

Project report on IOT

Name

Jairam J S

Email ID

jsjairam01@gmail.com

CONTENTS

s.no	Date	Project Name	Page.no
1.	11.5.2020	Blinking Two LED	3-7
2.	13.5.2020	Controlling of Servo Motor using Arduino	8-11
3.	15.5.2020	Automatic Room Lights using Arduino and PIR Sensor	12-18
4.	11.12.2020	Voice Controlled LEDs using Arduino and Bluetooth	19-21
5.	15.12.2020	Ultrasonic Security System	22-27
6.	02.07.2021	Google Assistant and Blynk in IoT based Home Automation	28-34
7.	04.07.2021	Arduino Ultrasonic sensor hc-sr04 Measure distance with LEDs	35-38

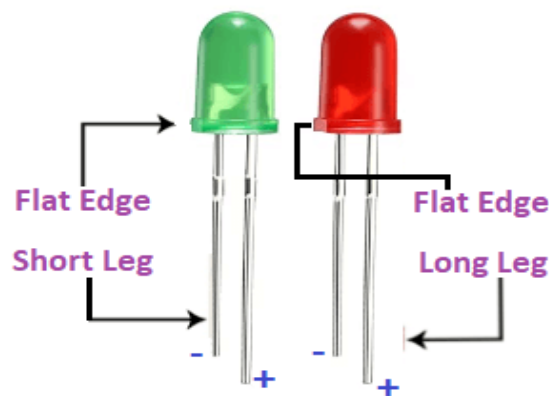
S.no:1

Blinking Two LED

We have already discussed a project of blinking an LED. Here, we will discuss a project of blinking two LED's. The concept of blinking two LED's is similar to the blinking of a single LED. As we know, we can use the resistance of any value, so let's take the resistors of 470 Ohms. The resistors reduce the amount of current reaching the LED, which saves the LED from being burnt. We can also use other resistors depending on the circuit limit and requirements. Let's start with the project.

Structure of two LED's

The structure of red and green LED is shown below:



The long terminal is called Anode (positive charged), and the short terminal is called Cathode (negative charged).

Components

The components used in the project are listed below:

1. 1 x Arduino UNO board.

We can also use other types of Arduino boards, such as Arduino Mega, Arduino Micro, etc.

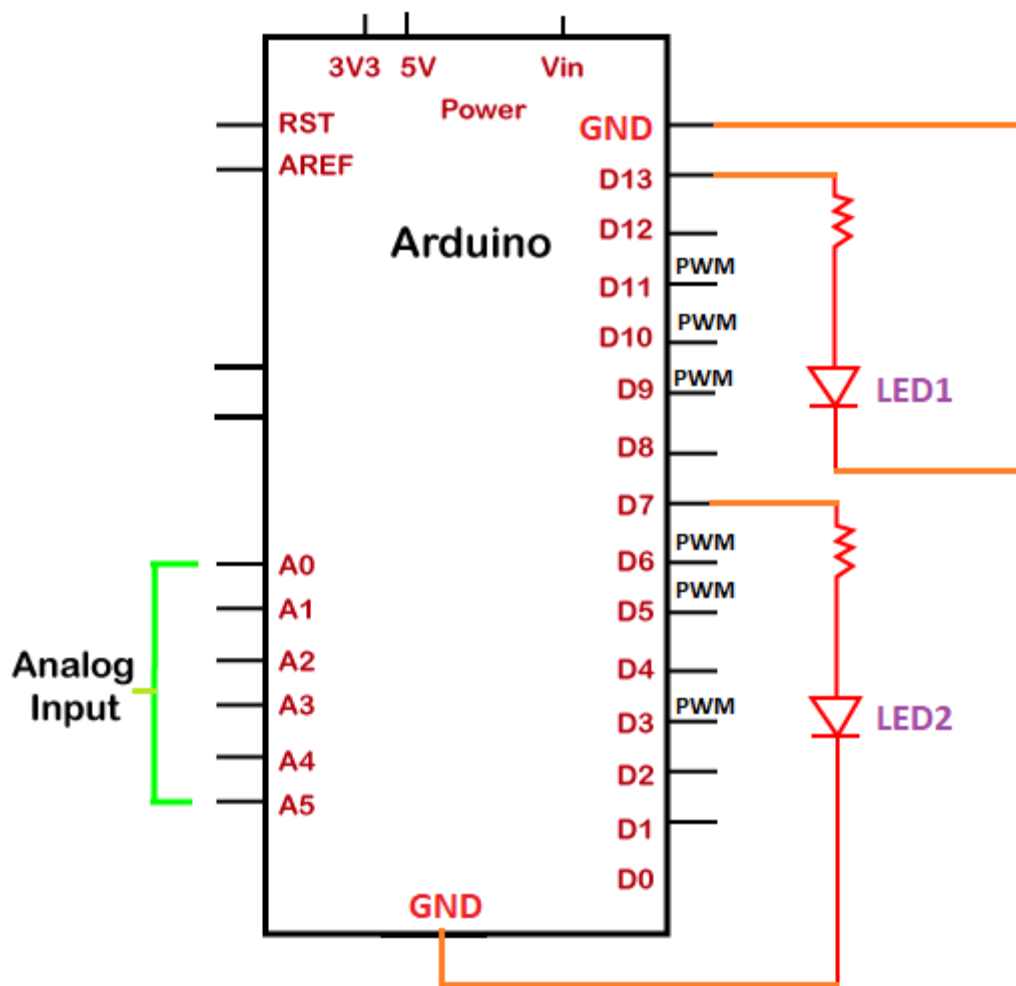
2. 1 x Breadboard
3. 4 x Jump wires

4. 1 x Red LED
5. 1 x Green LED
6. 2 x Resistor of 470 Ohm.

Structure of the project

Here, we will use the digital output pin number 13 and 7. The positive terminal of the red LED is connected to the PIN 13, and the negative terminal (anode) is connected to the ground. Similarly, the positive terminal (cathode) of the green LED is connected to PIN 7 and the negative terminal is connected to the ground. As mentioned, two resistors each of 470 Ohms, will be connected in series to the two LEDs in the project.

The structure will represent the pinout diagram of the project. It is shown below:



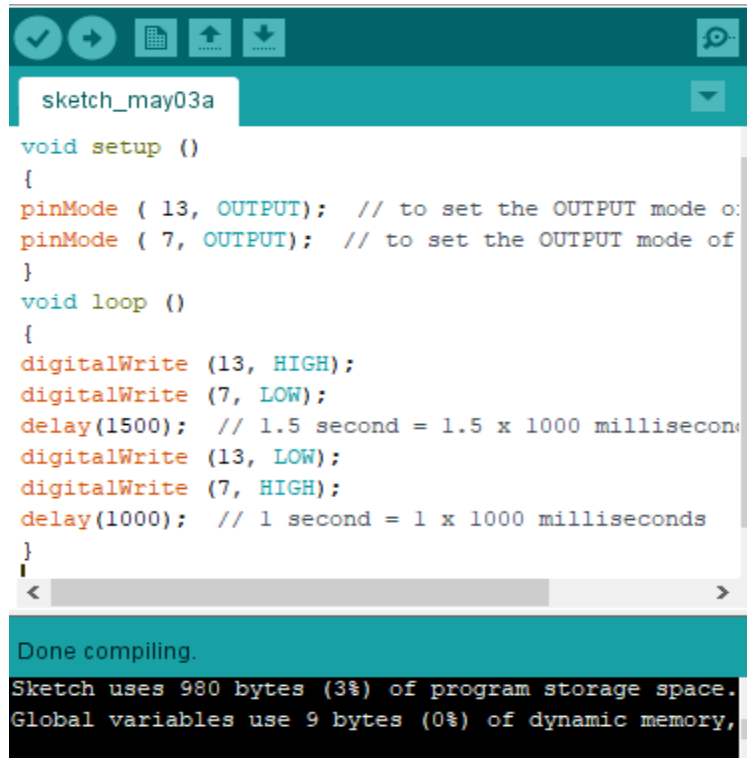
Sketch

Open the Arduino IDE (Integrated Development Environment) and start with the coding, which is given below:

```
1. void setup ()
2. {
3.   pinMode (13, OUTPUT);
4.   pinMode (7, OUTPUT);
5. }
6. void loop ()
7. {
8.   digitalWrite (13, HIGH);
9.   digitalWrite (7, LOW);
10. delay(1500); // 1.5 second = 1.5 x 1000 milliseconds
11. digitalWrite (13, LOW);
12. digitalWrite (7, HIGH);
13. delay(1000); // 1 second = 1 x 1000 milliseconds
14. }
```

We can modify the delay duration according to our choice or as per the requirements.

The sketch will be uploaded to the board after the correct compiling, as shown below:



```
void setup ()
{
  pinMode ( 13, OUTPUT); // to set the OUTPUT mode of
  pinMode ( 7, OUTPUT); // to set the OUTPUT mode of
}
void loop ()
{
  digitalWrite (13, HIGH);
  digitalWrite (7, LOW);
  delay(1500); // 1.5 second = 1.5 x 1000 milliseconds
  digitalWrite (13, LOW);
  digitalWrite (7, HIGH);
  delay(1000); // 1 second = 1 x 1000 milliseconds
}
}
```

Done compiling.

