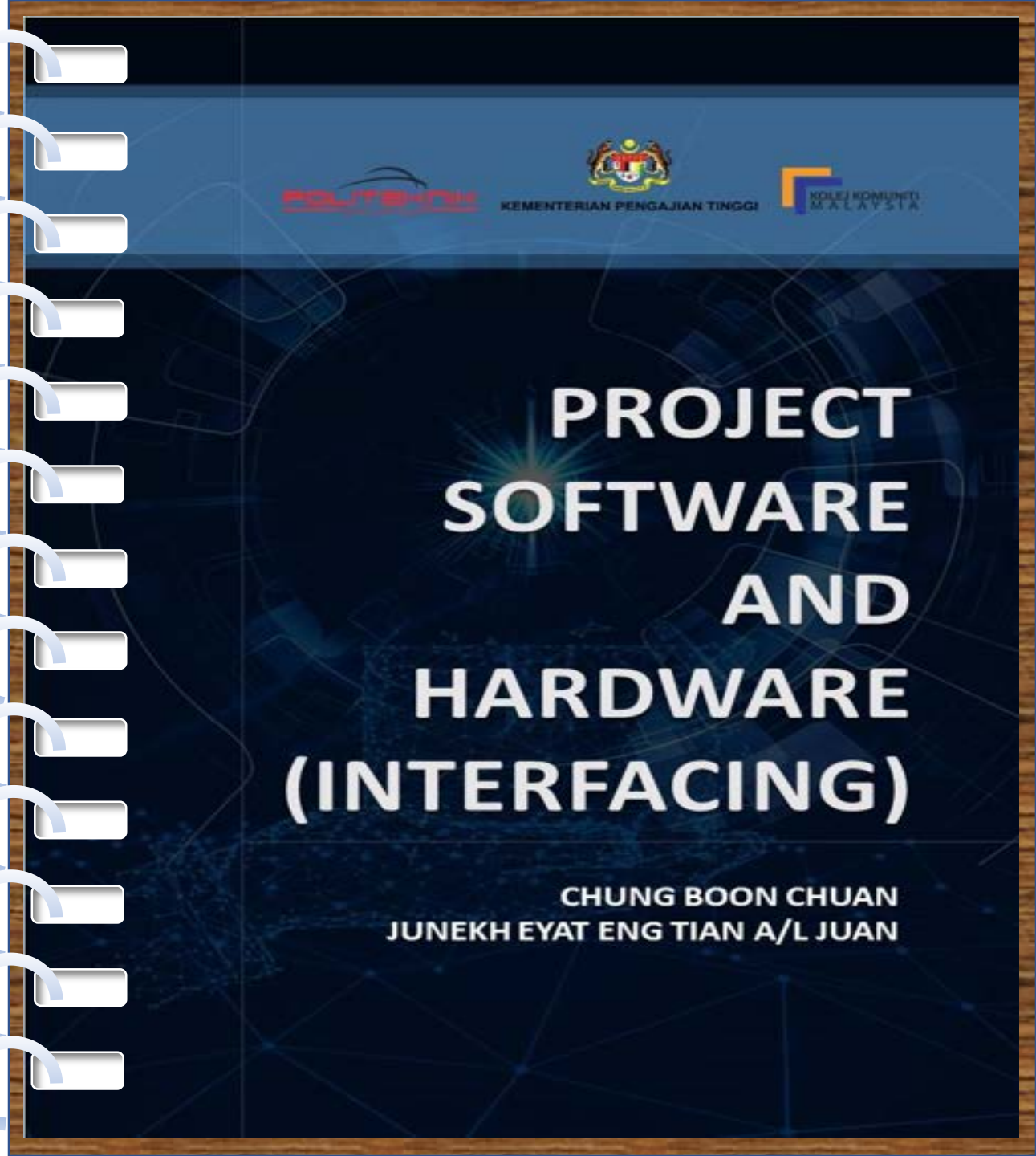

KEMENTERIAN PENGAJIAN TINGGI
KOLEJ KOMUNITI MALAYSIA
PROJECT SOFTWARE AND HARDWARE (INTERFACING)
CHUNG BOON CHUAN
JUNEKH EYAT ENG TIAN A/L JUAN

PROJECT SOFTWARE AND HARDWARE (INTERFACING)

CHUNG BOON CHUAN
JUNEKH EYAT ENG TIAN A/L JUAN

POLITEKNIK KOTA BHARU
POLITEKNIK IBRAHIM SULTAN

e ISBN 978-967-2702-04-7

9 7 8 9 6 7 2 7 0 2 0 4 7

I

Published and printed by:

Electrical Engineering Department Kota Bharu Polytechnic KM24 Kok Lanas, 16450 Ketereh Kelantan in collaboration with Electrical Engineering Department Ibrahim Sultan Polytechnic KM 10, Jalan Kong Kong, 81700 Pasir Gudang, Johor

Project Software And Hardware (Interfacing) First Edition 2021

B.C Chung, Junekh
Project Software And Hardware (Interfacing) For Student IPT
First Edition 2021 / B.C.Chung, Junekh

We would like to thanks Kota Bharu Polytechnic and Ibrahim Sultan Polytechnic for giving us the opportunity to produce this book. The book Project Software And Hardware (Interfacing) For IPT Students was successfully published after going through planning for a year. This book is written and published as a guide or reference to all electrical and electronic engineering students in polytechnics on the production of final semester projects. This book will give students exposure to Adriuno related projects. The Adruino project is very popular, inexpensive and easy to learn. Hopefully this book can be used by all lecturers and students.

Chung Boon Chuan
Electrical Engineering Department
Politeknik Kota Bharu
KM24 Kok Lanas, 16450 Ketereh
Kelantan

Junekh Eyat Eng Tian A/L Juan
Electrical Engineering Department
Politeknik Ibrahim Sultan
KM 10, Jalan Kong Kong,
81700 Pasir Gudang, Johor

SYNOPSIS

This Software and Hardware Book was produced to provide exposure for student learn the software and hardware to create a projects in Electrical and Electronic Engineering.

This book is suitable for use by students for Fundamental Programming and final semester projects. This book gives an idea and correct methods to to create project by using adruino with C programming . Adruino is a popular and user friendly boardcfor student to learn basic software and Hardware projects.

This book has been written and compiled specifically to facilitate students with basic knowledge related to programming to implement projects. By using easy to understand language, it clearly describes the steps to produce a project.

The author hopes for comments and suggestions from readers to improve for the next writing.

About Writer

Chung Boon Chuan

The author was born in 1972 in Tanah Merah, Kelantan. Has an academic qualification in Electrical Engineering First Degree (UTM, 97) and Bachelor of Education (UTM, 98). Started his career as a lecturer at Seberang Prai Polytechnic in August 1999 and currently works at Kota Bharu Polytechnic.

Junekh Eyat Eng Tian A/L Juan

The author was born in 1978 in Wakaf Bharu, Kelantan. He completed his Diploma in Electrical Engineering at Kota Bharu Polytechnic (PKB) in 2000. He then continued his bachelor's degree at Tun Hussein Onn University College (KUiTTHO) in 2002. He futher his studies in Master's Degree in Technical and Vocational Education at Universiti Tun Hussein Onn (UTHM) in 2005. Started his career as a lecturer in the Electrical Engineering Department in 2007 at Kota Bharu Polytechnic until 2016. He currently works at Ibrahim Sultan Polytechnic (PIS). Apart from that, he is also the project supervisor of Electrical Engineering students.

CONTENTS

CONTENTS

1. Introduction To Adruino

1.1 Introduction

a. Adruino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

b. Adruino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and otherphysical outputs.

c. Adruino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

d. The Adruino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

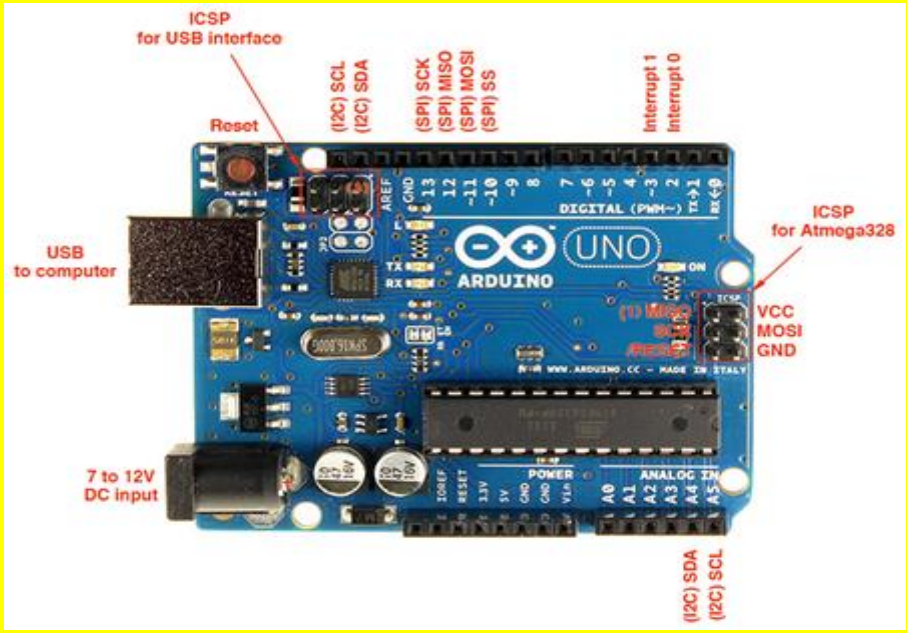


Figure 1.1 : Adruino Uno open-source microcontroller board

1. Introduction to **Adruino**

1.2 LED Blinking with Adruino Uno

```
// The setup function runs when you press reset or power the
board
void setup()
{
 // initialize digital pin 0 as an output.
pinMode(0, OUTPUT);
 }
// the loop function runs over and over again forever
void loop()
{
digitalWrite(0, HIGH);   // turn the LED on (HIGH is the voltage
level)
delay(500);              // wait for a second
digitalWrite(0, LOW);    // turn the LED off by making the voltage
LOW
delay(500);              // wait for a second
}
```

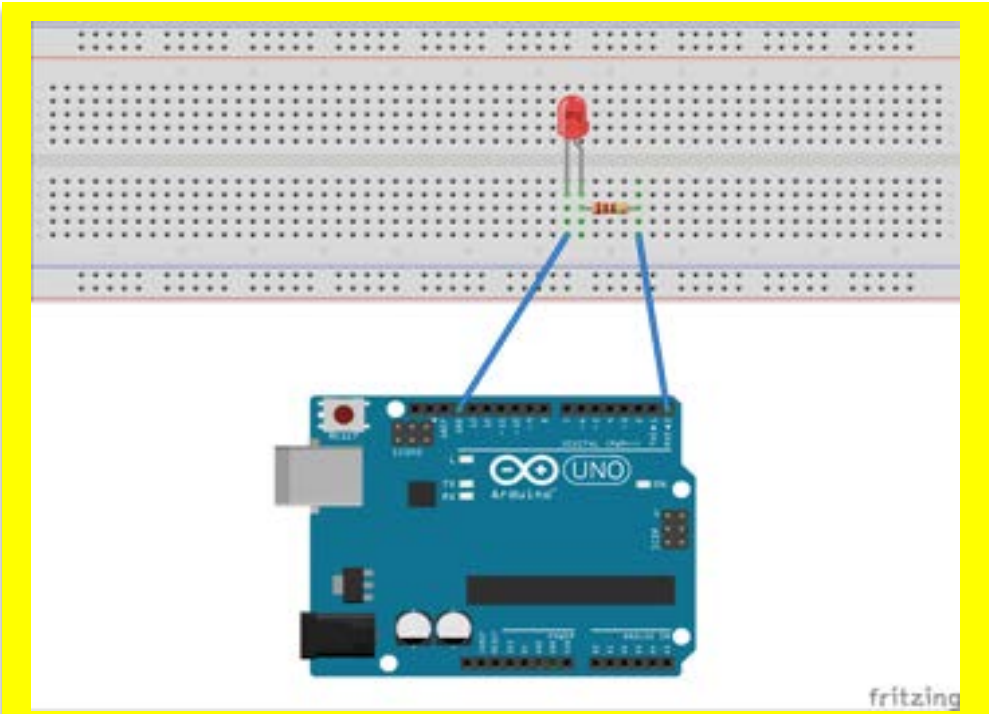


Figure 1.2 : Hardware Connection for LED Blinking with Adruino Uno

1. Introduction to **Adruino**

1.3 LED DICE

a. Hardware Connection

FirstlyLED is arangge like the shape of the dice. Then connect the LED1 to LED7 to digital port-2 until port-8. Connect a pull-up switch to control the operation of the LED dice which is connected to digital port 11. Every LEDs are connected with an 1K resistor to prevent over current LED which will make the LED damage.

```
//Defining LED Pins
int ledPins[7] = {2, 3, 4, 5, 6, 7, 8};
int dicePatterns[7][7] = {
{0, 0, 0, 0, 0, 0, 1}, // 1
{0, 0, 1, 1, 0, 0, 0}, // 2
{0, 0, 1, 1, 0, 0, 1}, // 3
{1, 0, 1, 1, 0, 1, 0}, // 4
{1, 0, 1, 1, 0, 1, 1}, // 5
{1, 1, 1, 1, 1, 1, 0}, // 6
{1, 1, 1, 1, 1, 1, 1}, // 7
{0, 0, 0, 0, 0, 0, 0} // BLANK
};

int switchPin =11;   //Defining button pin
int blank = 7;
void setup()
{
for (int i = 0; i < 8; i++)
{
pinMode(ledPins[i], OUTPUT);
digitalWrite(ledPins[i], LOW);
}

randomSeed(analogRead(0));
}
void loop()
{
  if (digitalRead(switchPin))
{
rollTheDice();
}
delay(100);
}
void rollTheDice()
{
```

```
int result = 0;
int lengthOfRoll = random(15, 25);
for (int i = 0; i < lengthOfRoll; i++)
{
result = random(0, 6); // result will be 0 to 5 not 1 to 6
show(result);
delay(50 + i * 10);
}
for (int j = 0; j < 3; j++)
{
show(blank);
delay(500);
show(result);
delay(500);
}
}
void show(int result)
{
for (int i = 0; i < 8; i++)
{
digitalWrite(ledPins[i], dicePatterns[result][i]);
}
}
```

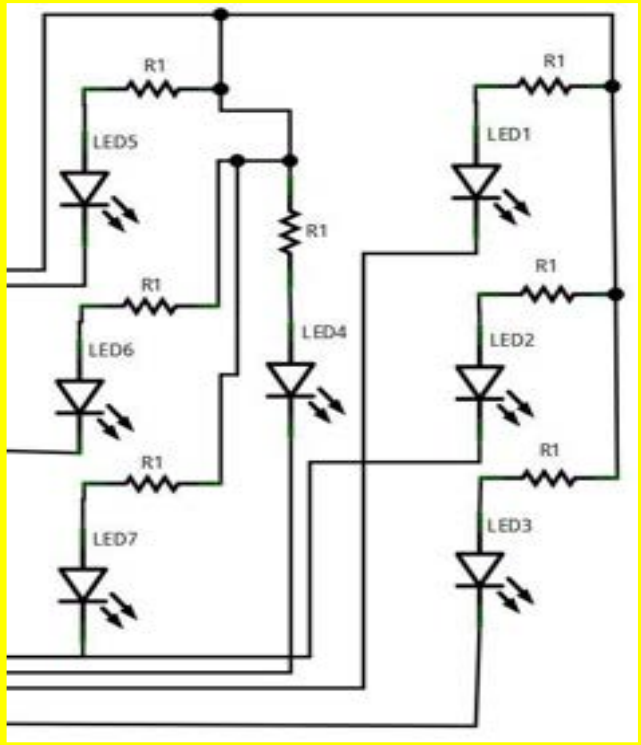



Figure 1.3.2 : Schematic Diagram For LED Dice

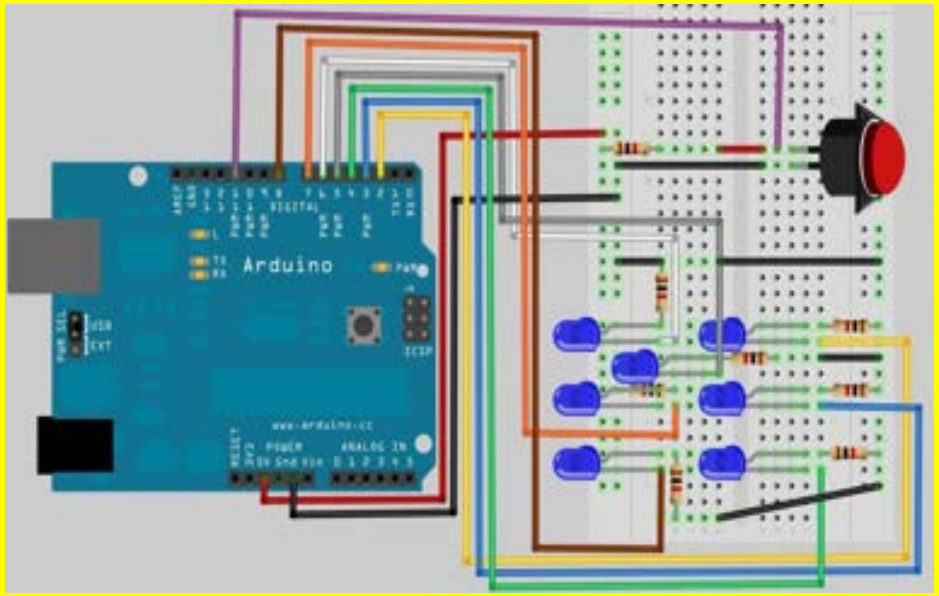


Figure 1.3.1 : Hardware Connection for LED Dice with Adruino Uno

1. Introduction to **Adruino**

1.4 Multiple Blink LED

In this lesson, it shown the first external circuit and control it from Adruino.It use digitalWrite command to turn LED on and off. The components require for this project are :

8 x LED
8 x 1KΩ Resistor
Adruino
1 x USB cable
Breadboard
Jumper Wires

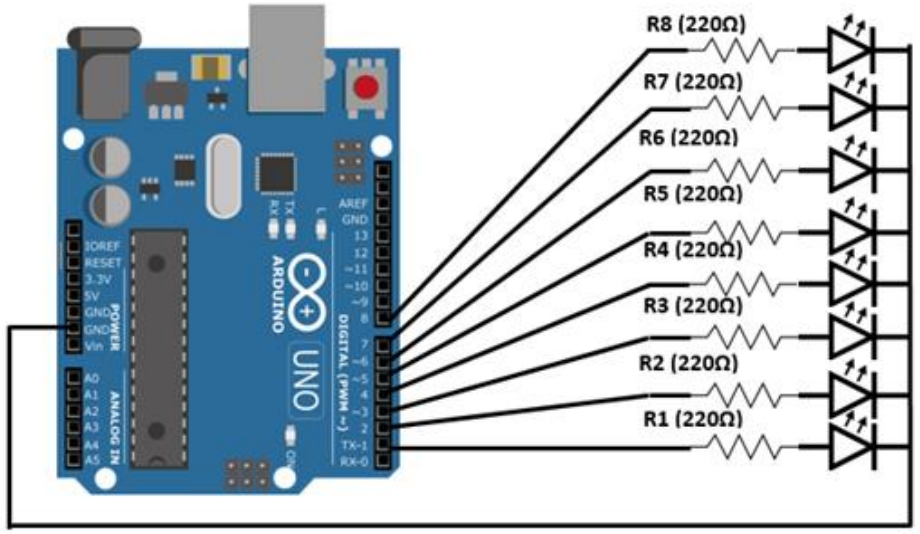


Figure 1.4.1 : Hardware Connection For Multiple Blink LED Project


```
int led1=1;
int led2=2;
int led3=3;
int led4=4;
int led5=5;
int led6=6;
int led7=7;
int led8=8;

void setup() {
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
pinMode(led3,OUTPUT);
pinMode(led4,OUTPUT);
pinMode(led5,OUTPUT);
pinMode(led6,OUTPUT);
pinMode(led7,OUTPUT);
pinMode(led8,OUTPUT);
}

void loop() {

digitalWrite ( led1, HIGH);
delay (500);
digitalWrite ( led1, LOW);
delay (500);
digitalWrite ( led2, HIGH);
delay (500);
digitalWrite ( led2, LOW);
```

```
delay (500);
digitalWrite ( led3, HIGH);
delay (500);
digitalWrite ( led3, LOW);
delay (500);
digitalWrite ( led4, HIGH);
delay (500);
digitalWrite ( led4, LOW);
delay (500);
digitalWrite ( led5, HIGH);
delay (500);
digitalWrite ( led5, LOW);
delay (500);

digitalWrite ( led6, HIGH);
delay (500);
digitalWrite ( led6, LOW);
delay (500);
digitalWrite ( led7, HIGH);
delay (500);
digitalWrite ( led7, LOW);
delay (500);
digitalWrite ( led8, HIGH);
delay (500);
digitalWrite ( led8, LOW);
delay (500);
}
```

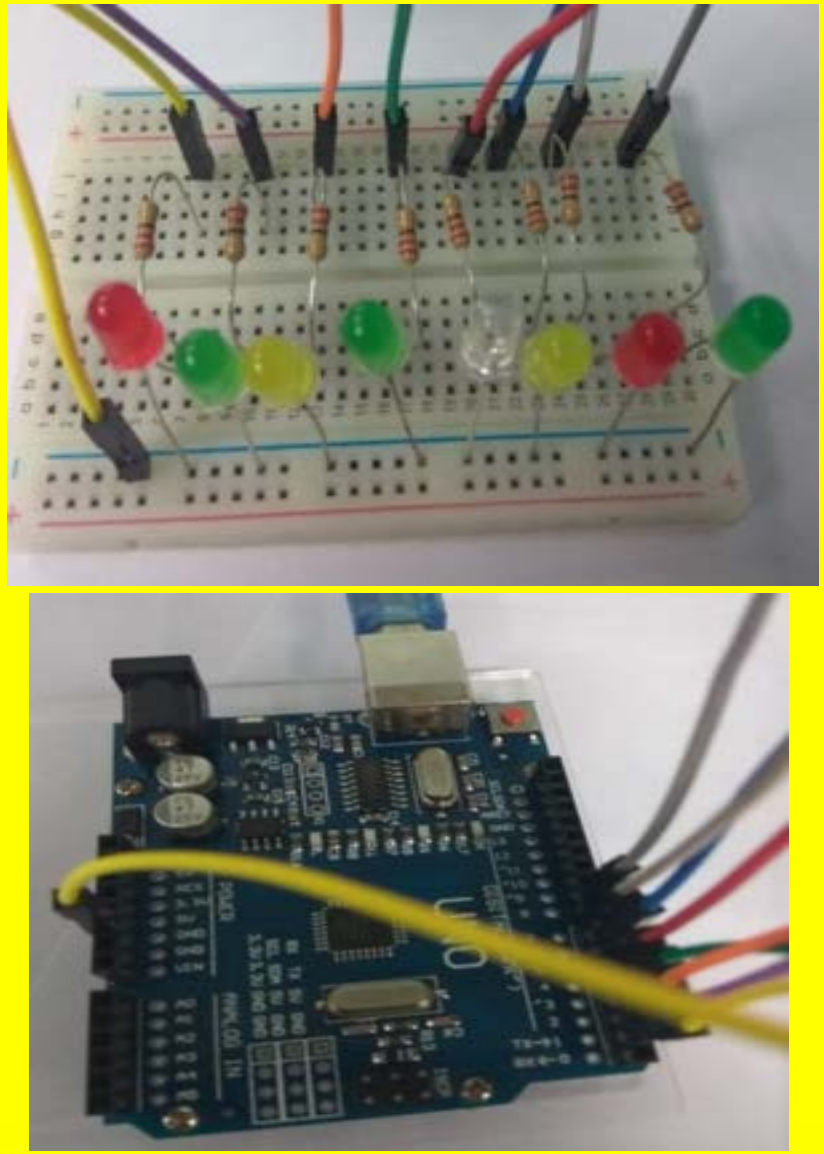


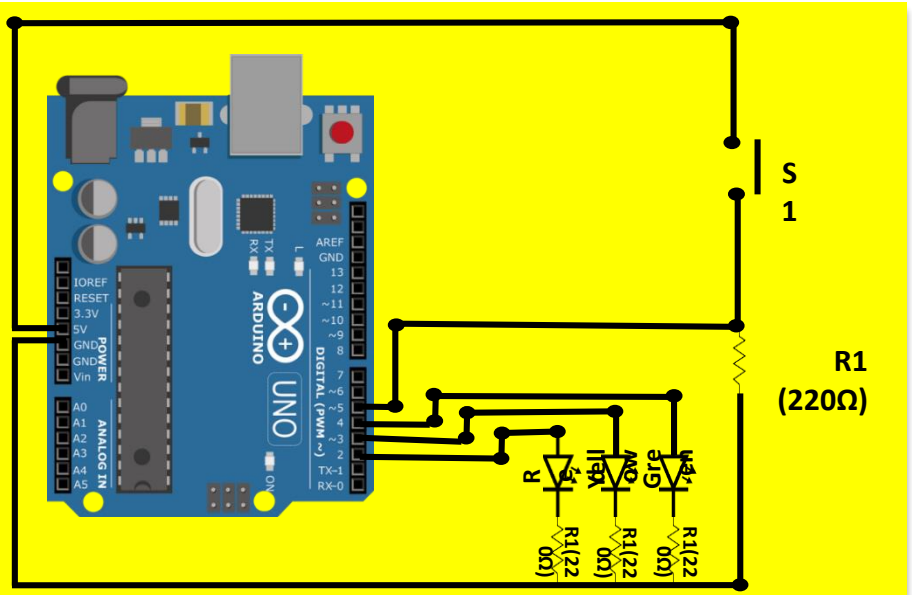
Figure 1.4.2 : Output For Multiple Blink LED Project

