa=0;
float mydistance;  // lokal variable
// suara ultrasonik on
triger1=1;
// jeda
delay_us(20);
// suara ultrasonik off
triger1=0;
// baca pulsa dan konvert ke cm
while(echo1==0);
while(echo1==1){
  pulsa++;
}
mydistance = (float)pulsa/160;
// nilai balik
return mydistance;
}

// fungsi baca sensor ultrasonik
float read_ultra2() {
unsigned int pulsa=0;
float mydistance;  // lokal variable
// suara ultrasonik on
```

```
triger2=1;
// jeda
delay_us(20);
// suara ultrasonik off
triger2=0;
// baca pulsa dan konvert ke cm
while(echo2==0);
while(echo2==1){
  pulsa++;
}
mydistance = (float)pulsa/160;
// nilai balik
return mydistance;
}
float bat;
  float jarak1;
  float jarak2,jarakterpendek;
  int time1=0,time2=0,time3=0,time4=0,time5=0;

void suara(int jeda){

if(time1>=jeda){
buzzer=1;
time2++;
if(time2>=jeda){
time1=0;
time2=0;
}
}
else{
time1++;
buzzer=0;
}
}
void getar(int jeda){
if(time3>=jeda){
motor=0;
time4++;
if(time4>=jeda){
time3=0;
time4=0;
```

```
}
}
else{
time3++;
motor=1;
}
}

void programku(){

bat = read_volt();
 jarak1 = read_ultra1();
 jarak2 = read_ultra2();
 jarakterpendek = min(jarak1,jarak2);

 if(jarakterpendek < 80){
 if(time5<10){
 time5++;
 }
 else{

 if(jarakterpendek < 50)suara(6);
 else suara(10);
 }
 }
 else{
 buzzer=0;
 time5=0;
 }
  if(ir==1) getar(15);
  else motor=0;

  if(bat<= 3.8)suara(3);
  else buzzer=0;

  delay_ms(10);
  }
void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
```

```
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=0
PORTB=0xFF;
DDRB=0x03;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=Out Func1=In Func0=In
// State6=T State5=T State4=T State3=P State2=0 State1=P State0=T
PORTC=0x0A;
DDRC=0x04;

DDRC.5=1;

// Port D initialization
// Func7=In Func6=Out Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=P State6=0 State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x80;
DDRD=0x40;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// 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
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
```

```
TCNT1L=0x00;
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=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

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

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// 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: 1000,000 kHz
// ADC Voltage Reference: AVCC pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
```

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

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

while (1)
    {
     // Place your code here

     programku();
     }
}
```