Sketch uses 980 bytes (3%) of program storage space.

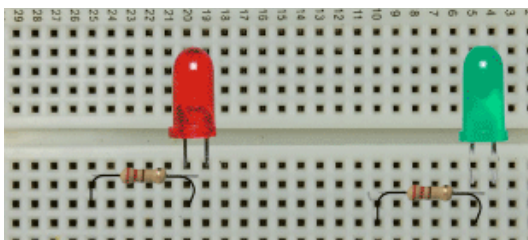
Global variables use 9 bytes (0%) of dynamic memory,

Click on the Verify button present on the toolbar to compile the code. The RX and TX LED on the board will light up after the successful uploading of the code.

Procedure

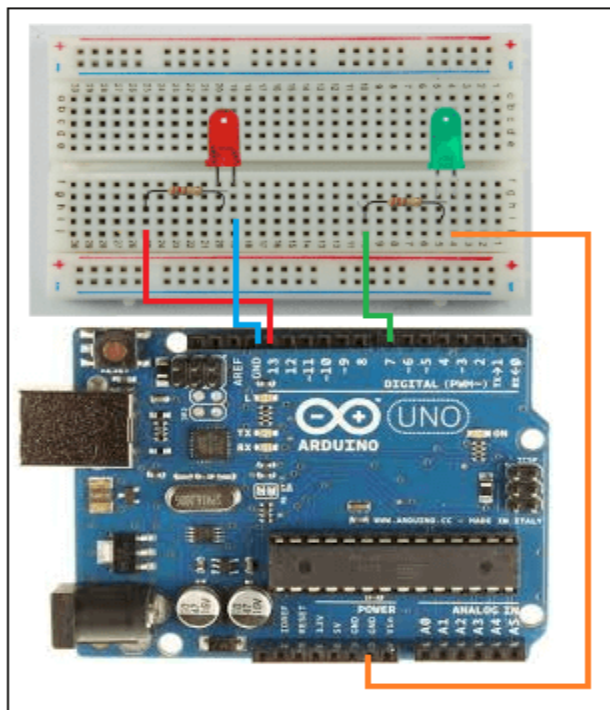
The procedure to join the components of the project is shown below:

- Plug-in the two LED adjacent to each other on the breadboard.
- Now, plug-in the resistors of 470 Ohm in series with the two LED, as shown below:



We need to check that the plug-in is performed correctly, as shown above. For any confusion, check the pin diagram shown above in the heading- Structure of project.

- Connect the left leg of the resistor (connected in series with **red** LED) to the digital output pin of the UNO board, i.e., PIN **13**.
- Connect the left leg of the resistor (connected in series with **green** LED) to the digital output pin of the UNO board, i.e., PIN **7**.
- Connect the negative/shorter terminal (Cathode) of the red and green LED to the **GND** pin of the UNO board using the wire, as shown below:



Here, the red wire is connected to the PIN 13, and the blue wire is connected to the GND. Similarly, the green wire is connected to the PIN 7, and the orange wire is connected to the GND.

S.no:2 Controlling of Servo Motor using Arduino

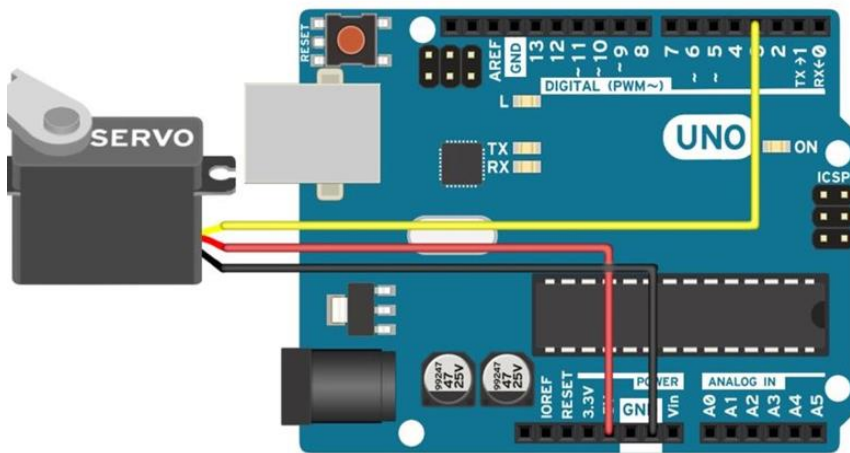
Step 1: How to Connect Them

Servo motors are great devices that can turn to a specified position. Usually, they have a servo arm that can turn 180 degrees. Using the Arduino, we can tell a servo to go to a specified position and it will go there. As simple as that! Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane. With time, they found their uses in robotics, automation, and of course, the Arduino world. Here we will see how to connect a servo motor and then how to turn it to different positions. We will need the following things:

1. An Arduino board connected to a computer via USB
2. A servo motor
3. Jumper wires

A servo motor has everything built in: a motor, a feedback circuit, and most important, a motor driver. It just needs one power line, one ground, and one control pin. Following are the steps to connect a servo motor to the Arduino:

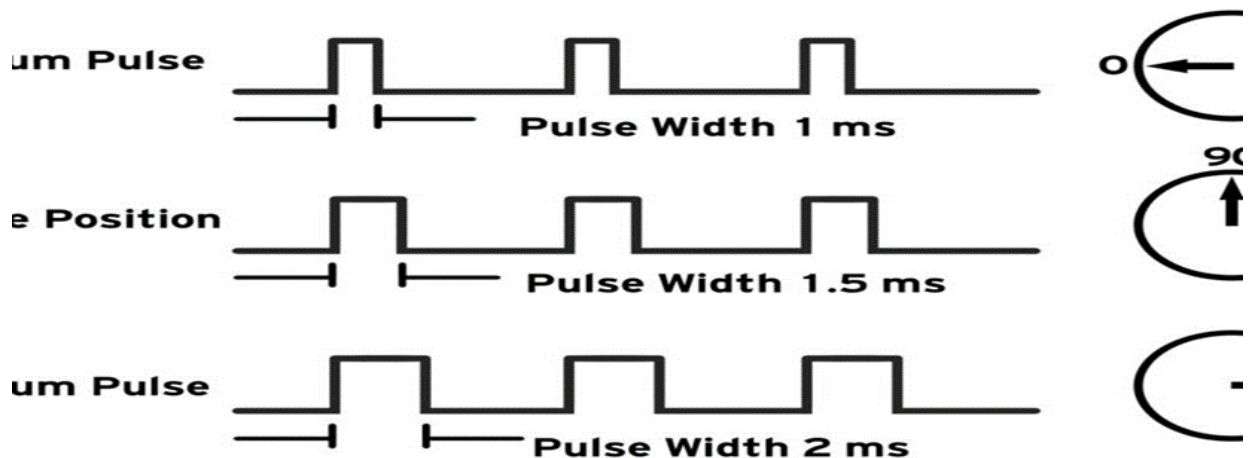
1. The servo motor has a female connector with three pins. The darkest or even black one is usually the ground. Connect this to the Arduino GND.
2. Connect the power cable that in all standards should be red to 5V on the Arduino.
3. Connect the remaining line on the servo connector to a digital pin on the Arduino.



Step 2: Code

If the servo motor is connected on another digital pin, simply change the value of servo Pin to the value of the digital pin that has been used.

Step 3: How It Works



Servos are clever devices. Using just one input pin, they receive the position from the Arduino and they go there. Internally, they have a motor driver and a feedback circuit that makes sure that the servo arm reaches the desired position. But what kind of signal do they receive on

the input pin?. It is a square wave similar to PWM. Each cycle in the signal lasts for 20 milliseconds and for most of the time, the value is LOW. At the beginning of each cycle, the signal is HIGH for a time between 1 and 2 milliseconds. At 1 millisecond it represents 0 degrees and at 2 milliseconds it represents 180 degrees. In between, it represents the value from 0–180. This is a very good and reliable method. The graphic makes it a little easier to understand. Remember that using the Servo library automatically disables PWM functionality on PWM pins 9 and 10 on the Arduino UNO and similar boards.

Code breakdown

The code simply declares the servo object and then initializes the servo by using the `servo.attach()` function. We shouldn't forget to include the servo library. In the `loop()`, we set the servo to 0 degrees, wait, then set it to 90, and later to 180 degrees.

Step 4: More Things About Servos

Controlling servos is easy, and here are a few more tricks we can use:

Controlling the exact pulse time

Arduino has a built-in function `servo.write(degrees)` that simplifies the control of servos. However, not all servos respect the same timings for all positions. Usually, 1 millisecond means 0 degrees, 1.5 milliseconds mean 90 degrees, and, of course, 2 milliseconds mean 180 degrees. Some servos have smaller or larger ranges. For better control, we can use the `servo.writeMicroseconds(us)` function, which takes the exact number of microseconds as a parameter. Remember, 1 millisecond equals 1,000 microseconds.