1. Introduction to **Adruino**

1.5 Traffic Signal Model

This project is design for traffic signals model using red, yellow, and green LEDs. Every time we press the button, the traffic signal will go to the next step in the sequence. The sequence of the traffic signals is red, red and amber together, green, amber, and then back to red. There a button connected. If the button is pressed, the lights will change in sequence with a delay between each step. The components require for this project are :

Adruino UNO
4x 220 ohm resistor
hook-up wires
breadboard
red LED
yellow LED
green LED
1x push switch



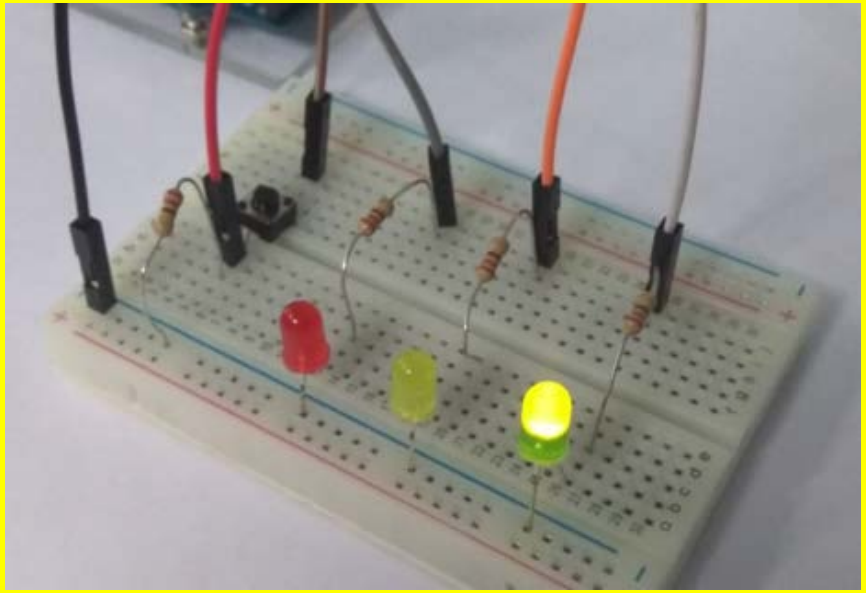


Figure 1.5 : Hardware Connection For Traffic Signal Model

```
int redPin = 2;
int yellowPin = 3;
int greenPin = 4;
int buttonPin = 5;
int state = 0;
void setup()
{
pinMode(redPin, OUTPUT);
pinMode(yellowPin, OUTPUT);
pinMode(greenPin, OUTPUT);
pinMode(buttonPin, INPUT);
}
void loop()
{
if (digitalRead(buttonPin))
{
if (state == 0)
{
setLights(HIGH, LOW, LOW);
state = 1;
}
else if (state == 1)
{
setLights(LOW, HIGH, LOW);
state = 2;
}
```



```
else if (state == 2)
{
setLights(LOW, LOW, HIGH);
state = 3;
}
else if (state == 3)

{
setLights(LOW, HIGH, LOW);
state = 0;
}
delay(1000);
}
}
void setLights(int red, int yellow,
int green)
{
digitalWrite(redPin, red);
digitalWrite(yellowPin, yellow);
digitalWrite(greenPin, green);
}
```

1. Introduction to **Adruino**

1.6 Traffic Signal Model (Automatic)

This project is design for automatic Traffic Signals Model using red, yellow, and green LEDs. The sequence of the traffic signals is red, red and amber together, green, amber, and then back to red. The components require for this project are :

Adruino UNO
Adruino UNO
3x 220 ohm resistor
hook-up wires
breadboard
red LED
yellow LED
green LED

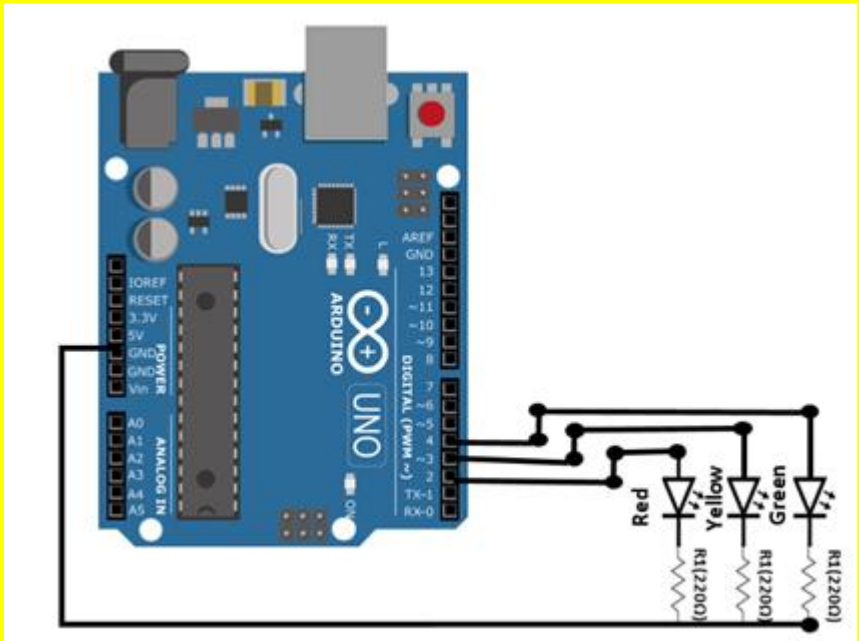
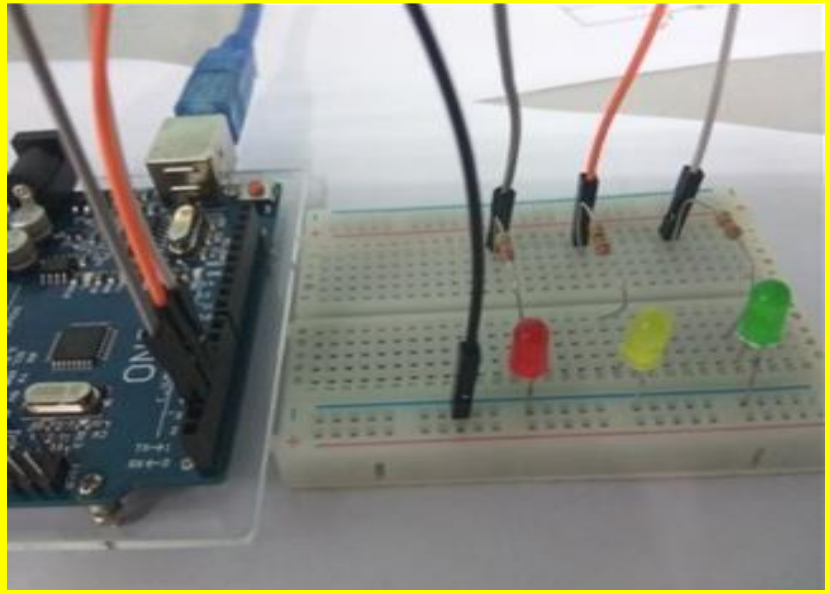


Figure 1.6.1 : Hardware Connection For Traffic Signal Model (Automatic)



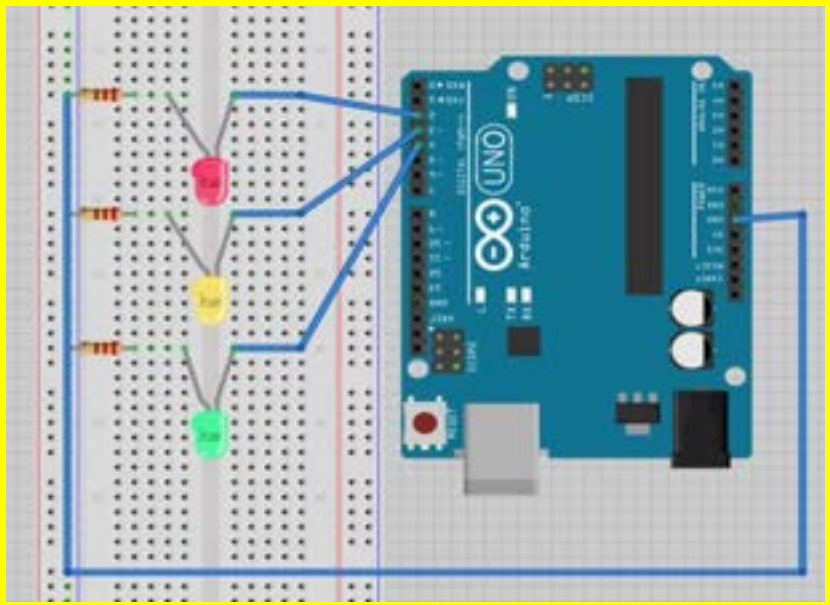


Figure 1.6.2 : Output For Traffic Signal Model (Automatic)

```
int red = 2;
int yellow = 3;
int green = 4;
int x;
void setup()
{
pinMode(red, OUTPUT);
pinMode(yellow, OUTPUT);
pinMode(green, OUTPUT);
}
void loop()
{
digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
digitalWrite (green, LOW);
delay(500);
digitalWrite (red, LOW);
for(x=1; x<=5;x++)

{
digitalWrite (yellow, LOW);
delay(500);
digitalWrite (yellow, HIGH);
delay(500);
}
digitalWrite (yellow, LOW);
digitalWrite (green, HIGH);
delay(500);
digitalWrite (green, LOW);
```

```
for(x=1; x<=4;x++)
{
digitalWrite (yellow, LOW);
delay(500);
digitalWrite (yellow, HIGH);
delay(500);
}
}
```


1. Introduction to **Adruino**

1.7 Traffic Signal Model (Automatic 2 Traffic Light)

This project Traffic Signals Model automatically using 2 Traffic Light. It consists of red, yellow, and green LEDs. The sequence of the traffic signals is red, red and amber together, green, amber, and then back to red. The components require for this project are :

Adruino UNO
2x 220 ohm resistor
hook-up wires
breadboard
2x red LED
2x yellow LED
2x green LED

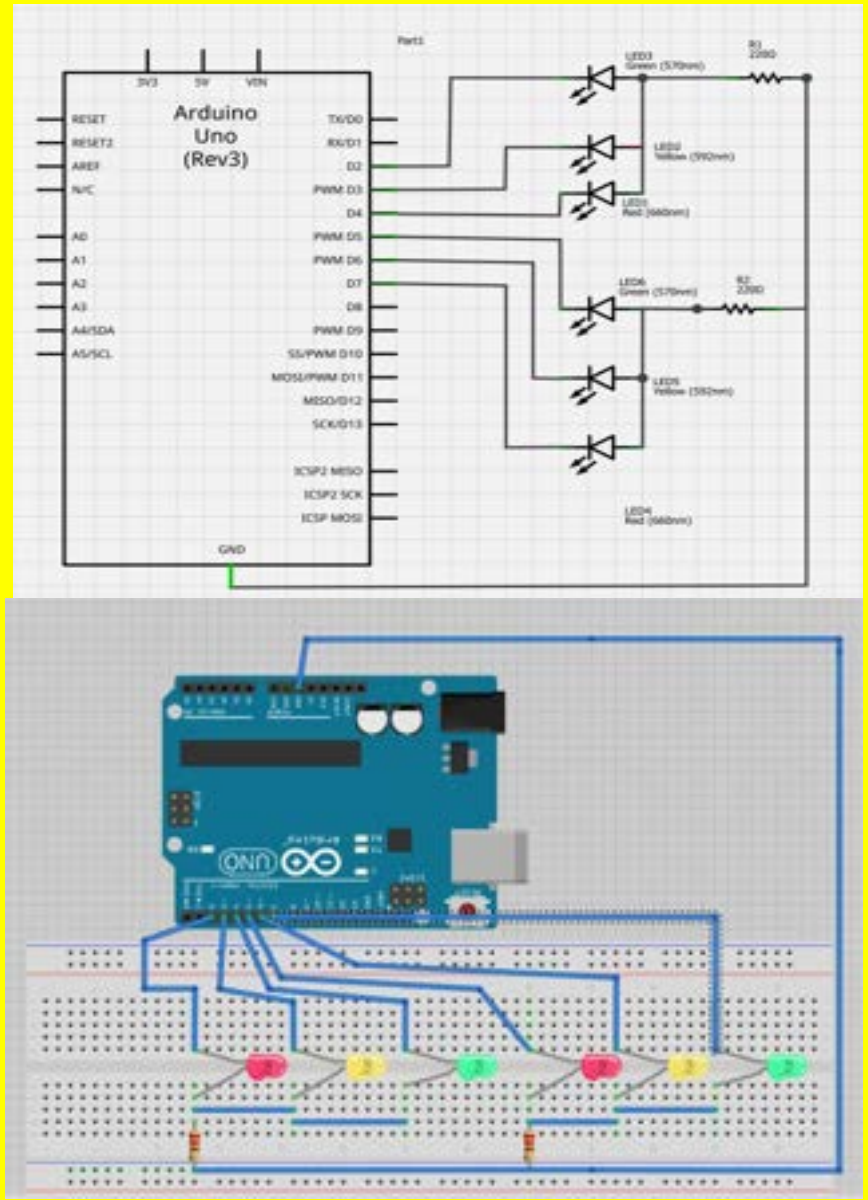


Figure 1.7 : Hardware Connection For Traffic Signal Model (Automatic 2 Traffic Light)


```
int red = 2;
int yellow = 3;
int green = 4;

int red2 = 5;
int yellow2 = 6;
int green2 = 7;


int x;
void setup()
{
pinMode(red, OUTPUT);
pinMode(yellow, OUTPUT);
pinMode(green, OUTPUT);
pinMode(red2, OUTPUT);
pinMode(yellow2, OUTPUT);
pinMode(green2, OUTPUT);

}

void loop()
{
digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
digitalWrite (green, LOW);

digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
delay(1000);
digitalWrite (red2, LOW);
digitalWrite (green2, HIGH);
```

```
delay(5000);
digitalWrite (green2, LOW);

for(x=1; x<=5;x++)
{
digitalWrite (yellow2, LOW);
delay(500);
digitalWrite (yellow2, HIGH);
delay(500);
}

digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
digitalWrite (green2, LOW);

digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
delay(1000);
digitalWrite (red, LOW);
digitalWrite (green,HIGH);

delay(5000);
digitalWrite (green,LOW);

for(x=1; x<=5;x++)
{
digitalWrite (yellow, LOW);
delay(500);
digitalWrite (yellow, HIGH);
delay(500);
}
}
```


1. Introduction to **Adruino**

1.8 Model Traffic Signal (auto 3 traffic light)

This project Traffic Signals Model automatically using 3 Traffic Light. It consists of red, yellow, and green LEDs. The sequence of the traffic signals is red, red and amber together, green, amber, and then back to red. The components require for this project are :

Adruino UNO
Adruino UNO
3x 220 ohm resistor
hook-up wires
breadboard
3x red LED
3x yellow LED
3x green LED

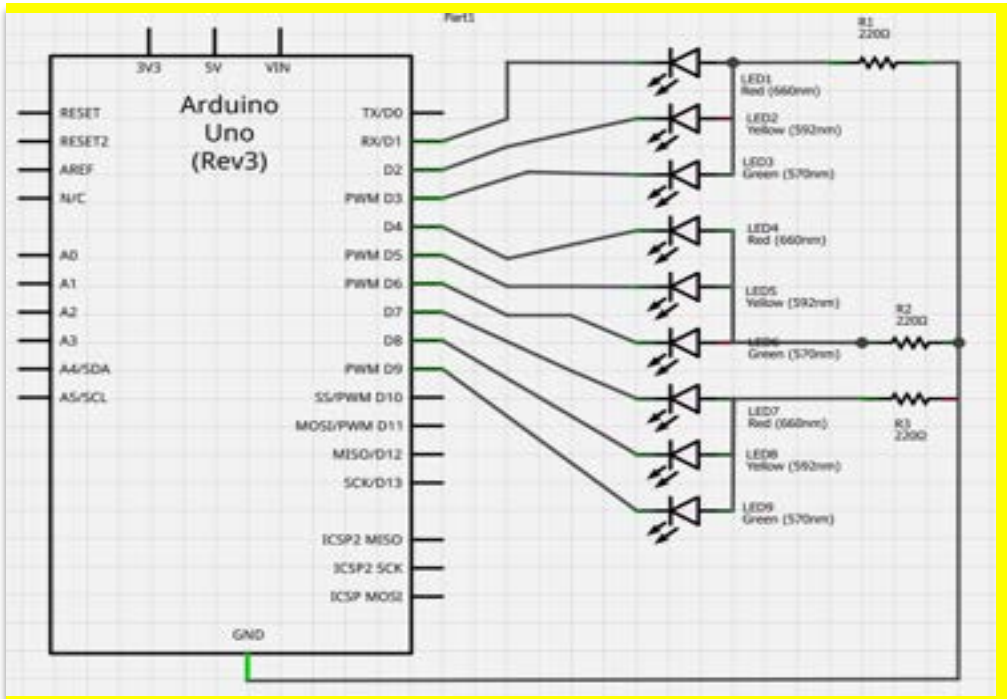
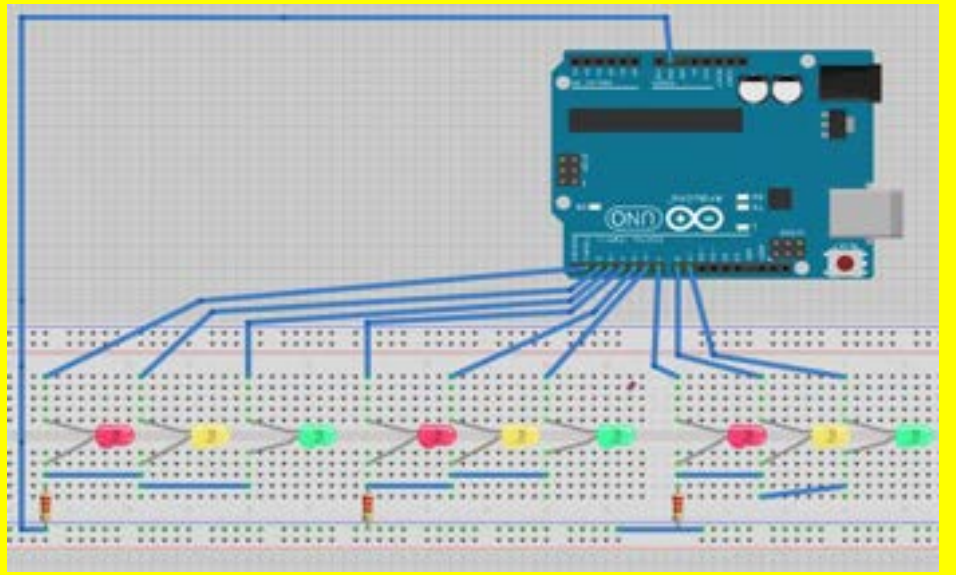


Figure 1.8.1 : Hardware Connection For Traffic Signal Model (Automatic 3 Traffic Light)



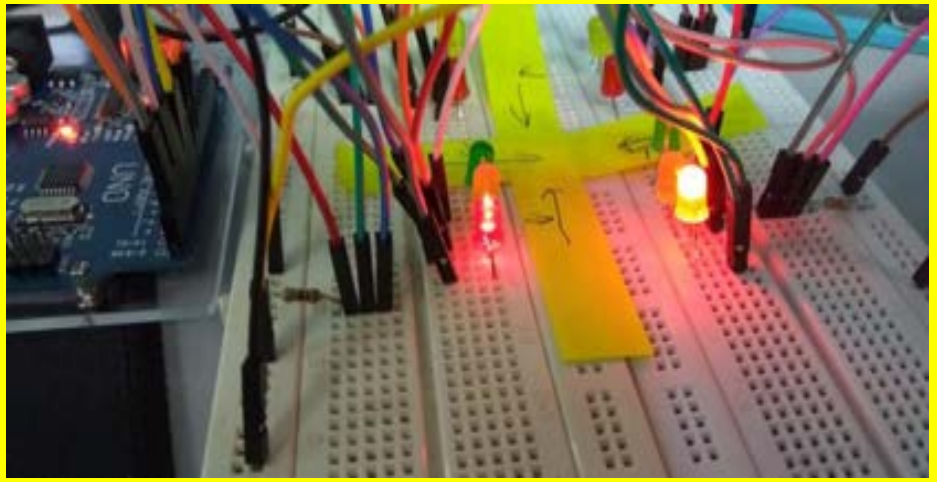


Figure 1.8.2 : OutputFor Traffic Signal Model (Automatic 3 Traffic Light)

```
int red = 1;
int yellow = 2;
int green = 3;

int red2 = 4;
int yellow2 = 5;
int green2 = 6;

int red3 = 7;
int yellow3 = 8;
int green3 = 9;

int x;
void setup()
{
pinMode(red, OUTPUT);
pinMode(yellow, OUTPUT);
pinMode(green, OUTPUT);
pinMode(red2, OUTPUT);
pinMode(yellow2, OUTPUT);
pinMode(green2, OUTPUT);
pinMode(red3, OUTPUT);
pinMode(yellow3, OUTPUT);
pinMode(green3, OUTPUT);
}
void loop()
{
digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
digitalWrite (green, LOW);

digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
digitalWrite (green3, LOW);
```



```
digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
delay(1000);
digitalWrite (red2, LOW);
digitalWrite (green2, HIGH);


delay(5000);
digitalWrite (green2, LOW);


//side ke 2
for(x=1; x<=5;x++)
{
digitalWrite (yellow2, LOW);
delay(500);
digitalWrite (yellow2, HIGH);
delay(500);
}

digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
digitalWrite (green2, LOW);

digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
digitalWrite (green, LOW);

digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
delay(1000);
digitalWrite (red3, LOW);
digitalWrite (green3,HIGH);
```

```
delay(5000);

digitalWrite (green3,LOW);

//side ke 3

for(x=1; x<=5;x++)
{
digitalWrite (yellow3, LOW);
delay(500);
digitalWrite (yellow3, HIGH);
delay(500);
}

digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
digitalWrite (green3, LOW);

digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
digitalWrite (green2, LOW);

digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
delay(1000);
digitalWrite (red, LOW);
digitalWrite (green,HIGH);


delay(5000);

digitalWrite (green,LOW);
```



```
for(x=1; x<=5;x++)
{
digitalWrite (yellow, LOW);
delay(500);
digitalWrite (yellow, HIGH);
delay(500);
}


}
```

1. Introduction to **Adruino**

1.9 Traffic Signal Model (Automatic 4 Sides)

This project is design for Automatic 4 Sides Traffic Signals Model using red, yellow, and green LEDs. The sequence of the traffic signals is red, red and amber together, green, amber, and then back to red. The components require for this project are :

Adruino UNO
4x 220 ohm resistor
hook-up wires
breadboard
4x red LED
4x yellow LED
4x green LED

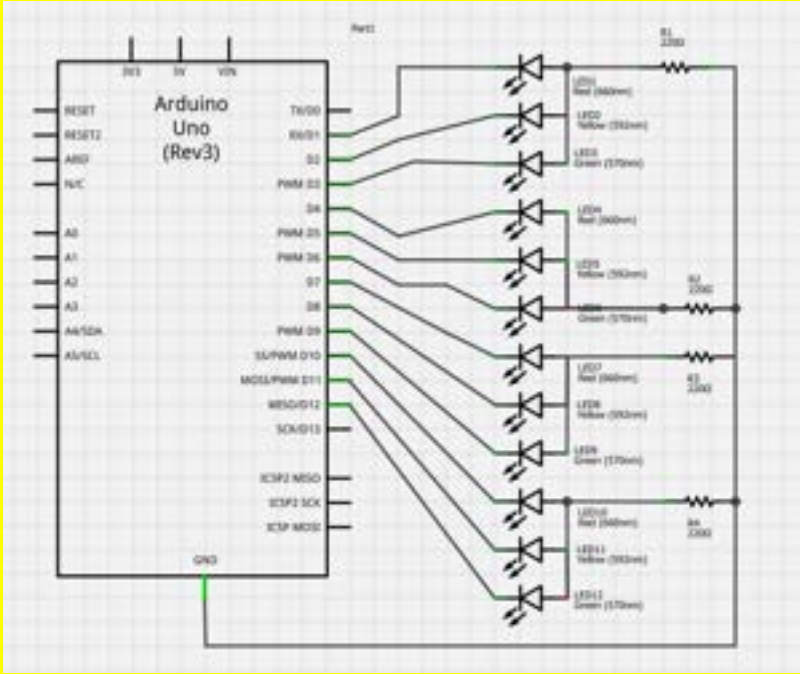
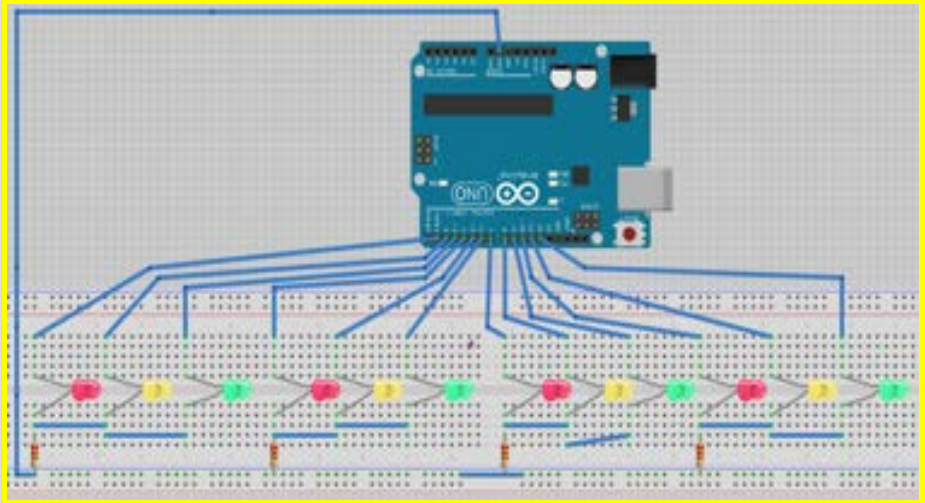


