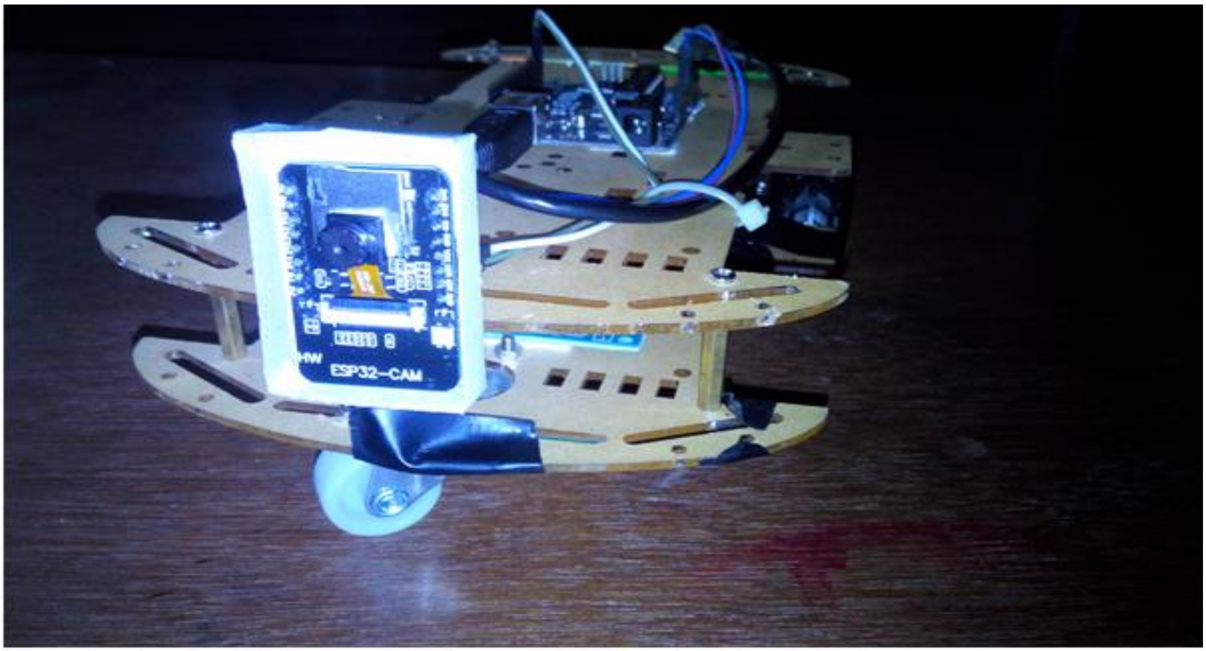
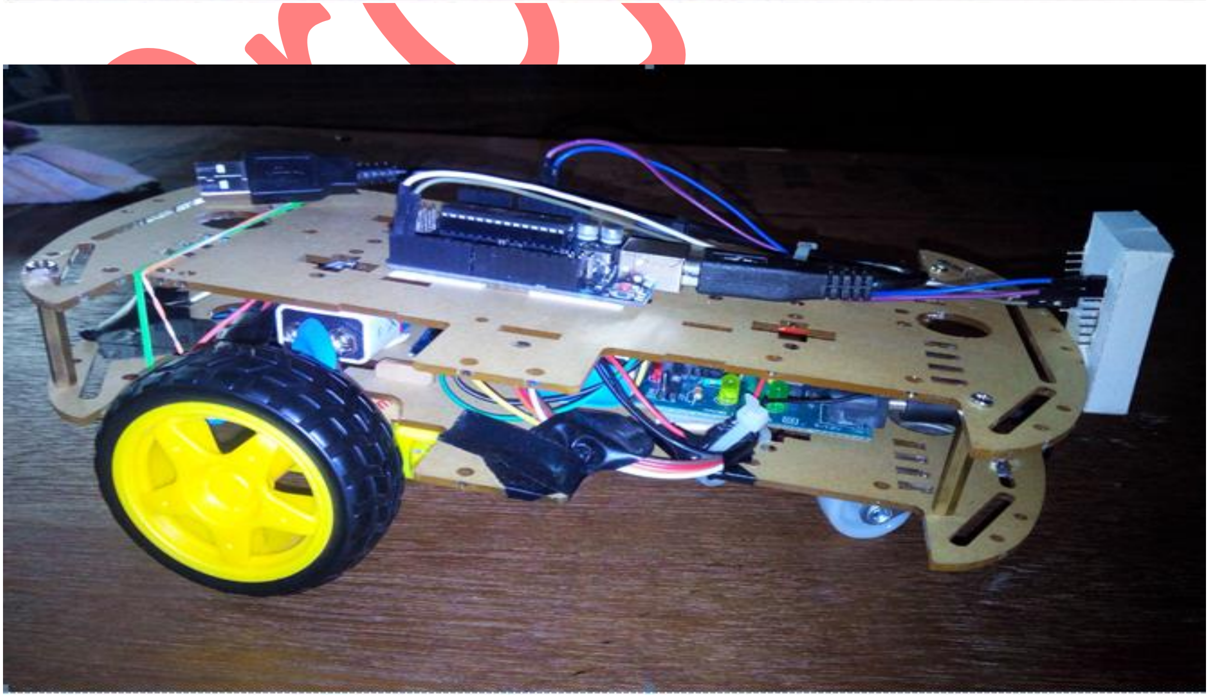
GESTURE CONTROLLED

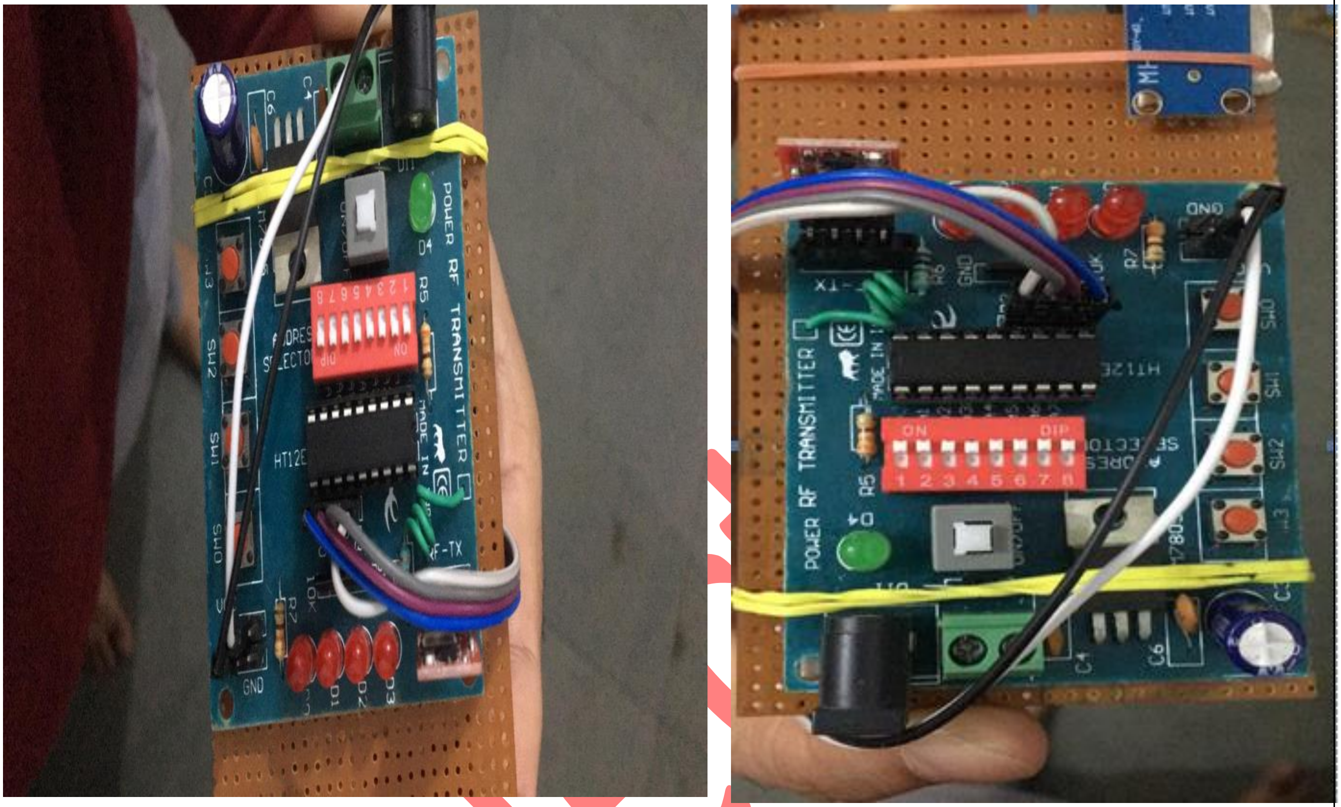
SPY ROBOT

RECEIVER: FINAL OUTPUT





TRANSMITTER:



Components used:- Arduino uno R3, 433Mhz transmitter and recievers , HT12E and HT12D ic’s ,

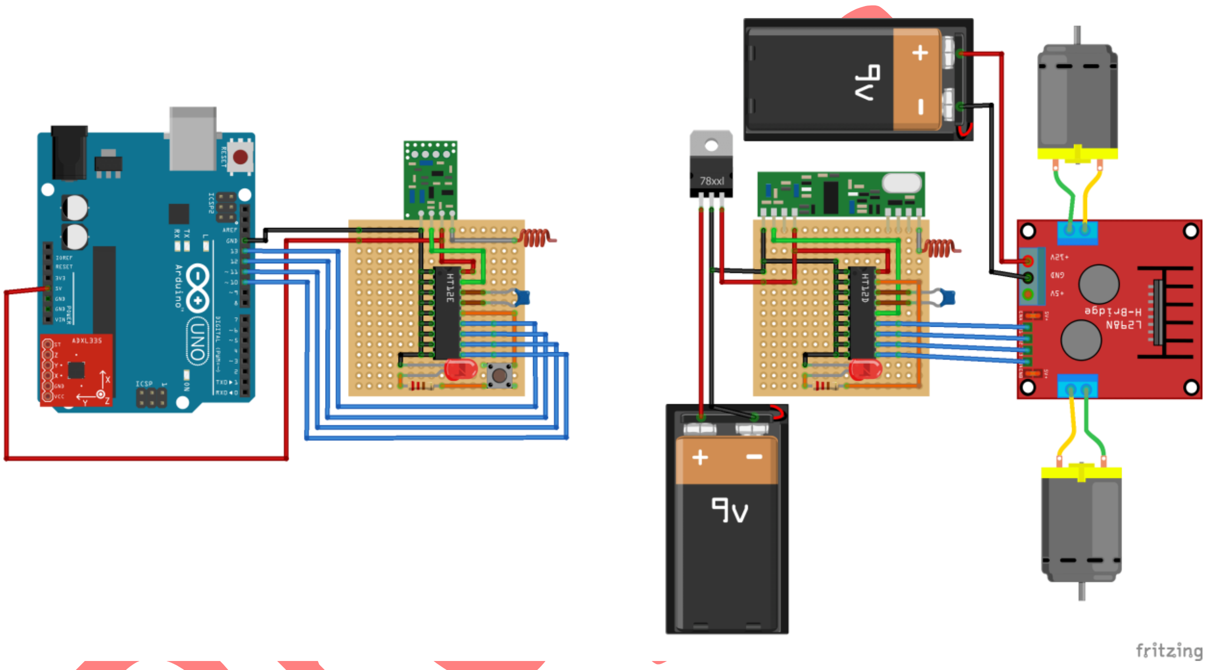
Accelerometer (ADXL335, ADXL345, MPU6050) , ESP32-CAM, L293D MOTOR DRIVER, Dc motor.

DESCRIPTION:- This project has two parts one is the transmitter and the other one is the receiver transmitter is used to generate triggering signals by using accelerometer and then these signals are than transmitted wirelessly using the transmitter and received by reciever than these signals are used to trigger the receiver circuit and to produce the final output.

<u>WORKING</u>:- In this project an accelerometer (adxl335) is used to send the analog values according to the gesture of the transmitter circuit to the arduino analog pins and after that the arduino read those signals and convert them to their corresponding digital values and fed this value to the encoder ic which convert parallel data into serial data and fed this encoded value to the transmitter than this data is received by the receiver and then that data is again fed to the decoder ic which convert the serial data to the parallel data again and finally this data is fed to the motor driver and after that according to the polarity of the pin car will move.

And for live video transmission and feedback I have used ESP-32 CAM which will provide the live feed over the server cloud created by this cam and we can acces that live feed through that server on phone as well as any other device connected to that cam server.

Circuit Diagram:-



APPLICATIONS:-

1. It can be used in remote surveillance & military.

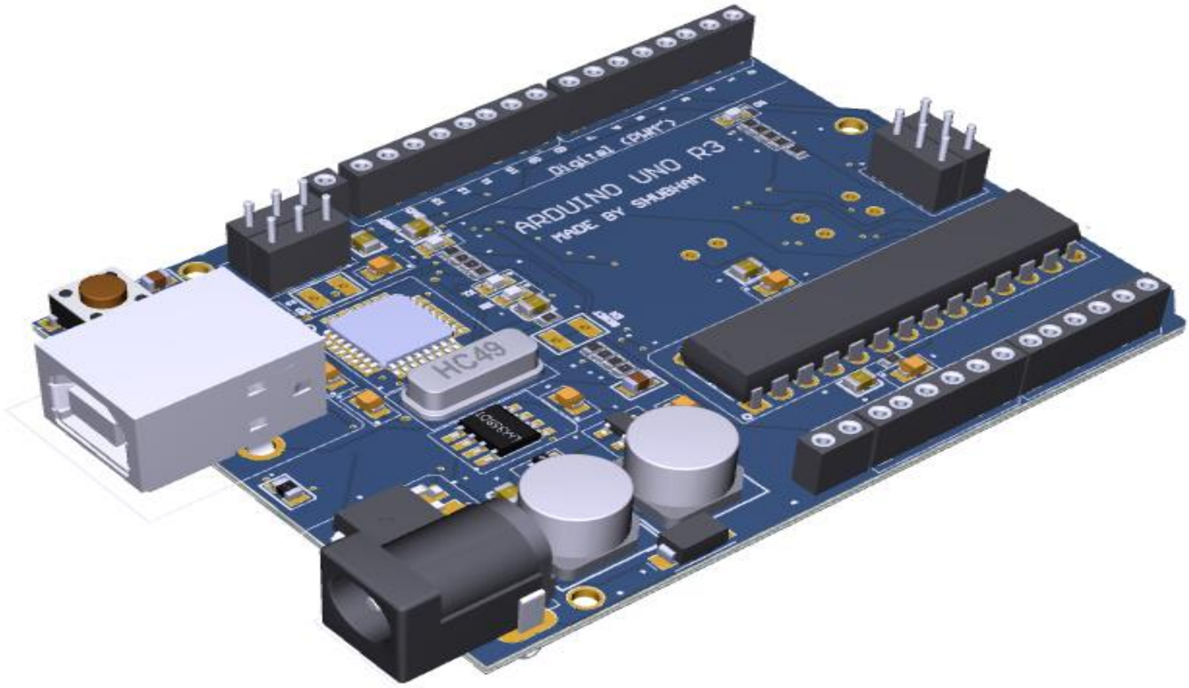
2. Gesture controlled grade robots can be made.

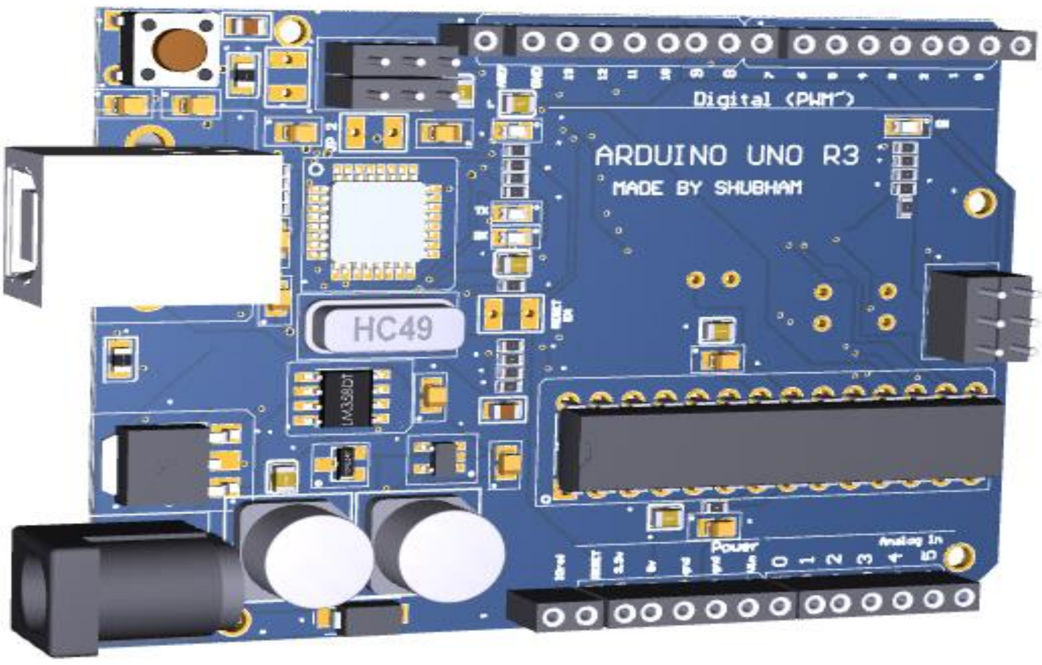
3. It can be used by physically challenged In Wheelchairs.

ARDUINO UNO DESIGN

USING ALTIUM DESIGNER

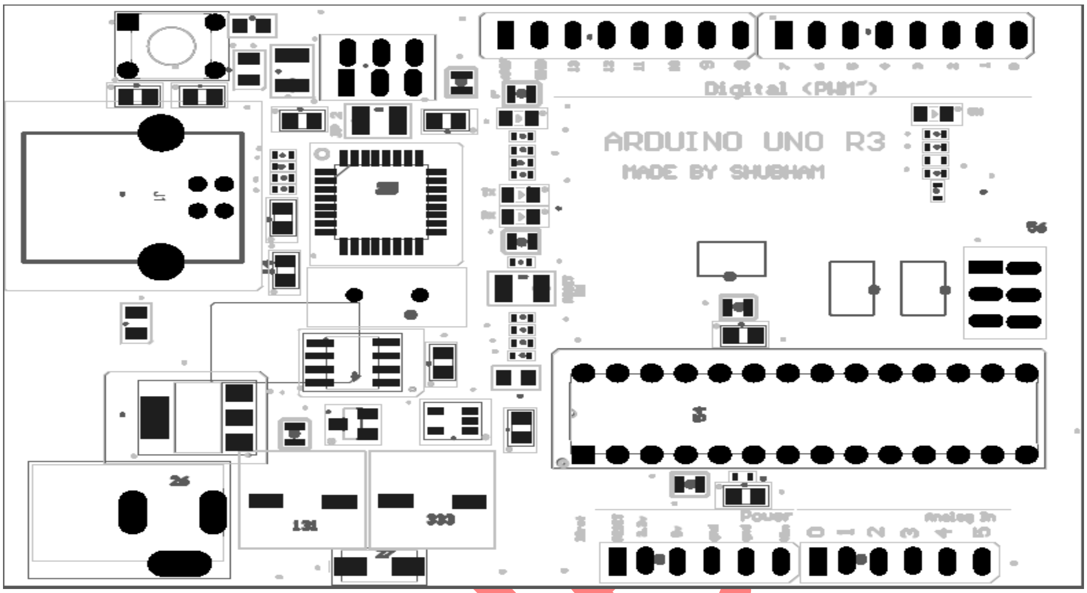
3d output:



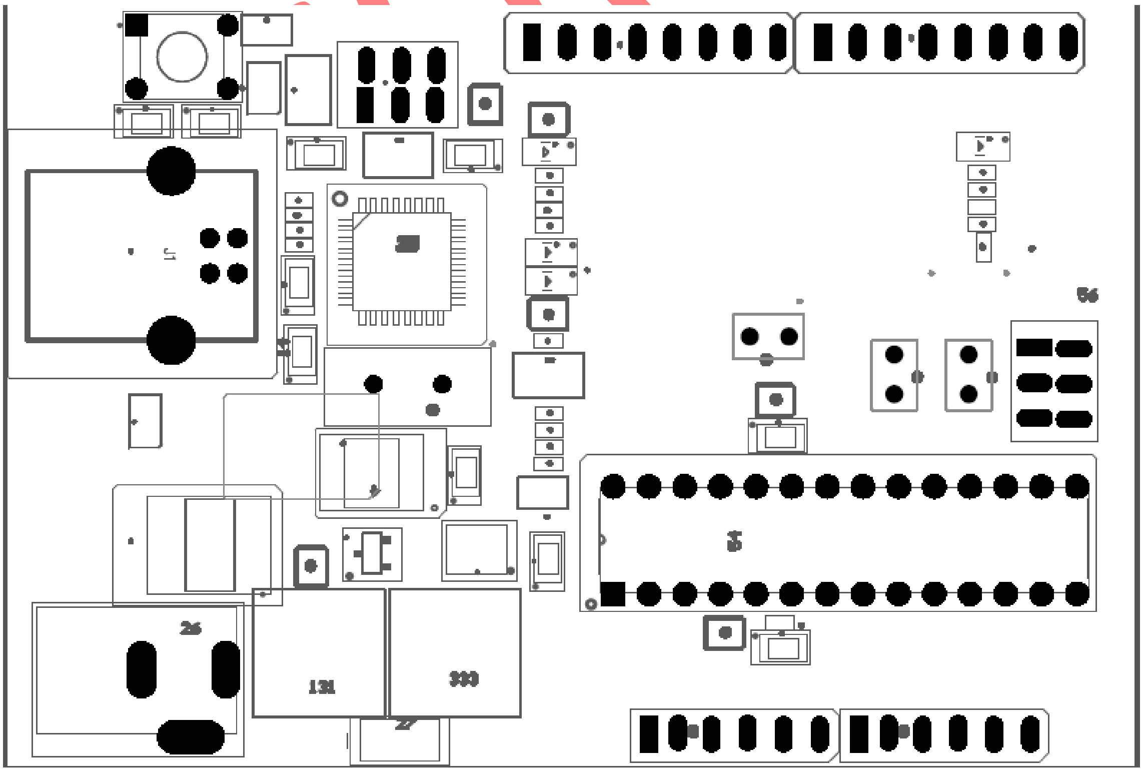


ASSEMBLY DRAWINGS:-

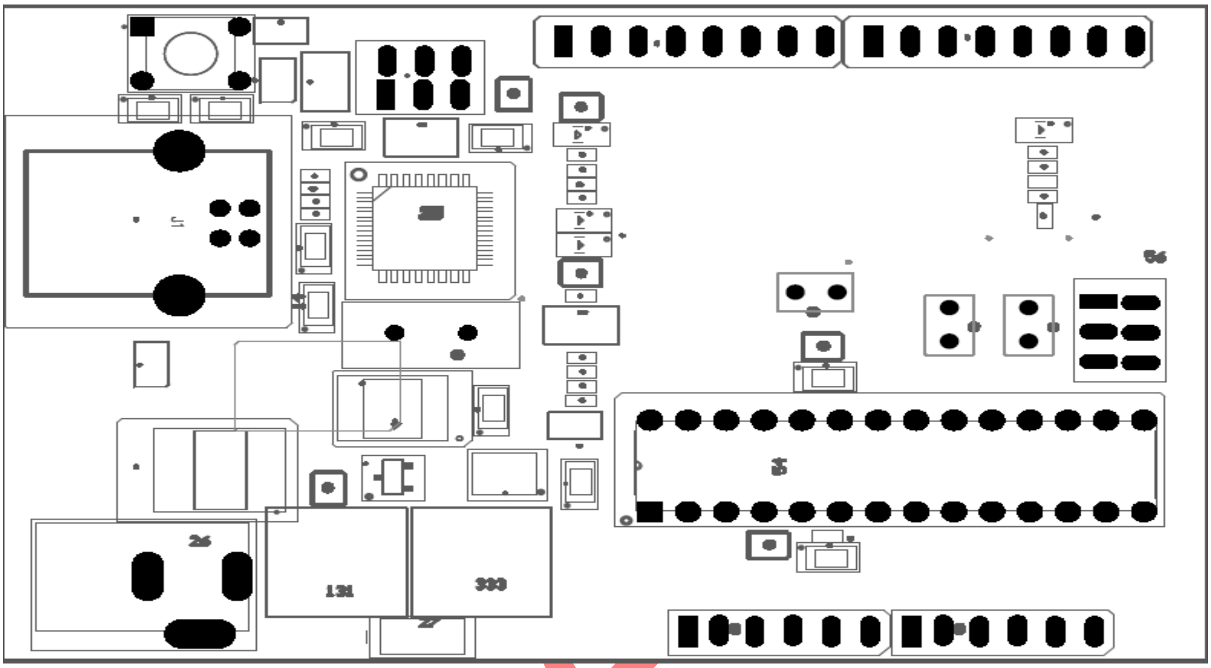
Layer1:



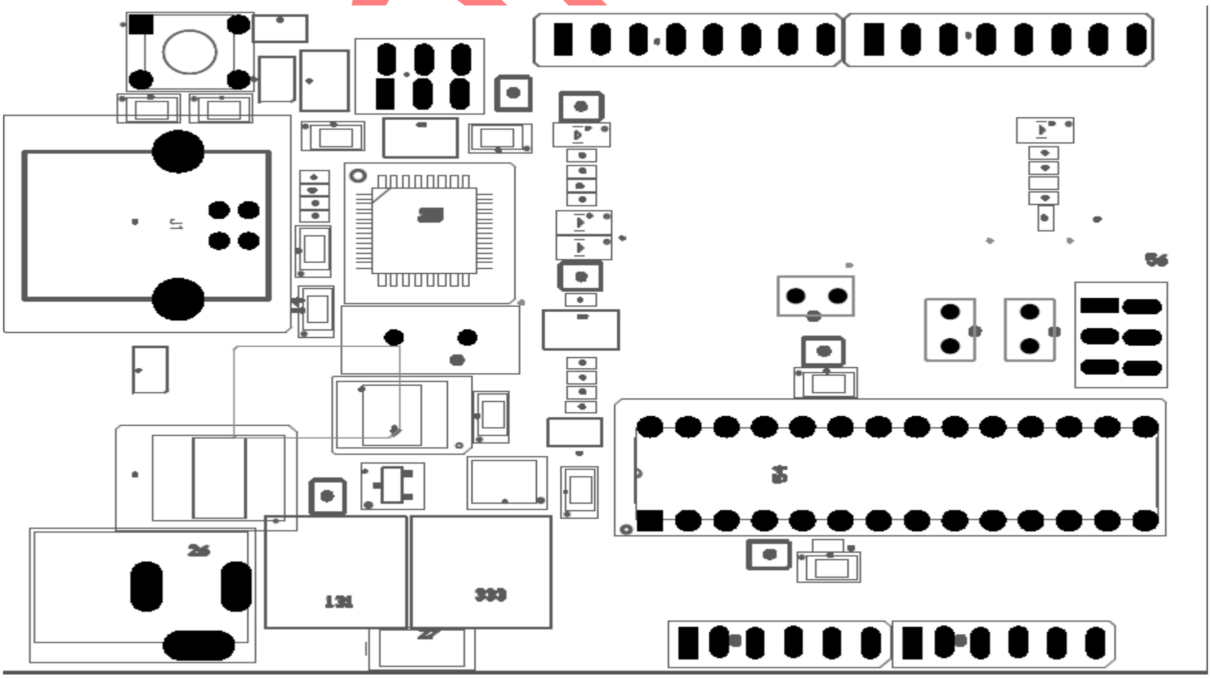
Layer2:



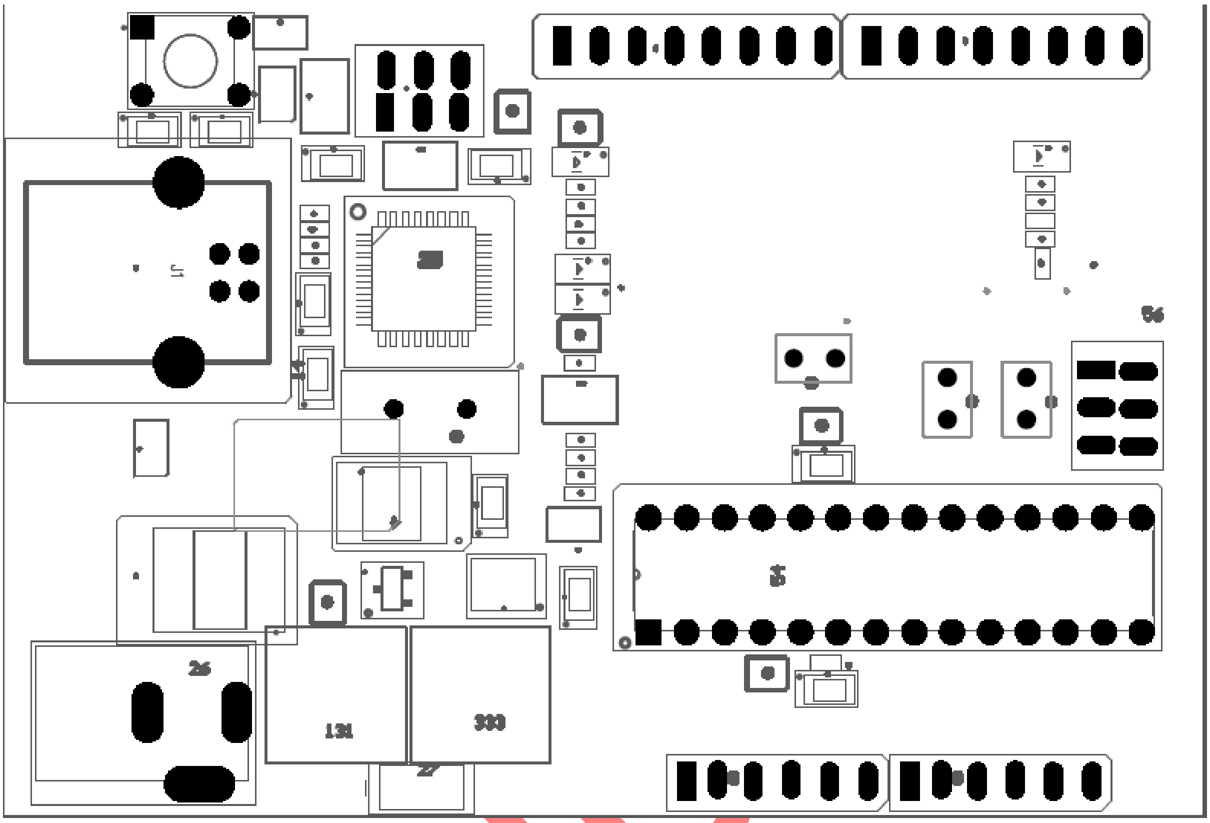
Layer3:



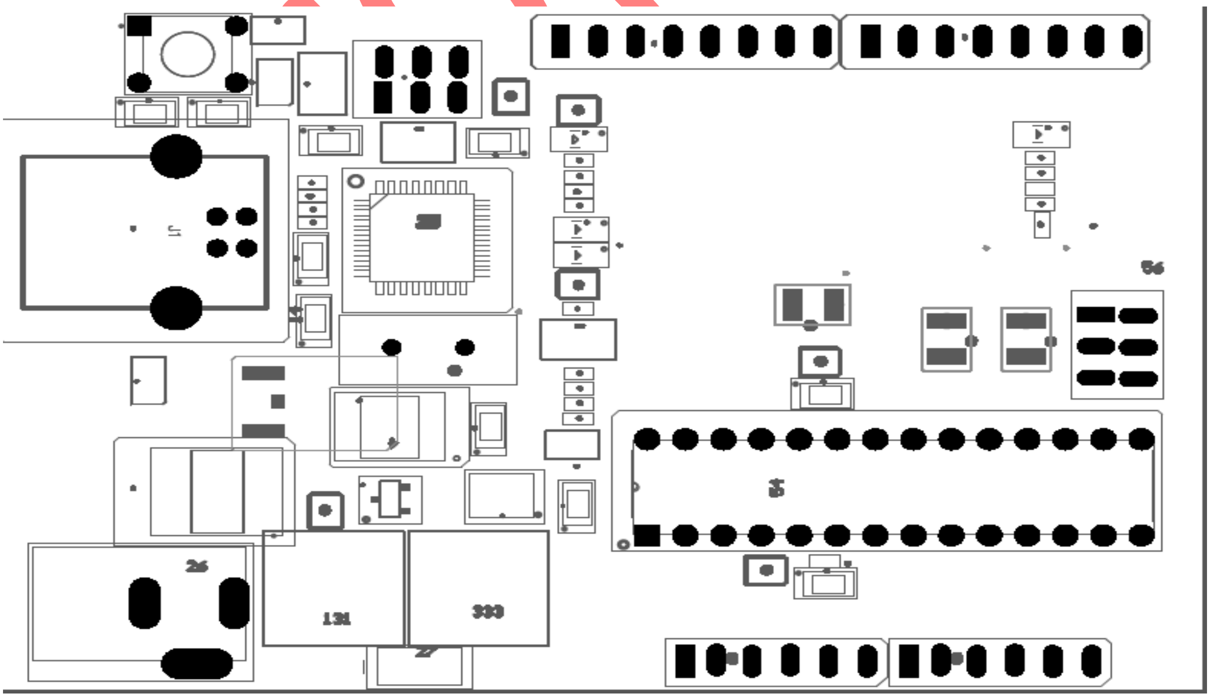
Layer4:



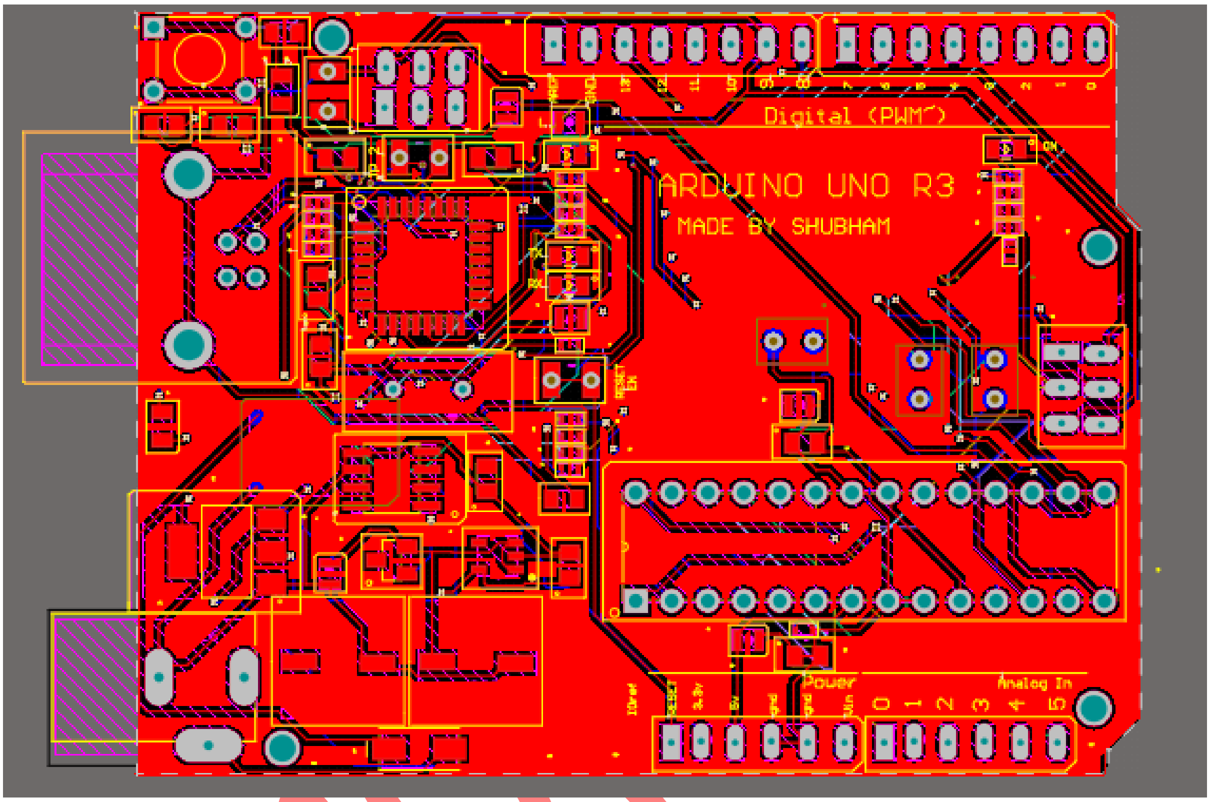
Layer5:



LAYER6:

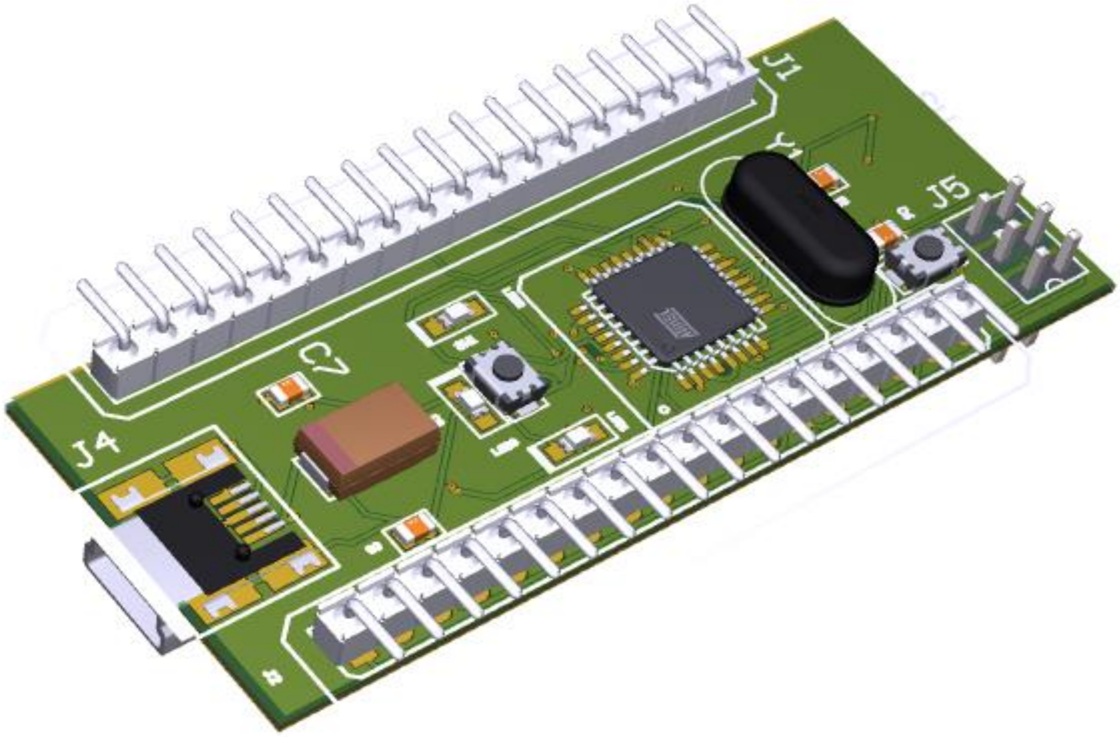


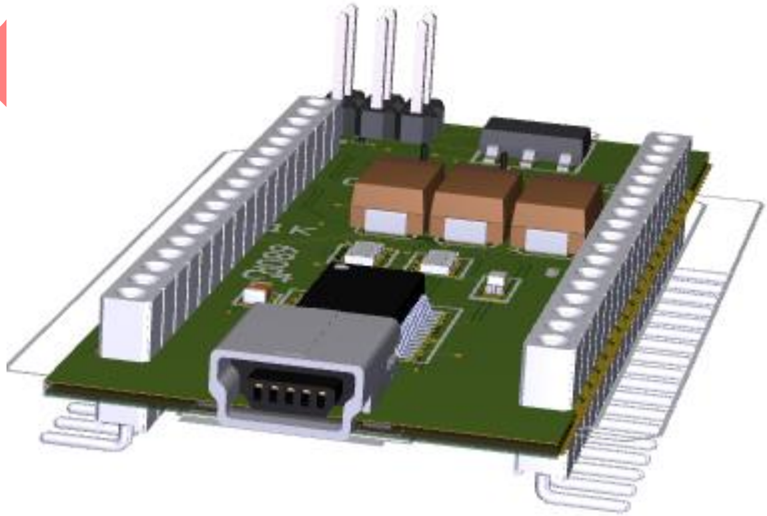
ROUTING DIAGRAM:-



ARDUINO NANO DESIGN USING ALTIUM DESIGNER

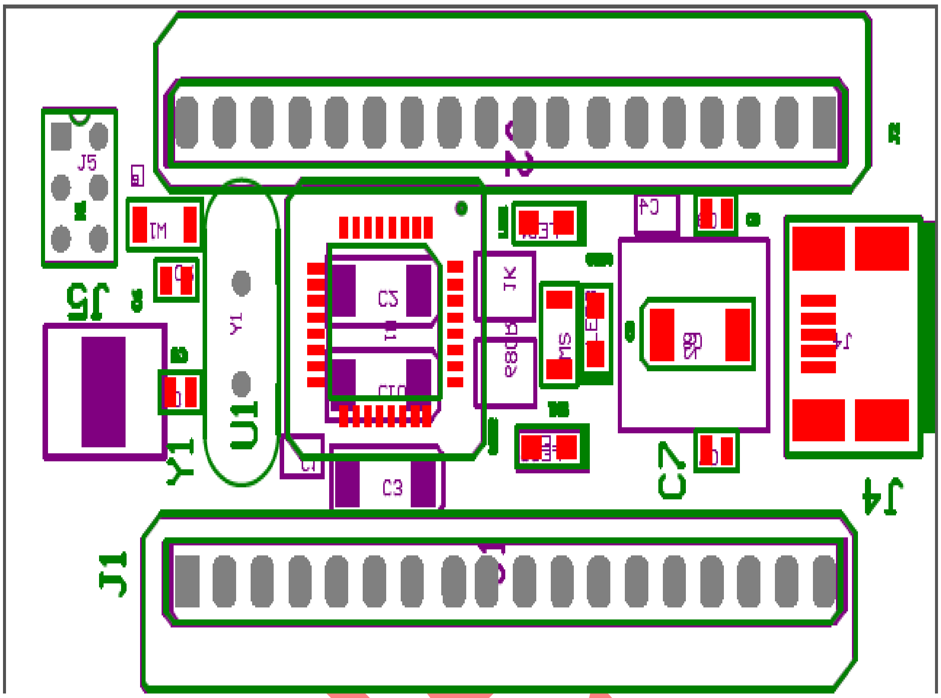
3d output:



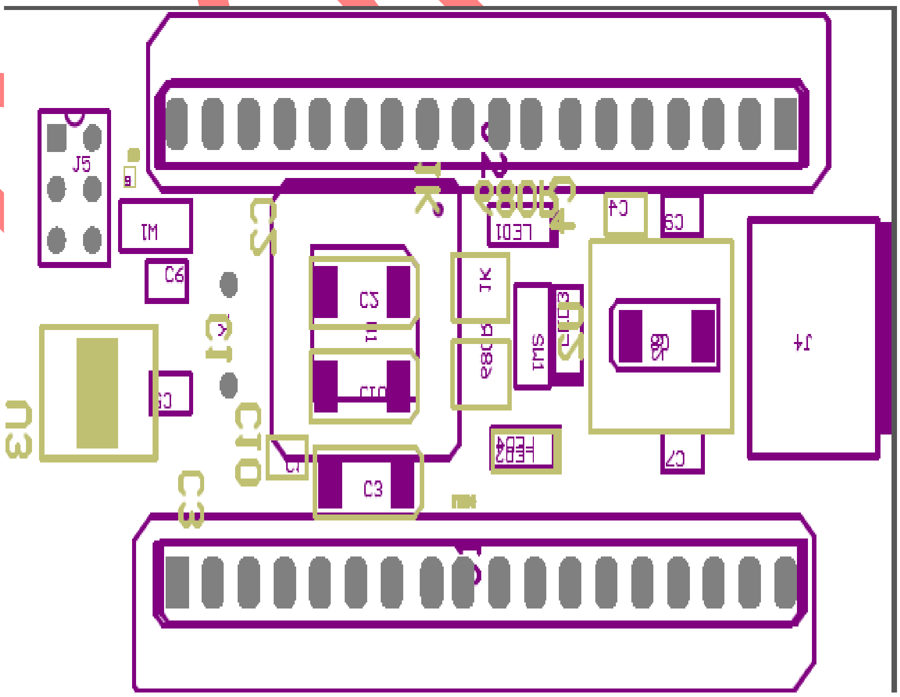


ASSEMBLY DRAWINGS:-

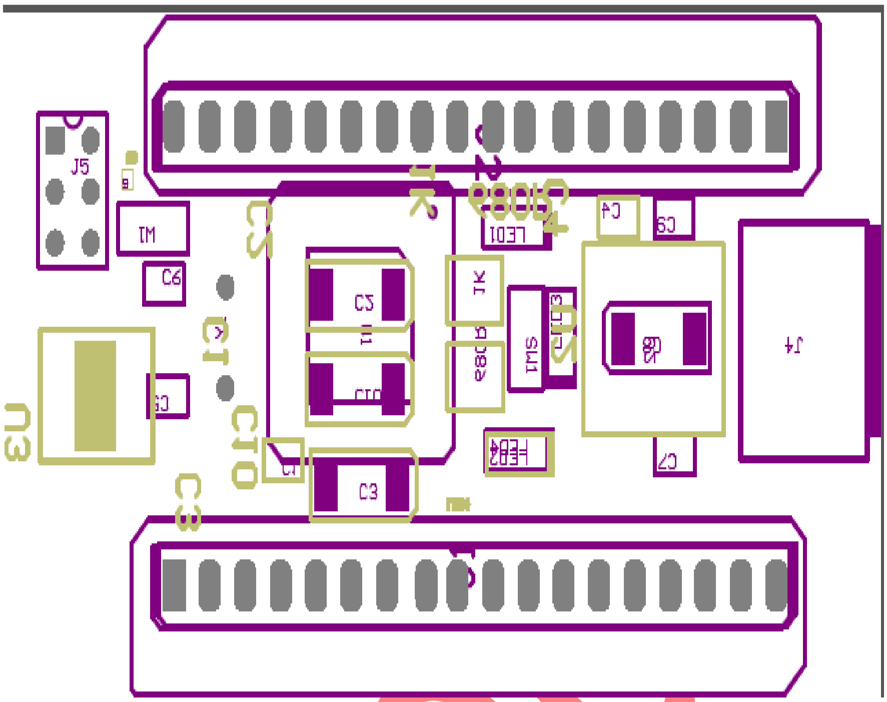
Layer1:



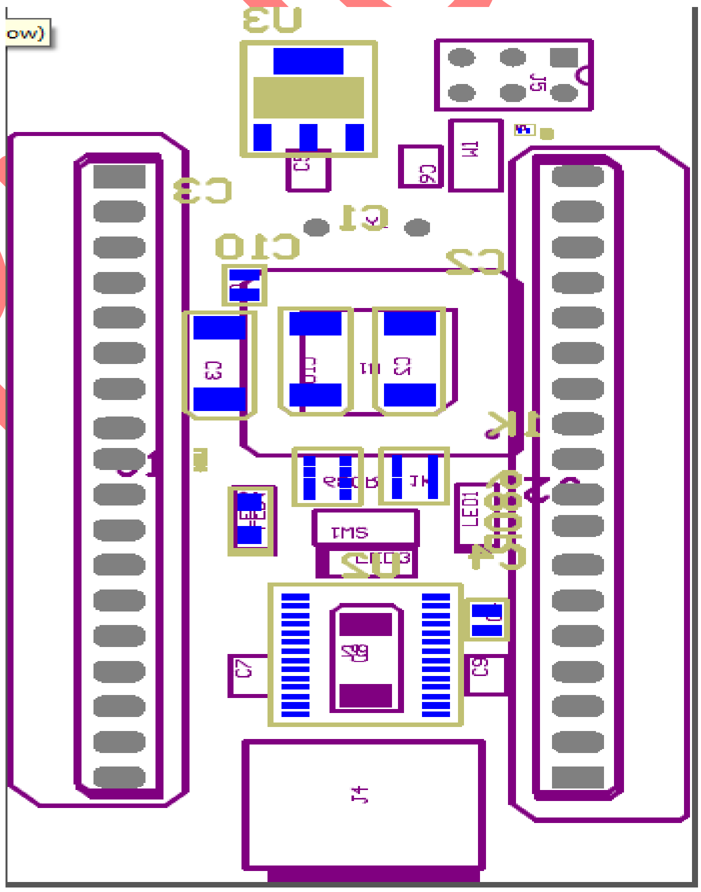
Layer2:



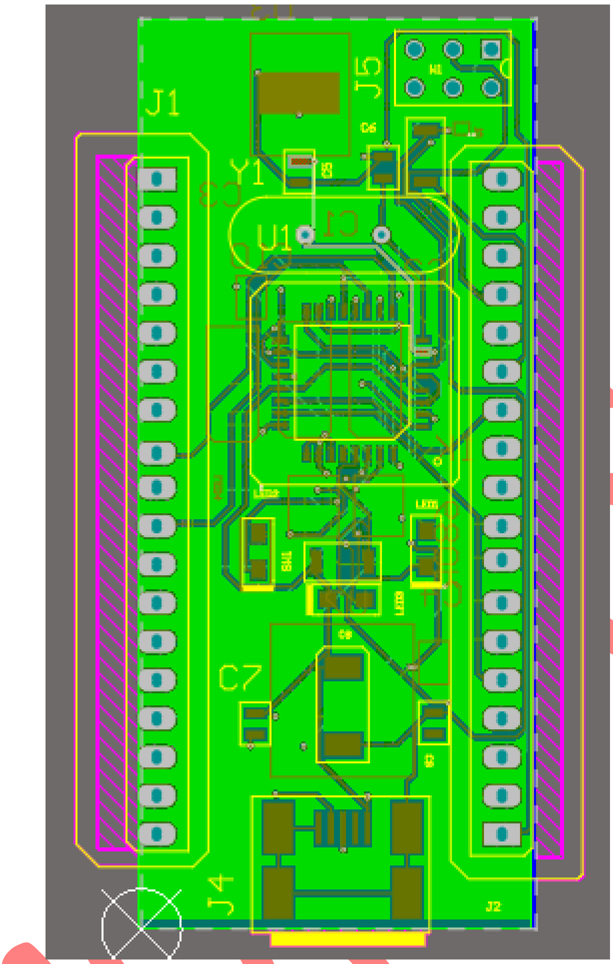
Layer3:-



Layer4:

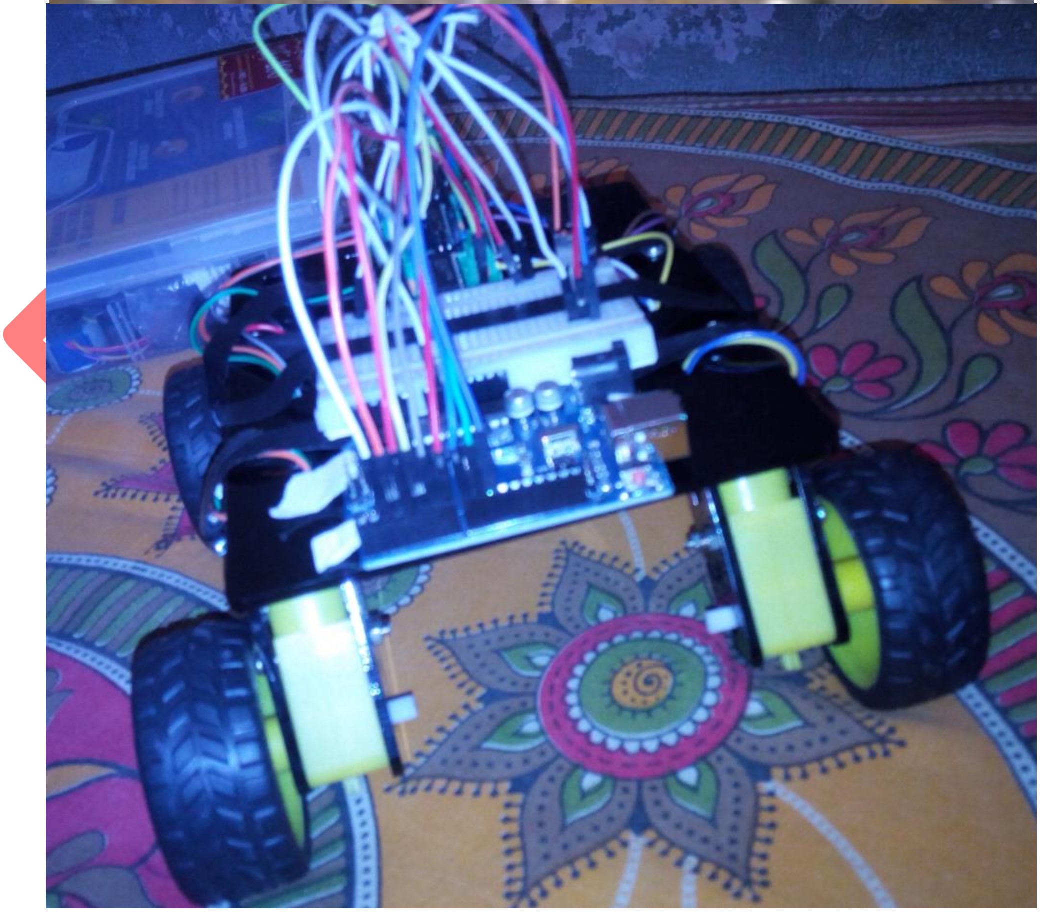


ROUTING DIAGRAM:-



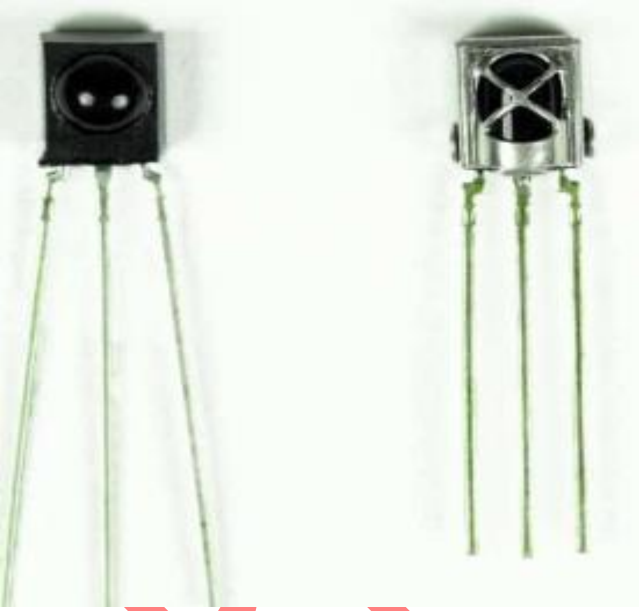
REMOTE CONTROLLED CAR

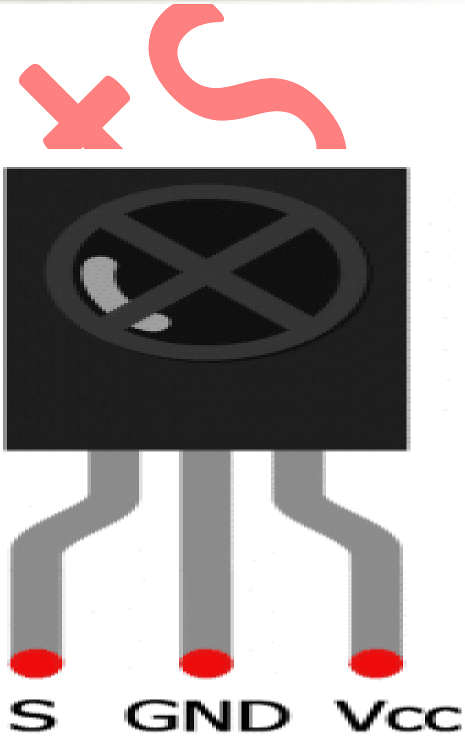
Final output:



REMOTE CONTROL:-

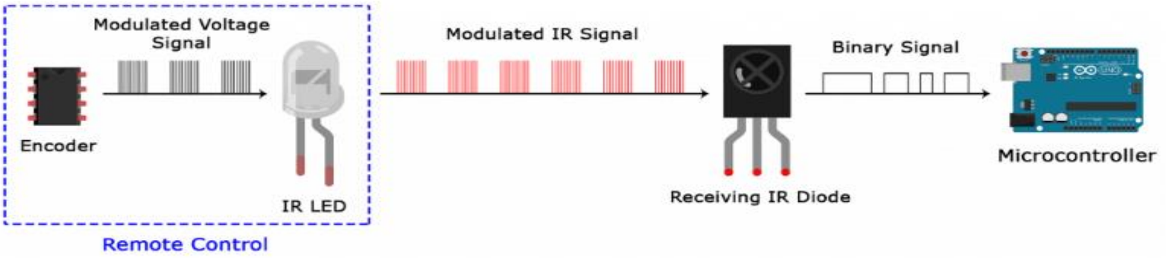
TRANSDUCER:-





Components used:- Arduino uno R3, transducer ,l293d motor driver ic ,remote control etc.

WORKING:-



CODE:-

```
#include <IRremote.h>


const int RECV_PIN = 10;

int timer = 100;          // The higher the number, the slower the timing.

int ledPins[] = {

  2,3,4,5,6,7,8,9

};

int pinCount = 8;

IRrecv irrecv(RECV_PIN);

decode_results results;

unsigned long key_value = 0;


void setup(){

  Serial.begin(9600);

   for (int thisPin = 0; thisPin < pinCount; thisPin++) {

    pinMode(ledPins[thisPin], OUTPUT);

  }

  irrecv.enableIRIn();

  irrecv.blink13(true);

}


void loop(){

  if (irrecv.decode(&results)){


        if (results.value == 0XFFFFFFFF)

          results.value = key_value;
```

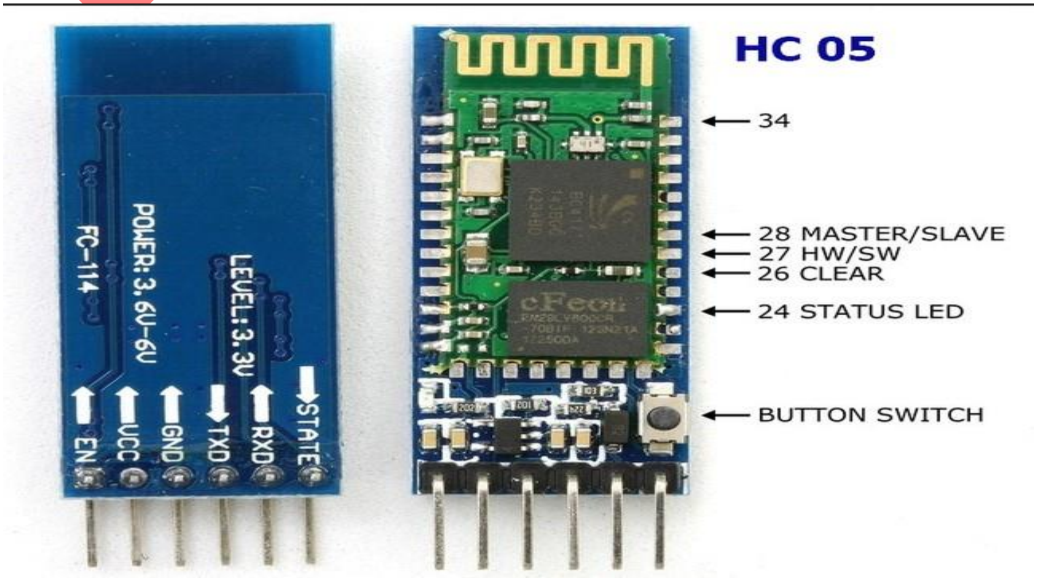
```
switch(results.value){
 case 0xFF30CF://1 forwARD
 digitalWrite(ledPins[0],HIGH);
 digitalWrite(ledPins[1],LOW);
 digitalWrite(ledPins[2],HIGH);
 digitalWrite(ledPins[3],LOW);
 digitalWrite(ledPins[4],HIGH);
 digitalWrite(ledPins[5],LOW);
 digitalWrite(ledPins[6],HIGH);
 digitalWrite(ledPins[7],LOW);
 delay(timer);
 break;
 case 0xFF18E7://2 backward
 digitalWrite(ledPins[0],LOW);
 digitalWrite(ledPins[1],HIGH);
 digitalWrite(ledPins[2],LOW);
 digitalWrite(ledPins[3],HIGH);
 digitalWrite(ledPins[4],LOW);
 digitalWrite(ledPins[5],HIGH);
 digitalWrite(ledPins[6],LOW);
 digitalWrite(ledPins[7],HIGH);
 delay(timer);
 break;
 case 0xFF7A85://3 right
 digitalWrite(ledPins[0],HIGH);
 digitalWrite(ledPins[1],LOW);
 digitalWrite(ledPins[2],LOW);
 digitalWrite(ledPins[3],LOW);
 digitalWrite(ledPins[4],HIGH);
```

```
    digitalWrite(ledPins[5],LOW);

    digitalWrite(ledPins[6],LOW);

    digitalWrite(ledPins[7],LOW);

    delay(timer);

    break;

    case 0xFF10EF://4 left

    digitalWrite(ledPins[0],LOW);

    digitalWrite(ledPins[1],LOW);

    digitalWrite(ledPins[2],HIGH);

    digitalWrite(ledPins[3],LOW);

    digitalWrite(ledPins[4],LOW);

    digitalWrite(ledPins[5],LOW);

    digitalWrite(ledPins[6],HIGH);

    digitalWrite(ledPins[7],LOW);

    delay(timer);

    break ;

    case 0xFFA857://stop

    digitalWrite(ledPins[0],LOW);

    digitalWrite(ledPins[1],LOW);

    digitalWrite(ledPins[2],LOW);

    digitalWrite(ledPins[3],LOW);

    digitalWrite(ledPins[4],LOW);

    digitalWrite(ledPins[5],LOW);

    digitalWrite(ledPins[6],LOW);

    digitalWrite(ledPins[7],LOW);

    delay(timer);

    break ; }       key_value = results.value;

   irrecv.resume();

}}
```

AVR BASED BLUETOOTH CONTROL CAR

Final output:

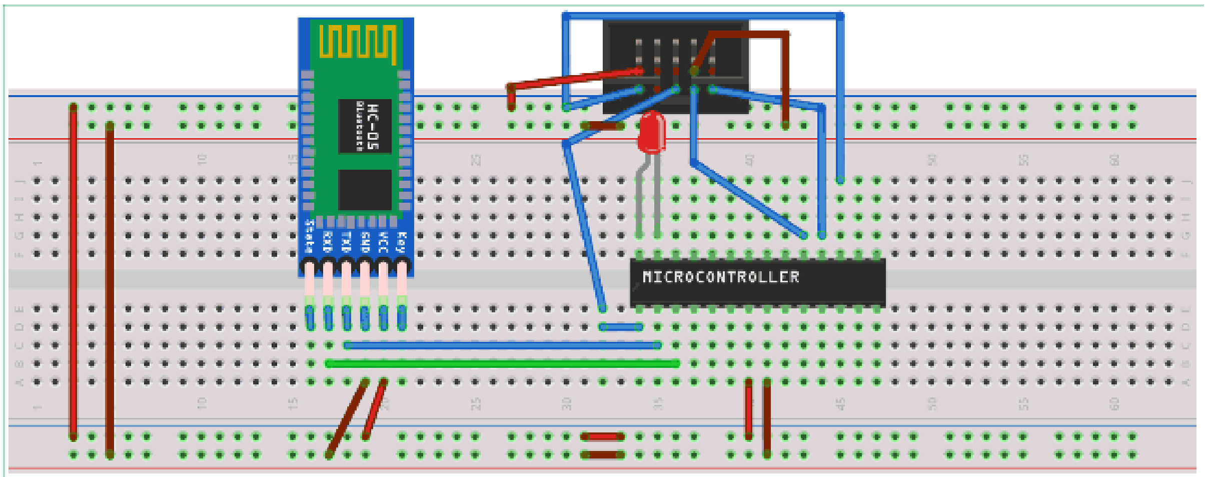
BLUETOOTH MODULE:-



Material required:- Atmega 16 , Bluetooth module (Hc-05) ,Motror driver L293d ic .