More servos

In order to use more than one servo, we need to declare multiple servo objects, attach different pins to each one, and address each servo individually. First, we need to declare the servo objects—as many as we need:

```
// Create servo objects
Servo Servo1, Servo2, Servo3;
```

Then we need to attach each object to one servo motor. Remember, every servo motor uses an individual pin:

```
Servo1.attach(servoPin1);  
Servo2.attach(servoPin2);  
Servo3.attach(servoPin3);
```

In the end, we just have to address each servo object individually:

```
Servo1.write(0); // Set Servo 1 to 0 degrees  
Servo2.write(90); // Set Servo 2 to 90 degrees
```

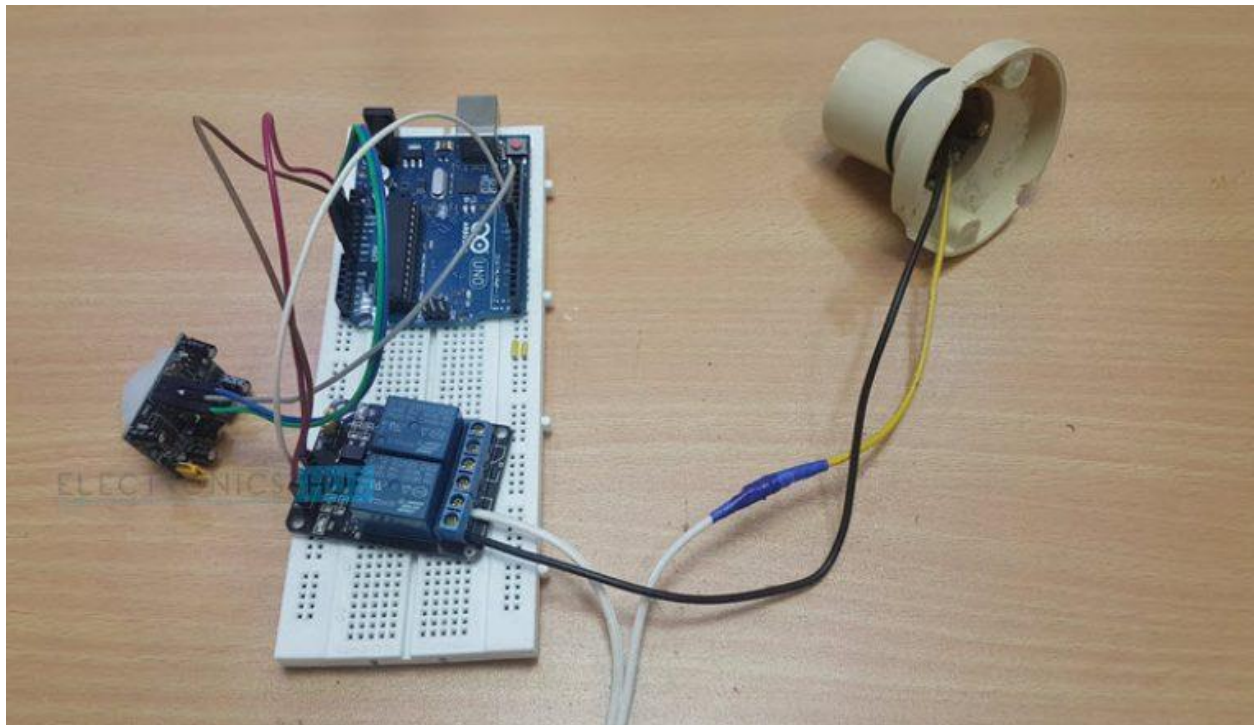
Connection-wise, the grounds from the servos go to GND on the Arduino, the servo power to 5V or VIN (depending on the power input), and in the end, each signal line has to be connected to a different digital pin. Contrary to popular belief, servos don't need to be controlled by PWM pins—any digital pin will work.

Continuous rotation servos

There is a special breed of servos labelled as continuous rotation servos. While a normal servo goes to a specific position depending on the input signal, a continuous rotation servo either rotates clockwise or counter-clockwise at a speed proportional to the signal. For example, the `Servo1.write(0)` function will make the servomotor spin counter-clockwise at full speed. The `Servo1.write(90)` function will stop the motor and `Servo1.write(180)` will turn the motor clockwise at full speed.

S.no:3 Automatic Room Lights using Arduino and PIR Sensor

In this project, we will see the Automatic Room Lights using Arduino and PIR Sensor, where the lights in the room will automatically turn ON and OFF by detecting the presence of a human. Such Automatic Room Lights can be implemented in your garages, staircases, bathrooms, etc. where we do not need continuous light but only when we are present. Also, with the help of an automatic room light control system, you need not worry about electricity as the lights get automatically off when there is no person. So, in this DIY project, we have implemented Automatic Room Lights using Arduino and PIR Sensor.

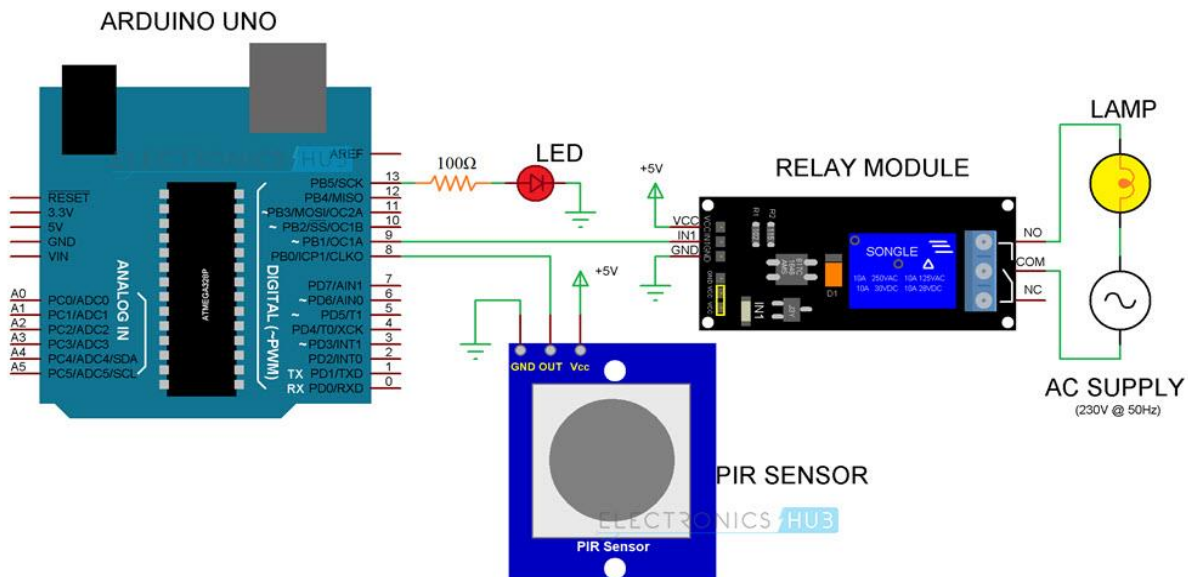


Overview

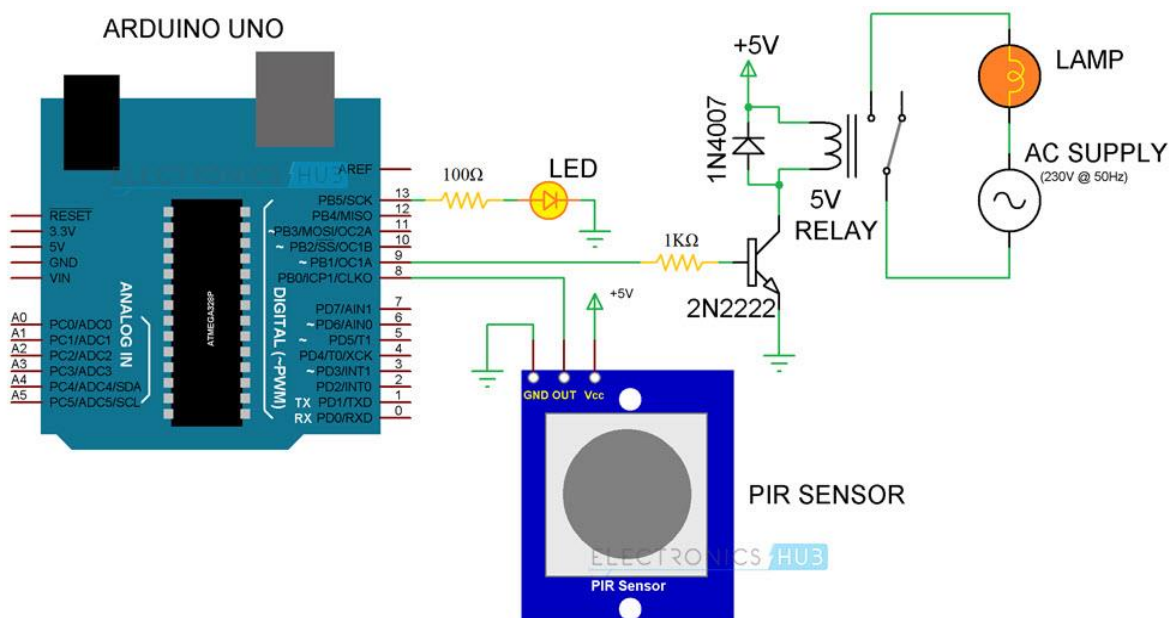
Automatic Room Lights System using Arduino is a very useful project as you need not worry about turning on and off the switches every time you want to turn on the lights. The main components of the Automatic Room Lights project are Arduino, PIR Sensor and the Relay Module. Out of the three components, the PIR Sensor is the one in focus as it is the main device that helps in detecting humans and human motion. In fact, the Automatic Room Lights project can be considered as one major application of the PIR Sensor. A similar concept is being already implemented in automatic toilet flush valves, hand dryers, etc.