Figure 1.9.1 : Hardware Connection For Traffic Signal Model (Automatic 4 Sides)



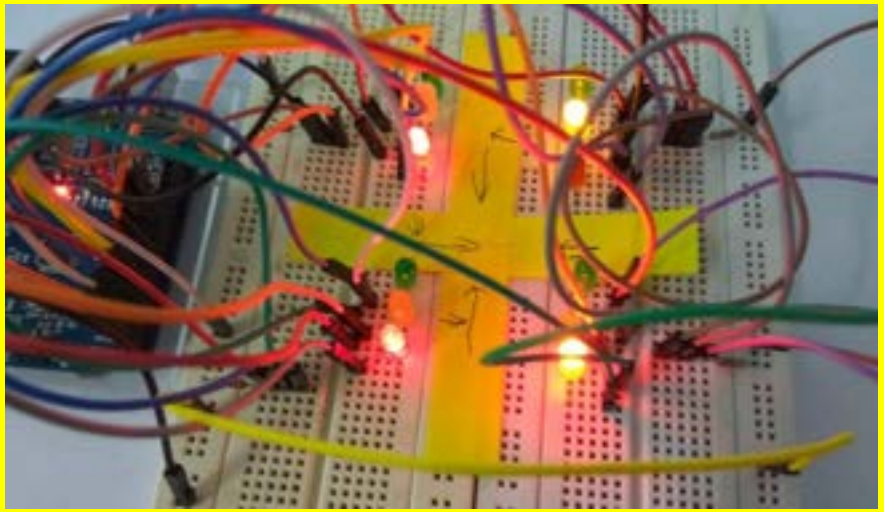


Figure 1.9.2 : Output For Traffic Signal Model (Automatic 4 Sides)


```
int red = 1;
int yellow = 2;
int green = 3;

int red2 = 4;
int yellow2 = 5;
int green2 = 6;

int red3 = 7;
int yellow3 = 8;
int green3 = 9;

int red4 = 10;
int yellow4 = 11;
int green4 = 12;

int x;
void setup()
{
pinMode(red, OUTPUT);
pinMode(yellow, OUTPUT);
pinMode(green, OUTPUT);
pinMode(red2, OUTPUT);
pinMode(yellow2, OUTPUT);
pinMode(green2, OUTPUT);

pinMode(red3, OUTPUT);
pinMode(yellow3, OUTPUT);
pinMode(green3, OUTPUT);

pinMode(red4, OUTPUT);
pinMode(yellow4, OUTPUT);
pinMode(green4, OUTPUT);
}
```

```
void loop()
{

 // side yg pertama
digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
digitalWrite (green, LOW);

digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
digitalWrite (green3, LOW);

digitalWrite (red4, HIGH);
digitalWrite (yellow4, LOW);
digitalWrite (green4, LOW);

digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
delay(1000);
digitalWrite (red2, LOW);
digitalWrite (green2, HIGH);

delay(5000);
digitalWrite (green2, LOW);

//side ke 2
for(x=1; x<=5;x++)
{
digitalWrite (yellow2, LOW);
delay(500);
digitalWrite (yellow2, HIGH);
delay(500);
}
```



```
digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
digitalWrite (green2, LOW);

digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
digitalWrite (green, LOW);

digitalWrite (red4, HIGH);
digitalWrite (yellow4, LOW);
digitalWrite (green4, LOW);

digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
delay(1000);
digitalWrite (red3, LOW);
digitalWrite (green3,HIGH);

delay(5000);

digitalWrite (green3,LOW);

//side ke 3

for(x=1; x<=5;x++)
{
digitalWrite (yellow3, LOW);
delay(500);
digitalWrite (yellow3, HIGH);
delay(500);
}
```

```
digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
digitalWrite (red4, HIGH);
delay(1000);
digitalWrite (red4, LOW);
digitalWrite (green4,HIGH);

delay(5000);

digitalWrite (green4,LOW);

//side ke 3

for(x=1; x<=5;x++)
{
digitalWrite (yellow4, LOW);
delay(500);
digitalWrite (yellow4, HIGH);
delay(500);
}

digitalWrite (red4, HIGH);
digitalWrite (yellow4, LOW);
digitalWrite (green4, LOW);

digitalWrite (red3, HIGH);
digitalWrite (yellow3, LOW);
digitalWrite (green3, LOW);
```



```
digitalWrite (red2, HIGH);
digitalWrite (yellow2, LOW);
digitalWrite (green2, LOW);

digitalWrite (red, HIGH);
digitalWrite (yellow, LOW);
delay(1000);
digitalWrite (red, LOW);
digitalWrite (green,HIGH);

delay(5000);
digitalWrite (green,LOW);
for(x=1; x<=5;x++)
{
digitalWrite (yellow, LOW);
delay(500);
digitalWrite (yellow, HIGH);
delay(500);
}

}
```

2. Intermediate Projects

2.1 Obstacle Sensor Using Adruino And HCSR04

The HC-SRO4 is an ultrasonic sensor which uses sonar to detect objects at a distance of 2 cm to 4 meters. This sensor is widely used in robotics to build robots that move and should divert or avoid obstacles.

The ultrasonic range finder can be used in several ways for range detection and robotics projects. It is able to detect the distance to obstacles are in front of a mobile robot, allowing a maneuver movements before a collision occurs.
Adruino and ultrasonic range finder can give user full control, allowing the user to schedule the most convenient way for for moving robots.

This project uses Adruino UNO R3 and one ultrasonic sensor HC-SRO4. The sensor HC-SRO4 is easy to find and Cheap. The components require for this project are :

Adruino UNO
220 ohm resistor
Jumpers wires
breadboard
HC- SR04;
Three LEDs of different colors;

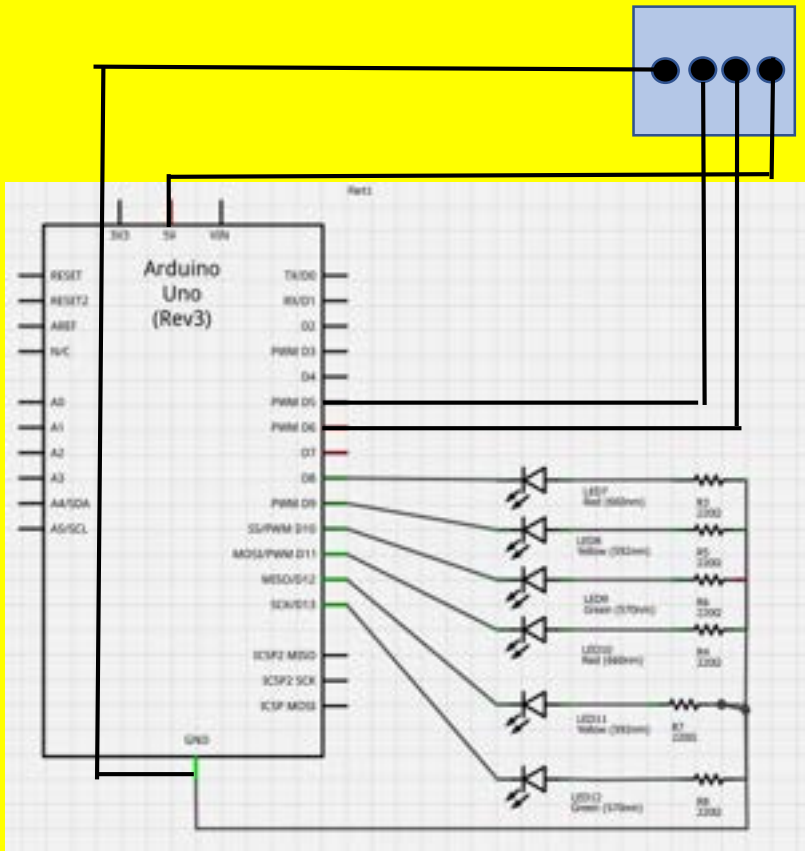
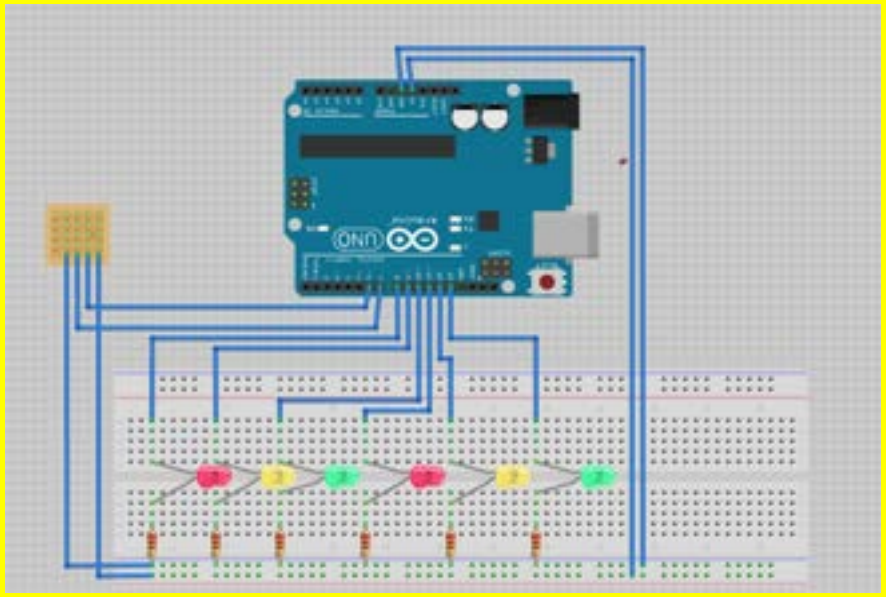


Figure 2.1.1 : Hardware Connection For Obstacle Sensor Using Adruino And HCSR04



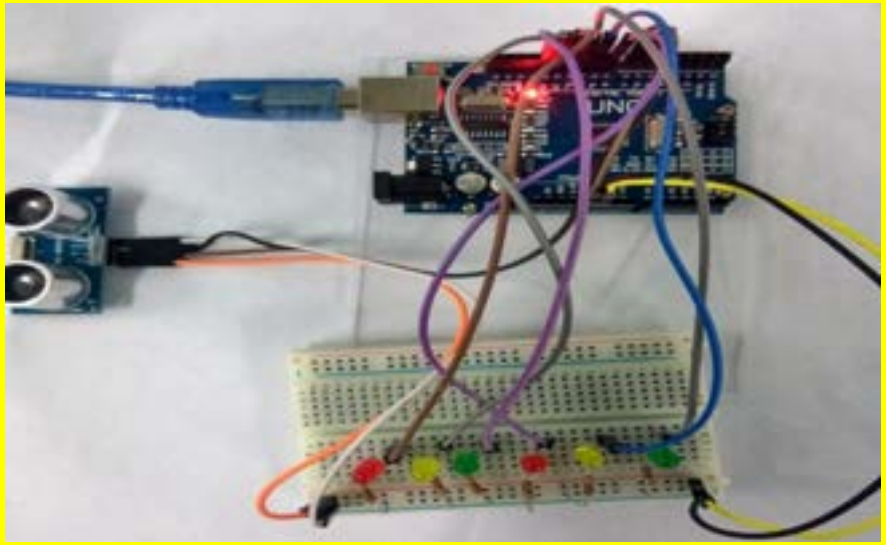


Figure 2.1.2 : Output For Obstacle Sensor Using Adruino And HCSR04

```
int trigPin = 7;    //Trig - green Jumper
int echoPin = 6;    //Echo - yellow Jumper
long cm,duration;
int ledgreen=10;
int ledyellow=9;
int ledred=8;
int ledgreen2=13;
int ledyellow2=12;
int ledred2=11;

void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledyellow, OUTPUT);
  pinMode(ledgreen, OUTPUT);
  pinMode(ledred, OUTPUT);
  pinMode(ledyellow2, OUTPUT);
  pinMode(ledgreen2, OUTPUT);
  pinMode(ledred2, OUTPUT);

}
```

```
void loop()
{

  // The sensor is triggered by a HIGH pulse of 10 or more
microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH
pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  cm = (duration/2) / 29.1;

if (cm > 30) {
digitalWrite(ledgreen,LOW);
digitalWrite(ledyellow,LOW);
digitalWrite(ledred,LOW);
digitalWrite(ledgreen2,LOW);
digitalWrite(ledyellow2,LOW);
digitalWrite(ledred2,LOW);
delay(500);
}
```



```
else if (cm > 25) {
digitalWrite(ledgreen2,HIGH);
digitalWrite(ledyellow2,LOW);
digitalWrite(ledred2,LOW);
digitalWrite(ledgreen,LOW);
digitalWrite(ledyellow,LOW);
digitalWrite(ledred,LOW);
delay(500);

}

else if (cm >20) {
digitalWrite(ledgreen2,LOW);
digitalWrite(ledyellow2,HIGH);
digitalWrite(ledred2,LOW);
digitalWrite(ledgreen,LOW);
digitalWrite(ledyellow,LOW);
digitalWrite(ledred,LOW);
delay(500);
}

else if (cm > 15) {
digitalWrite(ledgreen2,LOW);
digitalWrite(ledyellow2,LOW);
digitalWrite(ledred2,HIGH);
digitalWrite(ledgreen,LOW);
digitalWrite(ledyellow,LOW);
digitalWrite(ledred,LOW);
delay(500);
}
```

```
else if (cm > 10) {
digitalWrite(ledgreen2,LOW);
digitalWrite(ledyellow2,LOW);
digitalWrite(ledred2,LOW);
digitalWrite(ledgreen,HIGH);
digitalWrite(ledyellow,LOW);
digitalWrite(ledred,LOW);
delay(500);

}

else if (cm >5) {
digitalWrite(ledgreen2,LOW);
digitalWrite(ledyellow2,LOW);
digitalWrite(ledred2,LOW);
digitalWrite(ledgreen,LOW);
digitalWrite(ledyellow,HIGH);
digitalWrite(ledred,LOW);
delay(500);
}

else  {
digitalWrite(ledgreen2,LOW);
digitalWrite(ledyellow2,LOW);
digitalWrite(ledred2,LOW);
digitalWrite(ledgreen,LOW);
digitalWrite(ledyellow,LOW);
digitalWrite(ledred,HIGH);
delay(500);
}
}
```


2. Intermediate Projects
2.2 Seven segment Display

In this circuit and code the seven segment display is controlled by the Adruino. The pins are initialized to output by using the ddrd command(register operations). And, the pins are switched on and off by using the portd command.

In this project, 7 segment can display any character desired by the user. A 7 Segment LED Display can be drive by an Adruino. The user can program the circuit so that it displayed numerals 0-9 a second apart from each other. This project simply show go over (again) how it display any character (which can be displayed) on a 7 segment LED display.

To do thisuser should go over the internal makeup of a 7 segment LED display. The display is a device that is made up of 8 individual LEDs, including the decimal point at the bottom. Depending on which LED is lit decides what type of character will be shown.
As an example, look at the numbers shown below.
To understand how this program works, let's first look at the schematic makeup of a 7 segment LED display.

As an example, look at the numbers shown below. To understand how this program works, the schematic makeup of a 7 segment LED display.

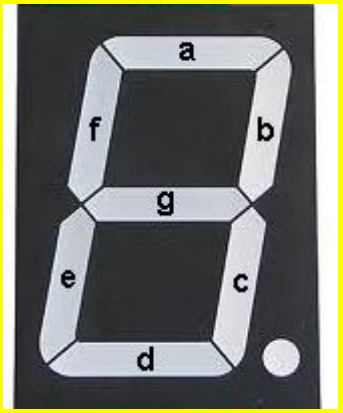


Figure 2.2.1 : 7 Segmen Display Schematic Diagram

The LED display, again, is made up of 8 individual LEDs, as shown above. The LEDs go in order of the alphabetical characters which you see. From first to last, the LEDs from 1-8 are a, b, c, d, e, f, g.
The decimal point would be the last pin.
To create an A, we would have to light LEDs, a,b,c,e,f,g. Thus to create it in code, it would be represented by B11101110.

The full list of all the alphabetical characters with their corresponding code we can display on the segment display is shown in the table below.

Alphabetical Character	Representation in Code
A	B11101110
b	B00111110
C	B10011100
c	B00011010
d	B01111010
E	B10011110
F	B10001110
H	B01101110
h	B00101110
L	B00011100
I	B01100000
O	B11111100
o	B00111010
P	B11001110
S	B10110110

Figure 2.2.2 : 7 Segmen Display Alphabetical characters Reprensentation In Code

The components require for this project are :

Adruino UNO
220 ohm resistor
Jumper wires
breadboard
Common Cathode 7 Segment LED Display

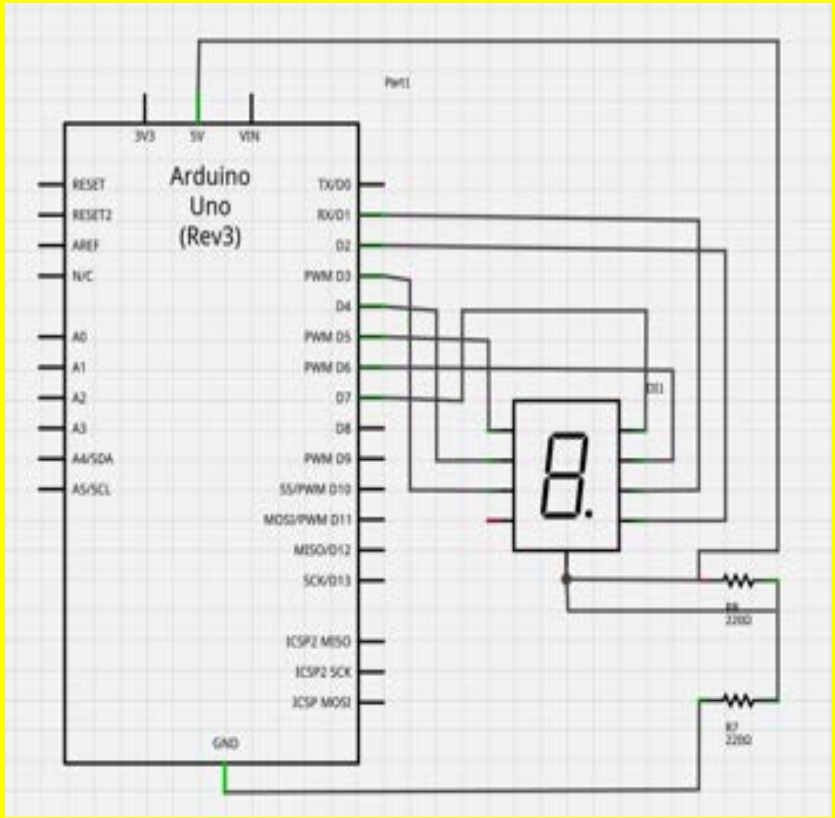
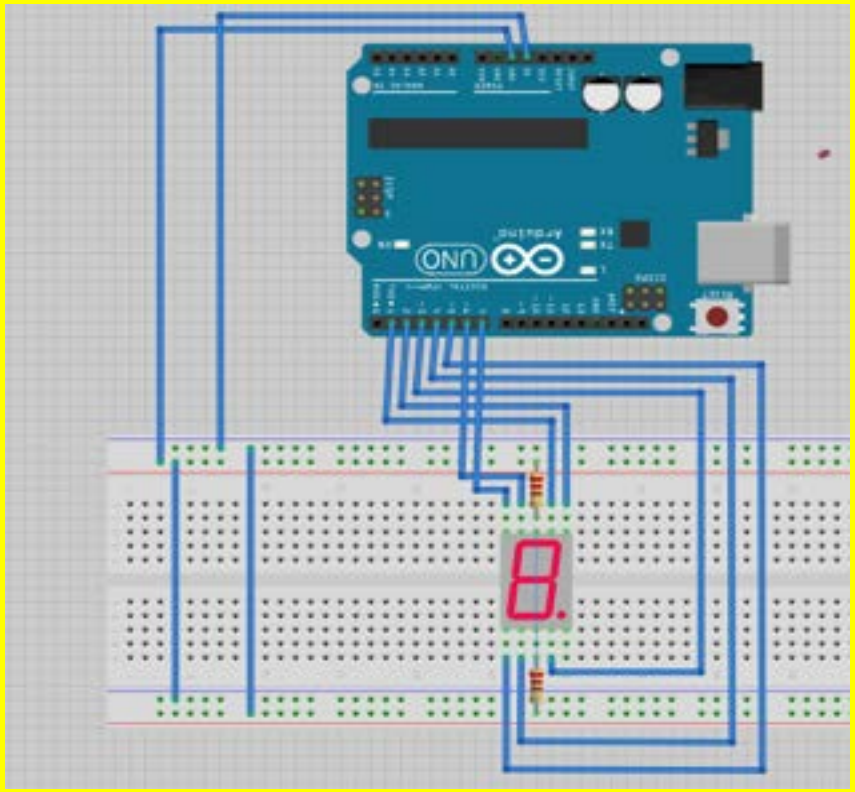


Figure 2.2.3 : Hardware Connection For 7 Segment Display



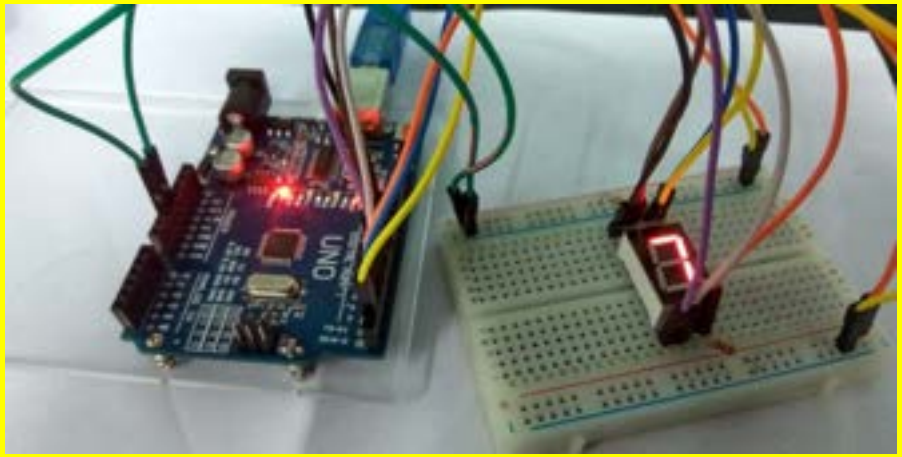


Figure 2.2.3 : Output For 7 Segment Display

```
int mynumbers[ ] = {
// B00000000, //
 B01111111, // 0
 B00001101, // 1
 B10110111, // 2
 B10011111, // 3
 B11001101, // 4
 B11011011, // 5
 B11111011, // 6
 B00001111, // 7
 B11111111, // 8
 B11011111, // 9


};
void setup()
{


 }
 void loop()
 {
 int i;
 DDRD = B11111111;
 for(i=0; i< 10; i++)
{
 PORTD = mynumbers[ i ];
delay(1000);
}
 }
```

2. Intermediate Projects
2.3 Infrared Barrier Module

The sensor module light is adaptable to the environment. It has a pair of infrared transmitting and receiving tube. Tube infrared emit a certain frequency, when detecting direction meet with obstacles (reflecting surface) it will reflected infrared receiving tube. After the comparator circuit processing, green indicator will light up, at the same time signal output interface to output digital signal (a low level signal). The sensor detection range can be through the potentiometer to adjust and have small interference. It is easy to assemble and easy to use. The applications of the Infrared Barrier Module are :

i) Robot obstacle avoidance
ii) Obstacle avoidance car
iii) Line count
iv) Black and white line tracking

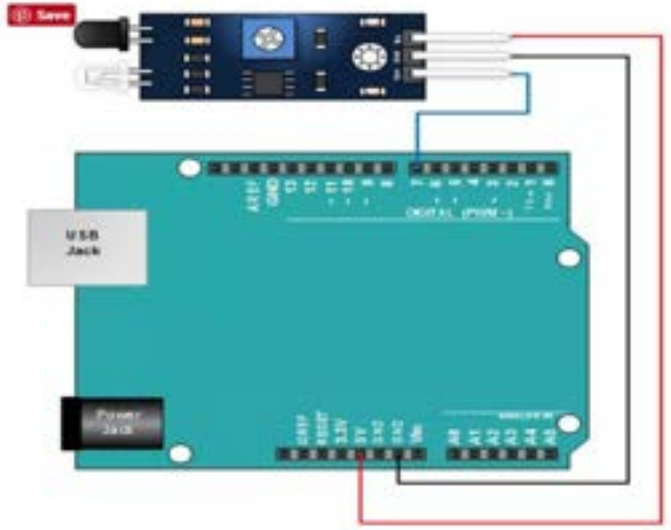
Technical Specifications
Working voltage: 3.3–5 V
Effective range: 2–30 cm (depends on object reflectivity)
Potentiometer adjustment direction: clockwise (detection distance increase), counterclockwise (detection distance decrease)
Dimensions: 47mm L × 19mm W x 80mm H
Weight: 1.66 oz (47 g)

