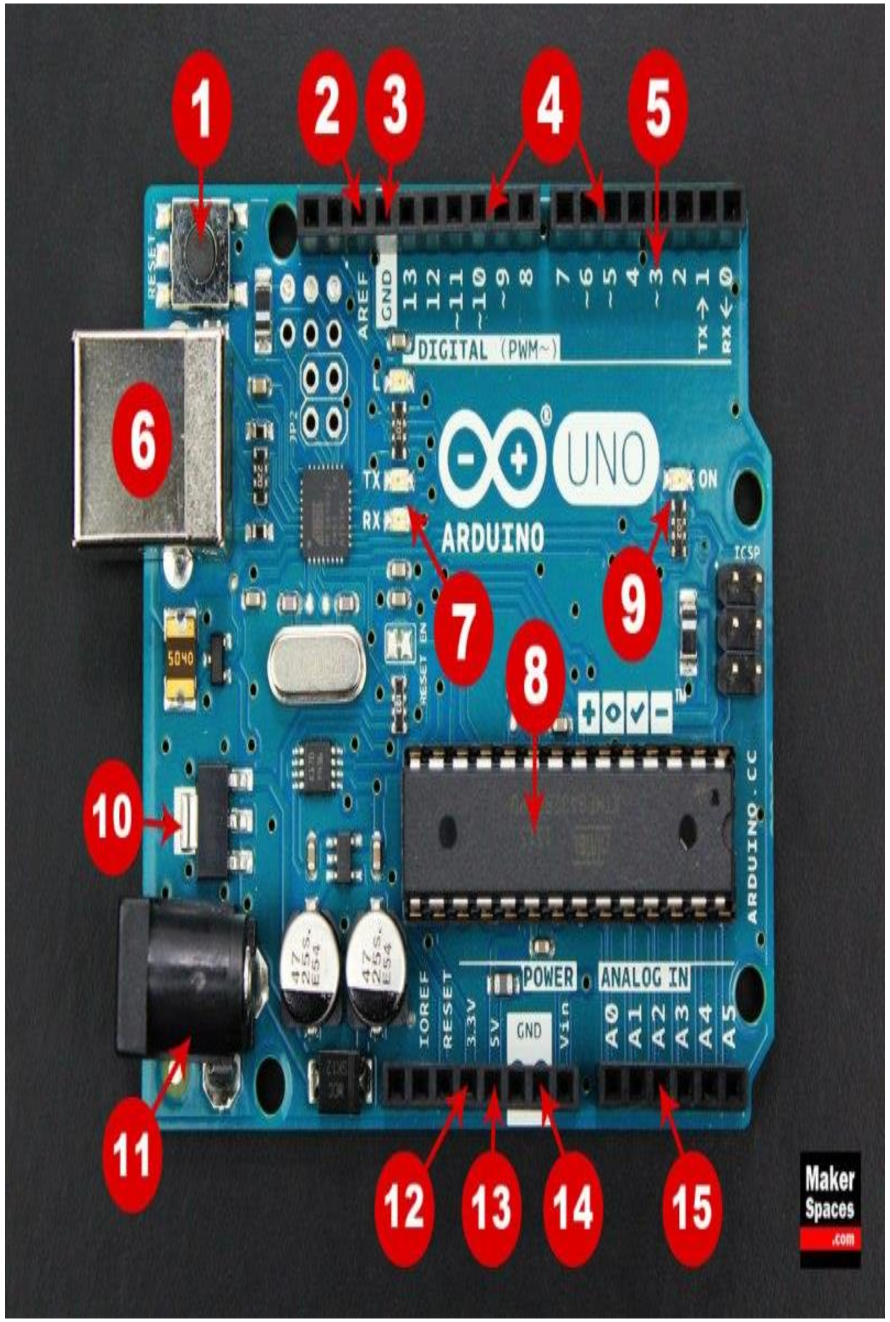
Arduino Uno

One of the most popular Arduino boards out there is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market. Because of its extreme popularity, the Arduino Uno has a ton of project tutorials and forums around the web that can help you get started or out of a jam. We're big fans of the Uno because of it's great features and ease of use.