Circuit Diagram of Automatic Room Lights using Arduino

The following image shows the circuit diagram of the project implemented using Arduino UNO, PIR Sensor and a Relay Module.



If you do not have a relay module, you can make one yourself using very simple hardware. The following circuit diagram shows the project being implemented with the help of discrete components for the Relay Module.



CAUTION: The project involves connection with 230V AC Mains (or 110V, depending on where you live!!!). Be extremely careful when connecting the bulb and Relay to mains supply. If you are unfamiliar with the connections, I strongly recommend having an adult supervision (or an expert supervision).

Components Required for Automatic Room Lights using Arduino

- Arduino UNO
- PIR Sensor
- 5V Relay Module (Relay Board)
- LED
- 100 Ω Resistor (1/4 Watt)
- Connecting Wires
- Breadboard
- Power Supply

If you do not have a Relay Module, use the following components:

- 5V Relay
- 2N2222 (or BC547) NPN Transistor
- 1N4007 PN Junction Diode
- 1K Ω Resistor (1/4 Watt)
- Component Description
- PIR Sensor

Component Description

- **PIR Sensor**

We have already seen about PIR Sensor in the PIR Motion Sensor Tutorial and also implemented in a variety of projects like Home Security System and Automatic Door Opener.

- **Relay Module**

A Relay Module is a very useful component as it allows Arduino, Raspberry Pi or other Microcontrollers to control big electrical loads. We have used a 2-channel Relay Module in this project but used only one relay in it. The relay module used in this project is shown below.



In order to control a single relay on the board, we need to use three pins of the relay module: VCC, GND and IN1.

NOTE: The relay module used in this project is as active LOW one i.e. when the IN1 pin is HIGH, the relay is OFF and when it is LOW, the relay is activated. This point is important while programming the Arduino UNO.

Circuit Design

PIR Sensor's Data OUT Pin is connected to Arduino's Digital I/O Pin 8. An LED is connected to pin 13 of Arduino to indicate whether the light is turned ON or OFF. The IN1 pin of the Relay Module is connected to Pin 9 of Arduino. A bulb is connected to mains supply through relay. One terminal of the bulb is connected to one wire of the mains supply. The other terminal of the bulb is connected to the NO (Normally Open) contact of the Relay Module. COM (Common) contact of the Relay is connected to the other wire of the mains supply. Be careful when connecting this part of the project.

Working of the Project

The Automatic Room Lights using Arduino and PIR Sensor is a simple project, where the lights in the room will automatically turn on upon detecting a human motion and stay turned on until the person has left or there is no motion. Working of this project is very simple and is explained here. Initially, when there is no human movement, the PIR Sensor doesn't detect any person and its OUT pin stays LOW. As the person enters the room, the change in infrared radiation in the room is detected by the PIR Sensor. As a result, the output of the PIR Sensor becomes HIGH. Since the Data OUT of the PIR Sensor is connected to Digital Pin 8 of Arduino, whenever it becomes HIGH, Arduino will activate the relay by making the relay pin LOW (as the relay module is an active LOW module). This will turn the Light ON. The light stays turned ON as long as there is movement in front of the sensor. If the person takes a nap or leaves the room, the IR Radiation will become stable (there will be no change) and hence, the Data OUT of the PIR Sensor will become LOW. This in turn will make the Arduino to turn OFF the relay (make the relay pin HIGH) and the room light will be turned OFF.

Source code:

```
1. int micPin = A0;
2. int gndPin = A1;
3. int powerPin = A2;
4. int micValue1 = 0;
5. int micValue2 = 0;
6. int led1 = 13;
7. boolean lightOn = false;
8. void setup() {
9.   pinMode(led1, OUTPUT);
10.  pinMode(powerPin, OUTPUT);
11.  pinMode(gndPin, OUTPUT);
12.  pinMode(micPin, INPUT);
13.  digitalWrite(gndPin, LOW);
14.  delay(500);
15.  digitalWrite(powerPin, HIGH);
```



```
16. Serial.begin(9600);
17. }
18. void loop() {
19. micValue1 = analogRead(micPin);
20. Serial.println(micValue1);
21. delay(1);
22. if (micValue1 >= 500){
23. lightOn = !lightOn;
24. delay(100);
25. digitalWrite(led1, lightOn);
26. }
```

Applications

- Garage Lights
- Bathroom Lights
- Hand Dryers
- Toilet Flushers
- Security Lights

S.no:4 Voice Controlled LEDs using Arduino and Bluetooth

Controlling LEDs with voice command seems to be a difficult task, but it's easy and you can quickly build it. We just need an Arduino UNO to serially communicate with HC-06 Bluetooth module and a smartphone to send voice command to Bluetooth module HC-06. For receiving voice command we are using “Arduino Bluetooth Voice Controller” android app which you can download from play store (link is given below).

Material Required

- Arduino UNO
- HC-06 Bluetooth Module
- LEDs (Red, and Green)
- Resistor 220 ohm (2 nos.)
- Arduino Bluetooth Voice Controller (Download from play store)
- Breadboard
- Connecting wires

HC-06 Bluetooth Module:

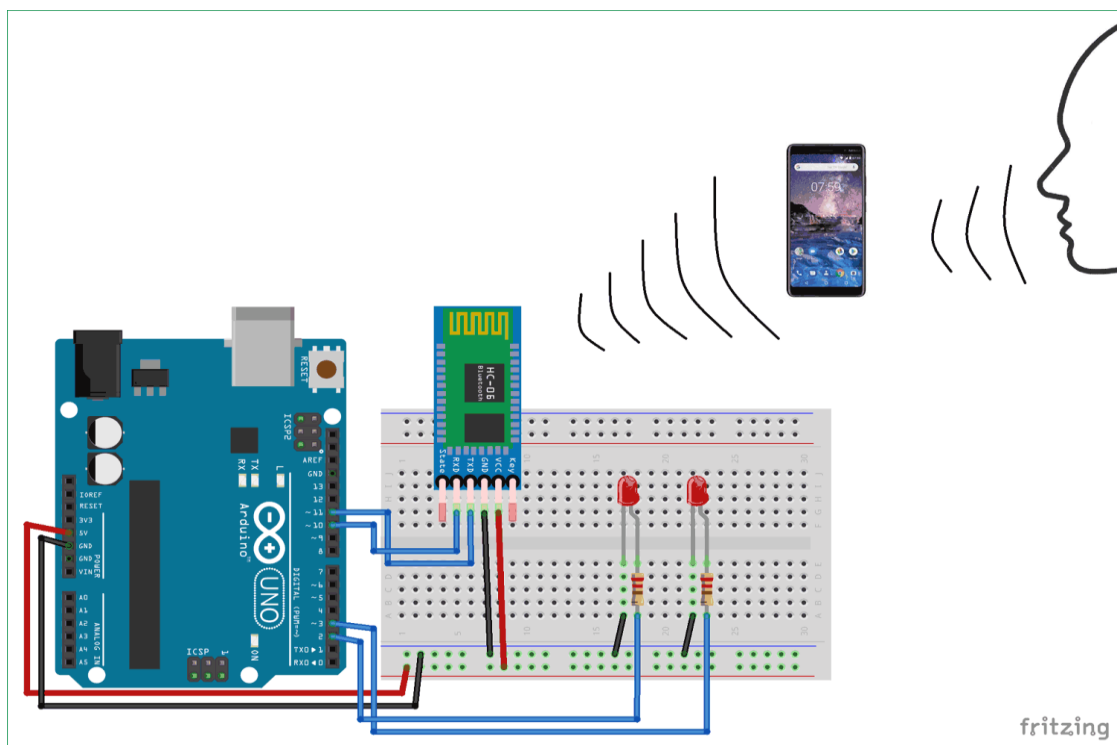
- Command Mode
- Operating Mode

In Command Mode we will be able to configure the Bluetooth properties like the name of the Bluetooth signal, its password, the operating baud rate etc. The Operating Mode is the one in which we will be able to send and receive data between the PIC Microcontroller and the Bluetooth module. Hence in this tutorial we will be toying only with the Operating Mode. The Command mode will be left to the default settings. The Device name will be HC-05 (I am using HC-06) and the password will be 0000 or 1234 and most importantly the default baud rate for all Bluetooth modules will be 9600. The module works on 5V supply and the signal pins operate on 3.3V, hence a 3.3V regulator is present in the module itself. Hence we need not worry about it. Out of the six pins only four will be used in the Operating mode. The pin connection table is shown below

S.no	Pin on HC-05/HC-06	Pin name on MCU	Pin number in PIC
1.	Vcc	Vdd	31st pin
2.	Vcc	Gnd	32st pin
3.	Tx	RC6/Tx/CK	25st pin
4.	Rx	RC7/Rx/DT	26st pin
5.	State	NC	NC
6.	EN(Enable)	NC	NC

Circuit Diagram

Circuit diagram for this Voice Controlled Lights is given below, while uploading the code in the Arduino UNO disconnect the Rx and Tx pins and connect again after the code is uploaded.