Figure 2.3.1 : Hardware Connection For Infrared Barrier Module

Connecting mode: VCC-VCC, GND-GND, OUT-IO
Module interface specification:
1. VCC port can connect to voltage between 3.3 V to 5 V converters
2. Ground port can connect to external GND
3. Output port can connect to IO port directly or can drive 5 V relay

When power on, the red power indicates light lit. When the module detects obstacles in front of the signal, the green indicator on the circuit board light level and at the same time the OUT port output have low level signal, the detection module from 2~30cm at 35° detection angle. Sensors is active infrared reflection detection, therefore the reflectivity and shape of the target are the key of the detection range. White is a minimum detection range while black is maximum detection range. Small area of the object distance will result large distance. The components require for this project are :

Adruino UNO
220 ohm resistor
hook-up wires
breadboard
SN-IRB-MOD: Infrared Barrier Module



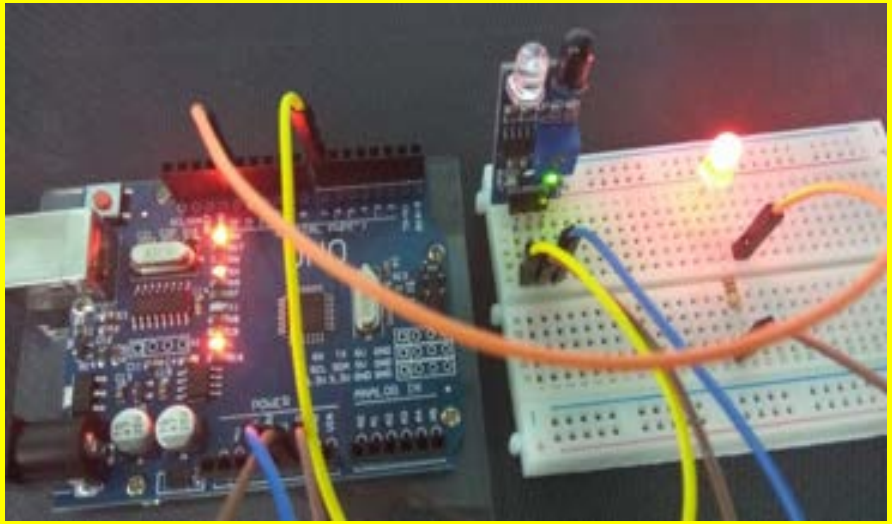


Figure 2.3.2 : Output For Infrared Barrier Module

The Code (on-off One LED)

```
int LED=13; // untuk LED pin 13
int isObstaclePin = 7; // untuk pin input bagi infrared
int isObstacle = HIGH; // infrared high

void setup() {

pinMode(LED,OUTPUT);
pinMode(isObstaclePin,INPUT);
Serial.begin(9600);
}

void loop() {

isObstacle = digitalRead (isObstaclePin);
if (isObstacle == HIGH)
{
  Serial.println("OBSTACLE!!, ABSTACLE!!");
  digitalWrite(LED,HIGH);
}
else

{
  Serial.println("clear");
  digitalWrite(LED,LOW);
}
delay(300);
}
```


The Code (on-off TWO LED)

```
int LED=13; // untuk LED pin 13
int LED2=12;
int isObstaclePin = 7; // untuk pin input bagi infrared
int isObstacle = HIGH; // infrared high

void setup() {

pinMode(LED,OUTPUT);
pinMode(LED2,OUTPUT);
pinMode(isObstaclePin,INPUT);
Serial.begin(9600);
}

void loop() {

isObstacle = digitalRead (isObstaclePin);
if (isObstacle == HIGH)
{
  Serial.println("OBSTACLE!!, ABSTACLE!!"); //paparan pada screen
  digitalWrite(LED,HIGH);
  digitalWrite(LED2,LOW);
}
else

{
  Serial.println("clear"); // paparan pada screen
  digitalWrite(LED,LOW);
  digitalWrite(LED2,HIGH);
}
delay(300);
```

2. Intermediate Projects

2.4 Adruino Infrared Collision Avoidance & Motor

This project is controlling a spinning motor. This requires the use of a transistor, which can switch a larger amount of current than the RedBoard or Adruino Uno R3. When using a transistor the maximum specs are high enough for the power. The transistor that are using for this circuit is rated at 40V max and 200 milliamps max, which is suitable for toy motor. When the motor is spinning and suddenly turned off, the magnetic field inside it collapses, generating a voltage spike. This can damage the transistor. To prevent this, we use a “flyback diode”, which diverts the voltage spike around the transistor. The components require for this project are :

1x Breadboard
1x RedBoard or Adruino Uno
1x Motor
1x 330Ω Resistor
1x NPN transistor
1x Diode 1N4148
6x Jumper Wires

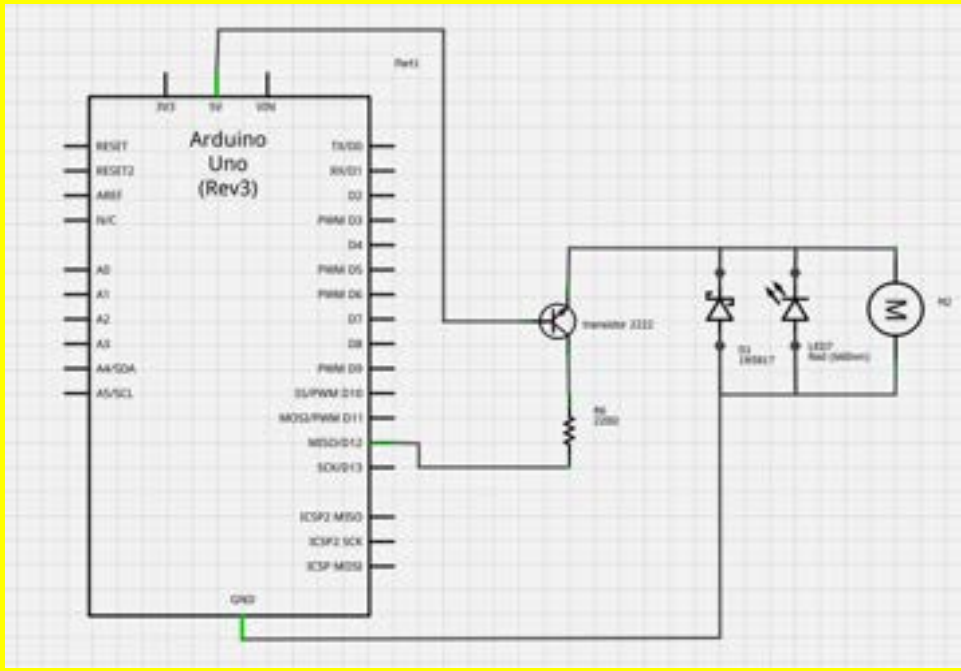
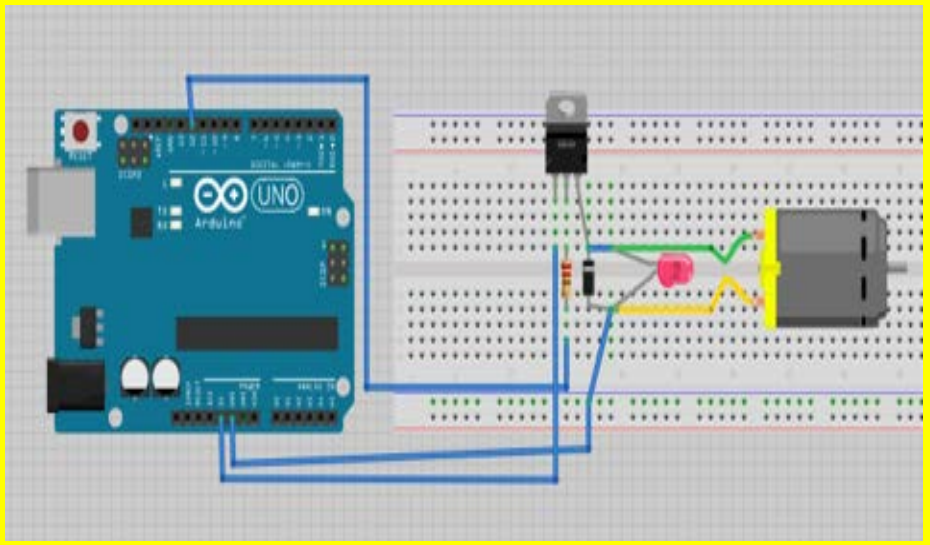


Figure 2.4.1 : Hardware Connection For Adruino Infrared Collision Avoidance & Motor



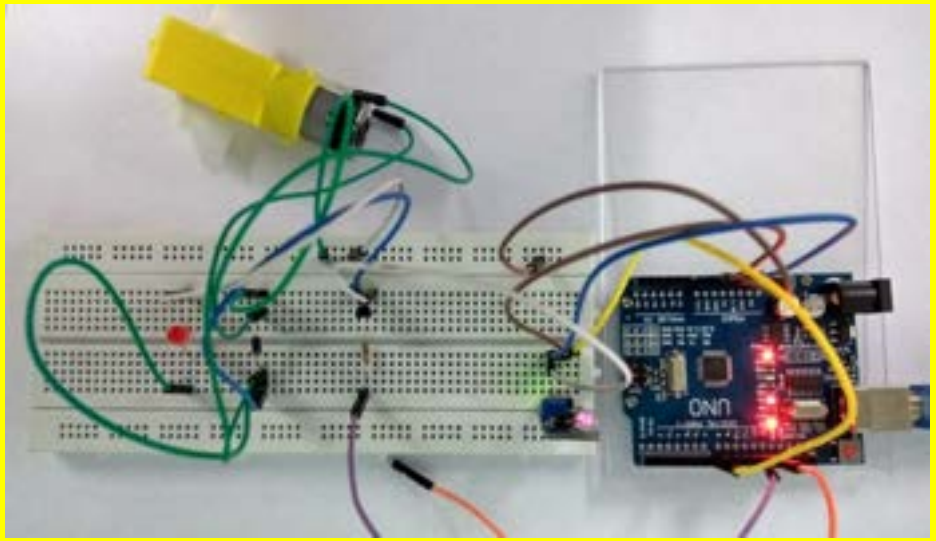


Figure 2.4.2 : Output For Adruino Infrared Collision Avoidance & Motor

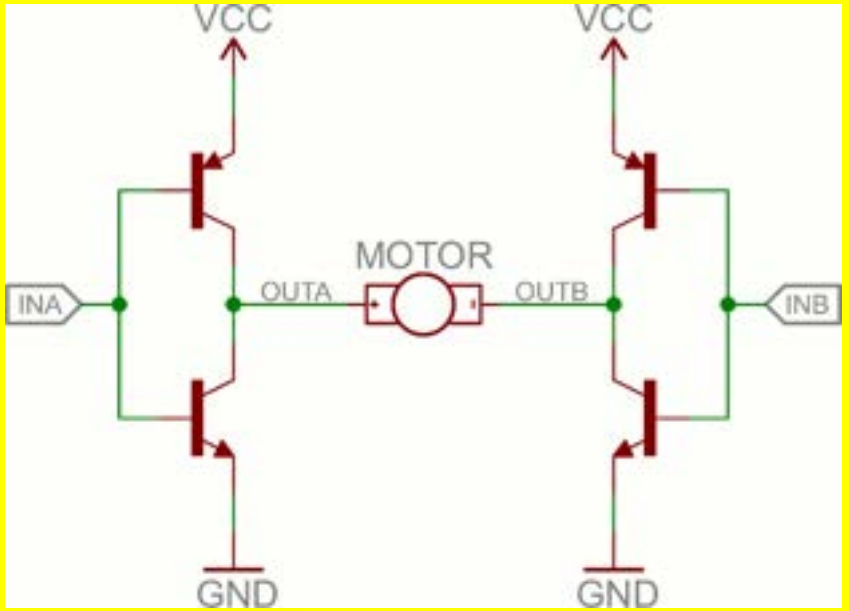


Figure 2.4.3 : H-Brige Connection For Adruino Infrared Collision Avoidance & Motor

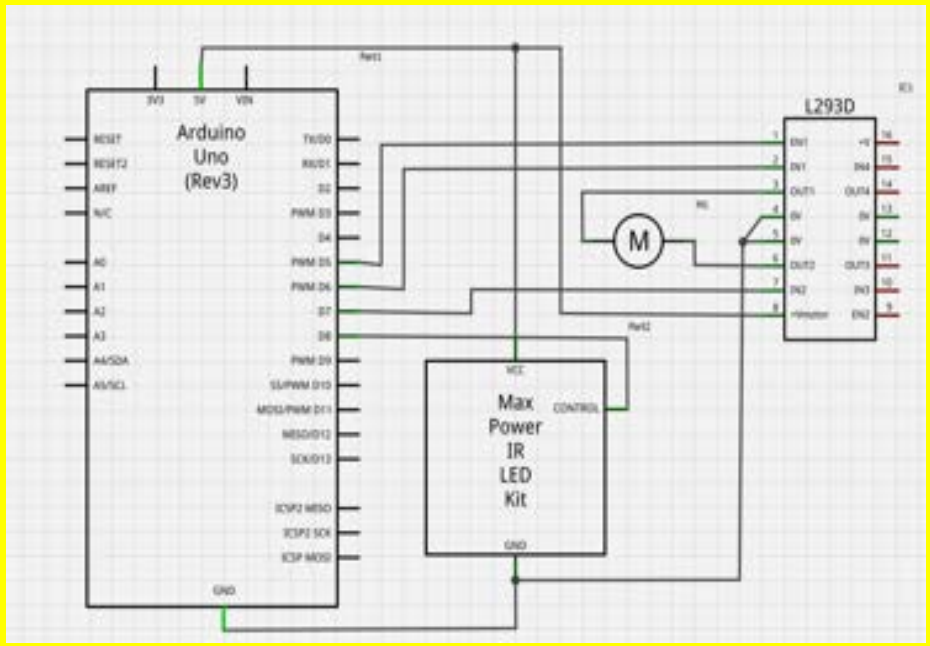
An H-bridge is a transistor-based circuit capable of driving motors both clockwise and counter-clockwise. It's an incredibly popular circuit – the driving force behind countless robots that must be able to move both forward and backward. Fundamentally, an H-bridge is a combination of four transistors with two inputs lines and two outputs. There's usually quite a bit more to a well-designed H-bridge including flyback diodes, base resistors and Schmidt triggers. If both inputs are the same voltage, the outputs to the motor will be the same voltage, and the motor won't be able to spin. But if the two inputs are opposite, the motor will spin in one direction or the other.

```
int LED=12; //to motor
int isObstaclePin = 7; // untuk pin input bagi infrared
int isObstacle = HIGH; // infrared high

void setup() {

pinMode(LED,OUTPUT);
pinMode(isObstaclePin,INPUT);
Serial.begin(9600);
}

void loop() {

isObstacle = digitalRead (isObstaclePin);
if (isObstacle == HIGH)
{
  Serial.println("OBSTACLE!!, ABSTACLE!!"); //paparan pada
screen
  digitalWrite(LED,HIGH);
}
else

{
  Serial.println("clear"); // paparan pada screen
  digitalWrite(LED,LOW);

}
delay(300);
}
```


2. Intermediate Projects
2.5 Adruino L293D DC motors control

The L293D Chip is a Dual H-Bridge Motor Driver for DC or Step motors. It can handle two Motors or one step motor. It can power motors until 36V and 600mA of steady current with Max of 1.2A. The chip is easy to use and takes little space. This project will use to power 1 or 2 DC Motors. The components require for this project are :

L293D chip
Adruino (I'm using UNO)
2x DC Motors
4x AA batteries and holder



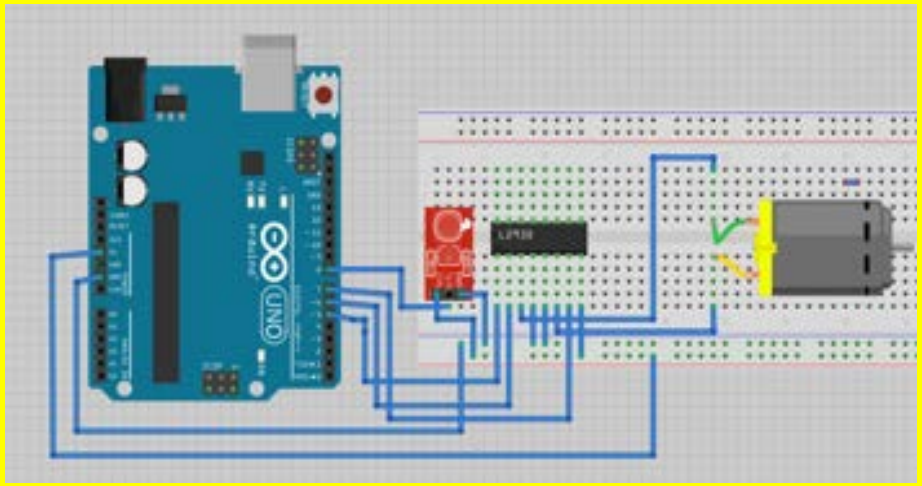


Figure 2.5.1 : Hardware Connection For Adruino L293D DC motors control


```
//Testing the DC Motors with
// L293D
 //Define Pins
//Motor A
int enableA = 5;
int MotorA1 = 6;
int MotorA2 = 7;
int isObstaclePin = 8; // untuk pin input bagi infrared
int isObstacle = HIGH; // infrared high
 void setup() {
   Serial.begin (9600);
  //configure pin modes
  pinMode (enableA, OUTPUT);
  pinMode (MotorA1, OUTPUT);
  pinMode (MotorA2, OUTPUT);
  pinMode(isObstaclePin,INPUT);
  }
 void loop() {
isObstacle = digitalRead (isObstaclePin);
if (isObstacle == HIGH)
{
Serial.println ("Enabling Motors");
  digitalWrite (enableA, HIGH);
  delay (500);
  //do something
   Serial.println ("Motion Forward");
  digitalWrite (MotorA1, LOW);
  digitalWrite (MotorA2, HIGH);
}
```

```
else
{
//reverse
delay(500);
Serial.println ("Enabling Motors");
  digitalWrite (enableA, HIGH);
  digitalWrite (MotorA1,HIGH);
  digitalWrite (MotorA2,LOW);
}
}
```


2. Intermediate Projects

2.6 Adruino HC-06 Serial Port Bluetooth Module

The HC-06 is used to add Bluetooth functionality to project devices & to replace serial port communication. The features for the Adruino HC-06 Serial Port Bluetooth Module are:

Operating voltage: 3.6 to 6Vdc.
Operating current: 30mA.
Communication distance: 10 meters.
Working mode: slave.
Passcode: 1234.
Full-duplex serial interface.
Communication format: Only supports 8 data bits, 1 stop bit, no parity.
Default baud rate: 9600 bps.
Dimension: 16 x 38 mm.
Weight: 5g.

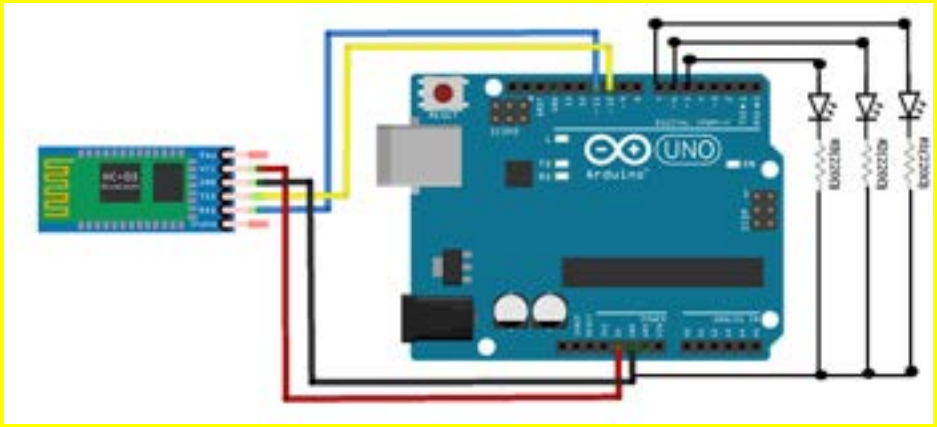
Status LED indicator:
i) LED flashes = Bluetooth is not connected.
ii) LED on = Bluetooth is connected.

Pinout:
i) RXD = Receive data (In) - Connect this pin to Adruino board's TXD pin.
ii) TXD = Transmit data (Out) - Connect this pin to Adruino board's RXD pin.
iii) GND = Ground
iv) VCC = 3.6 to 6V

The components require for this project are :

L293D chip
Adruino (I'm using UNO)
3x 220 ohm resistor
hook-up wires
breadboard
red LED
yellow LED
green LED
HC-06 Serial Port Bluetooth Module

Figure 2.6.1 : Adruino HC-06 Serial Port Bluetooth Module



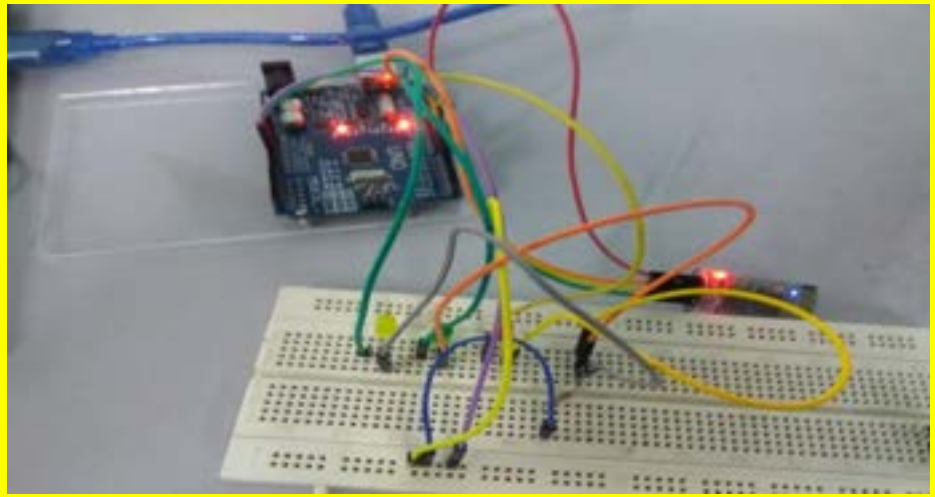


Figure 2.6.2 : Hardware Connection For Adruino HC-06 Serial Port Bluetooth Module

```
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11);
// creates a "virtual" serial port/UART
// connect BT module TX to D10
// connect BT module RX to D11
// connect BT Vcc to 5V, GND to GND
void setup()
{
  // set digital pin to control as an output
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
  // set the data rate for the SoftwareSerial port
  BT.begin(9600);
  // Send test message to other device
  BT.println("Hello from Adruino");
}
char a; // stores incoming character from other device
void loop()
{
  if (BT.available())
  // if text arrived in from BT serial...
  {
    a=(BT.read());
    if (a=='1')
    {
      digitalWrite(7, HIGH);
      BT.println("LED on");
    }
    if (a=='2')
    {
      digitalWrite(7, LOW);
      BT.println("LED off");
    }
```

```
if (a=='3')
  {
    digitalWrite(6, HIGH);
    BT.println("LED on");
  }

  if (a=='4')
  {
    digitalWrite(6, LOW);
    BT.println("LED off");
  }

   if (a=='5')
  {
    digitalWrite(5, HIGH);
    BT.println("LED on");
  }
  if (a=='6')
  {
    digitalWrite(5, LOW);
    BT.println("LED off");
  }

  if (a=='?')
  {
    BT.println("Send '1' to turn LED on");
    BT.println("Send '2' to turn LED on");
  }
  // you can add more "if" statements with other characters to add
more commands
  }
}
```

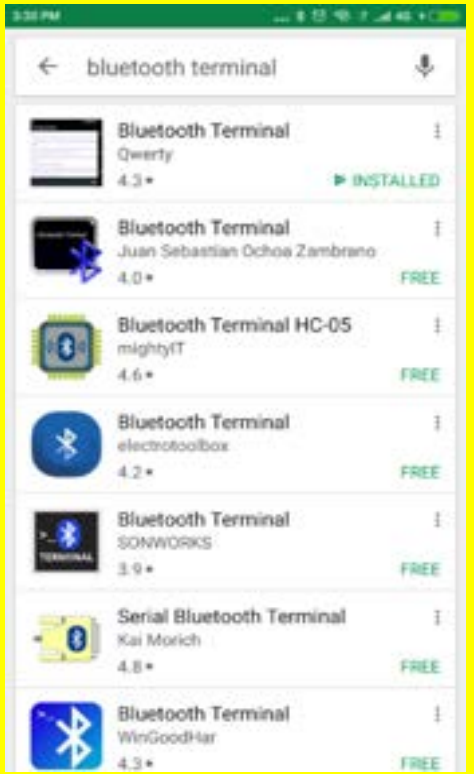



Figure 2.6.3 : Application Bluetooth From Play Store

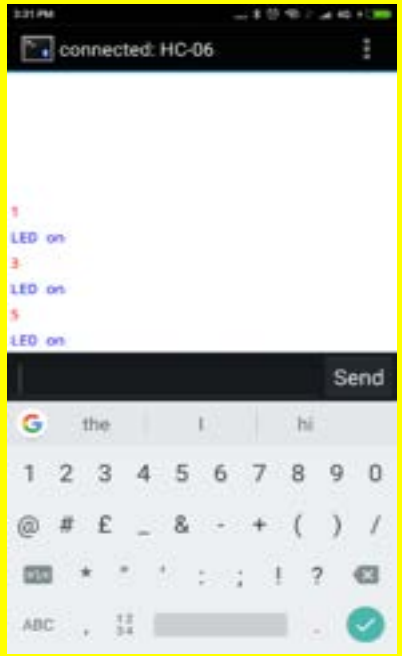


Figure 2.6.4 : Output For Adruino HC-06 Serial Port Bluetooth Module

2. Intermediate Projects

2.7 Adruino L293D DC motors control (2 Motor)

The L293D Chip is a Dual H-Bridge Motor Driver for DC or Step motors. It can handle two Motors or one step motor. It can power motors until 36V and 600mA of steady current – Max of 1.2A. The chip is easy to use and takes little space. This project will control 2 DC Motors. The components require for this project are :