Circuit Diagram:

Circuit diagram for **interfacing Bluetooth HC-05 with AVR** is given below.



CODE:-

```
#include<avr/io.h>

#define F_CPU 8000000

#include<util/delay.h>

#include<lcdavr.h>

#include<serial.h>


unsigned char v;

void main()

{

DDRB=0XFF;

lcd_init();

usart_init();

DDRA=0XFF;
```

```
while(1)

{

usart_string("WELCOME TO SERIAL");

for(int x=0XC0;x<=0XCF;x++)
{
v=usart_rec();
lcd_command(x);
lcd_data(v);
lcd_command(0x06);
_delay_ms(10);
if(v=='F')
{
lcd_clear();
lcd_command(0x80);
lcd_string("FORWARD ");
PORTA=0b00001010;
}
 else if(v=='B')

{
lcd_clear();
lcd_command(0x80);
lcd_string("BACKWARD ");
PORTA=0b00000101;
}
else if(v=='R')
{
lcd_clear();
lcd_command(0x80);
lcd_string("RIGHT ");
PORTA=0b00001000;
}
else if(v=='4')
{
lcd_clear();
lcd_command(0x80);
lcd_string("left ");
PORTA=0b00000010;
}
else if(v=='S')
{
lcd_clear();
lcd_command(0x80);
lcd_string("stop);
PORTA=0X00;
}}}}
```

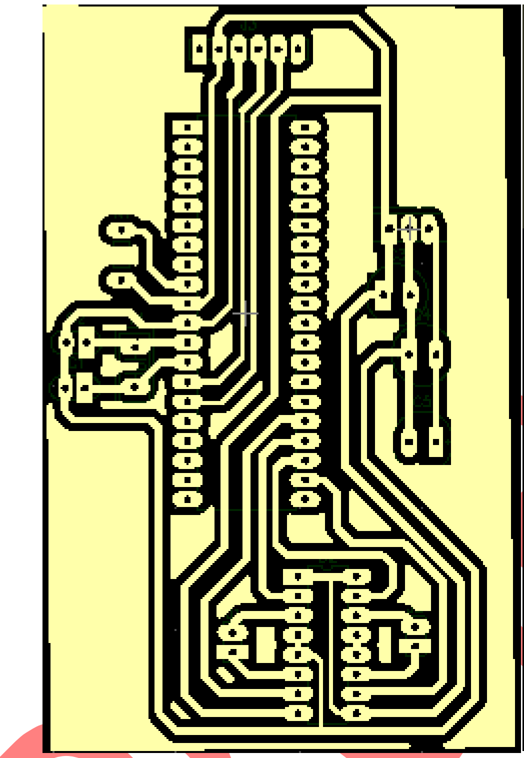
WORKING:-

In this project i used HC-0 bluetooth Module tonnect it through an android app and then controlling it's output

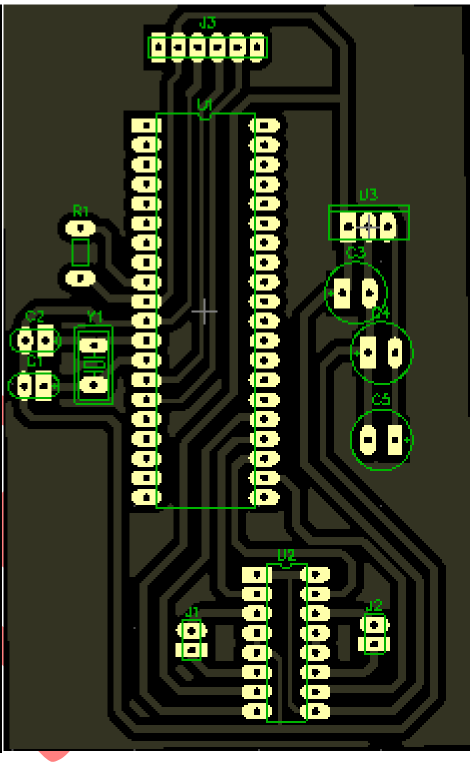
To control the microcontroller AtMEGA-16 which communicates with the bluetooth module via serial communication protocol using the transmitter and receiver pin in the microcontroller and after that this data is then fed to the motor driver correspondingly.

PCB LAYOUT:-

BOTTOM VIEW :



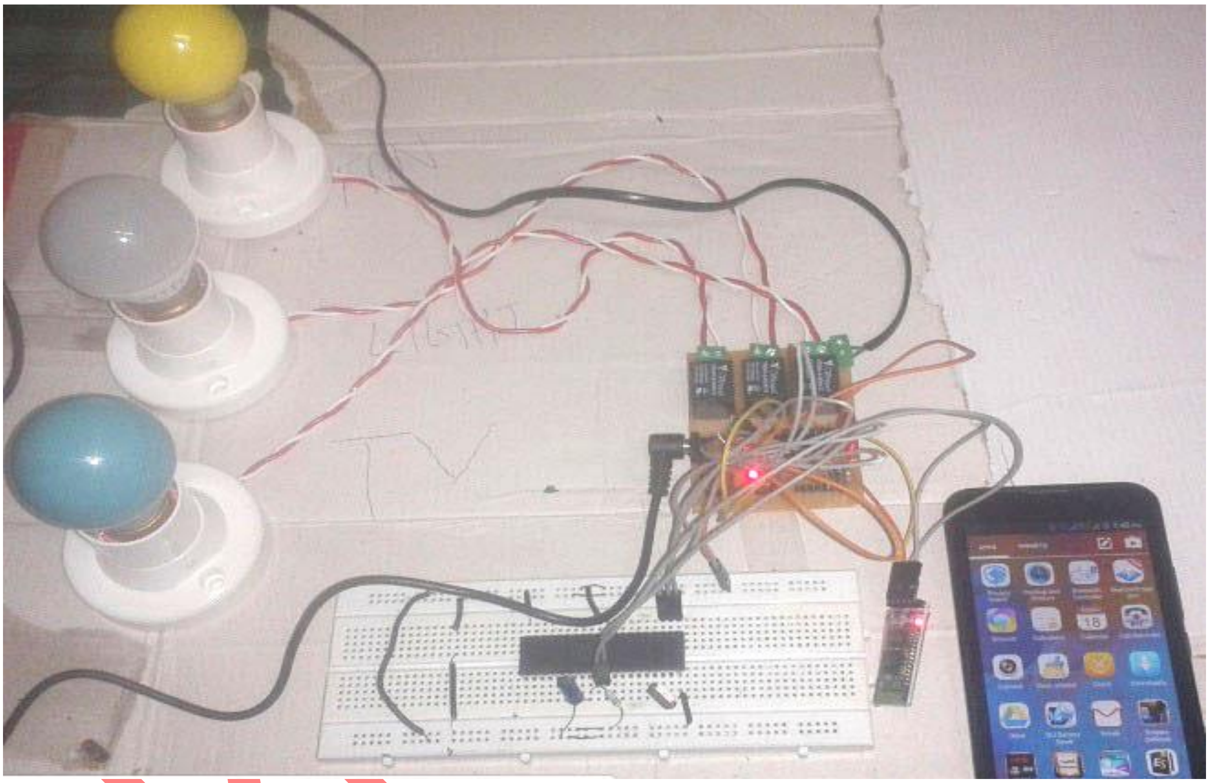
FRONT VIEW:



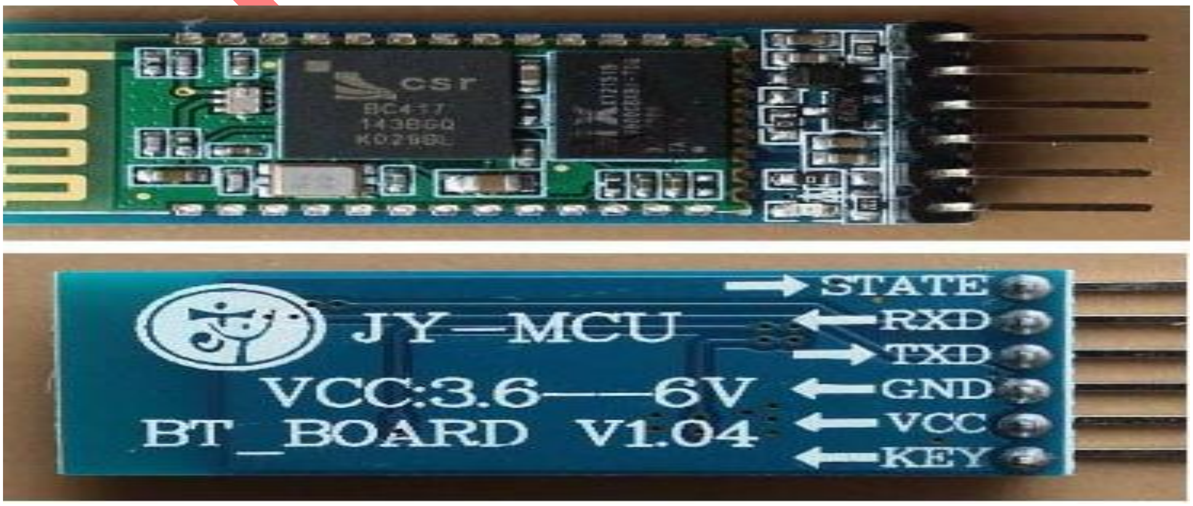
AVR BASED

HOME AUTOMATION

Final output:-

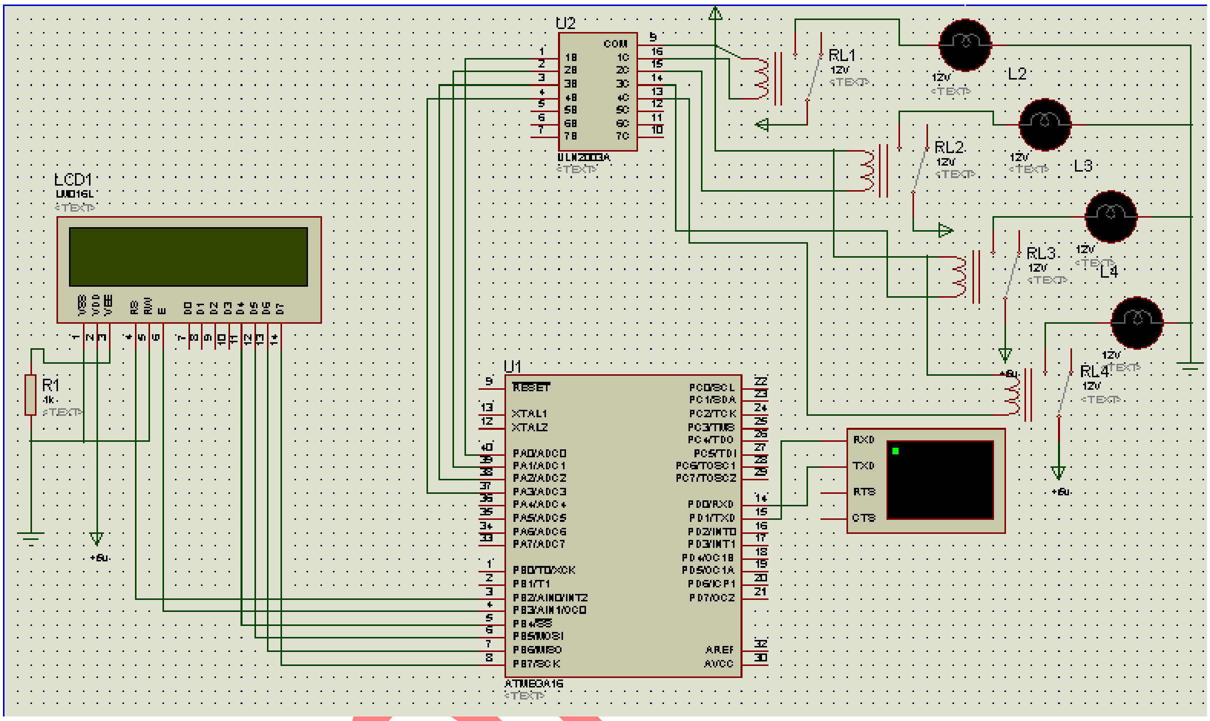


Bluetooth module:



Material used:- Atmega 16 , Uln2003A relay driver ,HC-05 bluetooth module ,light bulbs .

Schematic Diagram:-



CODE:

```
#include<avr/io.h>

#define F_CPU 8000000

#include<util/delay.h>

#include<lcdavr.h>

#include<serial.h>



unsigned char v;

int main()

{

DDRB=0XFF;

lcd_init();
```

```
usart_init();
DDRA=0XFF;

while(1)

{
usart_string("WELCOME TO SERIAL");
for(int x=0XC0;x<=0XCF;x++)
{
v=usart_rec();
lcd_command(x);
lcd_data(v);
lcd_command(0x06);
_delay_ms(10);

if(v=='1')
{
lcd_clear();
lcd_command(0x80);
lcd_string("LAMP 1 GLOW ");

PORTA|=(1<<PA0);
}

if(v=='6')
{
lcd_clear();
lcd_command(0x80);
lcd_string("LAMP 1 OFF  ");
```

```
if(PINA &=0x01)

{


PORTA |=(0<<PA0);}

}


if(v=='2')


{

lcd_clear();

lcd_command(0x80);

lcd_string("LAMP 2 GLOW ");


PORTA|=(1<<PA1);

}


if(v=='7')


{

lcd_clear();

lcd_command(0x80);

lcd_string("LAMP 2 OFF ");

if(PINA &=0x02)

{

PORTA|=(0<<PA1);}

}


if(v=='3')

{
```

```
lcd_clear();

lcd_command(0x80);

lcd_string("LAMP 3 GLOW");


PORTA|=(1<<PA2);

}


if(v=='8')

{

lcd_clear();

lcd_command(0x80);

lcd_string("LAMP 3 OFF");

if(PINA &=0x04)

{

PORTA|=(0<<PA2);}

}


if(v=='4')

{

lcd_clear();

lcd_command(0x80);

lcd_string(" LAMP 4 GLOW ");


PORTA|=(1<<PA3);

}


if(v=='9')

{

lcd_clear();
```

```
lcd_command(0x80);

lcd_string(" LAMP 4 OFF  ");

if(PINA &=0x08)

{

PORTA |=(0<<PA3);}}}}
```

WORKING:-

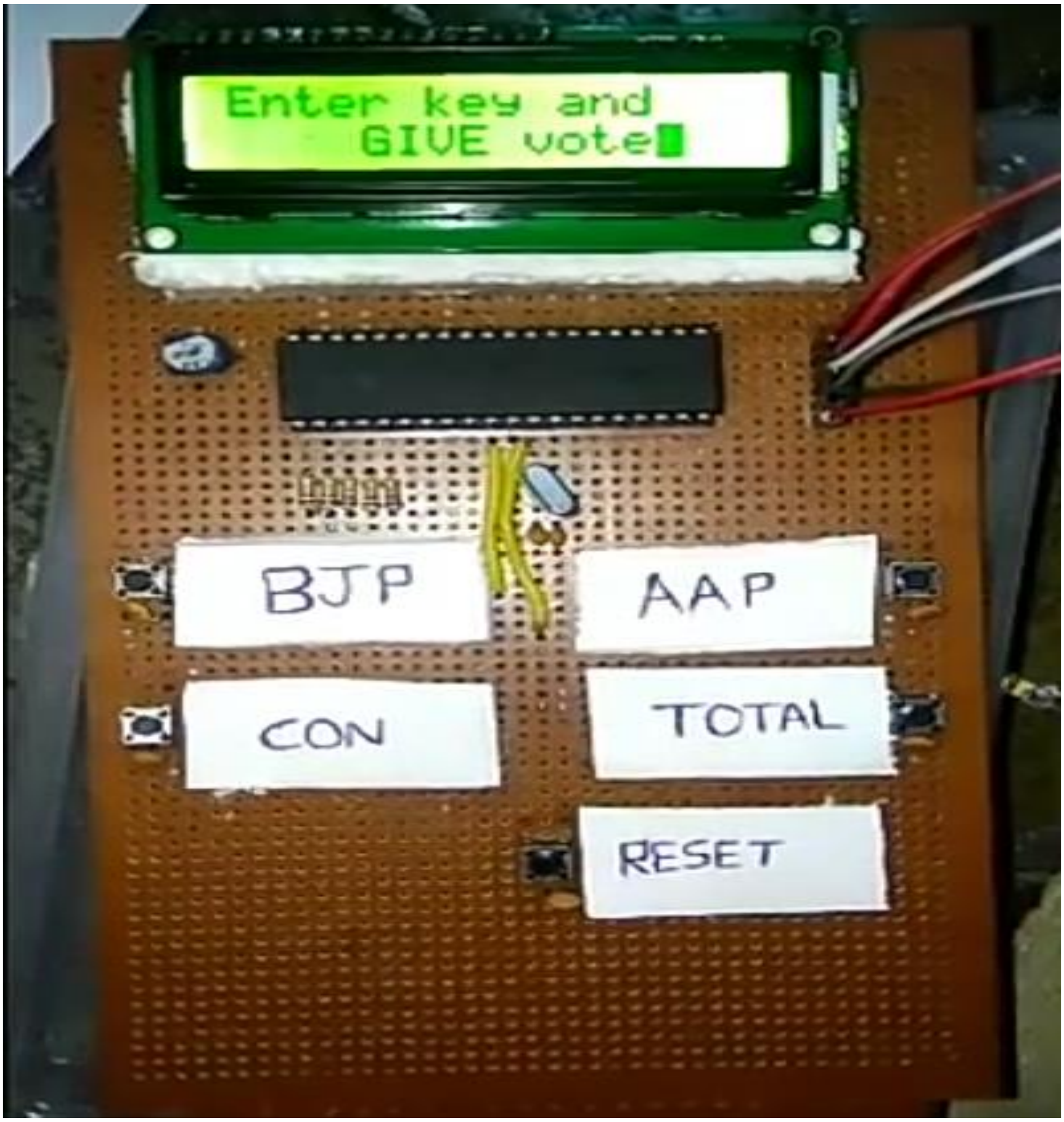
In this project i used HC-0 bluetooth Module tonnect it through an android app and then controlling it's output

To control the microcontroller AtMEGA-16 which communicates with the bluetooth module via serial communication protocol using the transmitter and receiver pin in the microcontroller and after that this data is then fed to the motor driver correspondingly.

AVR BASED

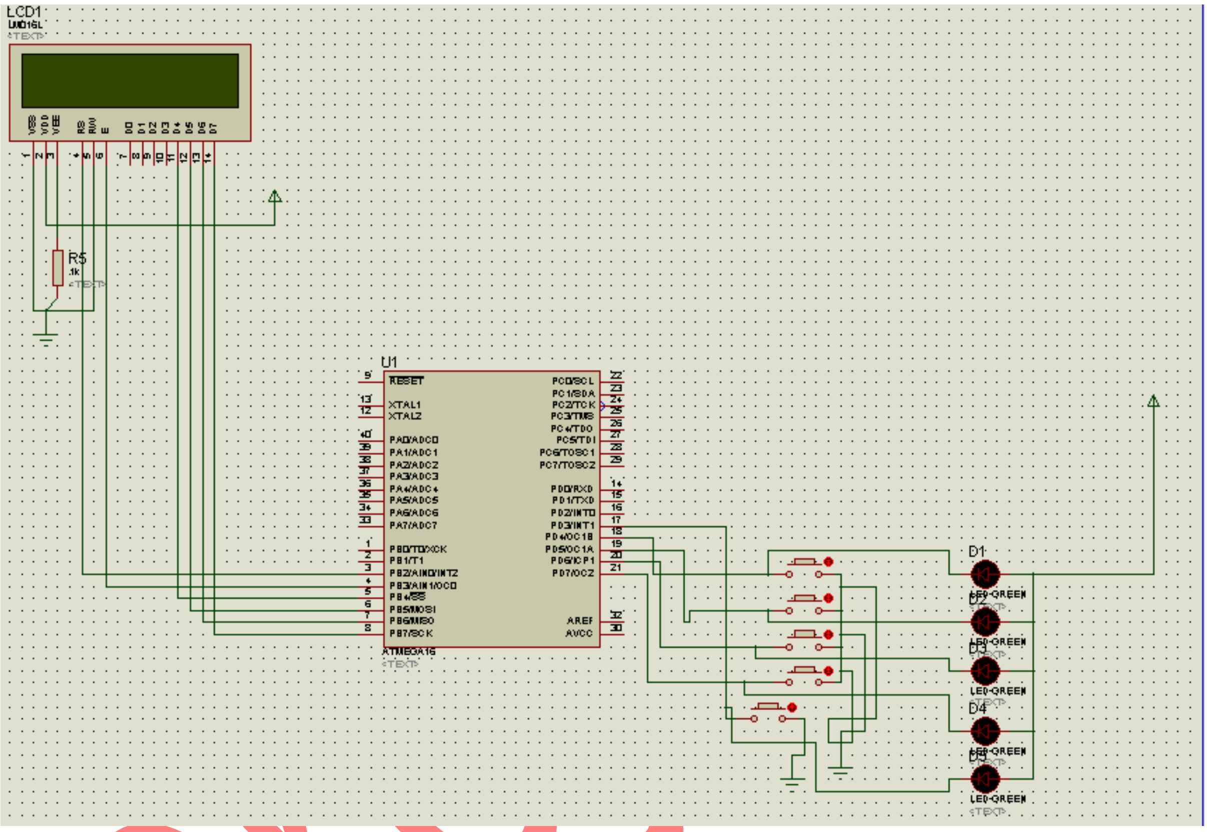
EVM MACHINE

Final output:



Material used:- 16x2 LCD ,Atmega 16 ,led , switches ,resistors 10k , 330ohm .