Working Procedure:

- Connect all components as per the circuit diagram; disconnect Rx and Tx pins while uploading the code.
- Download the app called “Arduino Bluetooth Voice Controller” which is free on play store.
- Open the app and follow the image below, like first click on “connect to Bluetooth device” and select your Bluetooth module and check if it is connected or not. Then click on the mic icon to speak and send the voice command to the HC-06 module.
- After setting up all the things, you just have to send the voice command by using the app which is further sent to Bluetooth module HC-06 and the HC-06 serially communicate with the Arduino UNO and then the task is performed as per the command. The below shows the command and the action to be performed by the command:

S.no	Command	Action
1.	all LED turn on	Both Red and Green LED turns ON
2.	all LED turn off	Both Red and Green LED turns OFF
3.	turn on Red LED	Red LED turns ON
4.	turn on green LED	Green LED turns ON
5.	turn off red LED	Red LED turns OFF
6.	turn off green LED	Green LED turns OFF

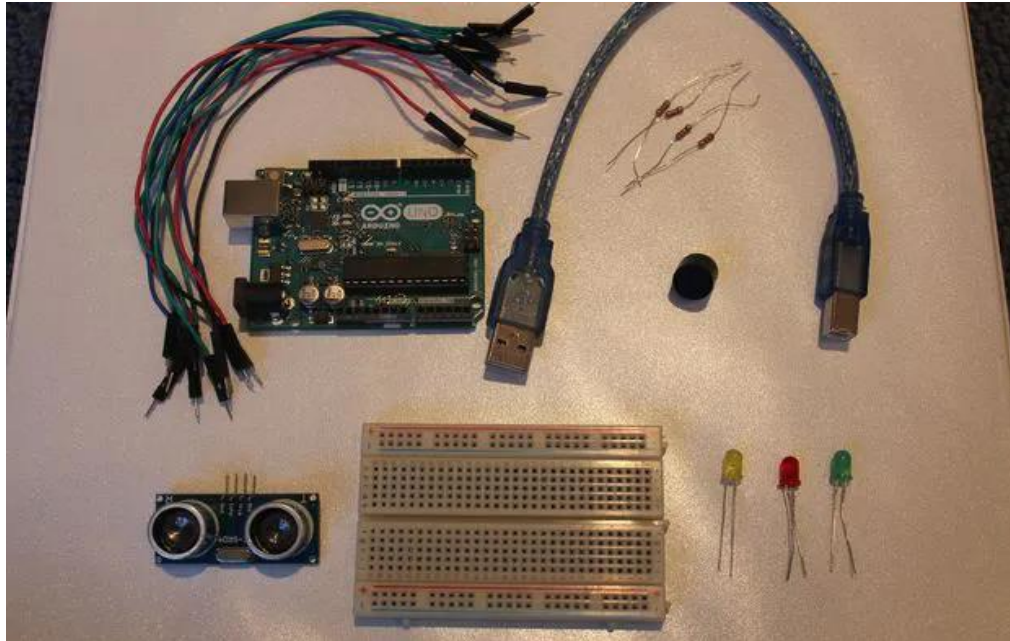
Source code

```
1.    int Green = 5;
2.    int Orange = 6;
3.    int Red = 7;
4.    void setup() {
5.        Serial.begin(9600);
6.        pinMode(Green,OUTPUT);
7.        pinMode(Orange,OUTPUT);
8.        pinMode(Red,OUTPUT);
9.    }
```

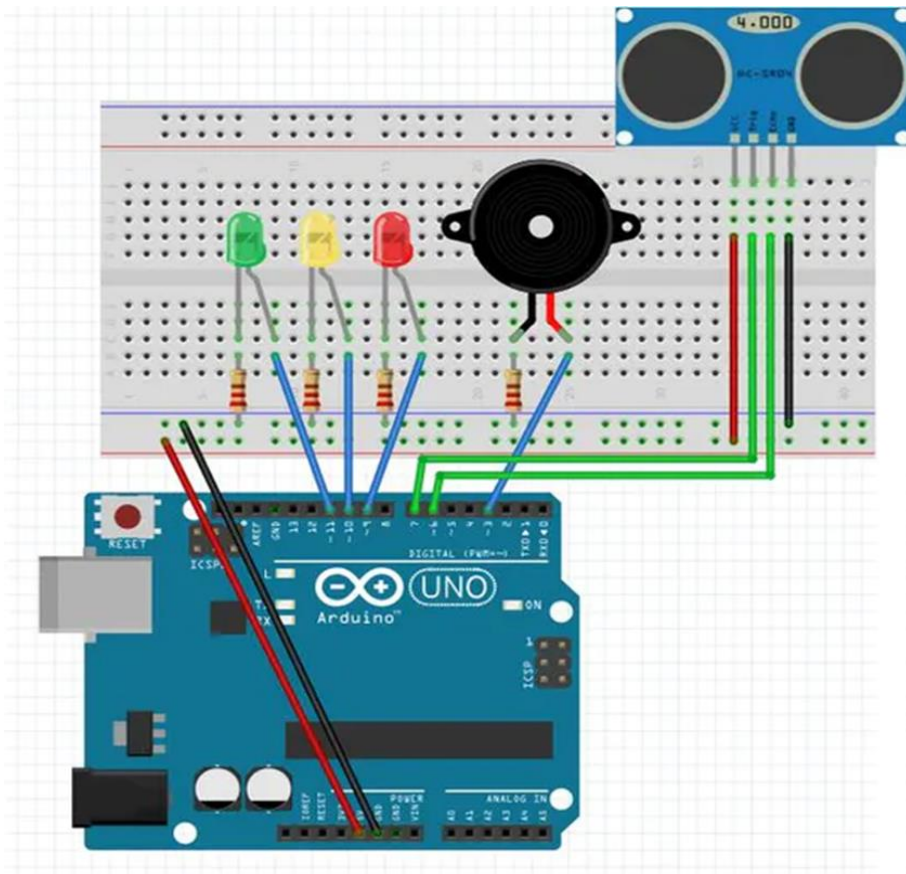
```
10.  char c;
11.  String voice;
12.  void loop() {
13.  if (Serial.available()>0)
14.  {
15.  voice="";
16.  voice=Serial.readString();
17.  Serial.print(voice+'\n');
18.  }
19.  if(voice=="green")
20.  {
21.  digitalWrite(Green,HIGH);
22.  } else if(voice=="green off")
23.  {
24.  digitalWrite(Green,LOW);
25.  }
26.  if(voice=="orange")
27.  {
28.  digitalWrite(Orange,HIGH);
29.  } else if(voice=="Orange off")
30.  {
31.  digitalWrite(Orange,LOW);
32.  }
33.  if(voice=="red")
34.  {
35.  digitalWrite(Red,HIGH);
36.  } else if(voice=="red off")
37.  {
38.  digitalWrite(Red,LOW);
39.  }
40.  }
```

S.no:5 Ultrasonic Security System

Step 1: Assemble Materials



Step 2: Setup



Connect a red wire from the 5V pin on the Arduino to the positive channel of the breadboard. Connect a black wire from the GND pin on the Arduino to the negative channel of the breadboard:

- Buzzer = pin 7

On Ultrasonic Sensor:

- Echo = pin 3
- Trig = pin 2

LEDs:

- RedLED = pin 4
- YellowLED = pin 5

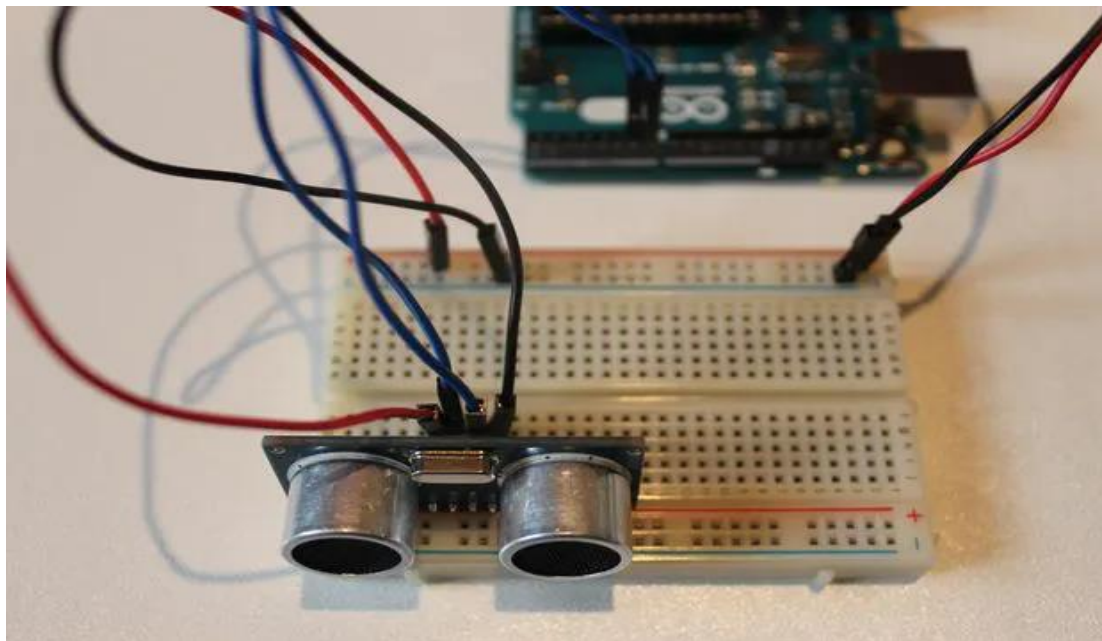
- GreenLED = pin 6

The green wires connected to the LEDs should be connected in line to the positive side of the LED, while the negative side of the LED should be connected to the negative channel of the breadboard using a 220 ohm resistor.

Step 3: Assembly – Breadboard

Firstly, let's connect the 5V and GND pin on the Arduino to the breadboard. As I mentioned before, be sure that the wire attached to the 5V pin is connected to the positive channel of the breadboard, and the wire attached to the GND pin is connected to the negative channel of the breadboard.

Step 4: Assembly - Ultrasonic Sensor



Time to connect the HC-SR04 ultrasonic sensor! A great tip is to place the ultrasonic sensor as far right to the breadboard as possible and make sure that it is facing out. Referring back to the setup picture, you should connect the GND pin on the ultrasonic sensor to the negative channel on the breadboard. Next connect the Trig pin on the sensor to pin 2 on the Arduino and connect the Echo pin on the sensor to pin 3 on the Arduino. Lastly, connect the VCC pin on the ultrasonic sensor to the positive channel on the breadboard. Refer to the picture above if anything gets confusing.

Step 5: Assembly - LEDs

The next step is to connect the LED's to the breadboard and Arduino. If you need to, I highly recommend that you refer back to the setup picture (Step 2), attaching the LEDs is pretty easy, there's a lot of repetition. Let's first attach the Green LED. So the way to do this, is to connect the anode (the longer leg) to pin 6 on the Arduino with a green wire, and to connect the cathode (the shorter leg) to the negative channel on the breadboard, using a 220 ohm resistor. Then repeat that step for the Yellow and then the Red LED, make sure to connect the anode (the longer leg) of the yellow LED to pin 5 on the Arduino and then connect the anode of the red LED to pin 6. Once you have done that, your setup should look similar to the picture above. Resistors are not absolutely necessary, however they are highly recommended to be used.

Step 6: Assembly - Buzzer

The last part of the setup for this, is connecting the buzzer to the breadboard and the Arduino. This is one of the easiest parts of the whole setup. All that is required to do is to connect the longer leg of the buzzer to pin 7 of the Arduino using a green wire and then connect the shorter leg of the buzzer to the negative channel of the breadboard using a 220 ohm resistor. It is HIGHLY recommended to use a resistor in connecting the shorter leg of the buzzer to the negative channel of the breadboard. This greatly reduces the volume of the buzzer and prevent it from dying to quickly.

Source code

```
1) #define trigPin 2
2) #define echoPin 3
3) #define LEDlampRed 4
4) #define LEDlampYellow 5
5) #define LEDlampGreen 6
6) #define soundbuzzer 7
7) int sound = 500;
8) void setup() {
9)   Serial.begin (9600);
10)  pinMode(trigPin, OUTPUT);
```

```

11) pinMode(echoPin, INPUT);
12) pinMode(LEDlampRed, OUTPUT);
13) pinMode(LEDlampYellow, OUTPUT);
14) pinMode(LEDlampGreen, OUTPUT);
15) pinMode(soundbuzzer, OUTPUT);
16) }
17) void loop() {
18) long durationindigit, distanceincm;
19) digitalWrite(trigPin, LOW);
20) delayMicroseconds(2);
21) digitalWrite(trigPin, HIGH);
22) delayMicroseconds(10);
23) digitalWrite(trigPin, LOW);
24) durationindigit = pulseIn(echoPin, HIGH);
25) distanceincm = (durationindigit/5) / 29.1;
26) if (distanceincm < 50) {
27) digitalWrite(LEDlampGreen, HIGH);
28) }
29) else {
30) digitalWrite(LEDlampGreen, LOW);
31) }
32) if (distanceincm < 20) {
33) digitalWrite(LEDlampYellow, HIGH);
34) }
35) else {
36) digitalWrite(LEDlampYellow,LOW);
37) }
38) if (distanceincm < 5) {
39) digitalWrite(LEDlampRed, HIGH);
40) sound = 1000;
41) }

```

```
42) else {  
43) digitalWrite(LEDlampRed,LOW);  
44) }  
45) if (distanceincm > 5 || distanceincm <= 0){  
46) Serial.println("Outside the permissible range of distances");  
47) noTone(soundbuzzer);  
48) }  
49) else {  
50) Serial.print(distanceincm);  
51) Serial.println(" cm");  
52) tone(soundbuzzer, sound);  
53) }  
54) delay(300);  
55) }
```

S.no:6 Google Assistant and Blynk in IoT based Home Automation

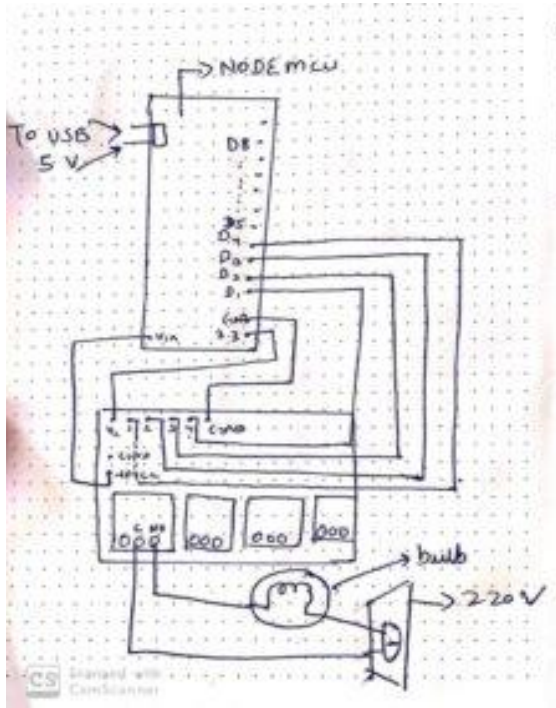
In this IoT project, I have shown how to control home appliances from Google Assistant by connecting the Google Assistant with Blynk App using the IFTTT. With this IoT based project, we can control the relay module from the smartphone using Blynk or Google Assistant.



How the project works:

1. If you say any phrase in the Google Assistant, then it will generate a trigger in IFTTT.
2. IFTTT will send the signal to Blynk Cloud server through Websockets
3. Blynk will control the NodeMCU to turn on or turn off the respective relays