L293D chip
Adruino (I'm using UNO)
2x DC Motors
4x AA batteries and holder

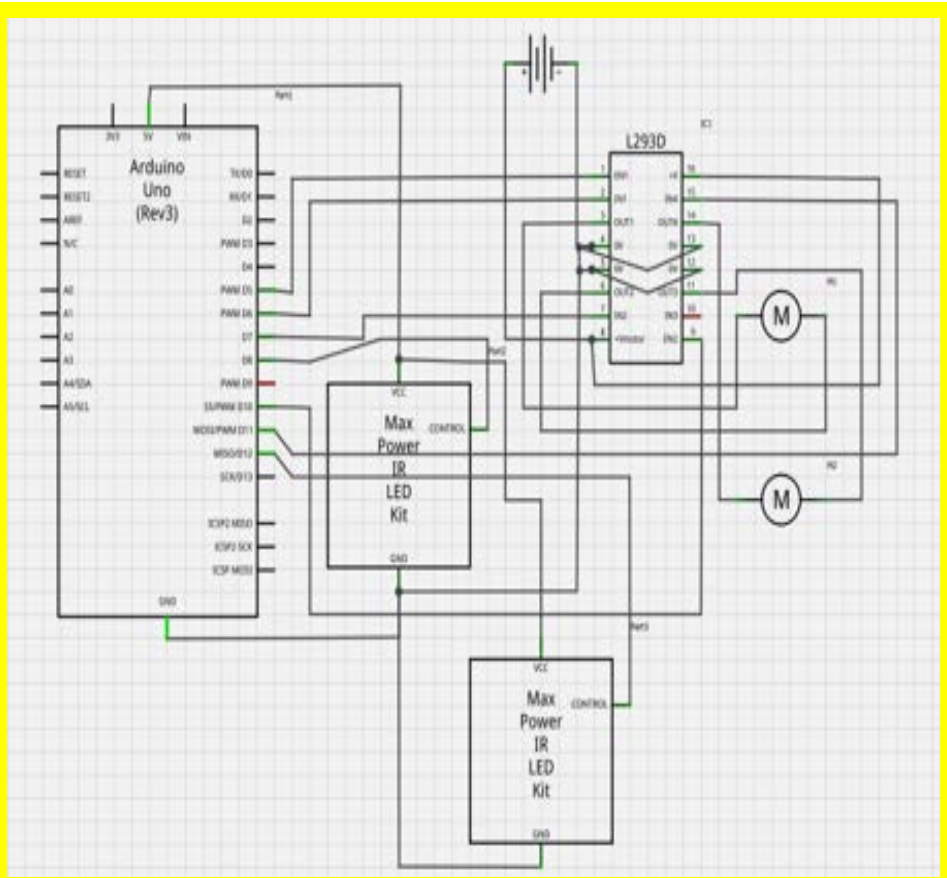
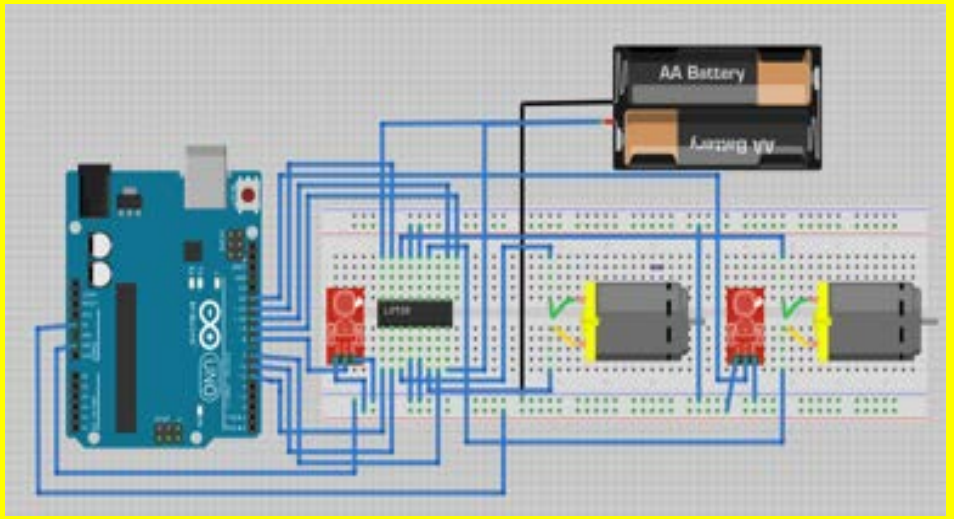


Figure 2.7.1 : Hardware Connection For Adruino L293D DC motors control (2 Motor)



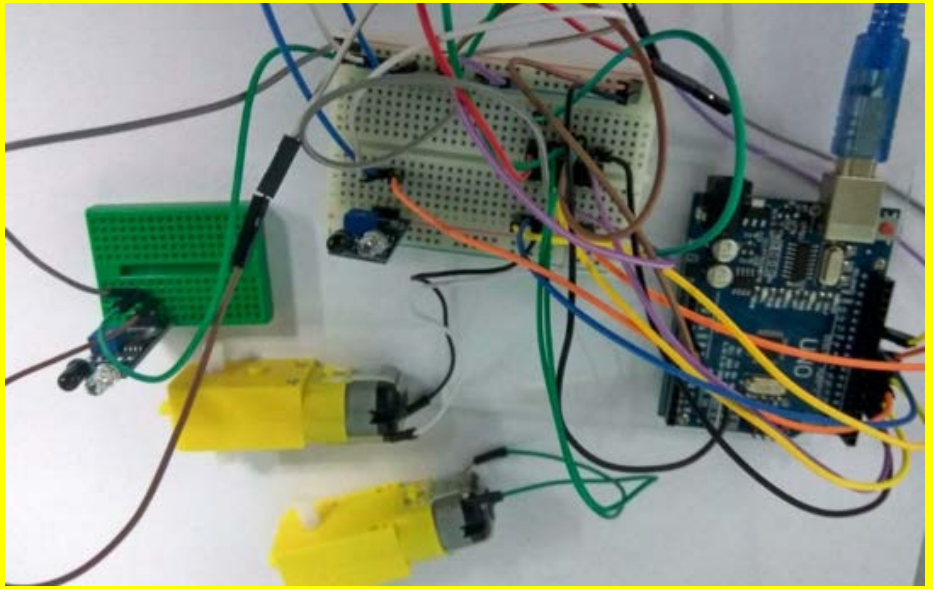


Figure 2.7.2 : Output For Adruino L293D DC motors control (2 Motor)

```
/**
* Bruno Santos, 2013
* feiticeir0@whatgeek.com.pt
* Small code to test DC motors - 2x with a L298 Dual H-Bridge Motor
Driver
* Free to share
**/

//Testing the DC Motors with
// L293D

//Define Pins
//Motor A
int enableA = 5;
int MotorA1 = 6;
int MotorA2 = 7;
int isObstaclePin = 8; // untuk pin input bagi infrared2
int isObstacle = HIGH; // infrared1 high

int enableB = 9;
int MotorB1 = 10;
int MotorB2 = 11;
int isObstaclePin2 = 12; // untuk pin input bagi infrared2
int isObstacle2 = HIGH; // infrared2 high

void setup() {
```

```
 Serial.begin (9600);
  //configure pin modes
  pinMode (enableA, OUTPUT);
  pinMode (MotorA1, OUTPUT);
  pinMode (MotorA2, OUTPUT);
  pinMode(isObstaclePin,INPUT);
  pinMode (enableB, OUTPUT);
  pinMode (MotorB1, OUTPUT);
  pinMode (MotorB2, OUTPUT);
  pinMode(isObstaclePin2,INPUT);

}

void loop() {

isObstacle = digitalRead (isObstaclePin);
isObstacle2 = digitalRead (isObstaclePin2);
if ( (isObstacle == HIGH)&&(isObstacle2 == HIGH))
{
Serial.println ("Enabling Motors");
  digitalWrite (enableA, HIGH);
  digitalWrite (enableB, HIGH);
  delay (500);
  //do something

  Serial.println ("Motion Forward");
  digitalWrite (MotorA1, LOW);
  digitalWrite (MotorA2, HIGH);
  digitalWrite (MotorB1, LOW);
  digitalWrite (MotorB2, HIGH);
}
```

```
else

{
//reverse
delay(500);
Serial.println ("Enabling Motors");
  digitalWrite (enableA, HIGH);
  digitalWrite (enableB, HIGH);
  digitalWrite (MotorA1,HIGH);
  digitalWrite (MotorA2,LOW);
  digitalWrite (MotorB1,HIGH);
  digitalWrite (MotorB2,LOW);
}

}
```


2. Intermediate Projects
2.8 Adruino L298D ROBOT CAR

This dual bidirectional motor driver is based on the very popular L298 Dual H-Bridge Motor Driver IC. This module will able to control two motors of up to 2A each in both directions. It is ideal for robotic applications and well suited for connection to a microcontroller requiring just a couple of control lines per motor. This project allow Adruino uno to control two dc motors.

The components require for this project are :

Adruino UNO
Adruino uno
Breadboard
L298 Module
4x AA batteries and holder

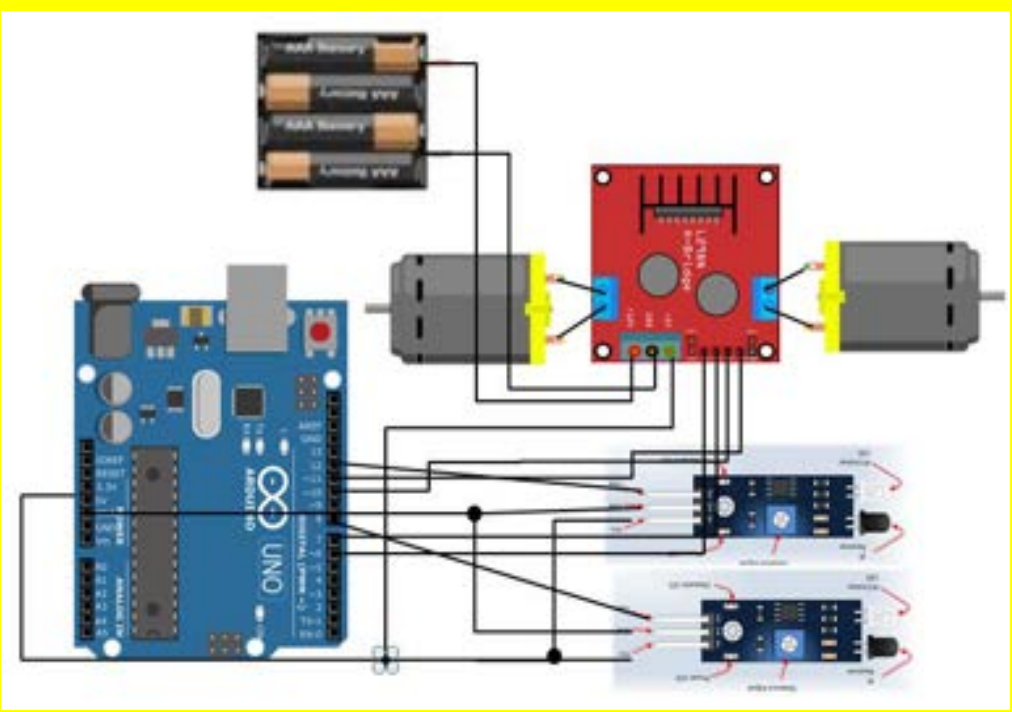


Figure 2.8.1 : Hardware Connection For Adruino L298D ROBOT CAR

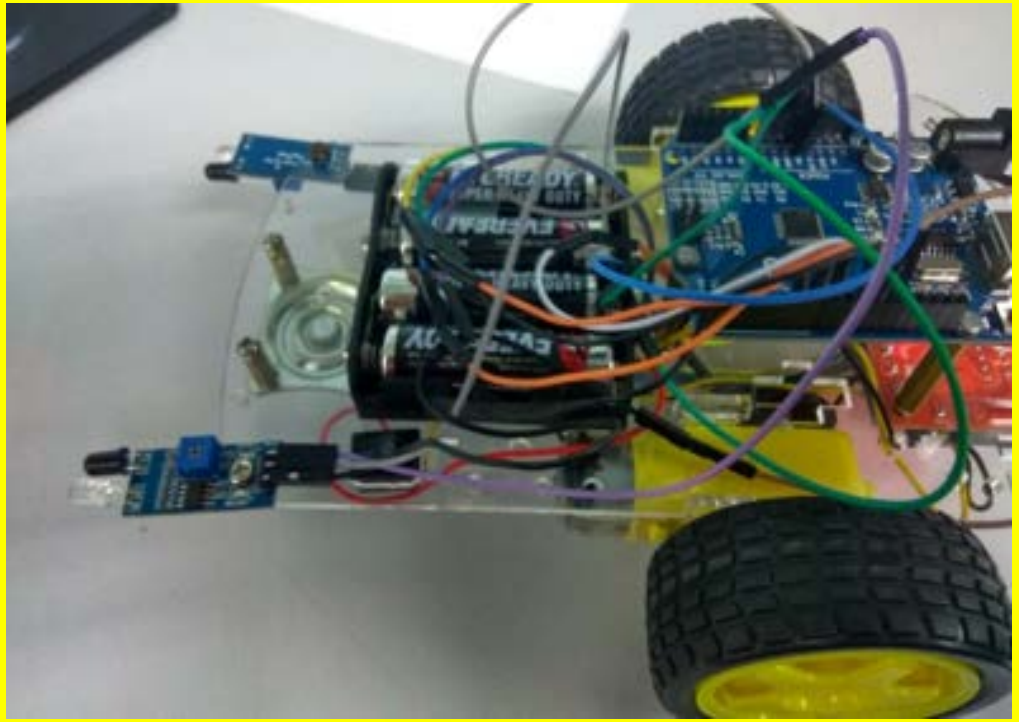


Figure 2.8.2 : Output Adruino L298D ROBOT CAR

```
//L293D
//Motor A
const int motorPin1  = 9;  // Pin 14 of L293
const int motorPin2  = 10;  // Pin 10 of L293
//Motor B
const int motorPin3  = 6; // Pin  7 of L293
const int motorPin4  = 5;  // Pin  2 of L293

//This will run only one time.
void setup(){

   //Set pins as outputs
   pinMode(motorPin1, OUTPUT);
   pinMode(motorPin2, OUTPUT);
   pinMode(motorPin3, OUTPUT);
   pinMode(motorPin4, OUTPUT);

   //Motor Control - Motor A: motorPin1,motorpin2 & Motor B:
motorpin3,motorpin4

   //This code  will turn Motor A clockwise for 2 sec.
   analogWrite(motorPin1, 180);
   analogWrite(motorPin2, 0);
   analogWrite(motorPin3, 180);
   analogWrite(motorPin4, 0);
   delay(5000);
   //This code will turn Motor A counter-clockwise for 2 sec.
   analogWrite(motorPin1, 0);
   analogWrite(motorPin2, 180);
   analogWrite(motorPin3, 0);
   analogWrite(motorPin4, 180);
   delay(5000);
```



```
   //This code will turn Motor B clockwise for 2 sec.
   analogWrite(motorPin1, 0);
   analogWrite(motorPin2, 180);
   analogWrite(motorPin3, 180);
   analogWrite(motorPin4, 0);
   delay(1000);
   //This code will turn Motor B counter-clockwise for 2 sec.
   analogWrite(motorPin1, 180);
   analogWrite(motorPin2, 0);
   analogWrite(motorPin3, 0);
   analogWrite(motorPin4, 180);
   delay(1000);

   //And this code will stop motors
   analogWrite(motorPin1, 0);
   analogWrite(motorPin2, 0);
   analogWrite(motorPin3, 0);
   analogWrite(motorPin4, 0);

}

void loop(){


}
```

3. Advance Projects
3.1 Pin selector

Common Widget Settings

This is one of the main parameters need to be set. It defines which pin to control or to read from.

Figure 3.1.1 : Pin Selector

Digital Pins - represent physical Digital IO pins on your hardware.
PWM-enabled pins are marked with the ~ symbol
Analog Pins - represent physical Analog IO pins on your hardware
Virtual Pins - have no physical representation. It used for any data transfer between Blynk App and project hardware. Virtual Pins are available at http://docs.blynk.cc/#blynk-main-operations-virtual-pins.

3. Advance Projects
3.2 Data Mapping

Common Widget Settings

Data mapping is to map incoming values to specific range by using mapping button :

Figure 3.2.1 : Data Mapping

If sensor sends values from 0 to 1023, but user want to display values in a range 0 to 100 in the application. With data mapping enabled, incoming value 1023 will be mapped to 100

3. Advance Projects
3.3 Split/Merge

Some of the Widgets can send more than one value. By using this switch, it can control how to send it.

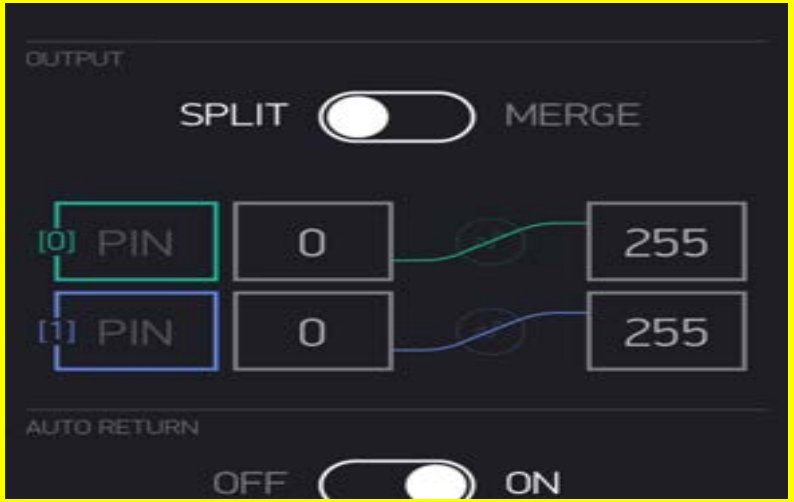


Figure 3.3.1 : Split/Merge

SPLIT: Each of the parameters is sent directly to the Pin on the hardware (e.g D7. No coding is needed.
NOTE: In this mode user are able to send multiple commands from one widget, which can reduce performance of the hardware.
Example: If using a Joystick Widget and it's set to D3 and D4, it will send 2 commands over the Internet:

```
digitalWrite(3, value);
digitalWrite(4, value);
```

Figure 3.3.2 : Split Output

MERGE: When MERGE mode is selected, if user sending just 1 message, consisting of array of values. But user need to parse it on the hardware. This mode can be used with Virtual Pins only. Example: Add a zeRGBa Widget and set it to MERGE mode. Choose Virtual Pin V1.

```
BLYNK_WRITE(V1) // There is a widget that WRITEs data to V1
{
  int r = param[0].asInt(); // get a RED channel value
  int g = param[1].asInt(); // get a GREEN channel value
  int b = param[2].asInt(); // get a BLUE channel value
}
```

Figure 3.3.3 : Merge Output

3. Advance Projects
3.4 Send On Release & Color Gradient

Send On Release

This option is available for most controller widgets and allows you to decrease data traffic on your hardware. For example, when you move joystick widget, commands are continuously streamed to the hardware, during a single joystick move you can send dozens of commands. There are use-cases where it's needed, however creating such a load may cause hardware reset. We recommend enabling **Send On Release** feature for most of the cases, unless you really need instant feedback. This option is enabled by default.

Color gradient

Some display widgets have ability to select gradient. Gradient allows you to colorize your widgets without any coding. At the moment we provide 2 types of gradients :
Warm: Green - Orange - Red;
Cold : Green - Blue - Violet;
Gradient changes color of your widget based on min/max properties. For example, you select warm gradient for your level display widget with min 0 and max 100 value. When value 10 comes to widget it will have green color, when value 50 comes you'll see orange color, when value 80 comes you'll see red color.

3. Advance Projects
3.5 IOT (Blynk)

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, vizualize it and do many other cool things.

There are three major components in the platform:

- Blynk App - allows to you create amazing interfaces for your projects using various widgets we provide.
- Blynk Server - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Now imagine: every time you press a Button in the Blynk app, the message travels to the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a blynk of an eye.

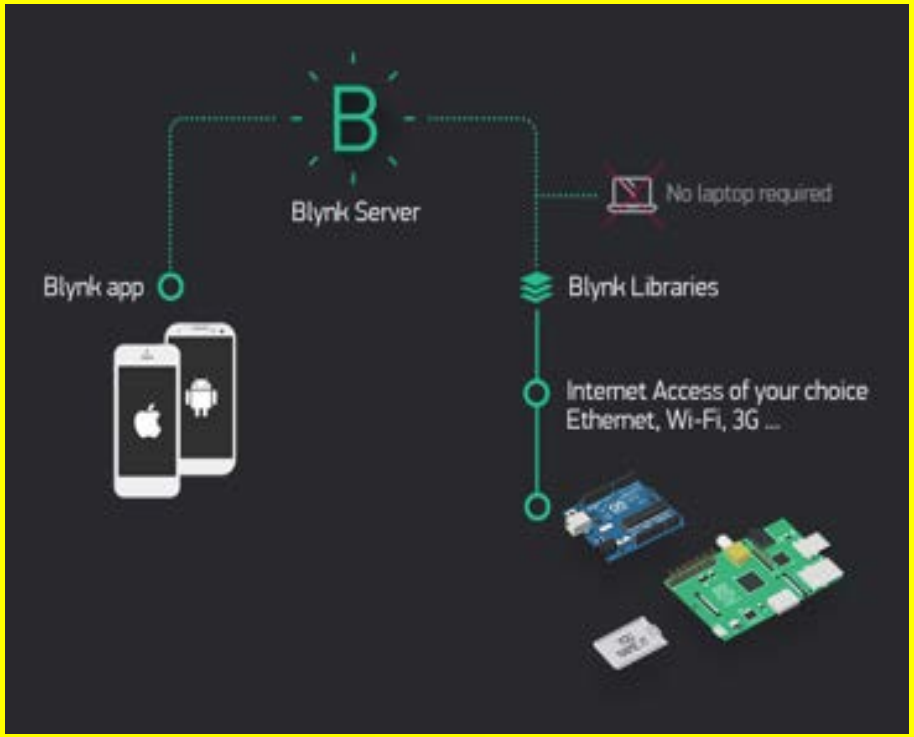


Figure 3.5.1 : IOT (Blynk)

3. Advance Projects
3.6 Installing Blynk Library

lynk library should be installed manually
Follow the instructions:
i. Download **Blynk_Release_vXX.zip** (scroll down to the Downloads section)
ii. Unzip the archive. You will notice that archive contains several folders and several libraries.
iii. Copy all of these libraries to your_sketchbook_folder of Adruino IDE. To find the location of your_sketchbook_folder, go to top menu in Adruino IDE:
Windows: File → Preferences
Mac OS: Adruino → Preferences

The structure of your your_sketchbook_folder should now look like this, (*along with your other sketches if you have them*):
your_sketchbook_folder/libraries/Blynk
your_sketchbook_folder/libraries/BlynkESP8266_Lib
...
your_sketchbook_folder/tools/BlynkUpdater
your_sketchbook_folder/tools/BlynkUsbScript
...