Schematic diagram:



Code:-

```
#include<avr/io.h>

#define F_CPU 8000000

#include<util/delay.h>

#include<lcdavr.h>

int a,b,c;//abc is generated internally so use lcd_number() to display them


int main()

{  int a=0;b=0;c=0;

    DDRB=0xff;//portb as output
```

```
                                DDRD=0x00;

                                //portd as input

                                lcd_init();

                                while(1)

                                { PORTD=0b11111000;


                                  lcd_command(0x80);

                                  lcd_string("press any key");


if((PIND &0b11111000)==0b11101000)

                                  {

                                      // a=a+1;// a=BJP votes


                                                     lcd_clear();


                                                     lcd_command(0x80);

                                                     lcd_string("  thank you");

                                                     _delay_ms(20);


                                                     lcd_clear();

                                                       a=a+1;

                                                     }

                                      if((PIND &0b11111000)==0b11011000)

                                            {     //b=b+1;// b=CONG votes


                                                               lcd_clear();

                                                               lcd_command(0x80);

                                                               lcd_string("  thank you");

                                                     _delay_ms(20);
```

```
                                             lcd_clear();
                                        b=b+1;// b=party2 votes
                                                             }
if((PIND &0b11111000)==0b10111000)
                                    {
                                      //  c=c+1;//AAP votes

                                                    lcd_clear();
                                                    lcd_command(0x80);
                                                    lcd_string("  thank you");
                                         _delay_ms(20);

                                             lcd_clear();
                                                    c=c+1;//party3 votes
  }
if((PIND &0b11111000)==0b01111000)
                                    {  //result



                                      if((a>>b) &&(a>>c))
                                            {     lcd_clear();
                                                  lcd_command(0x80);
                                                              lcd_string("  BJP wins ");
                                                              _delay_ms(20);   }
  if((b>>a) &&(b>>c))
   {     lcd_clear();
```

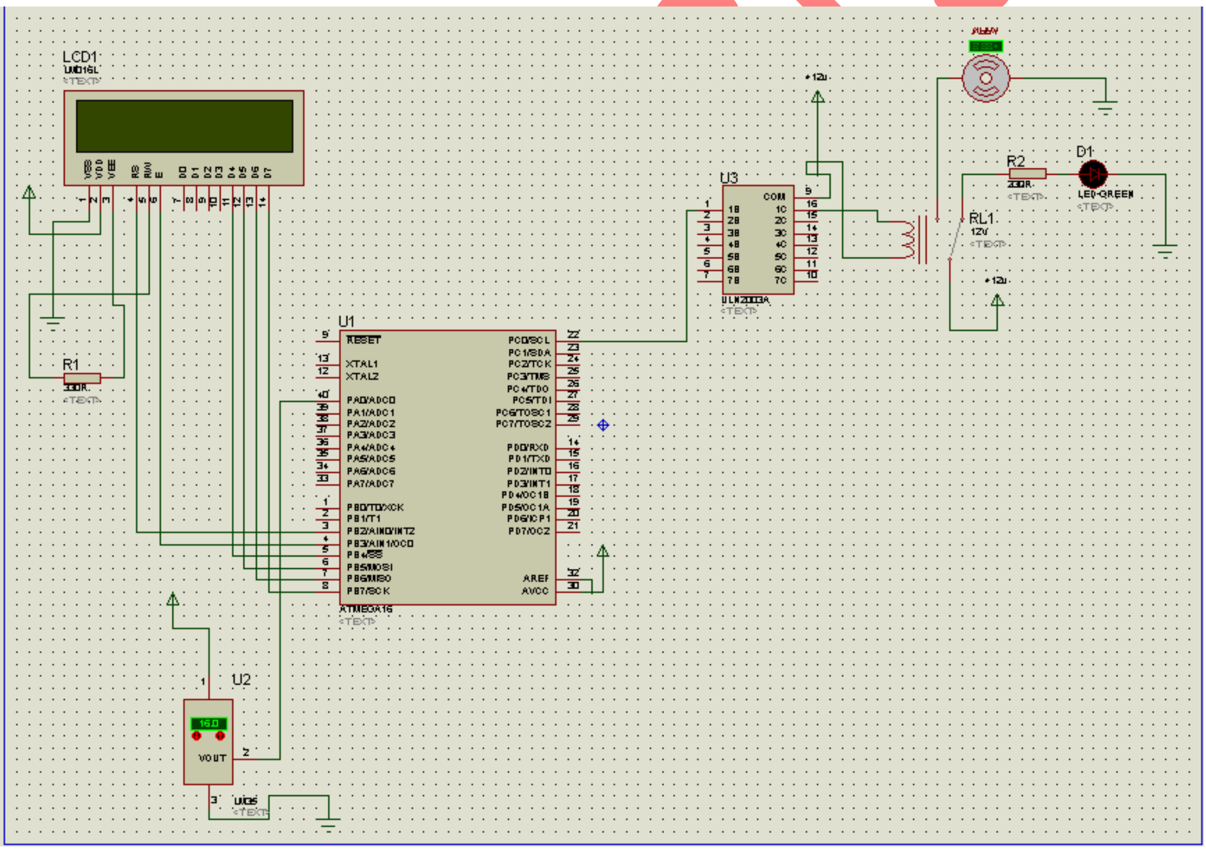
```
                                        lcd_command(0x80);

                                               lcd_string("  CONG wins ");

                                               _delay_ms(20);   }

          if((c>>a) &&(c>>b))

             {     lcd_clear();

                                        lcd_command(0x80);

                                               lcd_string("  party 3 wins ");

                                               _delay_ms(20);   }



                                          if((a==b))

                                               {

                                                lcd_clear();

                                                lcd_command(0x80);

                                                lcd_string("part1=party2 votes");

                                                _delay_ms(10);

                                                lcd_clear();}


                                                if((a==c))

                                                {

                                                   lcd_clear();

                                                lcd_command(0x80);

                                                lcd_string("part1=party3 votes");

                                                _delay_ms(10);

                                                lcd_clear();}

            if((b==c))

                                               {

                                                  lcd_clear();

                                                lcd_command(0x80);
                                               lcd_string("part2=party3 votes");
                                               _delay_ms(10);
                                               lcd_clear();
                }
                                              }
if((PIND & 0b11111000)==0b11110000)
                      {
                        //reset
                             a=0;
                             b=0;
                             c=0;
                             }
   else { lcd_command(0x80);
                      lcd_string("press any key");
}}}
```

Working: In this project every parties vote input is taken with the help of switches and every parties vote is been stored and counted with a separate variable for every party and the final result is shown on Lcd an reset button is also used to restart the whole process.

AVR BASED TEMPERATURE CONTROLLED FAN

MATERIAL USED:-ATMEGA 16, an AC FAN ,RELAY ,ULN2003A ,16X2 LCD ,LED,Resistor, TEMPERATURE SENSOR.

SCHEMATIC DIAGRAM:-



CODE:-

```
#include<avr/io.h>

#define F_CPU 8000000

#include<util/delay.h>

#include<lcdavr.h>

#include<stdio.h>


void adc_init()

{

                    ADMUX = 0B01000000;//for vcc +5v aNd vref =5v| ADC0 is selected for
displaying

                    ADCSRA = ((1<<ADEN) | (1<<ADPS2)| (1<<ADPS1)| (1<<ADPS0));

                    }

unsigned int adc_read(unsigned char channel)

{



                    ADCSRA  |= (1<<ADSC);//for starting the conversion

                    while(!(ADCSRA & (1<<ADIF)));//for progress report //for converting the
data


                    ADCSRA  |= (1<<ADIF);//for updating the data the data and previous task is
done

                    return ADC;

}


int main()

{



                    unsigned char i[5];
```

```
                unsigned int value;

                DDRB = 0XFF;

                DDRC=0xff;

                lcd_init();

                adc_init();

while(1)

{

  value=adc_read(0);//0th pin is used 0f adc channel

                lcd_clear();

                lcd_command(0x80);

                sprintf(i,"%d",value);//convrets adc value to i's data type

                lcd_string("adc value:     ");

                lcd_command(0x8b);

                lcd_string(i);

                lcd_command(0x8e);

                lcd_string("mv");

                _delay_ms(100);

                if((value>=0)&&(value<=56))

                {

                  PORTC=(1<<PC1);

                  }


   else if((value>=199)&&(value<=308))

                 {

                     PORTC=(1<<PC0);

                          }


                          else

                          {
```

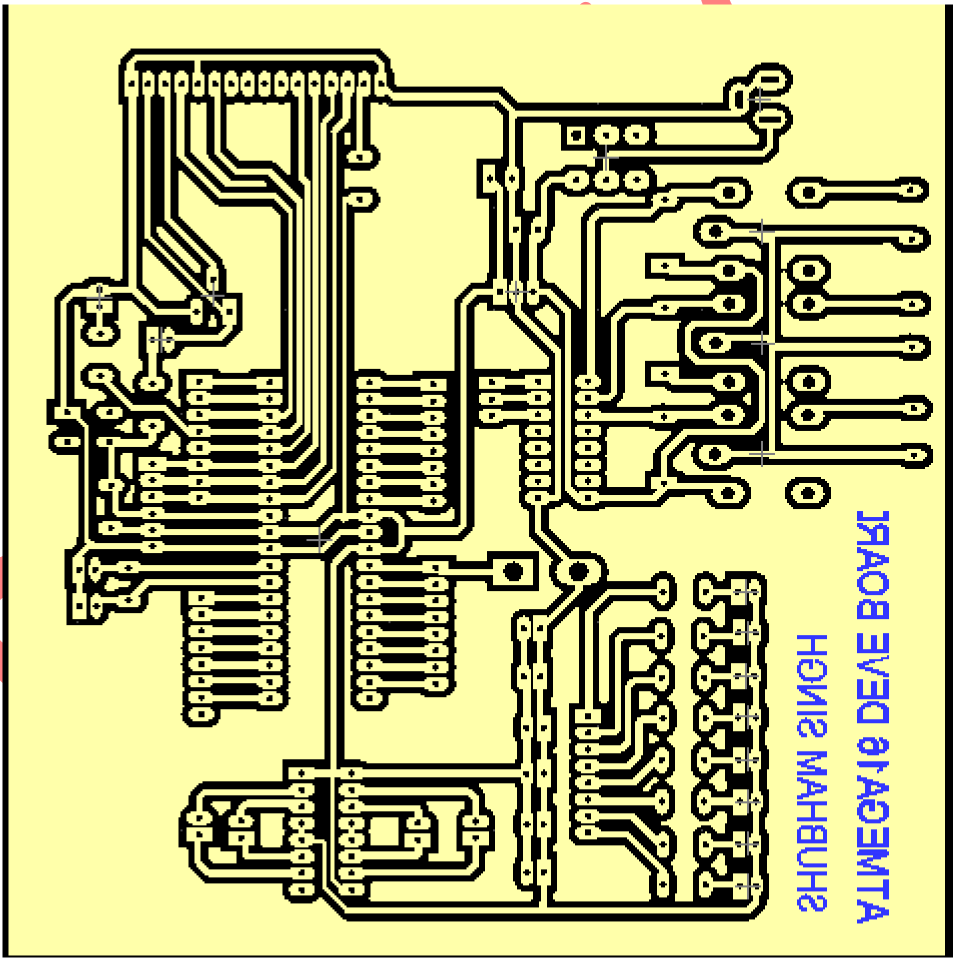
```
                                        PORTC=0x00;
                                                }

}
                        return 0;
}
```

WORKING :- In this project an temperature sensor is placed where the temperature has to be monitored than it is connected to the atmega-16 via it’s analog pins and it constantly measuring its analog feed back from the sensor and converting it’s values to the corresponding digital values and according to our need the speed or even the moment of the fan can be controlled within a certain range of the digital values from the temperature sensor . Fan is connected to the microcontroller with a relay which is further connected with the uln2003a relay driver module to protect the microcontroller from any back current from the relay as it is going to be used to drive a 220v appliance which is far mor greater than the operating voltage of the atmega-16 which can cause some serious damage to it.

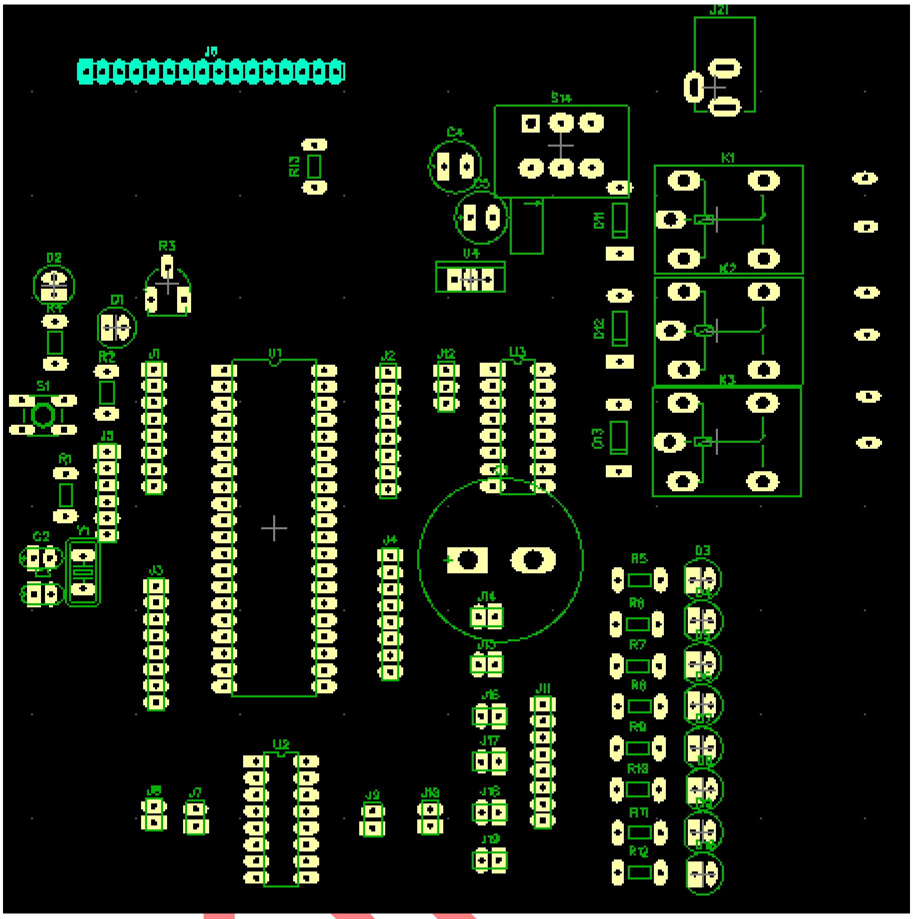
AVR DEVELOPMENT BOARD USING DIPTRACE

FINAL OUTPUT

BOTTOM VIEW :-

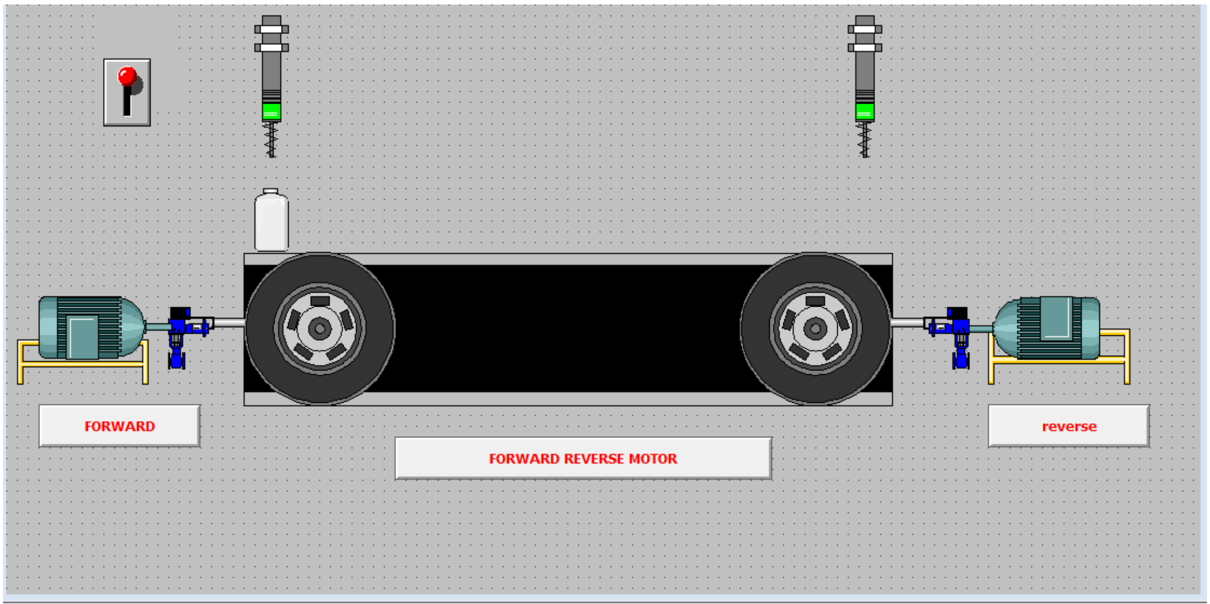


FRONT VIEW:-


J21
S14
C4
C5
K1
K2
K3
R13
D11
D12
D13
U4
R3
D2
R4
D1
R2
S1
J1
U1
J2
J12
U3
R1
J3
C2
Y1
J5
J4
J14
J15
J16
J17
J18
J19
J11
U2
J6
J7
J9
J10
R5
R6
R7
R8
R9
R10
R11
R12
D3

SCADA PROJECTS

MOTOR ROTATION :-



CODES:-

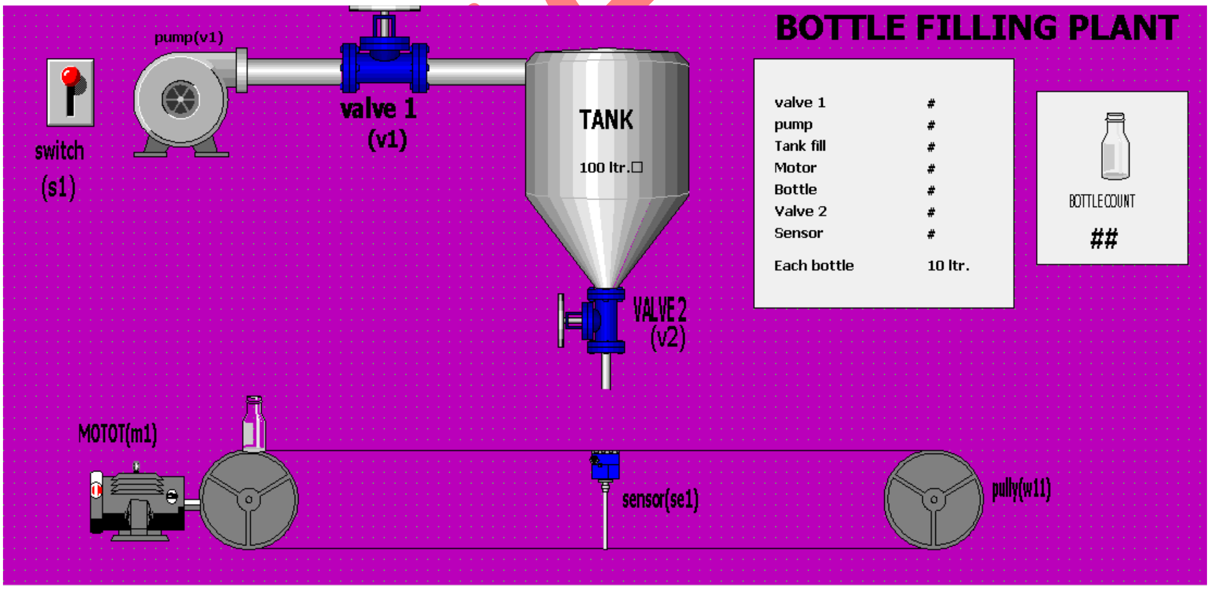
On show :-

```
sw=0;

s1=0;

s2=0;

m1=0;

m2=0;
```