Circuit for this IoT project

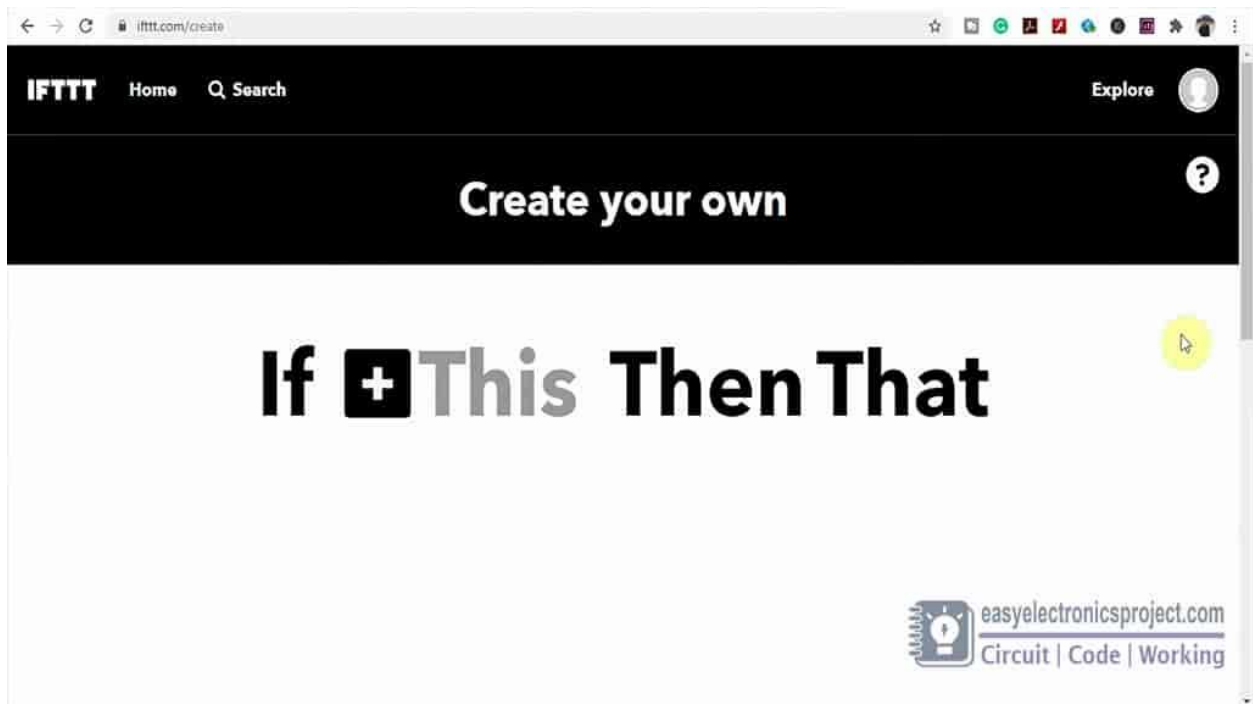


Source code

```
1. #define BLYNK_PRINT Serial
2. #include <ESP8266WiFi.h>
3. #include <BlynkSimpleEsp8266.h>
4. char auth[] = "KG6_oJPNA5ipxYATC4Hk7QKyboEMyD_i"; // the auth code that you
   got on your gmail
5. char ssid[] = "NETGEAR_JNVR"; // username or ssid of your WI-FI
6. char pass[] = "25353116"; // password of your Wi-Fi
7. void setup()
8. {
9. // Debug console
10. Serial.begin(9600);
11. pinMode(D1,OUTPUT); //extend these to D8 if you are using a 8 pin relay
12. pinMode(D2,OUTPUT);
13. pinMode(D3,OUTPUT);
14. pinMode(D4,OUTPUT);
```

```
15. digitalWrite(D1,HIGH);// Make it low if you want everything to go off
16. digitalWrite(D2,HIGH); // in case of a power cut
17. digitalWrite(D3,HIGH);
18. digitalWrite(D4,HIGH);
19. Blynk.begin(auth, ssid, pass);
20. }
21. void loop()
22. {
23. Blynk.run();
24. }
```

Connecting Blynk with Google Assistant



Here, I will use IFTTT to connect google assistant with Blynk.

Steps to connect Google Assistant with Blynk:

← → ↻ ifttt.com/create/if-say-a-simple-phrase?sid=3

← Back

Complete trigger fields

Step 2 of 6

What do you want to say?

Switch on light

What's another way to say it? (optional)

light on

And another way? (optional)

What do you want the Assistant to say in response?

easyelectronicproject.com
Circuit | Code | Working

Open IFTTT and search for Google Assistant and enter what do you want to say to control the relay module and the response from the google assistant. Then click on Create Trigger.

← → ↻ ifttt.com/create/if-say-a-simple-phrase-then-make-a-web-request?sid=8

← Back

Complete action fields

Step 5 of 6

URL

http://188.166.206.43/ISPTShUpdpJXHi1R3NeRXMcIqCcRbrE5/update/D5

Surround any text with "<<>>" to escape the content

Add ingredient

Method

PUT

The method of the request e.g. GET, POST, DELETE

Content Type

Please select

easyelectronicproject.com
Circuit | Code | Working

After that search for Webhooks. Now, to generate the URL I need the IP of the Blynk-cloud. So to get the IP, I have to open the command prompt and type ping blynk-cloud.com.

URL structure: http://(Blynk-cloud IP)/(Auth Token)/update/D(GPIO pin)

Method: PUT

Content Type: Application/json

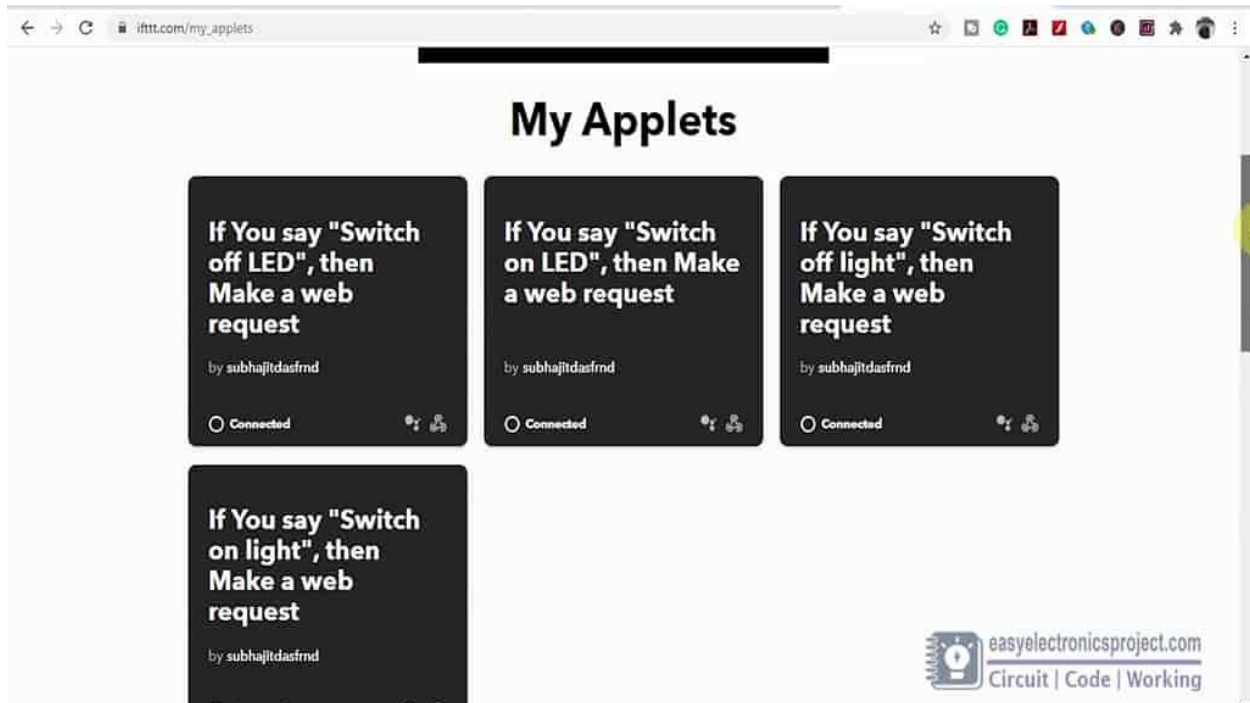
Body:

[“0”] → To turn on the Relay (Active Low)

[“1”] → To turn off the Relay (Active Low)

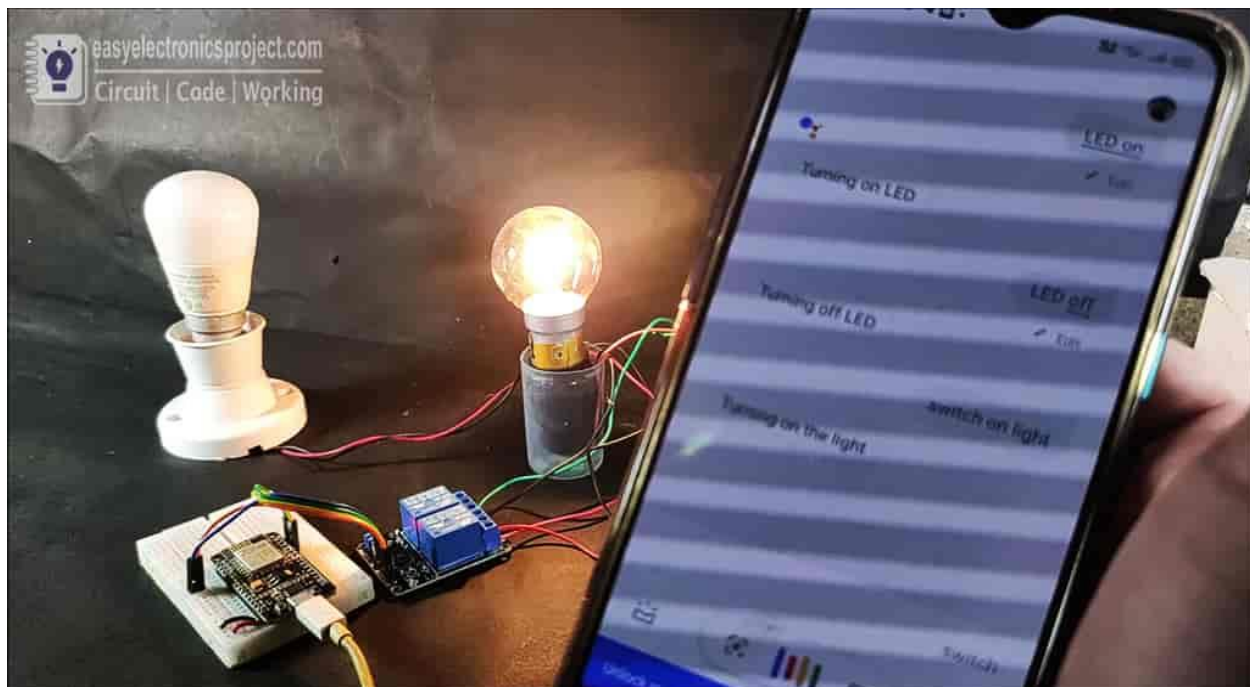
Best Pins for Output (Best to worst)	
Board Label	Raw Pin Number
D1	GPIO5
D2	GPIO4
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15

Then click on the Create action.