Note that:
all libraries should go to libraries folder
all tools should go to tools folder

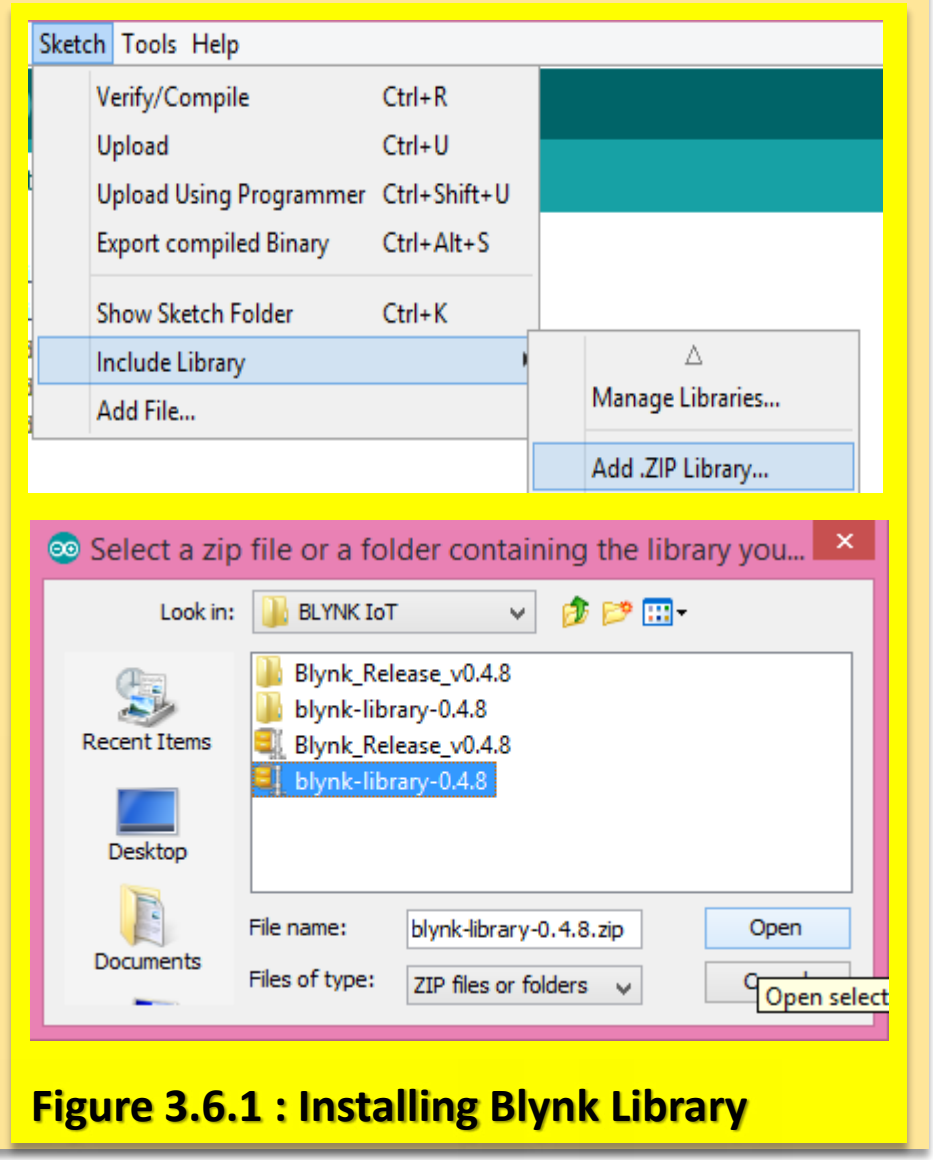


Figure 3.6.1 : Installing Blynk Library

Connect circuit as shown on Figure below :

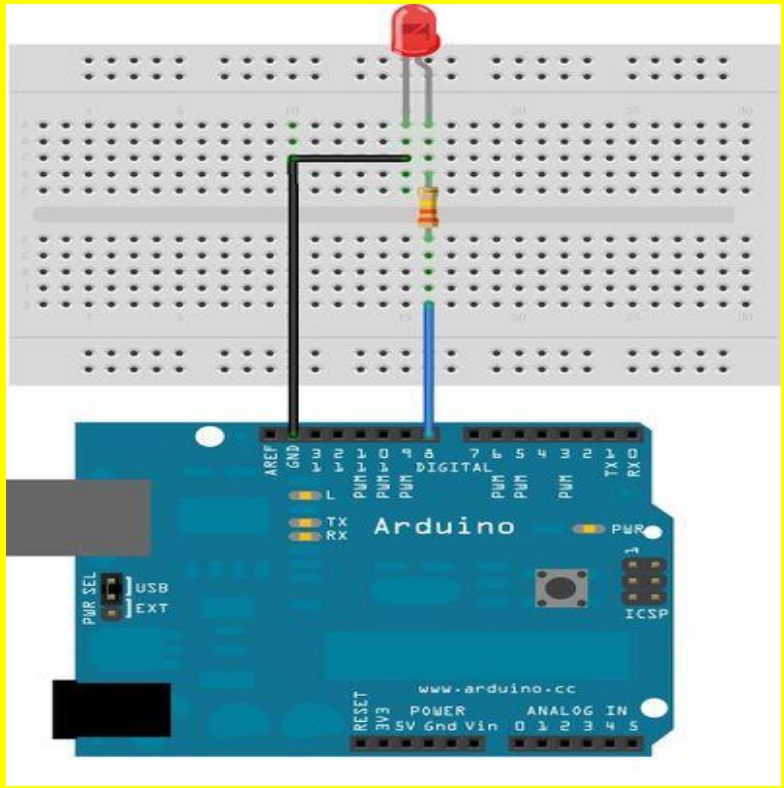


Figure 3.7.1 : Hardware Connection For Blynk App

```
/*************************************************************
  Download latest Blynk library here:
    https://github.com/blynkkk/blynk-library/releases/latest

  Blynk is a platform with iOS and Android apps to control
  Adruino, Raspberry Pi and the likes over the Internet.
  You can easily build graphic interfaces for all your
  projects by simply dragging and dropping widgets.

    Downloads, docs, tutorials: http://www.blynk.cc
    Sketch generator:           http://examples.blynk.cc
    Blynk community:            http://community.blynk.cc
    Social networks:            http://www.fb.com/blynkapp
                                http://twitter.com/blynk_app

  Blynk library is licensed under MIT license
  This example code is in public domain.


 *************************************************************
  =>
  =>        USB HOWTO: http://tiny.cc/BlynkUSB
  =>

  Feel free to apply it to any other example. It's simple!

 *************************************************************/
```

```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT DebugSerial


// You could use a spare Hardware Serial on boards that have it (like Mega)
#include <SoftwareSerial.h>
SoftwareSerial DebugSerial(2, 3); // RX, TX

#include <BlynkSimpleStream.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";


void setup()
{
  // Debug console
  DebugSerial.begin(9600);

  // Blynk will work through Serial
  // Do not read or write this serial manually in your sketch
  Serial.begin(9600);
  Blynk.begin(Serial, auth);
}

void loop()
{
  Blynk.run();
}
```


3. Advance Projects
3.8 Adruino over USB (no shield)

If you don't have any shield and your hardware doesn't have any connectivity, you can still use Blynk – directly over USB :

i. Open Adruino Serial USB example and change Auth Token.

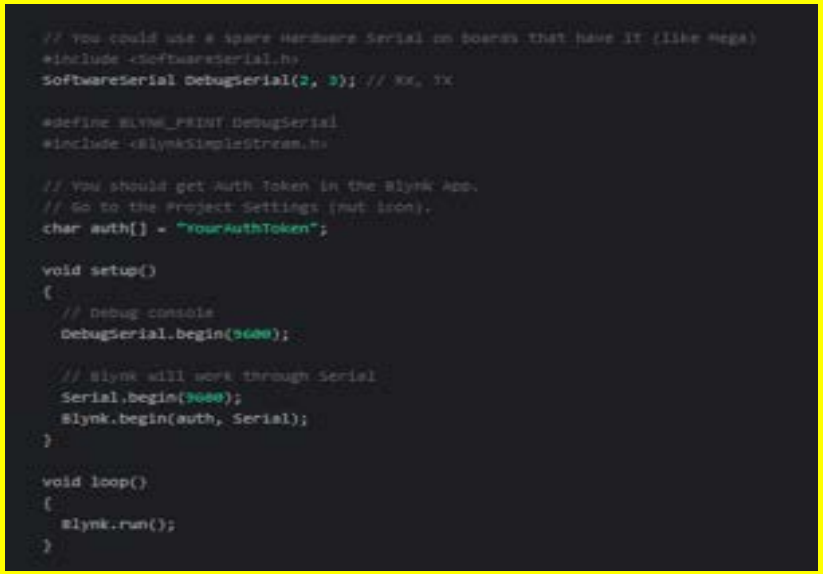


Figure 3.8.1 : Using Blnyk With No Shield

ii. Run the script which is usually located in /scripts folder:
- Windows:My Documents\Adruino\libraries\Blynk\scripts
- Mac User$/Documents/Adruino/libraries/Blynk/scripts

On Windows:
Open cmd.exe

Write your path to blynk-ser.bat folder. For example:
cd C:\blynk-library-0.3.1\blynk-library-0.3.1\scripts

Run blynk-ser.bat file. For example : blynk-ser.bat -c COM4 (where COM4 is port with your Adruino)
And press “Enter”, press “Enter” and press “Enter”

```
C:\Users\USER\Documents\Arduino\libraries\Blynk\scripts>blynk-ser.bat -c COM7
Connecting device at COM7 to blynk-cloud.com:8442...
OpenC0C("\\.\COM7", baud=9600, data=8, parity=no, stop=1) - OK
Connect("blynk-cloud.com", "8442") - OK
InOut() START
DSR is OFF
EVENT_CLOSE
InOut() - STOP
Disconnect() - OK
Connect("blynk-cloud.com", "8442") - OK
InOut() START
DSR is OFF
```

Figure 3.8.2 : Command Prompt For Blynk

Or user can use this method :

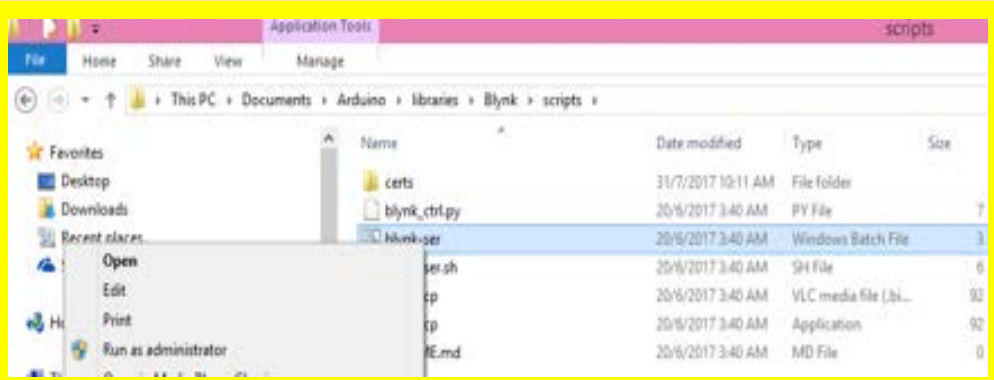


Figure 3.8.3 : Other method For Blynk

Figure 3.8.4 : Appl Blynk

3. Advance Projects
3.9 Adruino/USB/simple Time

Timer
Timer triggers actions at a specific time. Even if smartphone is offline. Start time sends 1 (HIGH). Stop time sends 0 (LOW). Recent Android version also has improved Timer within Eventor widget. With Eventor Time Event you can assign multiple timers on same pin, send any string/number, select days and timezone. It is recommended to use Eventor over Timer widget. However Timer widget is still suitable for simple timer events.

Figure 3.9.1 : Simple Timer

```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT SwSerial

 #include <SoftwareSerial.h>
SoftwareSerial SwSerial(10, 11); // RX, TX

#include <BlynkSimpleStream.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

BLYNK_WRITE(V1) {
  long startTimeInSecs = param[0].asLong();
  SwSerial.println(startTimeInSecs);
  SwSerial.println();
}

void setup()
{
  // Debug console
  SwSerial.begin(9600);

  // Blynk will work through Serial
  // Do not read or write this serial manually in your sketch
  Serial.begin(9600);
  Blynk.begin(Serial, auth);
}

void loop()
{
  Blynk.run();
}
```


3. Advance Projects

3.10 Adruino ESP8266 Remote Serial Port WIFI Transceiver Wireless Module

.

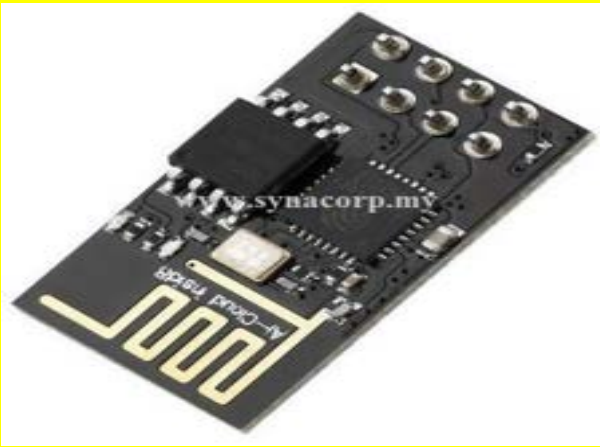


Figure 3.10.1 : WIFI Transceiver Wireless Module

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Adruino device and get about as much WiFi-ability as a WiFi Shield offers.

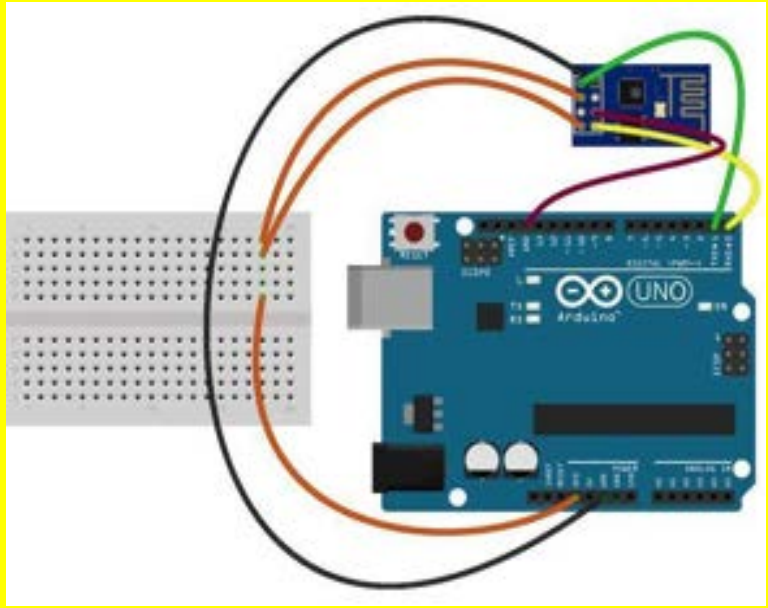
This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module which is designed to occupy minimal PCB area.

Specification:
- 32-bit RISC CPU at 80 MHz.
- External Flash 512 KiB to 4 MiB
- 802.11 b/g/n protocol
- WiFi Direct (P2P) and soft-AP
- Integrated Protocol stack.
- 16 GPIO pins (not available on all models).
- 1 10 bit ADC (= analog port with 1024 values)
- SPI, I2C
- Vcc and logical levels differ between 1.6-3.3V, some accept higher voltages through a voltage regulator or level shifters, so check the specks of your module.

Step 1:
To ensure that ESP 8266 can function well, we must firmware it into version 0.9.2.4 (Old firmware).

Next, connect the Adruino Uno and upload the “BareMinimum” sketch in Adruino.ide software, this is to ensure that nothing will disturb the flasher software. To find “BareMinimum” sketch, go to “File” > “Examples” > “01.Basics” > “BareMinimum”

**Please take note of what COM-Port that Adruino Uno uses. (See your COM >“Tools” >“Port”)
Then close the Adruino.ide after uploading the sketch and unplug the USB.



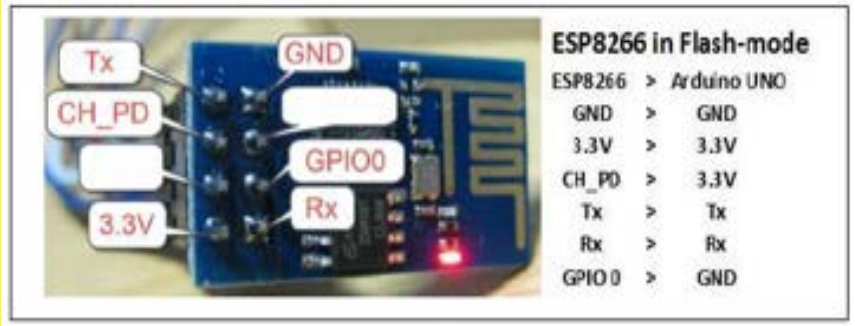


Figure 3.10.1 : Wiring For ESP8266

Step 2:

Download flasher-software and the 0.9.2.4 firmware BIN-file here: http://www.mediafire.com/download/zcw2gy07s2z60y6/ESP8266_flasher_and_0.9.2.4.zip

Unzip the file then run the flasher software (esp8266_flasher). Then Click on the "BIN"-button and find the BIN-file (**v0.9.2.4 AT Firmware-ESPFIX.bin**) in the folder.

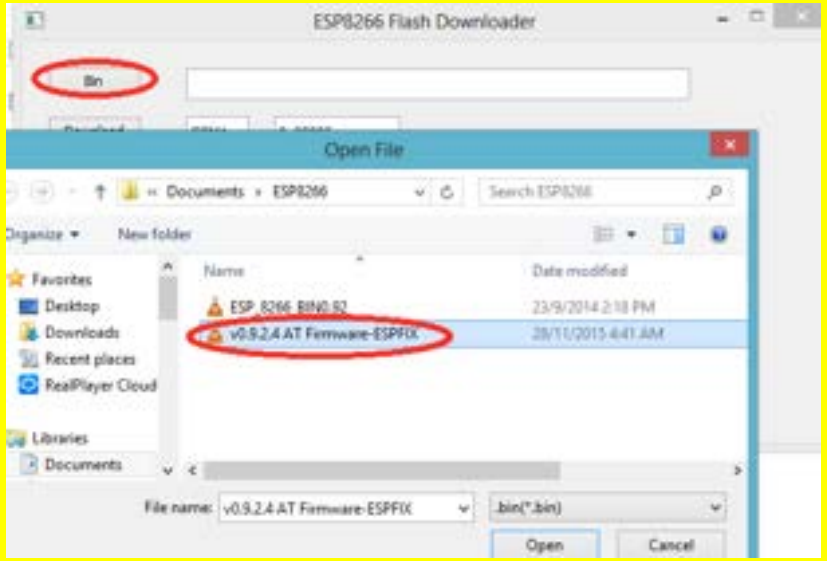


Figure 3.10.2 : Zip File For 0.9.2.4 firmware

Next is to check whether it is the correct COM port or change it to the correct port that used by Adruino.
To start the upload, click on the "Download"-button. If everything is running well, the flasher software will show the flashing and the blue LED on the ESP8266 will blink very fast.

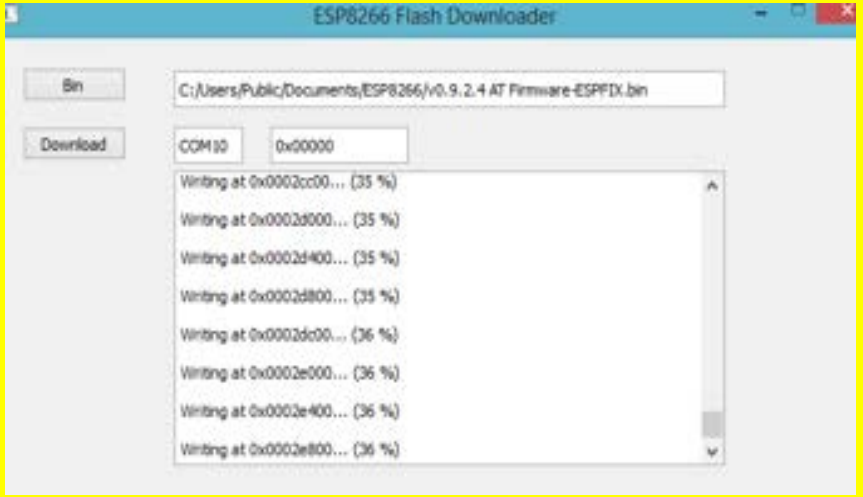


Figure 3.10.3 : Running ESP8266 Software

Wait until the software finish running and in the end at 99%, it will show some error but this is ok. The ESP8266 now has firmware version: 0.9.2.4.

Step 3 :'

Rewiring the ESP8266 as shown in diagram below :

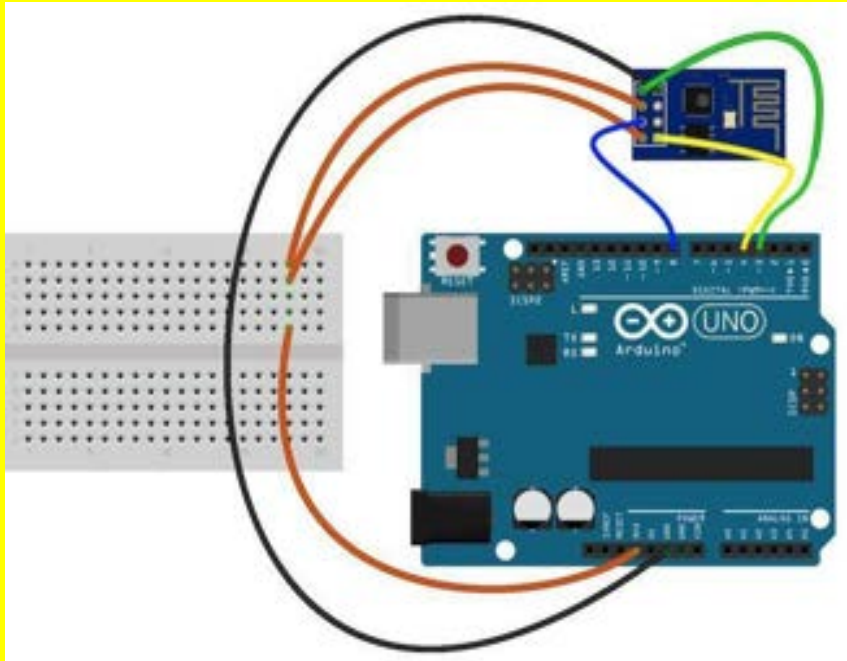


Figure 3.10.4 : Hardware Connection For ESP8266

Figure 3.10.5 : Wiring For ESP8266

3. Advance Projects
3.11 LED Control

A simple LED for indication. You need to send 0 in order to turn LED off. And 255 in order to turn LED on. Or just use Blynk API as described below :

```
WidgetLED led1(V1); //register to virtual pin 1
led1.off();
led1.on();
```

Figure 3.11.1 : Blynk API

All values between 0 and 255 will change LED brightness :

```
WidgetLED led2(V2);
led2.setValue(127); //set brightness of LED to 50%.
```

Figure 3.11.2 : Change LED Brightness

.

Figure 3.11.3 : LED Setting Interface


```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial


#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

WidgetLED led1(V1);

BlynkTimer timer;

// V1 LED Widget is blinking
void blinkLedWidget()
{
  if (led1.getValue()) {
    led1.off();
    Serial.println("LED on V1: off");
  } else {
    led1.on();
    Serial.println("LED on V1: on");
  }
}
```

```
void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth);

  timer.setInterval(1000L, blinkLedWidget);
}

void loop()
{
  Blynk.run();
  timer.run();
}
```


3. Advance Projects
3.12 Slider

Similar to potentiometer. Allows to send values between MIN and MAX.

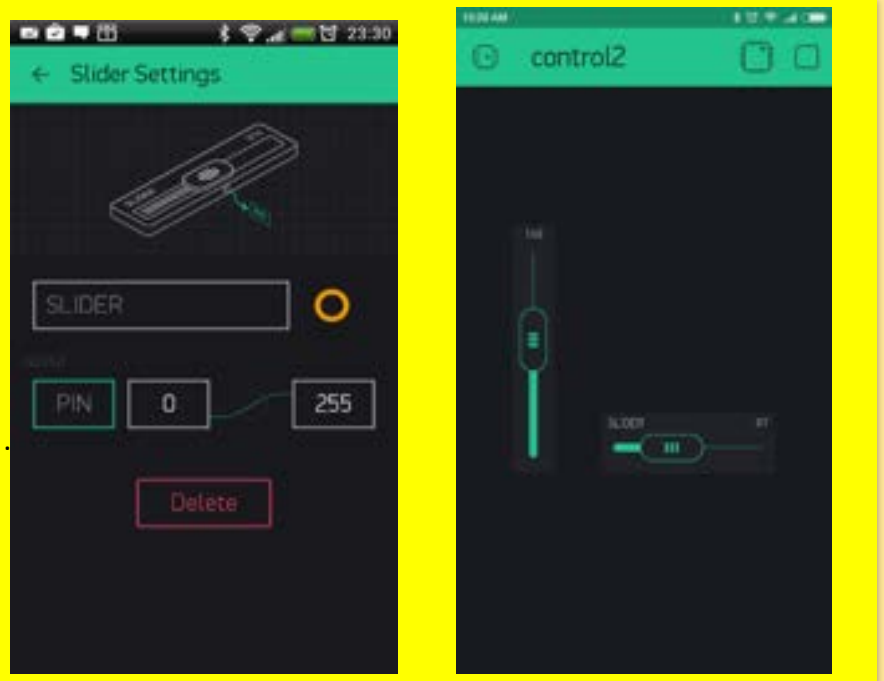


Figure 3.12.1 : Slider Interface

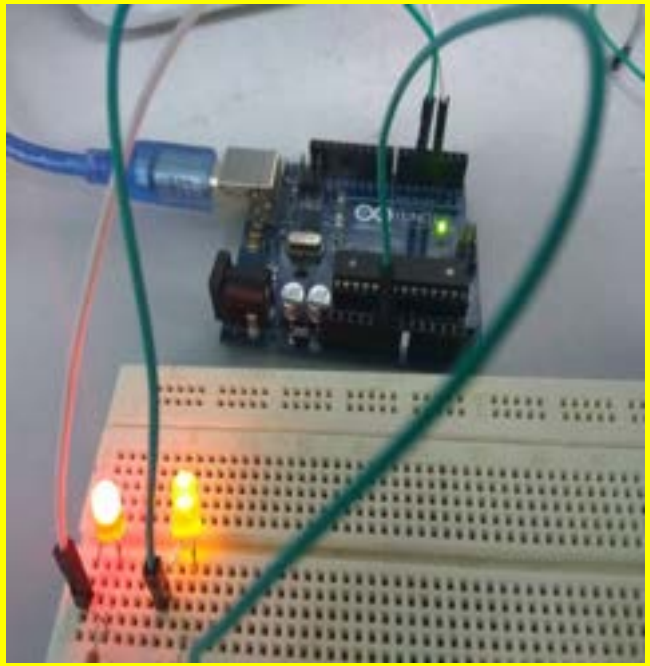


Figure 3.12.2 : Slider Output

3. Advance Projects
3.13 Joystick

Control servo movements in 4 directions

Settings:

SPLIT/MERGE modes

Rotate on Tilt

When it's ON, Joystck will automatically rotate if you use your smartphone in landscape orientation

Auto-Return

When it's OFF, Joystick handle will not return back to center position. It will stay where you left it.