WHILE SHOW:-

```
IF b1==0 THEN x=0;ENDIF;

IF b1==560 THEN x=1;ENDIF;


IF b1==50 THEN y=0;ENDIF;

IF b1==510 THEN y=1;ENDIF;
IF sw==1  AND x==0  THEN  m1=1;ELSE m1=0;ENDIF;

IF sw==1  AND x==0  THEN  b1=b1+5;ENDIF;


IF sw==1  AND x==1 THEN  m2=1;ELSE m2=0;ENDIF;

IF sw==1  AND x==1  THEN  b1=b1 - 5;ENDIF;


IF sw==1  AND y==0  AND b1<50 THEN  s1=1;ELSE s1=0;ENDIF;

IF sw==1  AND y==0  AND  b1<50 THEN s1=1; ENDIF;


IF sw==1  AND y==1  AND b1>510 THEN  s2=1;ELSE s2=0;ENDIF;

IF sw==1  AND y==1  AND  b1>510 THEN s2=1; ENDIF;
```

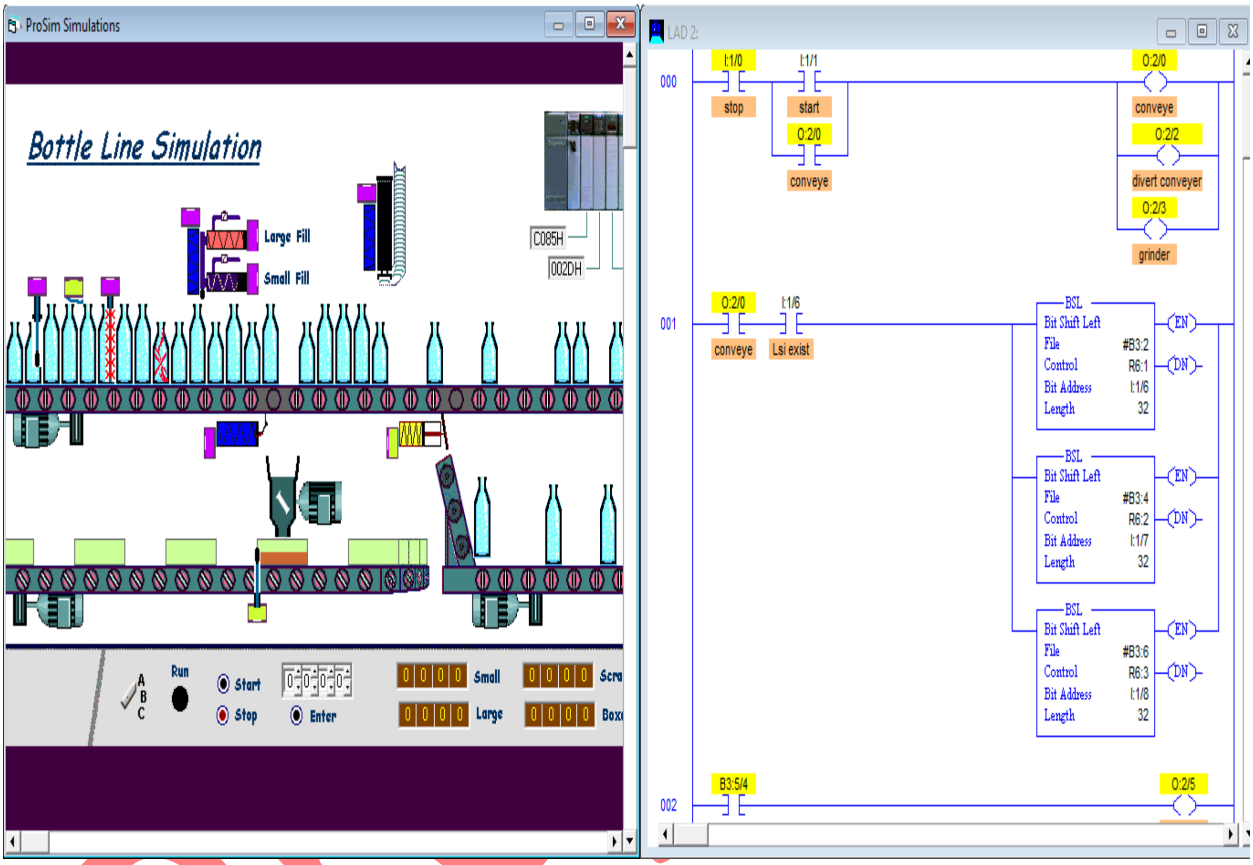
BOTTLE FILLING PLANT :-



valve 1	#
pump	#
Tank fill	#
Motor	#
Bottle	#
Valve 2	#
Sensor	#
Each bottle	10 ltr.

LOGIX PRO PROJECT :-

BOTTLE LINE SIMULATION:-



In this simulation the bigger bottles will be separated and collected through the bottom conveyer and the broken bottle will be crushed using the grinder.

MATLAB BASED CAR NUMBER PLATE DETECTION

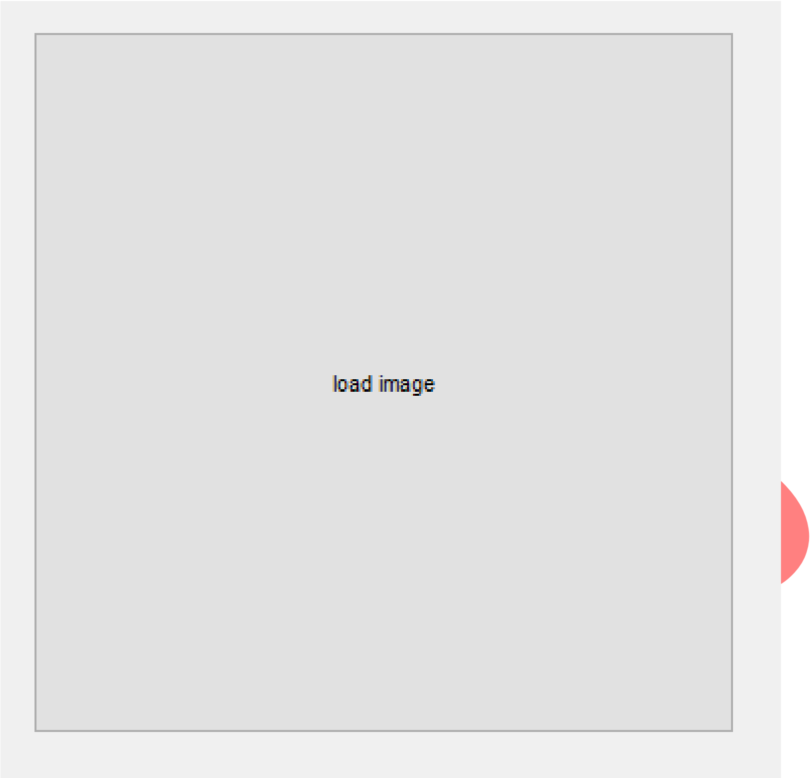
Method 1:-

CODE:

```
function varargout = car_plate_detection(varargin)
% CAR_PLATE_DETECTION MATLAB code for car_plate_detection.fig
%      CAR_PLATE_DETECTION, by itself, creates a new
CAR_PLATE_DETECTION or raises the existing
%      singleton*.
%
%      H = CAR_PLATE_DETECTION returns the handle to a new
CAR_PLATE_DETECTION or the handle to
%      the existing singleton*.
%
%
CAR_PLATE_DETECTION('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in CAR_PLATE_DETECTION.M with
the given input arguments.
%
%      CAR_PLATE_DETECTION('Property','Value',...) creates a
new CAR_PLATE_DETECTION or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before
car_plate_detection_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to
car_plate_detection_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help
car_plate_detection

% Last Modified by GUIDE v2.5 07-Jul-2019 17:48:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@car_plate_detection_OpeningFcn, ...
                   'gui_OutputFcn',
@car_plate_detection_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before car_plate_detection is made
visible.
function car_plate_detection_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to car_plate_detection
(see VARARGIN)

% Choose default command line output for car_plate_detection
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes car_plate_detection wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command
line.
function varargout = car_plate_detection_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in car_image.
function car_image_Callback(hObject, eventdata, handles)
% hObject    handle to car_image (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
close all;
clear all;
[file,path]=uigetfile('*.*');

i=fullfile(path,file);
im=imread(i);
im = imresize(im, [480 NaN]);
imgray = rgb2gray(im);
imbin = imbinarize(imgray);
im = edge(imgray, 'sobel');


im = imdilate(im, strel('diamond', 2));
im = imfill(im, 'holes');
im = imerode(im, strel('diamond', 10));


Iprops=regionprops(im,'BoundingBox','Area', 'Image');
area = Iprops.Area;
count = numel(Iprops);
maxa= area;
boundingBox = Iprops.BoundingBox;
for i=1:count
   if maxa<Iprops(i).Area
       maxa=Iprops(i).Area;
       boundingBox=Iprops(i).BoundingBox;
   end
```

```
end

%all above step are to find location of number plate

im = imcrop(imbin, boundingBox);

%resize number plate to 240 NaN
im = imresize(im, [240 NaN]);

%clear dust
im = imopen(im, strel('rectangle', [4 4]));

%remove some object if it width is too long or too small than
500
im = bwareaopen(~im, 500);
%%%get width
 [h, w] = size(im);



imshow(im);


%read letter
Iprops=regionprops(im,'BoundingBox','Area', 'Image');
count = numel(Iprops);

noPlate=[]; % Initializing the variable of number plate
string.

for i=1:count
   ow = length(Iprops(i).Image(1,:));
   oh = length(Iprops(i).Image(:,1));
   if ow<(h/2) & oh>(h/3)
       letter=readLetter(Iprops(i).Image); % Reading the
letter corresponding the binary image 'N'.
       figure;imshow(Iprops(i).Image);
       noPlate=[noPlate letter]; % Appending every subsequent
character in noPlate variable.

   end

end
disp('no.plate is:');disp(noPlate);
f = msgbox({'no. plate of choosed car is',noPlate});
```

FIGURE:



Templates:-

```
%CREATE TEMPLATES
%Letter
A=imread('char/A.bmp');B=imread('char/B.bmp');
C=imread('char/C.bmp');D=imread('char/D.bmp');
E=imread('char/E.bmp');F=imread('char/F.bmp');
G=imread('char/G.bmp');H=imread('char/H.bmp');
I=imread('char/I.bmp');J=imread('char/J.bmp');
K=imread('char/K.bmp');L=imread('char/L.bmp');
M=imread('char/M.bmp');N=imread('char/N.bmp');
O=imread('char/O.bmp');P=imread('char/P.bmp');
Q=imread('char/Q.bmp');R=imread('char/R.bmp');
S=imread('char/S.bmp');T=imread('char/T.bmp');
U=imread('char/U.bmp');V=imread('char/V.bmp');
W=imread('char/W.bmp');X=imread('char/X.bmp');
Y=imread('char/Y.bmp');Z=imread('char/Z.bmp');
Afill=imread('char/fillA.bmp');
Bfill=imread('char/fillB.bmp');
Dfill=imread('char/fillD.bmp');
Ofill=imread('char/fillO.bmp');
Pfill=imread('char/fillP.bmp');
Qfill=imread('char/fillQ.bmp');
Rfill=imread('char/fillR.bmp');
```

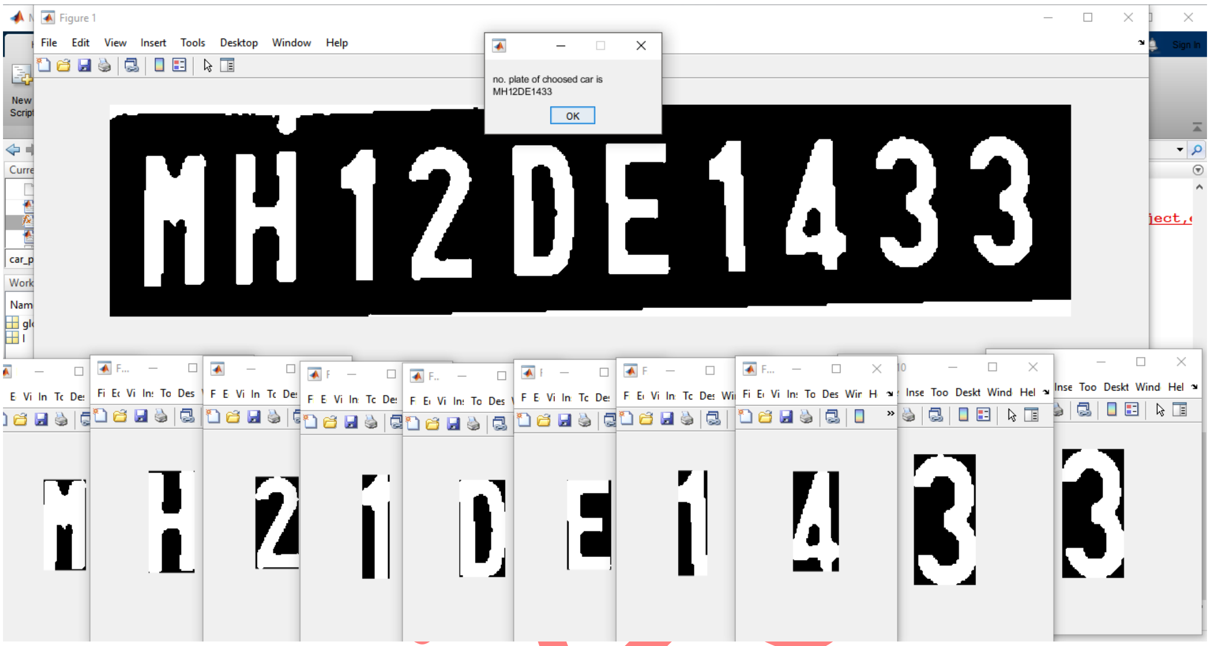
Letter reading:

```
function letter=readLetter(snap)
%READLETTER reads the character fromthe character's binary
image.
%   LETTER=READLETTER(SNAP) outputs the character in class
'char' from the
%   input binary image SNAP.

load NewTemplates % Loads the templates of characters in the
memory.
snap=imresize(snap,[42 24]); % Resize the input image so it
can be compared with the template's images.
comp=[ ];
for n=1:length(NewTemplates)
    sem=corr2(NewTemplates{1,n},snap); % Correlation the input
image with every image in the template for best matching.
    comp=[comp sem]; % Record the value of correlation for
each template's character.
    %display(sem);

end
vd=find(comp==max(comp)); % Find the index which correspond to
the highest matched character.
%display(max(comp));
%*_*_*_*_*_*_*_*_*_*_*_*_*_
% Accodrding to the index assign to 'letter'.
% Alphabets listings.
if vd==1 || vd==2
    letter='A';
elseif vd==3 || vd==4
    letter='B';
elseif vd==5
    letter='C';
elseif vd==6 || vd==7
    letter='D';
elseif vd==8
    letter='E';
elseif vd==9
    letter='F';
elseif vd==10
    letter='G';
elseif vd==11
    letter='H';
elseif vd==12
    letter='I';
elseif vd==13
    letter='J';
elseif vd==14
    letter='K';
```

```
elseif vd==15
    letter='L';
elseif vd==16
    letter='M';
elseif vd==17
    letter='N';
elseif vd==18 || vd==19
    letter='O';
elseif vd==20 || vd==21
    letter='P';
elseif vd==22 || vd==23
    letter='Q';
elseif vd==24 || vd==25
    letter='R';
elseif vd==26
    letter='S';
elseif vd==27
    letter='T';
elseif vd==28
    letter='U';
elseif vd==29
    letter='V';
elseif vd==30
    letter='W';
elseif vd==31
    letter='X';
elseif vd==32
    letter='Y';
elseif vd==33
    letter='Z';
    %*_*_*_*_*
% Numerals listings.
elseif vd==34
    letter='1';
elseif vd==35
    letter='2';
elseif vd==36
    letter='3';
elseif vd==37 || vd==38
    letter='4';
elseif vd==39
    letter='5';
elseif vd==40 || vd==41 || vd==42
    letter='6';
elseif vd==43
    letter='7';
elseif vd==44 || vd==45
    letter='8';
elseif vd==46 || vd==47 || vd==48
    letter='9';
else
```

```
    letter='0';
end
end
```

OUTPUT:



no. plate of choosed car is
MH12DE1433

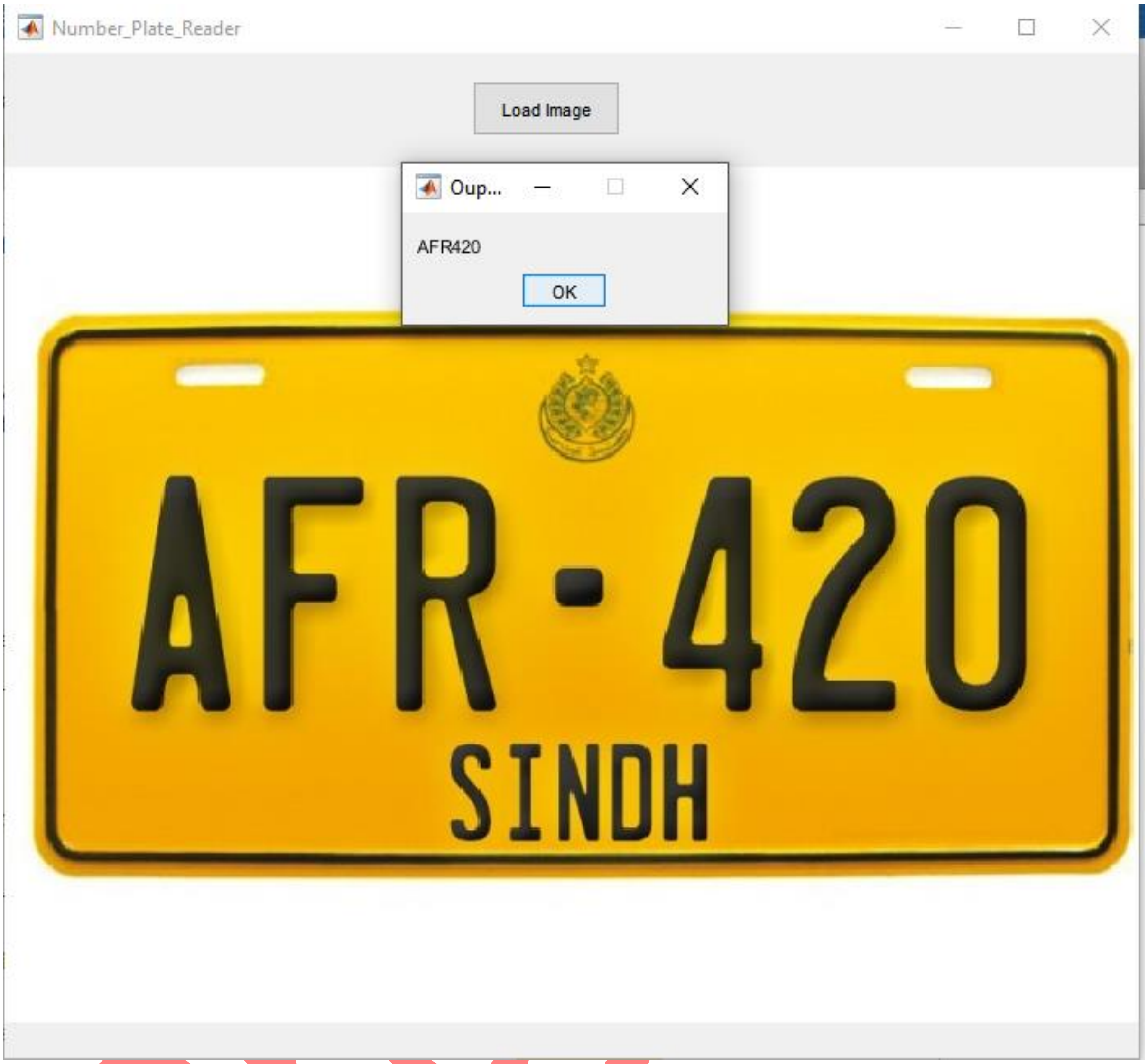
Method 2:

CODE :

```
function varargout = Number_Plate_Reader(varargin)
clc;


gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Number_Plate_Reader_OpeningFcn, ...
                   'gui_OutputFcn',
@Number_Plate_Reader_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Number_Plate_Reader is made
visible.
function Number_Plate_Reader_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to Number_Plate_Reader
(see VARARGIN)

% Choose default command line output for Number_Plate_Reader
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

initialize_gui(hObject, handles, false);

% UIWAIT makes Number_Plate_Reader wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = Number_Plate_Reader_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all
properties.

% --- Executes on button press in calculate.
% --- Executes on button press in reset.

% --- Executes when selected object changed in unitgroup.
function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is
false, it means
% we are we are just re-initializing a GUI by calling it from
the cmd line
% while it is up. So, bail out as we dont want to reset the
data.
if isfield(handles, 'metricdata') && ~isreset
    return;
end

% Update handles structure
guidata(handles.figure1, handles);


% --- Executes on button press in pushbutton10.
function[text1]= pushbutton10_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
    [baseFileName,folder]=uigetfile('*.*','Specify an image
file','on');

    fullimageFileName=fullfile(folder,baseFileName);

    axes1=imread(fullimageFileName);

    axes(handles.axes1);

    image(axes1)
    prompt={'Enter the number of charcters to read:'};
    dlg_title = 'Number of Characters';
    num_lines = 1;
    def = {'0'};
    answer = inputdlg(prompt,dlg_title,num_lines,def);
    no = str2num(answer{1});

    text1 = final(fullimageFileName,no);
    msgbox(text1,'Ouput');
```

OUTPUT:



AFR420

WORKING:- In this project first I have created a data set of all the alphabets and numbers in separate files fo comparing them to the car images and then before starting anything the input RGB image is converted in binary image which will only contain black and white dots and after that these binary images is than compared with the data set just created and according to that the result wil be shown respectively.