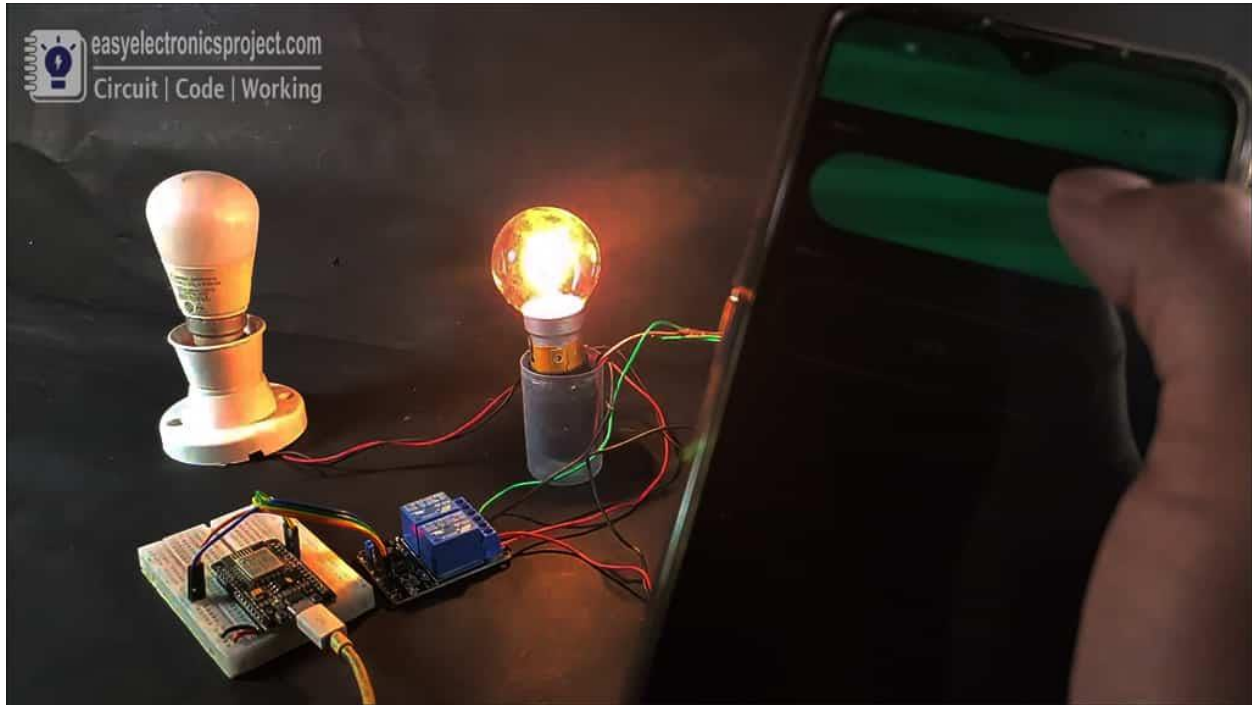
Thus, I have created 4 applets to control 2-channel relay module from google assistant. You can create multiple applets as per the requirement.

Google Assistant control Relay:



As per the applets I have created on the IFTTT, if I say “Switch on Light” the relay-1 will turn on, and to turn off the relay-1, I have to say “Switch off Light“.And to control the relay-2, I have to say “Switch on LED” or “Switch off LED”.

Blynk control Relay:



I can also control the relay module from Blynk App. So if I have internet in my smartphone, then I can control the relay module from anywhere in the world.

S.no:7 Arduino Ultrasonic sensor hc-sr04 Measure distance with LEDs

The Ultrasonic sensor is very popular and used to measure the distance to an object. It comes complete with ultrasonic transmitter and receiver modules. Hc-sr04 transmit high-frequency sound and when an object detects then reflect the signal to echo.

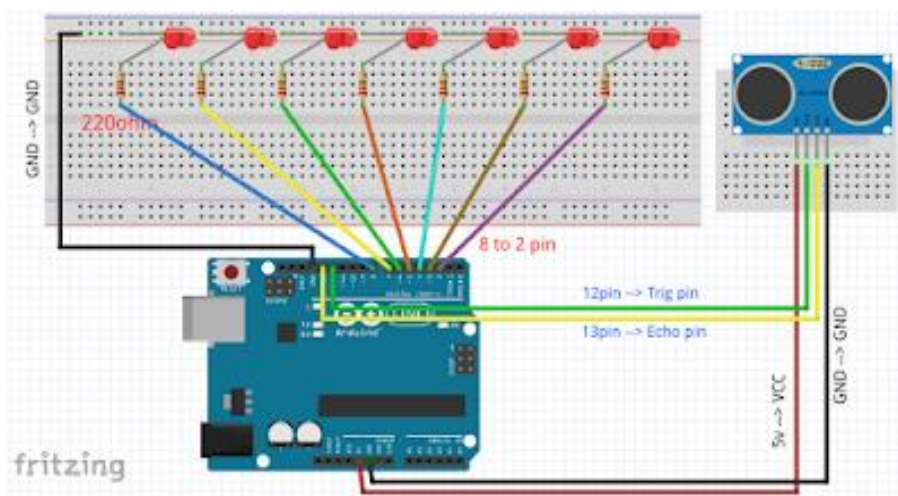
Features:

1. Distance measure: 2cm to 400cm.
2. Power Supply: +5V DC.
3. Measuring Angle: 30 degree.
4. uses as: Obstacle avoiding robot, distance measure, measure water level and many more.

Requirement:

1. Arduino Uno
2. Ultrasonic sensor Hc-sr04
3. 7 x LEDs
4. 7 x 220ohm resistor
5. Breadboard
6. jumper wire

Circuit for this IoT project:



Source code:

```
1. const int trig = 12;
2. const int echo = 13;
3. const int LED1 = 8;
4. const int LED2 = 7;
5. const int LED3 = 6;
6. const int LED4 = 5;
7. const int LED5 = 4;
8. const int LED6 = 3;
9. const int LED7 = 2;
10. int duration = 0;
11. int distance = 0;
12. void setup()
13. {
14.   pinMode(trig , OUTPUT);
15.   pinMode(echo , INPUT);
16.   pinMode(LED1 , OUTPUT);
17.   pinMode(LED2 , OUTPUT);
18.   pinMode(LED3 , OUTPUT);
19.   pinMode(LED4 , OUTPUT);
20.   pinMode(LED5 , OUTPUT);
21.   pinMode(LED6 , OUTPUT);
22.   pinMode(LED7 , OUTPUT);
23.   Serial.begin(9600);
24. }
25. void loop()
26. {
27.   digitalWrite(trig , HIGH);
28.   delayMicroseconds(1000);
29.   digitalWrite(trig , LOW);
30.   duration = pulseIn(echo , HIGH);
```

```
31. distance = (duration/2) / 28.5 ;
32. Serial.println(distance);
33. if ( distance <= 7 )
34. {
35. digitalWrite(LED1, HIGH);
36. }
37. else
38. {
39. digitalWrite(LED1, LOW);
40. }
41. if ( distance <= 14 )
42. {
43. digitalWrite(LED2, HIGH);
44. }
45. else
46. {
47. digitalWrite(LED2, LOW);
48. }
49. if ( distance <= 21 )
50. {
51. digitalWrite(LED3, HIGH);
52. }
53. else
54. {
55. digitalWrite(LED3, LOW);
56. }
57. if ( distance <= 28 )
58. {
59. digitalWrite(LED4, HIGH);
60. }
61. else
```

```
62. {
63. digitalWrite(LED4, LOW);
64. }
65. if ( distance <= 35 )
66. {
67. digitalWrite(LED5, HIGH);
68. }
69. else
70. {
71. digitalWrite(LED5, LOW);
72. }
73. if ( distance <= 42 )
74. {
75. digitalWrite(LED6, HIGH);
76. }
77. else
78. {
79. digitalWrite(LED6, LOW);
80. }
81. if ( distance <= 49 )
82. {
83. digitalWrite(LED7, HIGH);
84. }
85. else
86. {
87. digitalWrite(LED7, LOW);
88. }
89. }
```