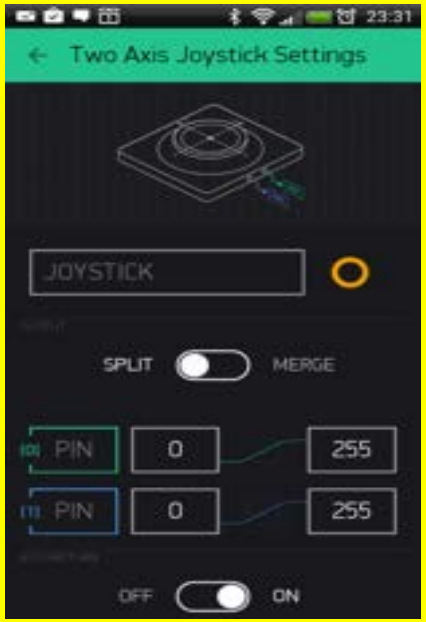


Figure 3.13.1 : Joystick Interface

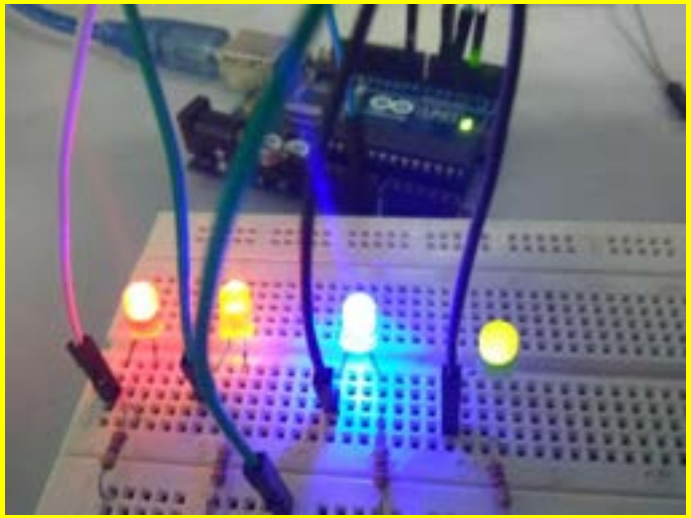


Figure 3.13.2 : Joystick Output

3. Advance Projects
3.14 Button

Works in push or switch modes. Allows to send 0/1 (LOW/HIGH) values. Button sends 1 (HIGH) on press and sends 0 (LOW) on release.

Figure 3.14.1 : Button Interface

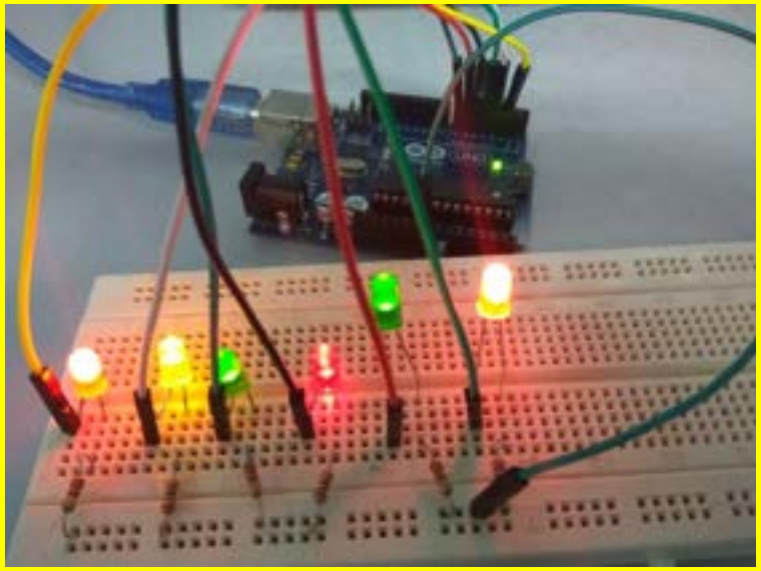


Figure 3.14.2 : Button Output

3. Advance Projects
3.15 Timer

Timer triggers actions at a specific time. Even if smartphone is offline. Start time sends 1 (HIGH). Stop time sends 0 (LOW). Recent Android version also has improved Timer within Eventor widget. With Eventor Time Event you can assign multiple timers on same pin, send any string/number, select days and timezone. It is recommended to use Eventor over Timer widget. However Timer widget is still suitable for simple timer events.

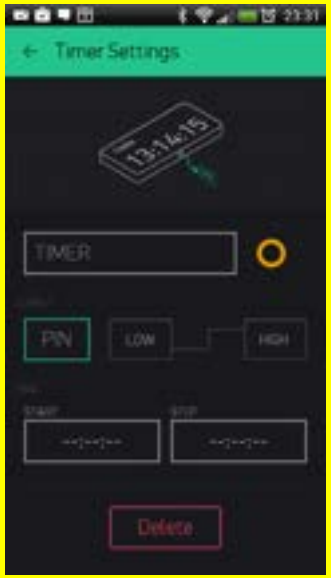
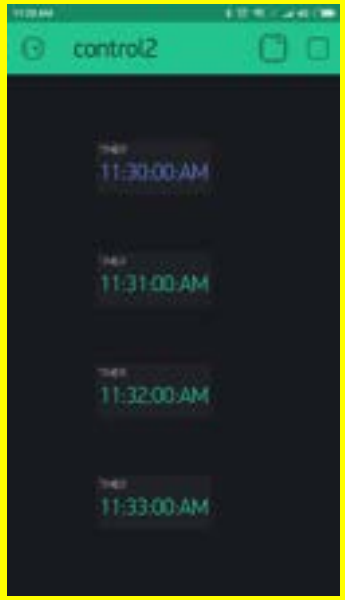


Figure 3.15.1 : Timer Interface



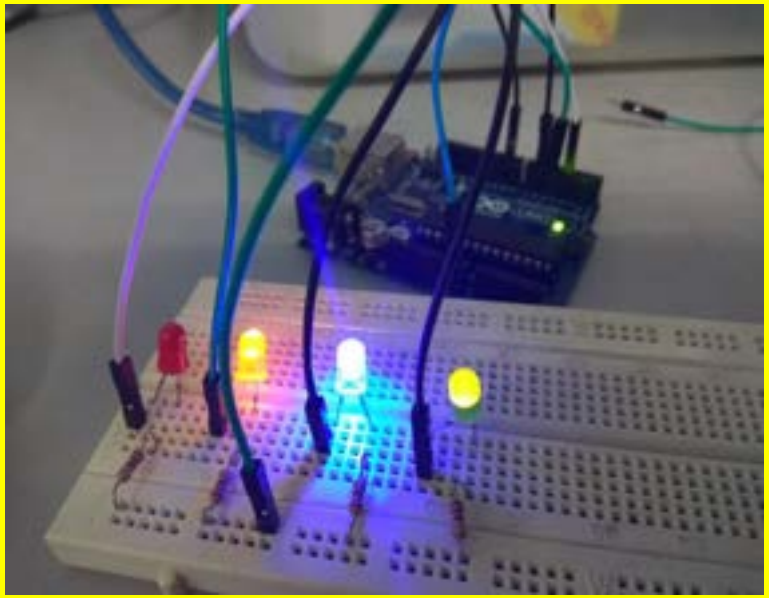


Figure 3.15.2 : Timer Output

3. Advance Projects
3.16 zeRGBa

zeRGBa is usual RGB controller (color picker).

Settings:
SPLIT: Each of the parameters is sent directly to the Pin on your hardware (e.g D7). You don't need to write any code.

NOTE: In this mode you send multiple commands from one widget, which can reduce performance of your hardware.
Example: If you have a zeRGBa Widget and it's set to D1, D2, D3 it will send 3 commands over the Internet:

```
digitalWrite(1, r);
digitalWrite(2, g);
digitalWrite(3, b);
```

Figure 3.16.1 : Split

MERGE: When MERGE mode is selected, you are sending just 1 message, consisting of array of values. But you'll need to parse it on the hardware.

This mode can be used with Virtual Pins only.
Example: Add a zeRGBa Widget and set it to MERGE mode. Choose Virtual Pin V1.

```
BLYNK_WRITE(V1) // zeRGBa assigned to V1
{
    // get a RED channel value
    int r = param[0].asInt();
    // get a GREEN channel value
    int g = param[1].asInt();
    // get a BLUE channel value
    int b = param[2].asInt();
}
```

Figure 3.16.2 : Merge

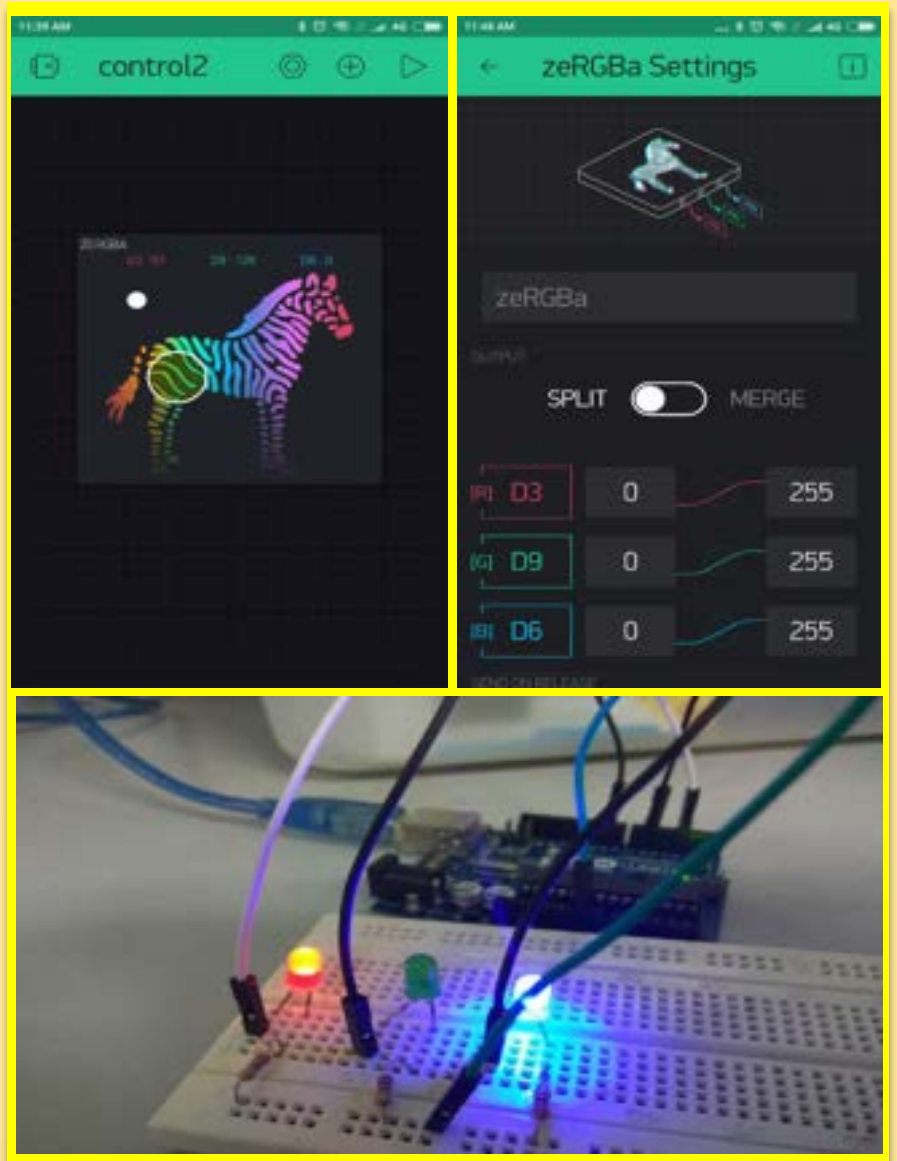


Figure 3.16.3 : zeGBRa

3. Advance Projects
3.17 Step H

Step control is like 2 buttons assigned to 1 pin. One button increments your value by desired step and another one decrements it. It is very useful for use cases where you need to change your values very precisely and you can't achieve this precision with slider widget.

Send Step option allows you to send step to hardware instead of actual value of step widget. **Loop value** option allows you to reset step widget to start value when maximum value is reached.

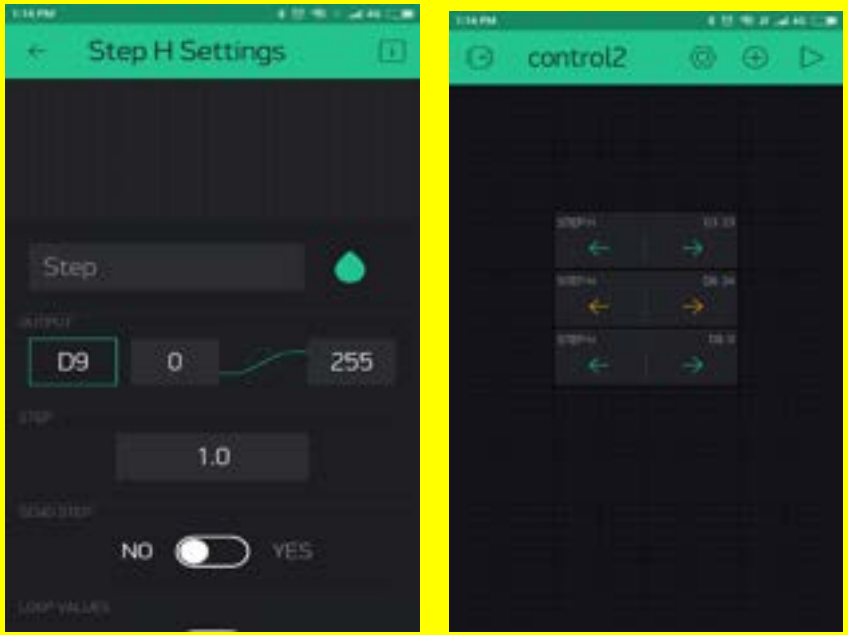


Figure 3.17.1 : Step H Interface

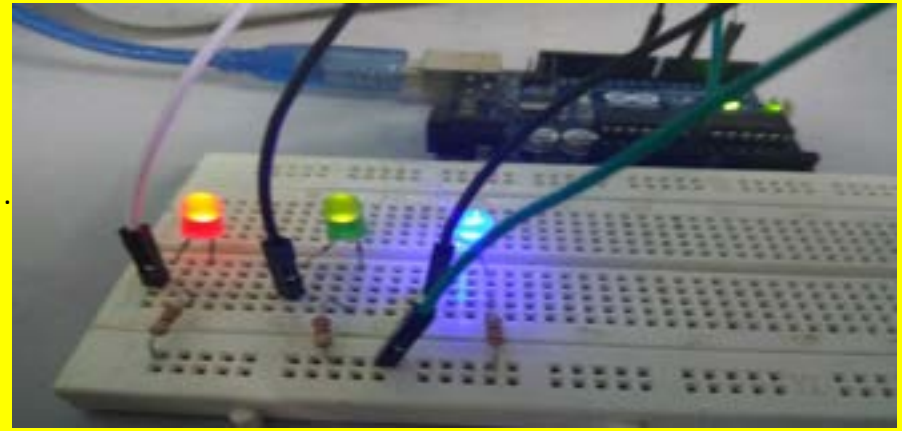


Figure 3.17.2 : Step H Output

3. Advance Projects
3.18 Step V

Step control is like 2 buttons assigned to 1 pin. One button increments your value by desired step and another one decrements it. It is very useful for use cases where you need to change your values very precisely and you can't achieve this precision with slider widget.

Send Step option allows you to send step to hardware instead of actual value of step widget. **Loop value**option allows you to reset step widget to start value when maximum value is reached.

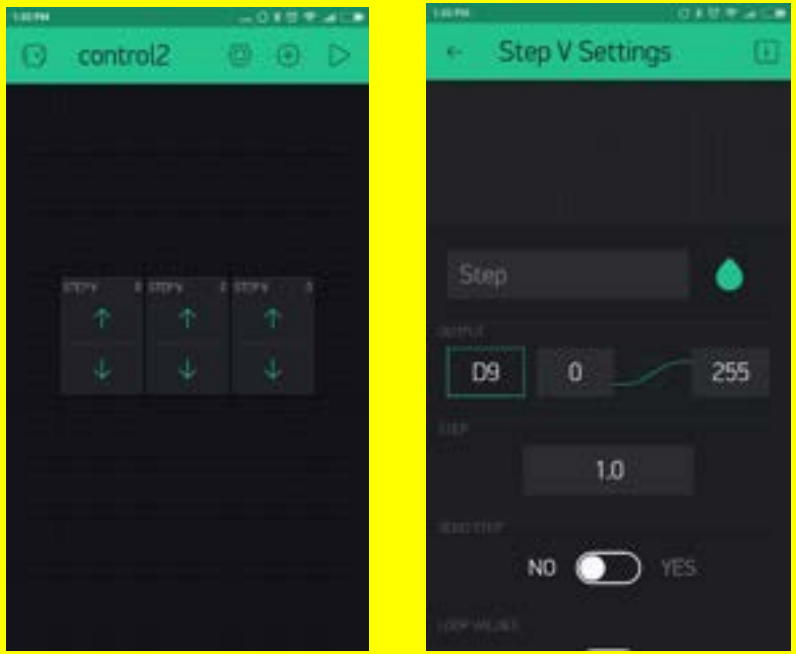


Figure 3.18.1 : Step V Interface

Figure 3.18.2 : Step V Output

3. Advance Projects
3.19 Labelled Value

Displays incoming data from your sensors or Virtual Pins. It is a better version of 'Value Display' as it has a formatting string, so you could format incoming value to any string you want.

Figure 3.19.1 : Labelled Value

Sketch: BlynkBlink

Formatting options
Let's assume, your sensor sends number 12.6789 to Blynk application. Next formatting options are supported:
/pin/ - displays the value without formatting (12.6789)
/pin./ - displays the value without decimal part (13)
/pin.#/ - displays the value with 1 decimal digit (12.7)
/pin.##/ - displays the value with two decimal places (12.68)

3. Advance Projects
3.20 NodeMCU Esp8266 12E

http://www.instructables.com/id/Simple-Led-Control-With-Blynk-and-NodeMCU-Esp8266-/

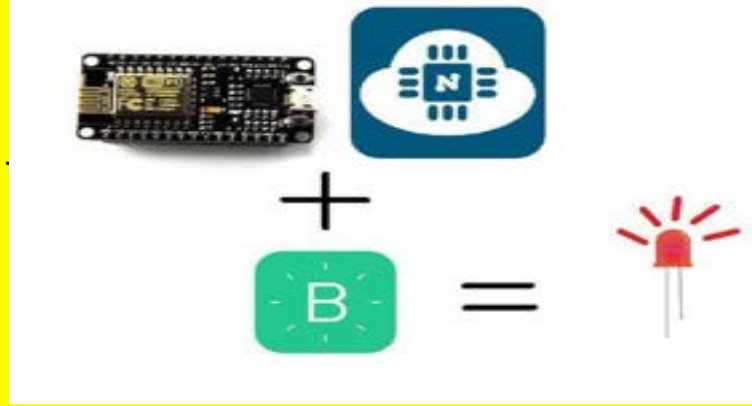
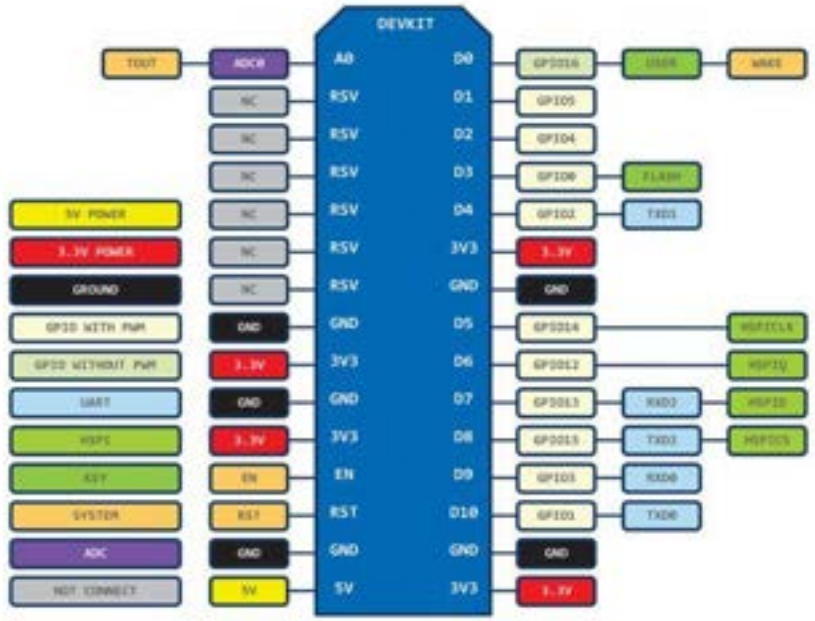


Figure 3.20.1 : NodeMCU Esp8266 12E



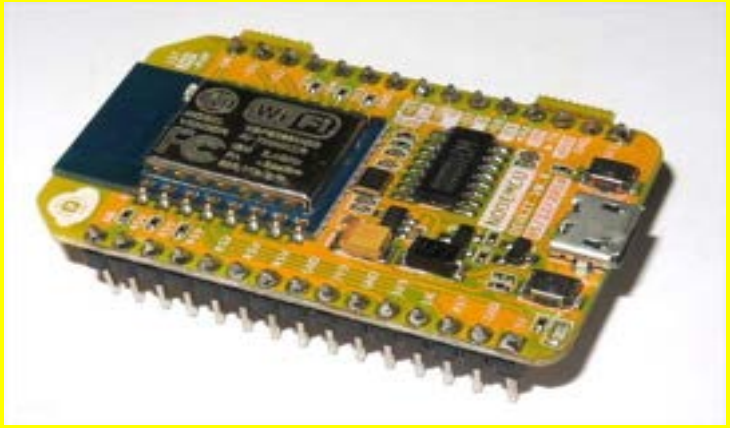


Figure 3.20.2 : NodeMCU Board

3. Advance Projects
3.21 Getting to Know

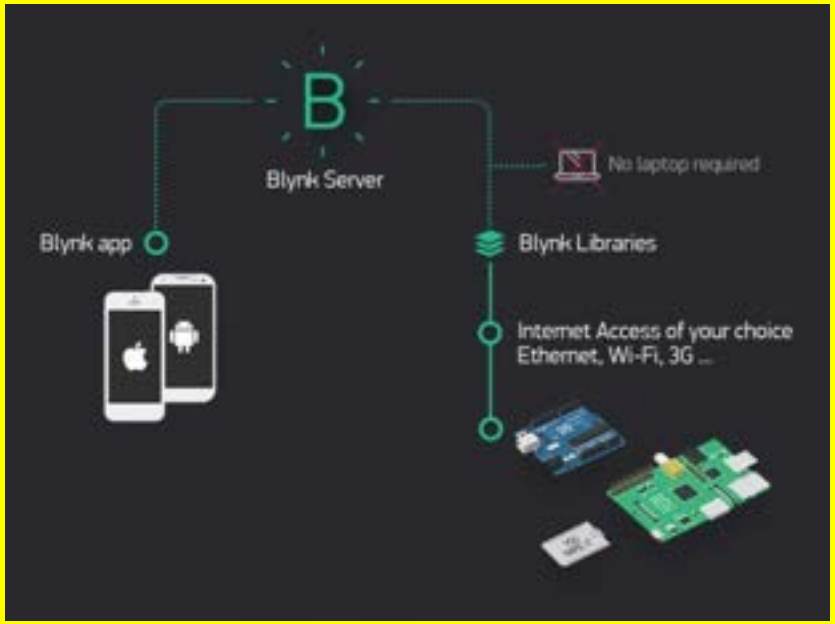


Figure 3.21.1 : Blnyk Platfrom

Blynk is a Platform with iOS and Android apps to control Adruino, Raspberry Pi and the likes over the Internet.
Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, vizualize it and do many other cool things.

It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets. It's really simple to set everything up and you'll start tinkering in less than 5 mins. Blynk is not tied to some specific board or shield. Instead, it's supporting hardware of your choice. Whether your Adruino or Raspberry Pi is linked to the Internet over Wi-Fi, Ethernet or this new ESP8266 chip, Blynk will get you online and ready for the Internet Of Your Things.

How does it work?
There are three major components in the platform:

Blynk App - allows to you create amazing interfaces for your projects using various widgets we provide.

Blynk Server - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It’s open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.

Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Its features:
*Supports majority of development boards like Adruino ,RPI, esp8266
* Easy to use
* Awesome widgets like LCD, push buttons, labelled value, graphs
* Not restricted to local Wifi network
*Direct pin manipulation with no code writing
*Easy to integrate and add new functionality using virtual pins
I guess this should be more than enough to build the project:)

3. Advance Projects

3.22 Simple Led Control With Blynk and NodeMCU Esp8266 12E

Materials Required
Now that we have some insights about the hardware and the app, we require the following components

1. **Node MCU Esp8266 12E** development board
for my friends in India :follow this link if you want to buy
http://www.amazon.in/ESP8266-NodeMcu-WiFi-Development-Board/dp/B00UY8C3N0
and for my buddies in around the world:
https://www.amazon.com/Diymall%C2%AE-NodeMCU-Devkit-CP2102-Apples/dp/B00UY8C3N0?ie=UTF8&*Version*=1&*entries*=0

2. Smart Phone with Blynk App installed
Note : You better charge your mobile before use:)

3. Led with an 330 ohm resistor

4. Breadboard

5. Adruino IDE v1.6.6

And that's pretty much the list to be satisfied!!