MATLB BASED REAL TIME FACE DETECTION

Code:

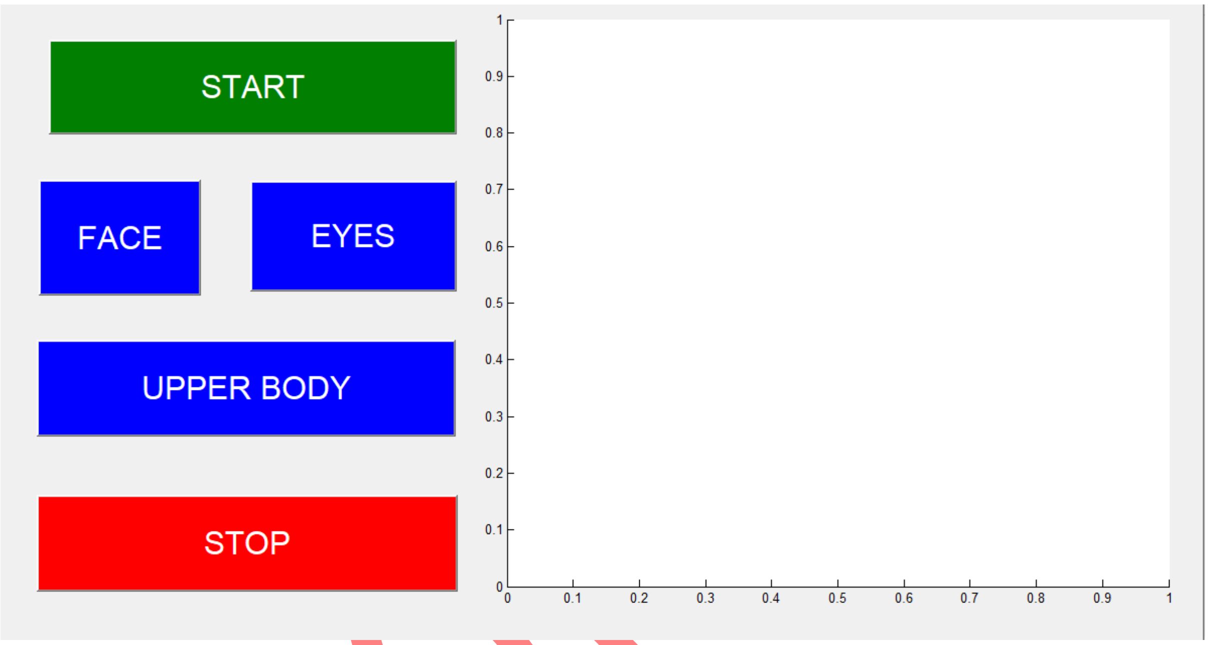
```
function varargout = testing(varargin)
% TESTING MATLAB code for testing.fig
%      TESTING, by itself, creates a new TESTING or raises the
existing
%      singleton*.
%
%      H = TESTING returns the handle to a new TESTING or the
handle to
%      the existing singleton*.
%
%      TESTING('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in TESTING.M with the given
input arguments.
%
%      TESTING('Property','Value',...) creates a new TESTING
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before testing_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to testing_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help testing

% Last Modified by GUIDE v2.5 20-Aug-2013 16:34:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @testing_OpeningFcn, ...
                   'gui_OutputFcn',  @testing_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before testing is made visible.
function testing_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to testing (see VARARGIN)

% Choose default command line output for testing
handles.output = hObject;
axes(handles.axes1);
imshow('blank.jpg');
axis off;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes testing wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = testing_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
handles.vid = videoinput('winvideo' , 1, 'YUY2_640X480');
%preview(handles.vid);
guidata(hObject, handles);

% --- Executes on button press in face.
function face_Callback(hObject, eventdata, handles)
% hObject    handle to face (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
%handles.vid = videoinput('winvideo' , 1, 'YUY2_640X480');
triggerconfig(handles.vid ,'manual');
set(handles.vid, 'TriggerRepeat',inf);
set(handles.vid, 'FramesPerTrigger',1);
handles.vid.ReturnedColorspace = 'rgb';
 handles.vid.Timeout = 5;
start(handles.vid);
while(1)

facedetector = vision.CascadeObjectDetector;
trigger(handles.vid);
handles.im = getdata(handles.vid, 1);
bbox = step(facedetector, handles.im);
hello =
insertObjectAnnotation(handles.im,'rectangle',bbox,'Face');
imshow(hello);
end
guidata(hObject, handles);


% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
handles.output = hObject;
```

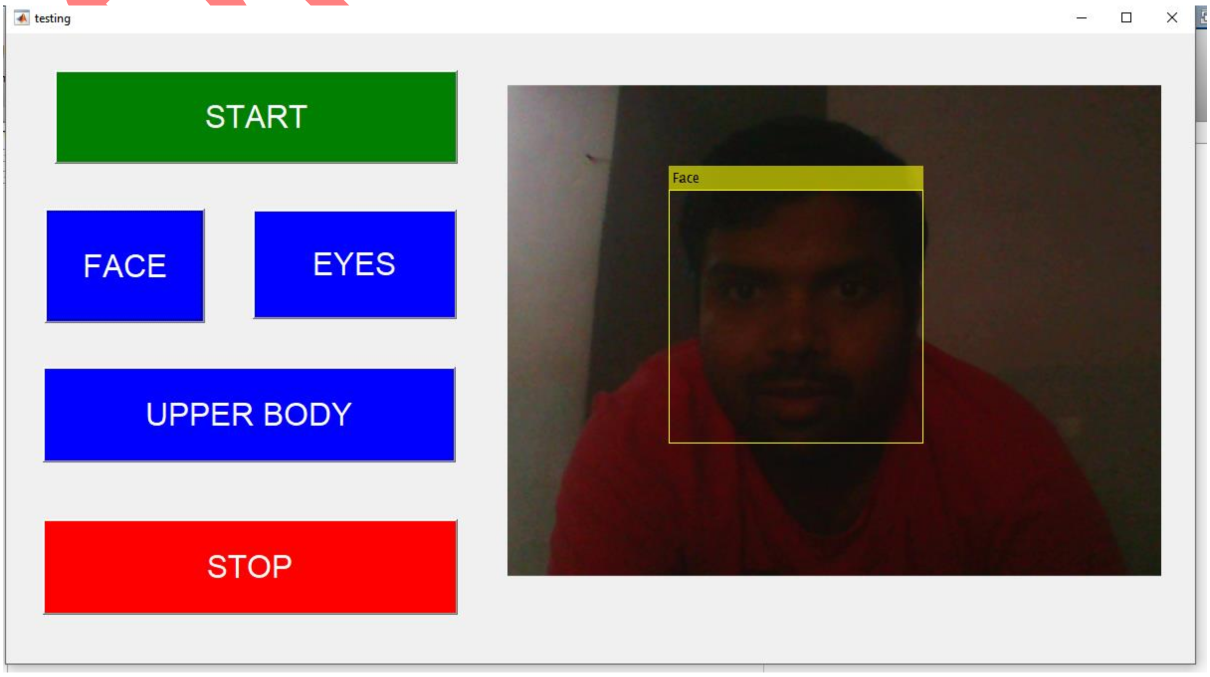
```
stop(handles.vid),clear handles.vid %, ,delete(handles.vid)
guidata(hObject, handles);


% --- Executes on button press in eyes.
function eyes_Callback(hObject, eventdata, handles)
% hObject    handle to eyes (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
triggerconfig(handles.vid ,'manual');
set(handles.vid, 'TriggerRepeat',inf);
set(handles.vid, 'FramesPerTrigger',1);
handles.vid.ReturnedColorspace = 'rgb';
 handles.vid.Timeout = 2;
start(handles.vid);
while(1)
bodyDetector = vision.CascadeObjectDetector('EyePairBig');
bodyDetector.MinSize = [11 45];
%bodyDetector.ScaleFactor = 1.05;
trigger(handles.vid);
handles.im = getdata(handles.vid, 1);
bbox = step(bodyDetector, handles.im);
hello =
insertObjectAnnotation(handles.im,'rectangle',bbox,'EYE');
imshow(hello);
end
guidata(hObject, handles);


% --- Executes on button press in upperbody.
function upperbody_Callback(hObject, eventdata, handles)
% hObject    handle to upperbody (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
triggerconfig(handles.vid ,'manual');
set(handles.vid, 'TriggerRepeat',inf);
set(handles.vid, 'FramesPerTrigger',1);
handles.vid.ReturnedColorspace = 'rgb';
 handles.vid.Timeout = 5;
start(handles.vid);
while(1)
bodyDetector = vision.CascadeObjectDetector('UpperBody');
bodyDetector.MinSize = [60 60];
bodyDetector.ScaleFactor = 1.05;
trigger(handles.vid);
handles.im = getdata(handles.vid, 1);
bbox = step(bodyDetector, handles.im);
```

```
hello =
insertObjectAnnotation(handles.im,'rectangle',bbox,'UpperBody'
);
imshow(hello);
end
guidata(hObject, handles);
```

FIGURE:-



OUTPUT:



BLOOD CANCER DETECTION USING MATLAB

CODE:-

```
function varargout = BloodCancer(varargin)
% BLOODCANCER MATLAB code for BloodCancer.fig
%       BLOODCANCER, by itself, creates a new BLOODCANCER or
raises the existing
%       singleton*.
%
%       H = BLOODCANCER returns the handle to a new BLOODCANCER
or the handle to
%       the existing singleton*.
%
%       BLOODCANCER('CALLBACK',hObject,eventData,handles,...)
calls the local
%       function named CALLBACK in BLOODCANCER.M with the given
input arguments.
%
%       BLOODCANCER('Property','Value',...) creates a new
BLOODCANCER or raises the
%       existing singleton*.  Starting from the left, property
value pairs are
%       applied to the GUI before BloodCancer_OpeningFcn gets
called.  An
%       unrecognized property name or invalid value makes
property application
%       stop.  All inputs are passed to BloodCancer_OpeningFcn
via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
BloodCancer

% Last Modified by GUIDE v2.5 06-Dec-2018 10:48:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BloodCancer_OpeningFcn,
...
                   'gui_OutputFcn',  @BloodCancer_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before BloodCancer is made visible.
function BloodCancer_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% varargin   command line arguments to BloodCancer (see
VARARGIN)

% Choose default command line output for BloodCancer
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes BloodCancer wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = BloodCancer_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
% clc
% clear all
[x,y] = uigetfile('*');
f=strcat(y,x);
I=imread(f);
axes(handles.axes1);
imshow(I);


% --- Executes on button press in Grayscaled.
function Grayscaled_Callback(hObject, eventdata, handles)
% hObject    handle to Grayscaled (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
d=getimage(handles.axes1);
G=rgb2gray(d);
axes(handles.axes2);
imshow(G);

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
d=getimage(handles.axes2);
B=im2bw(d); %convert to BW
[~,threshold]=edge(B,'sobel'); %calculate threshold
fudgeFactor=.5; %to tune threshold value
BWs=edge(B,'sobel',threshold*fudgeFactor);
axes(handles.axes3);
imshow(BWs);
```

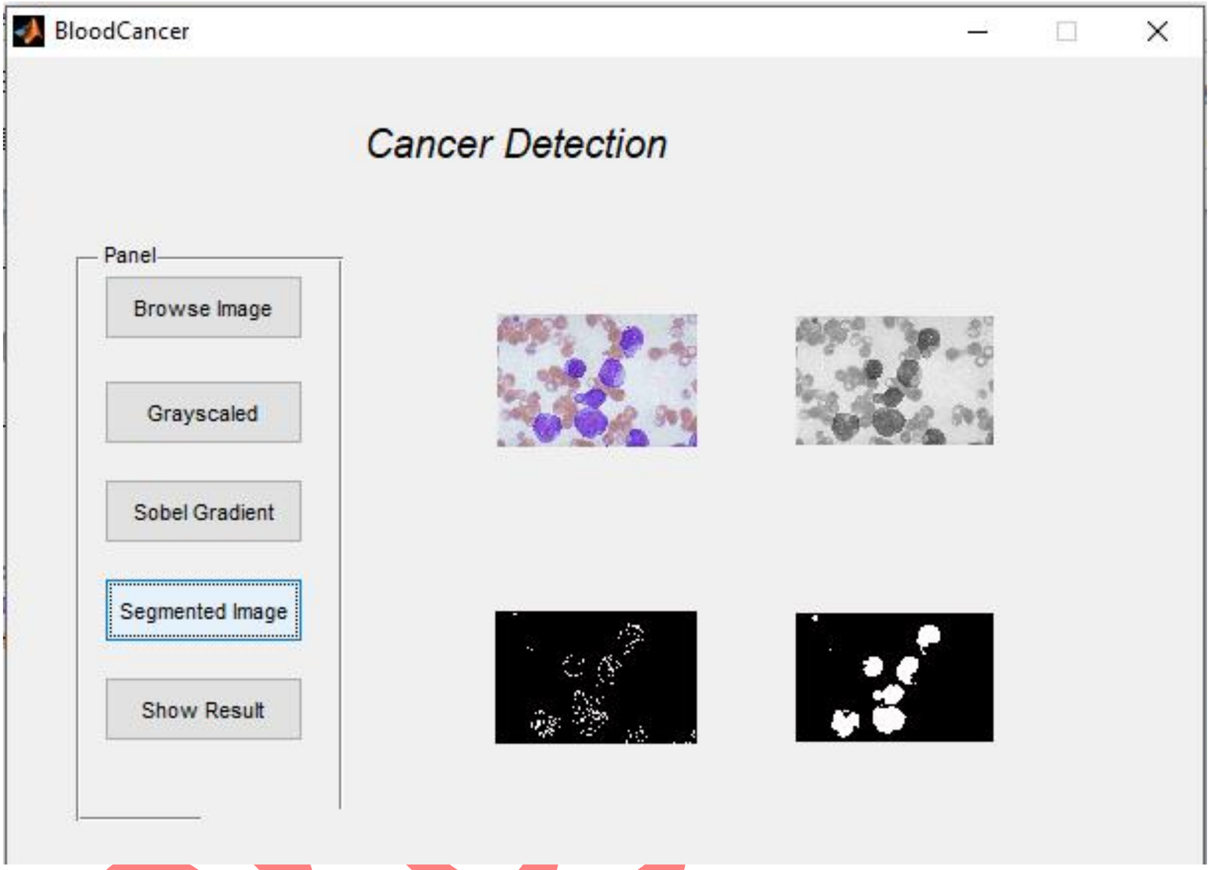
```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
d=getimage(handles.axes3);
se90=strel('line', 3, 90);
se0=strel('line', 3, 0);
BWsdil=imdilate(d, [se90,se0]); %dilate the gradient image
BWdfill=imfill(BWsdil, 'holes'); %fill holes in gradient image
BWnobord=imclearborder(BWdfill, 6); %clear borderline
seD=strel('diamond', 1); %to smoothen image
BWfinal=imerode(BWnobord, seD);
BWfinal=imerode(BWfinal, seD);
axes(handles.axes4);
imshow(BWfinal);


% --- Executes on button press in Result.
function Result_Callback(hObject, eventdata, handles)
% hObject    handle to Result (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
d=getimage(handles.axes4);
[centers, radii] = imfindcircles(d,[35
60],'ObjectPolarity','dark',...
    'Sensitivity',0.9);
imshow(d);
h = viscircles(centers,radii);
cell=length(d);

[l,NUM]=bwlabel(d,4);
cancer=(NUM/cell)*1000;
if  (ge(cancer,51))
    disp('Not recoverable')
    if(ge(cancer, 76))
        disp('4th stage')
        disp('cancer % is')
        disp(cancer)
    else
        disp('3rd stage')
        disp('cancer % is')
        disp(cancer)
    end;
elseif (le(cancer, 50))
    disp('Recoverable')
    if(le(cancer,25))
        disp('1st stage')
        disp('cancer % is')
        disp(cancer)
    else
        disp('2nd stage')
        disp('cancer % is')
        disp(cancer)
    end;
    set(handles.edit6,'string',cancer)
end;


% --- Executes during object creation, after setting all
properties.
function text1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called


% --- Executes during object creation, after setting all
properties.
function text_4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
d = getimage(handles.axes4)
[centers, radii] = imfindcircles(d,[35
60],'ObjectPolarity','dark',...
    'Sensitivity',0.9);
imshow(d);
h = viscircles(centers,radii);
cell=length(d);

[l,NUM]=bwlabel(d,4);
cancer=(NUM/cell)*1000;
if (ge(cancer,51))
    fprintf(handles.text4,'Not Recoverable');
else
    set(handles.text4,'string','recoverable');
end;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as
text
%        str2double(get(hObject,'String')) returns contents of
edit1 as a double


% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as
text
%        str2double(get(hObject,'String')) returns contents of
edit2 as a double


% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

```
% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as
text
%        str2double(get(hObject,'String')) returns contents of
edit3 as a double


% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
```

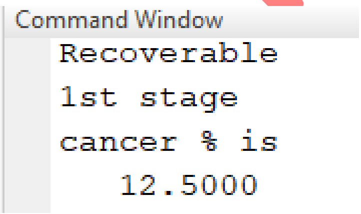
```
% Hints: get(hObject,'String') returns contents of edit4 as
text
%        str2double(get(hObject,'String')) returns contents of
edit4 as a double


% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as
text
%        str2double(get(hObject,'String')) returns contents of
edit6 as a double


% --- Executes during object creation, after setting all
properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Figure:-



OUTPUT:-



```
Recoverable
1st stage
cancer % is
   12.5000
```

STEGANOGRAPHY USING MATLAB

Code:

```
clc;
clear all;
close all;
cover = input('Enter cover image: ', 's');
message = input('Enter message image name: ', 's');
x = imread(cover);          % cover message
y = imread(message);     % message image
n = input('Enter the no of LSB bits to be subsituted- ');
S = uint8(bitor(bitand(x,bitcmp(2^n-1,8)),bitshift(y,n-8)));
%Stego
E = uint8(bitand(255,bitshift(S,8-n))); %Extracted
origImg = double(y);    %message image
distImg = double(E);    %extracted image
[M N] = size(origImg);
distImg1=imresize(distImg,[M N]);
error = origImg - distImg1;
MSE = sum(sum(error .* error)) / (M * N);
if(MSE > 0)
    PSNR = 10*log10(M*N./MSE);
else
    PSNR = 99;
end
disp('PSNR of message image to extracted image is')
disp(abs(PSNR))
disp('MSE is')
disp(abs(MSE))
figure(1),imshow(x);title('1.Cover image')
figure(2),imshow(y);title('2.Message to be hide')
figure(3),imshow((abs(S)),[]);title('3.Stegnographic image')
figure(4),imshow(real(E),[]); title('4.Extracted image')
figure(5),imhist(x); title('Histogram of cover image')
figure(6),imhist(S); title('Histogram of transformed stego
image')
```

OUTPUT:

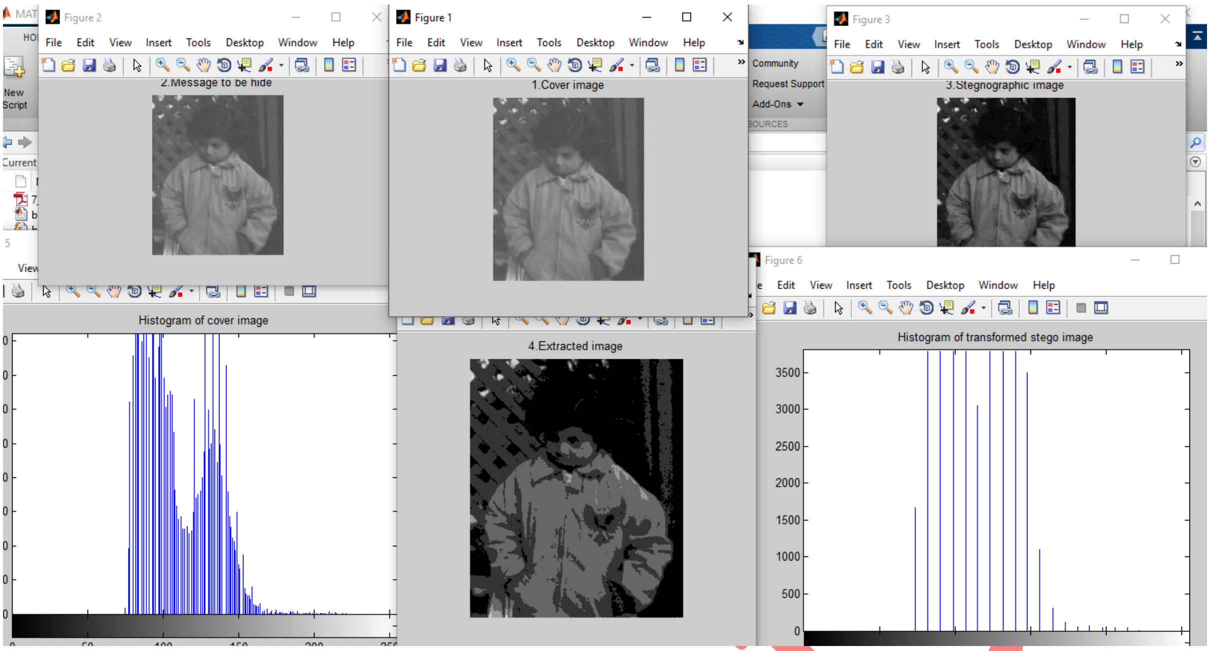


Figure 2
File Edit View Insert Tools Desktop Window Help
2.Message to be hide
Figure 1
File Edit View Insert Tools Desktop Window Help
1.Cover image
Community
Request Support
Add-Ons
Figure 3
File Edit View Insert Tools Desktop Window Help
3.Stegnographic image
Figure 6
Edit View Insert Tools Desktop Window Help
Histogram of cover image
50
100
150
200
4.Extracted image
Histogram of transformed stego image
3500
3000
2500
2000
1500
1000
500
0

<u>WORKING</u> :- In this project an hidden message image same as the size of the overlapping image is being encoded within that image and therefore getting transmitted as an hidden image and the resultant histrograms for both the images is produced correspondingly.