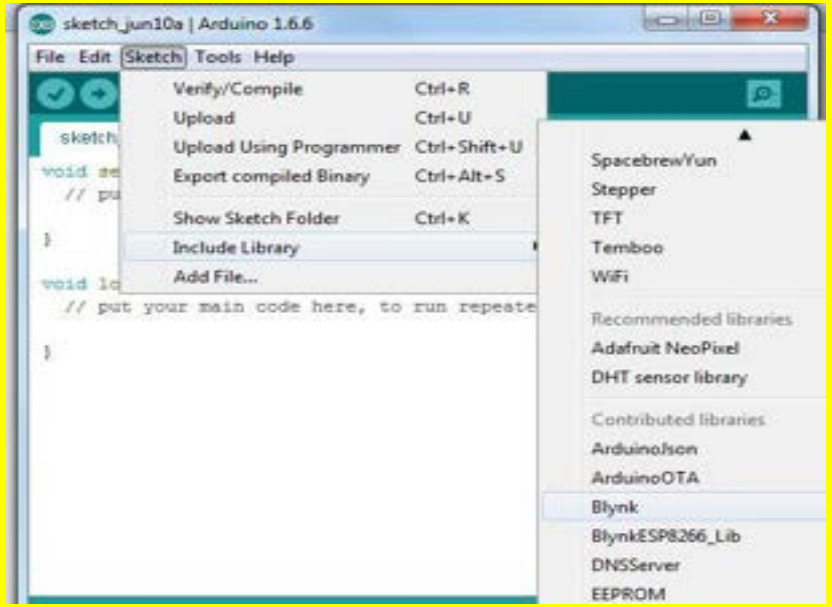


Figure 3.22.1 : Setting Up Blynk With Adruino IDE

This blynk app has set of library files which have to be included in the Adruino IDE environment before the project is executed

1. Follow the link to install libraries
http://www.blynk.cc/getting-started/

2. Once the Zip file is downloaded ,extract it and individually copy all the folder to your libraries folder of your Adruino

3. Once done just open Adruino IDE and go to **Sketch-> Include libraries** and you would see blynk in the menu

4. If you see that then libraries have been included successfully

***Now it is time to include the board configuration in the Adruino IDE**
What is board configuration?
Ok , a simple answer is that it contains all the essential parameters which required to get the board booted and configured.
for example in if you go to Tools->Board Menu you would see a list of boards . All this boards listed have different configuration settings. Therefore we should also include NodeMCU's board configurations which typically contain the board architecture , clock speed, baud rate etc.

Lets start. In the Adruino IDE go to **File->Preferences**
Now Copy the below link and paste it in the Additional Boards Manager Url text box
http://Adruino.esp8266.com/stable/package_esp8266c...
Restart the Adruino IDE after that.
Now after restarting the Adruino IDE , go to **Tools->Boards** and select Node MCU board , mine was version 0.9

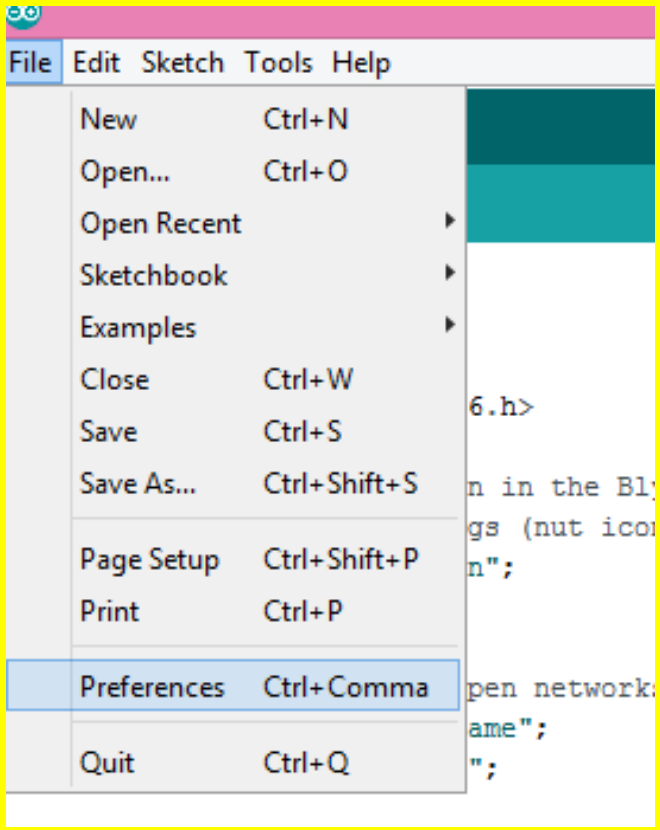


Figure 3.22.2 : File Menu

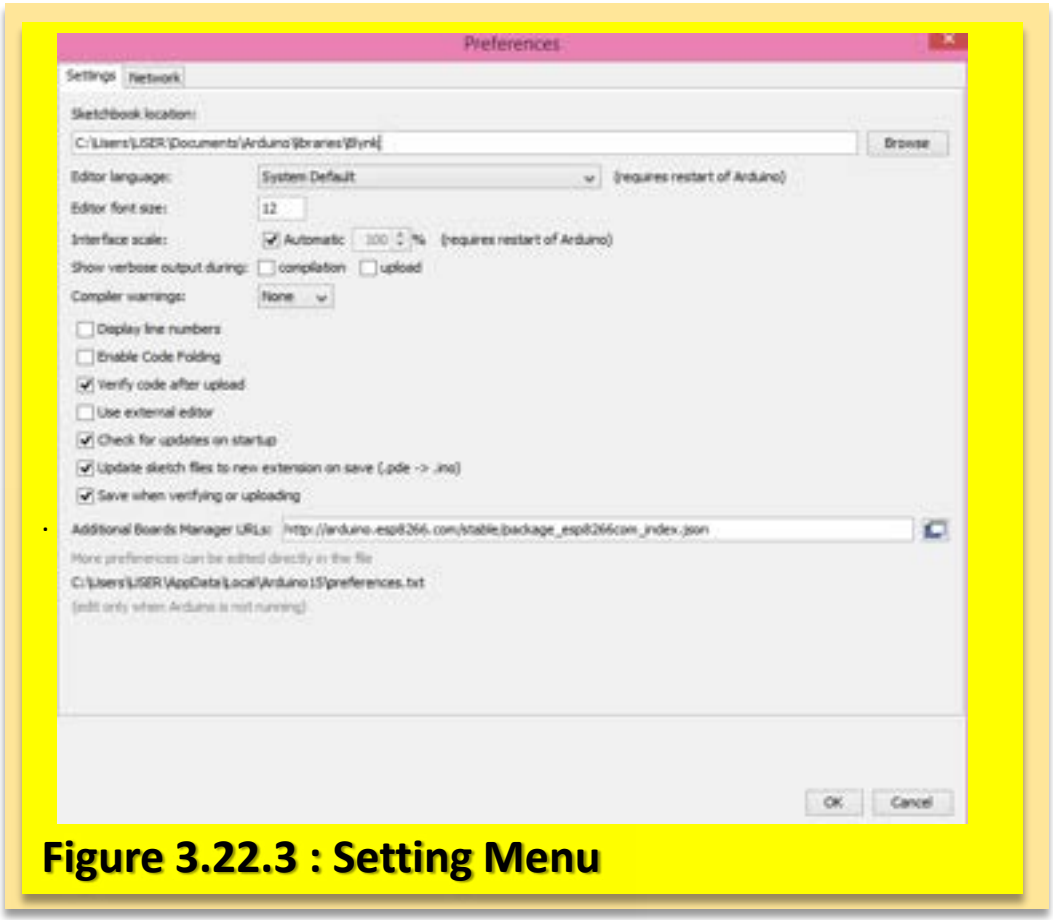


Figure 3.22.3 : Setting Menu

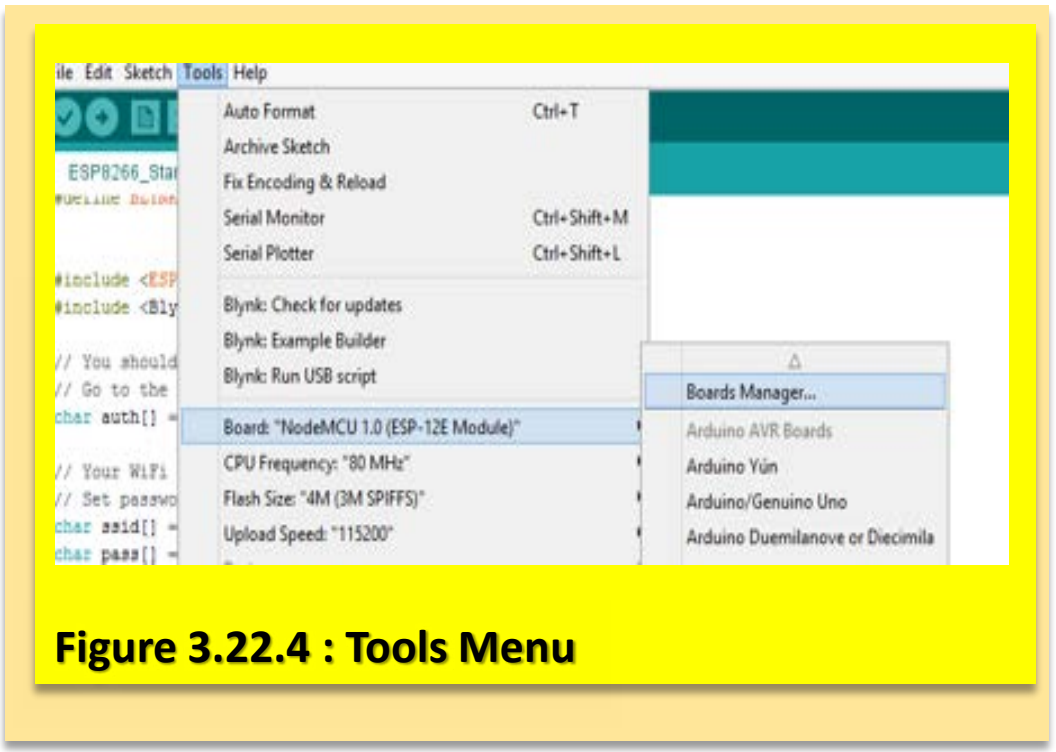


Figure 3.22.4 : Tools Menu

Figure 3.22.5 : Board Manager

3. Advance Projects

3.23 Simple Led Control With Blynk and NodeMCU Esp8266 12E

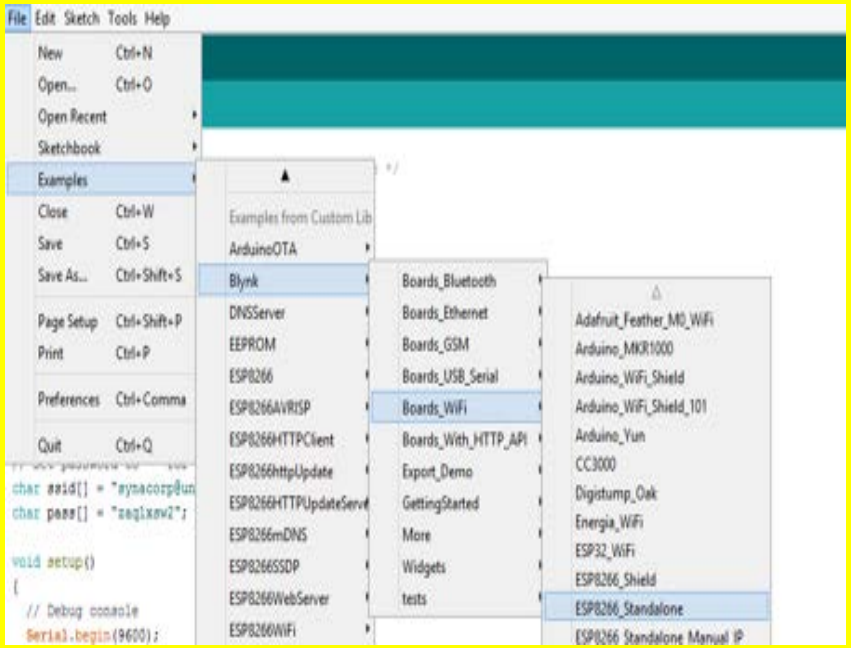


Figure 3.22.6 : Open new file for ESP8266

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "7c8e49e91e6b4a619687670372df3c86";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "synacorp@unifi";
char pass[] = "zaq1xsw2";

void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin("7c8e49e91e6b4a619687670372df3c86","synacorp@unifi","zaq1xsw2");
}
```

Figure 3.22.7 : BlynkSimpleEsp8266.h File

Program:

```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "7c8e49e91e6b4a619687670372df3c86";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "synacorp@unifi";
char pass[] = "zaq1xsw2";
void setup()
{
  // Debug console
  Serial.begin(9600);

Blynk.begin("7c8e49e91e6b4a619687670372df3c86","synacorp@uni
fi","zaq1xsw2");
}

void loop()
{
  Blynk.run();
}
```

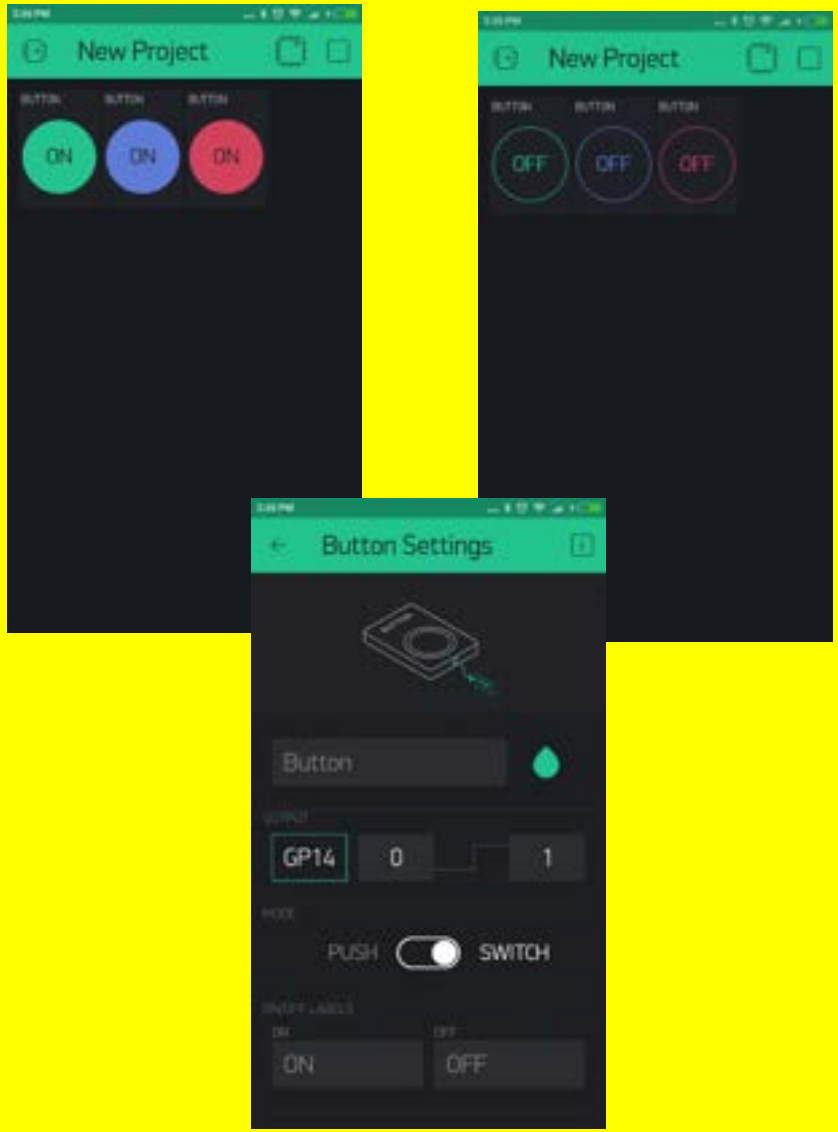
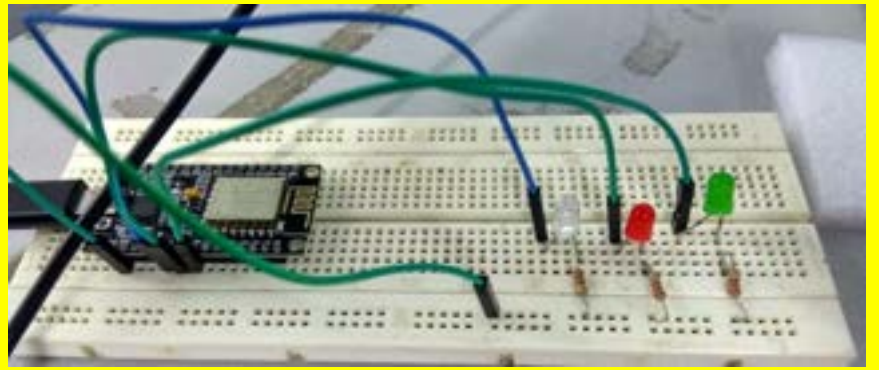
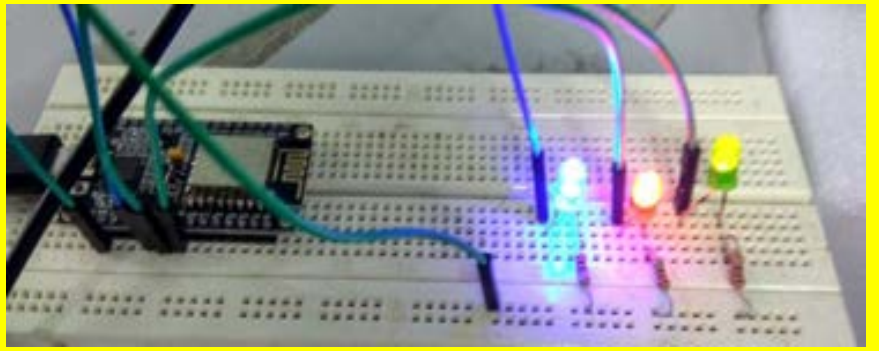


Figure 3.22.8 : Blnyk Interface





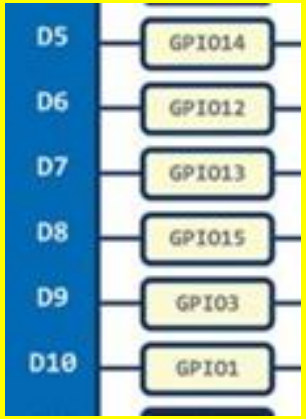


Figure 3.22.9 : Circuit Connection

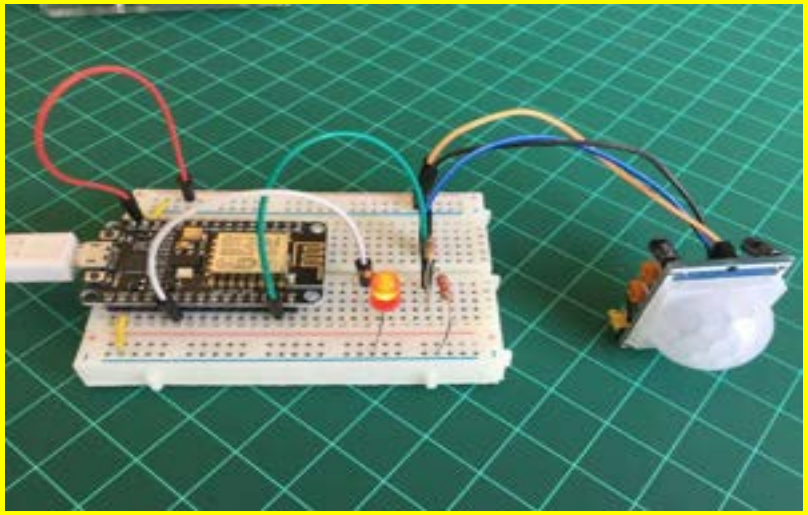


Figure 3.22.10 : IoT Motion Detector

Bill of Material

- NodeMCU ESP12-E
- Motion Sensor HC-SR501
- Resistor (1K, 2.2K and 330ohm)
- LED
- Breadboard
- Cables

Figure 3.22.11 : Motion Detector Manual

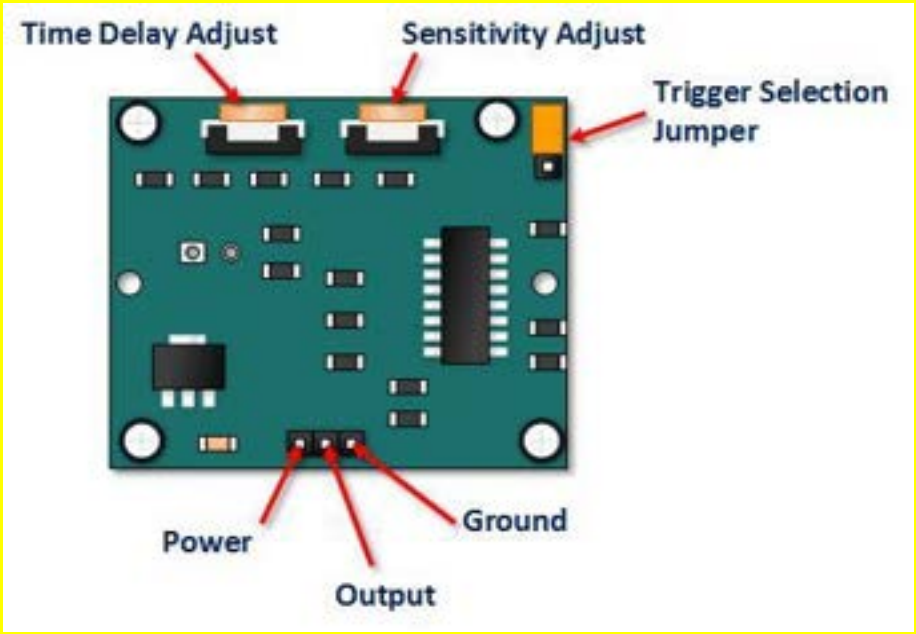


Figure 3.22.12 : Motion Detector Board

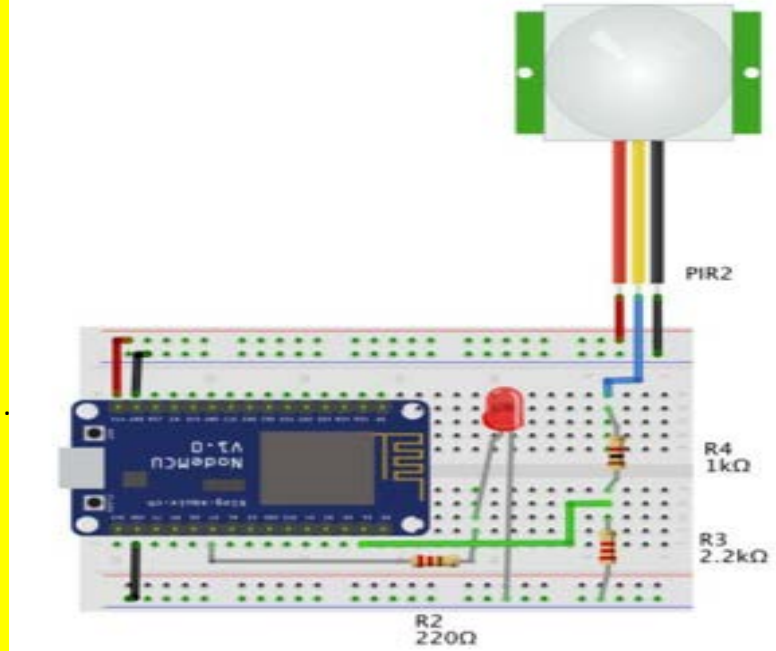


Figure 3.22.13 : The HW

The HW is very simple. The sensor has 3 pins (+5V, GND and Output).
It is important to point that the output of the HC-SR501 generates a logic signal of + 3.3V (HIGH) or 0V (LOW), which is COMPATIBLE with the input levels of the NodeMCU, which also works with the 3.3V level. I kept the circuit with a level converter to ensure if any other sensor with 5V output is used. If you are sure that you sensor generates 3.3V and not 5V as output you can eliminate the level converter, connecting NodeMCU Pin D1 (GPIO 5) directly to HC-SR501 Output.
We have also included a LED at pin D7 (GPIO13), for a local visualization

REFERENCES

1. Monk, S. (2016). *Programming Arduino: Getting Started with Sketches ((2nd Edition)*. Mc Graw Hill education.
2. Boxall, J. (2013). *Arduino Workshop: A Hands-On Introduction with 65 Projects*. San Francisco.
3. Margolis, M. (2012) *Arduino Cookbook, 2nd Edition 2nd Edition*. O'Reilly.
4. Richard, B. (2015). *Arduino Programming in 24 Hours, Sams Teach Yourself 1st Edition*. Sams.
5. Thorpe, E. (2020). Ard*uino: Advanced Methods and Strategies of Using Arduino.* Kindle Edition.
6. Nussey, J. (2018). *Arduino For Dummies (For Dummies (Computer/Tech)) 2nd Edition.* Kindle Edition.
7. David W.J., Adams, J. & Molle, H. (2011). *Arduino Robotics (Technology in Action) 1st ed. Edition.* Apress.
8. Parker, D. (2020).Arduino Programming: The Ultimate Guide For Making The Best Of Your Arduino Programming Projects. Kindle Edition

KEMENTERIAN PENGAJIAN TINGGI

KOLEJ KOMUNITI MALAYSIA

e ISBN 978-967-2702-04-7

9 7 8 9 6 7 2 7 0 2 0 4 7