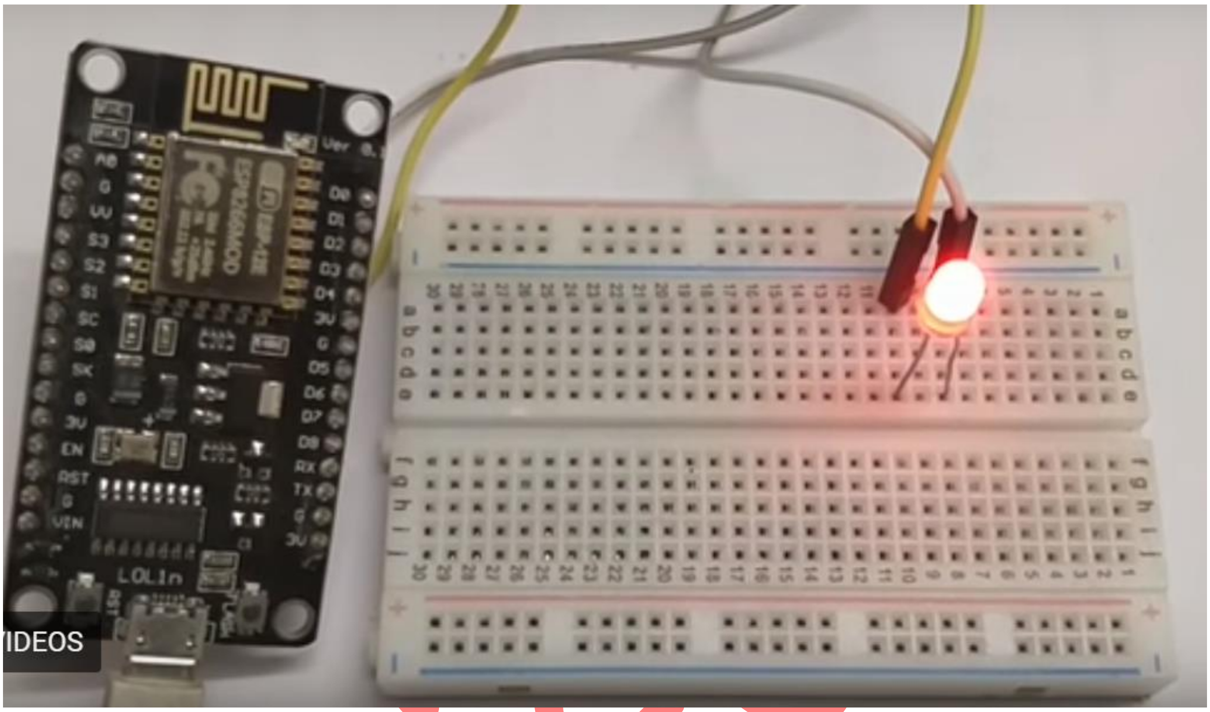
CONTROLLING ARDUINO PINS USING SERVER OF ESP8266

CODE:

```
#include <ESP8266WiFi.h>

const char* ssid = "shubham"; // SSID i.e. Service Set Identifier is the name of your WIFI
const char* password = "shubham123"; // Your Wifi password, in case you have open network comment the
whole statement.

int ledPin = 13; // GPIO13 or for NodeMCU you can directly write D7
WiFiServer server(80); // Creates a server that listens for incoming connections on the specified port, here in
this case port is 80.

void setup() {
  Serial.begin(115200);
  delay(10);

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.print("Use this URL to connect: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP()); //Gets the WiFi shield's IP address and Print the IP address of serial monitor
  Serial.println("/");

}
```

```
void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Match the request

  int value = LOW;
  if (request.indexOf("/LED=ON") != -1)  {
    digitalWrite(ledPin, HIGH);
    value = HIGH;
  }
  if (request.indexOf("/LED=OFF") != -1)  {
    digitalWrite(ledPin, LOW);
    value = LOW;
  }

// Set ledPin according to the request
//digitalWrite(ledPin, value);

  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); //  do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");

  client.print("Led pin is now: ");

  if(value == HIGH) {
    client.print("On");
  } else {
    client.print("Off");
  }
  client.println("<br><br>");
  client.println("<a href=\"/LED=ON\"\"><button>Turn On </button></a>");
  client.println("<a href=\"/LED=OFF\"\"><button>Turn Off </button></a><br />");
  client.println("</html>");

  delay(1);
  Serial.println("Client disonnected");
  Serial.println(""); }
```

OUTPUT:



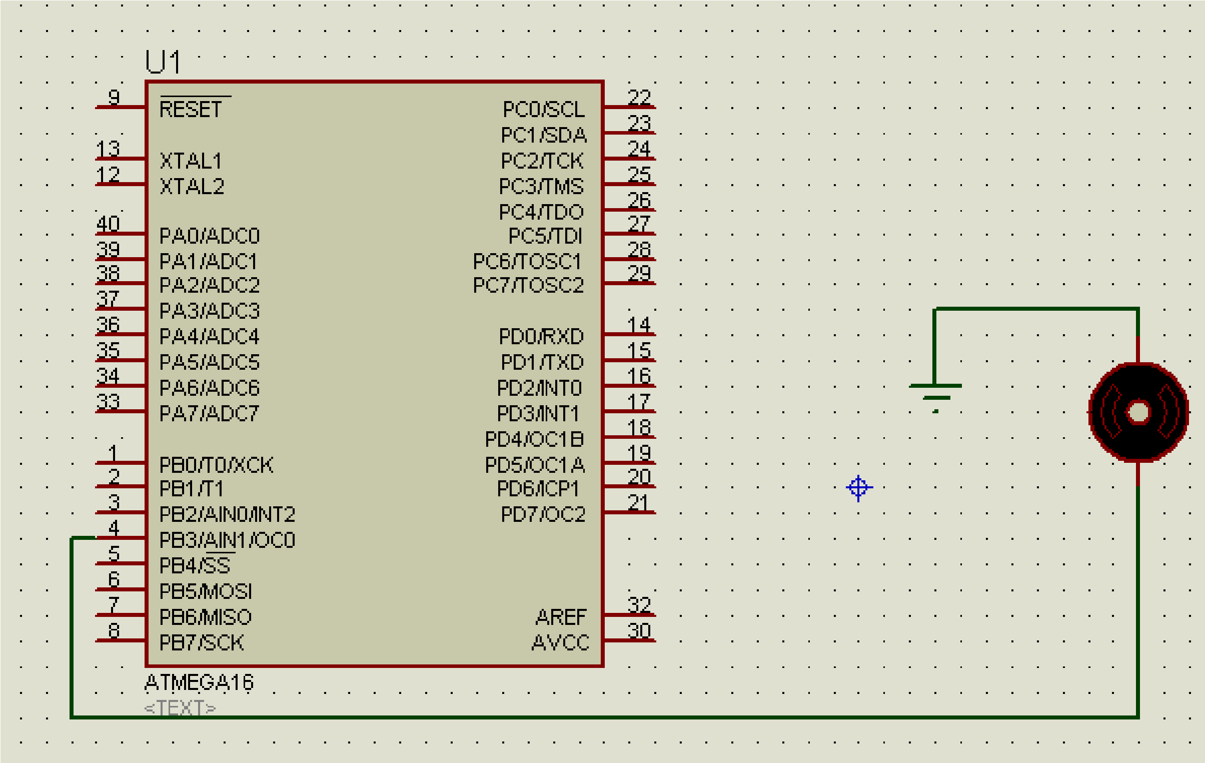
WORKING: In this project the arduino pins are getting controlled by an mobile app (Blynk) connected to the server created by nodemcu(ESP8266).Nodemcu is programmed with the help of arduino ide here but we can program it with it’s own ide also .

CONTROLLING SPEED OF MOTOR USING PWM IN AVR

CODE:

```
#include<avr/io.h>
#define F_CPU 1000000
#include<util/delay.h>
#include"lcdavr.h"



void init_PWM()
{
        TCCR0 |= (1<<WGM00)|(1<<WGM01)|(1<<COM01)|(1<<COM00)|(1<<CS00);
        DDRB |= 1<<PB3;

}
int main()
{
DDRA=0<<PA0;
PORTA=1<<PA0;
        init_PWM();
        while(1)
        {
        int i;
        OCR0=i  ;
          for(int i=0;i<250;i++)
   {
    OCR0=i;
          _delay_ms(300);

   }

  for( i=250;i>0;i--)
   {
    OCR0=i; //output at pb3

          _delay_ms(300);
           }


    }
        return 0;
}


//************************************END*************************************************
*
```

SCHEMATIC DIAGRAM:



WORKING: In this project I am using the in built PWM(Pulse Width Modulation) for controlling the speed of the motor connected at the pin PB3 with the help of different amplitudes pulses according to which the speed of the Motor can be controlled.

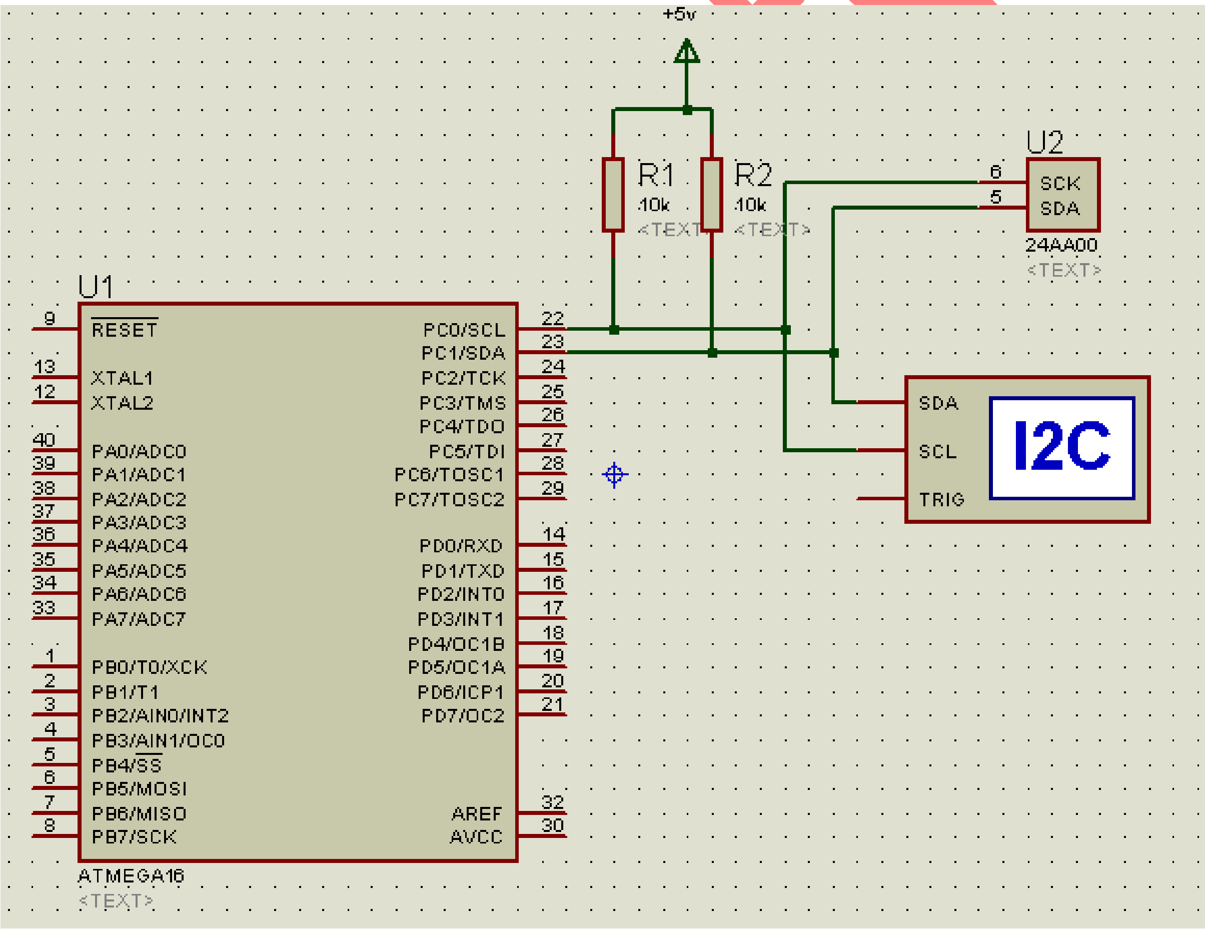
I2C COMMUNICATION USING ATMEGA 16

CODE:

```
#include<avr/io.h>
#define F_CPU 8000000
#include<util/delay.h>

//#################### I2C WRITE ###############

void i2c_write(unsigned char data)
{
TWDR=data;
TWCR=(1<<TWINT)|(1<<TWEN);
while((TWCR & (1<<TWINT))==0);
}


//##################### I2C START ##############

void i2c_start()
{
TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
while((TWCR & (1<<TWINT))==0);
}

//##################### I2C STOP ###############

void i2c_stop()
{
TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}

//##################### I2C INIT ###############

void i2c_init()
{
TWSR=0X00;   // SET PRESCALER BITS TO ZERO
TWBR=0X47;   // SCL FREQUENCY IS 50K FOR XTAL=8M
TWCR=0X04;   // ENABLE THE TWI MODULE
}

//#################### VOID MAIN ###############
char dataa;
int main()
{
```

```
i2c_init();
i2c_start();            //TRANSMIT START CONDITION
i2c_write(0XA0);     // TRANSMIT SLA + W(0)
i2c_write(0b11110000);      // TRANSMIT DATA
i2c_stop();             //TRANSMIT STOP CONDITION
while(1)
{
dataa=TWDR;
PORTA=dataa;
}
return 0;

}
```

SCHEMATIC DIAGRAM:



WORKING:In this project Atmega 16 is communicating with an I2C device having SCL & SDA PINs which is serial clock and serial data pins and the final data is shown on atmega 16 pins.I2C is a two wire communication protocol it stands for **Inter Integerated Circuit.**

COMMUNICATION BETWEEN TWO ATMEGA 16 USING SPI PROTOCOL

CODE:

SPI MASTER :

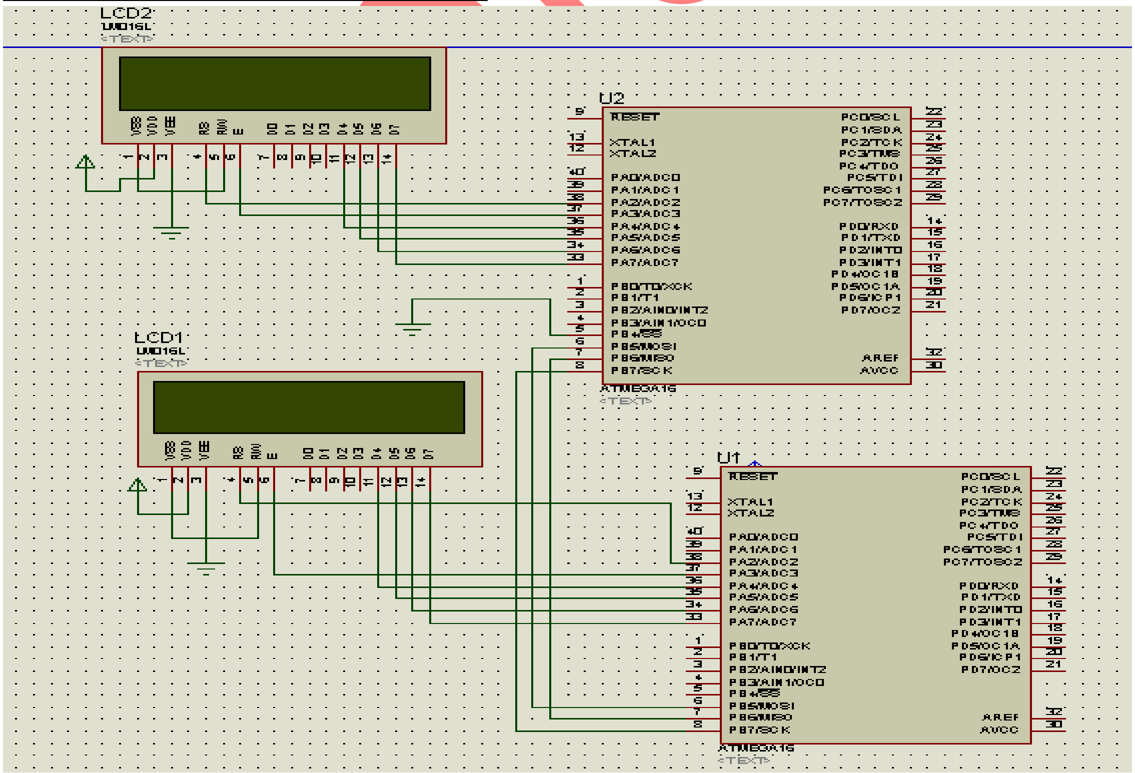
```
#define F_CPU 8000000UL                /* Define CPU Frequency 8MHz */
#include <avr/io.h>                    /* Include AVR std. library file */
#include <util/delay.h>                /* Include Delay header file */
#include <stdio.h>                     /* Include Std. i/p o/p file */
#include<lcdavr.h>

#define MOSI PB5
#define MISO PB6
#define SCK PB7
#define SS PB4

void SPI_Init()                                  /* SPI Initialize function */
{
        DDRB |= (1<<MOSI)|(1<<SCK)|(1<<SS);      /* Make MOSI, SCK, SS
                                                 as Output pin */
        DDRB &= ~(1<<MISO);                      /* Make MISO pin
                                                 as input pin */
        PORTB |= (1<<SS);                        /* Make high on SS pin */
        SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);     /* Enable SPI in master mode
                                                 with Fosc/16 */
        SPSR &= ~(1<<SPI2X);                     /* Disable speed doubler */
}

void SPI_Write(char data)         /* SPI write data function */
{
        char flush_buffer;
        SPDR = data;                   /* Write data to SPI data register */
        while(!(SPSR & (1<<SPIF)));    /* Wait till transmission complete */
        flush_buffer = SPDR;           /* Flush received data */

/* Note: SPIF flag is cleared by first reading SPSR (with SPIF set) and then accessing SPDR hence flush buffer
used here to access SPDR after SPSR read */
}

char SPI_Read()                        /* SPI read data function */
{
        SPDR = 0xFF;
        while(!(SPSR & (1<<SPIF)));         /* Wait till reception complete */
        return(SPDR);                       /* Return received data */
}


int main(void)
{
DDRA=0XFF;
        uint8_t count;
        char buffer[5];

        lcd_init();
        SPI_Init();

        lcd_string("MASTER DEVICE");
        lcd_command(0xc0);
        lcd_string("SENDING DATA:-");


        //SS_Enable;
        count = 0;
        while (1)                       /* Send Continuous count */
        {
                SPI_Write(count);
                sprintf(buffer, "%d  ", count);
                lcd_command(0xce);
                lcd_string(buffer);
                count++;
                _delay_ms(100);
        }

}
```

SPI SLAVE :

```
#include<avr/io.h>
#define F_CPU 8000000
#include<util/delay.h>
#include<stdio.h>
#include<lcdavr.h>

#define MOSI PB5
#define MISO PB6
#define SCK PB7
#define SS PB4

void SPI_Init()                                         /* SPI Initialize function */
{
        DDRB &= ~((1<<MOSI)|(1<<SCK)|(1<<SS));  /* Make MOSI, SCK, SS as
                                                input pins */
        DDRB |= (1<<MISO);                      /* Make MISO pin as
                                                output pin */
        SPCR = (1<<SPE);                        /* Enable SPI in slave mode */
}
```

```
char SPI_Receive()                          /* SPI Receive data function */
{
        while(!(SPSR & (1<<SPIF)));         /* Wait till reception complete */
        return(SPDR);                       /* Return received data */
}

int main(void)
{
        uint8_t count;
        char buffer[5];
        DDRA=0XFF;

        lcd_init();
        SPI_Init();

        lcd_string("Slave Device");
        lcd_command(0xc0);
        lcd_string("Receive Data:   ");
        while (1)                       /* Receive count continuous */
        {
                count = SPI_Receive();
                sprintf(buffer, "%d   ", count);
                lcd_command(0xce);
                lcd_string(buffer);
        }

}
```

SCHEMATIC DIAGRAM :



WORKING :- In this project I used the SPI protocol for communication between two Atmega 16 ‘s also known as **Serial Peripheral Interface** which use MOSI,MISO,SCK & SDA pins SCK & SDA are for data and clock signal and MOSI & MISO pins are used for selecting the modes,

MOSI – **MASTER OUT SLAVE IN**

MISO – **MASTER IN SLAVE OUT**

By using these pins we can select for a particular device whether it will be working as a master and send data to the slave or it will work as a slave and receive data from the master it is connected to.