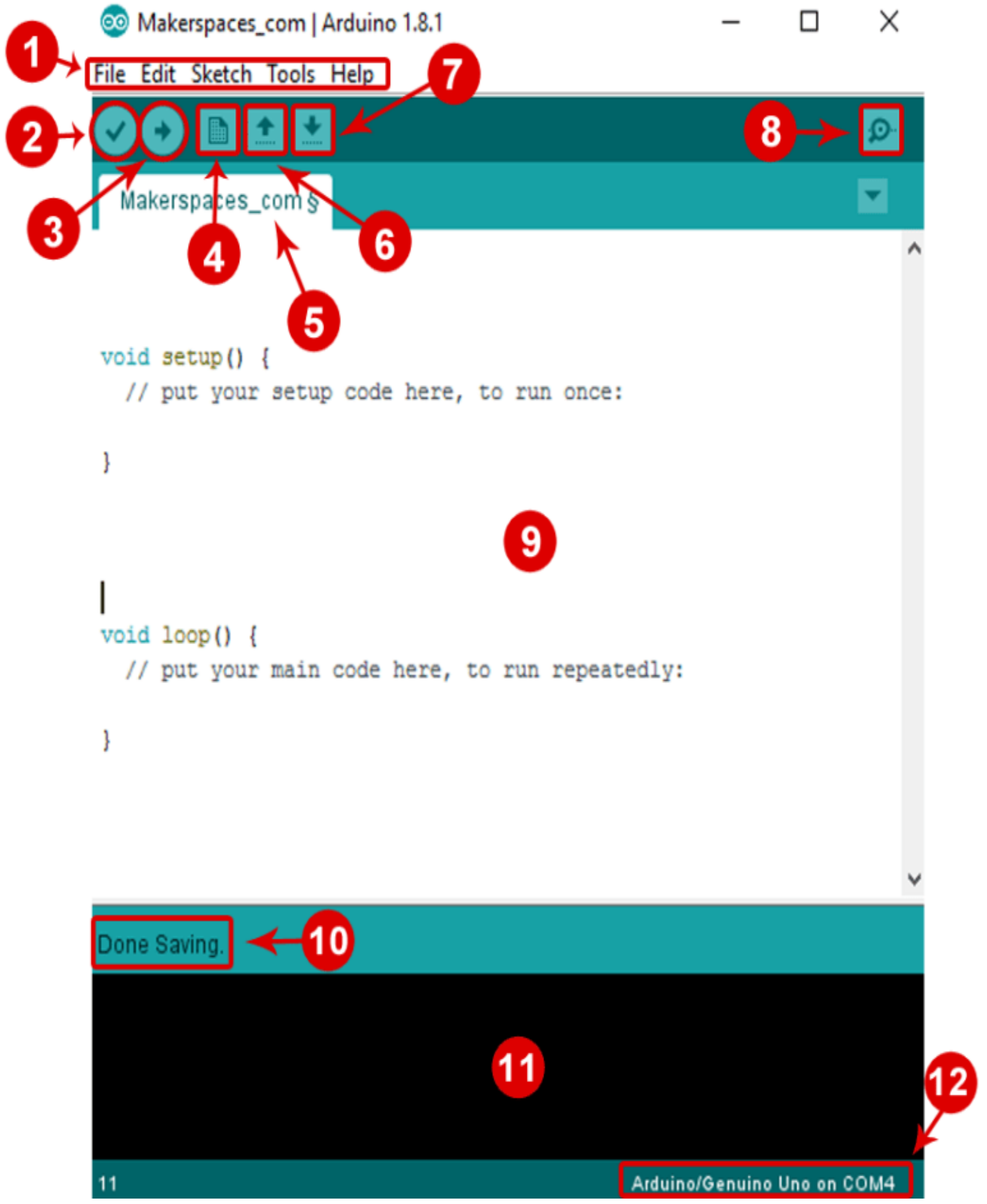


Board Breakdown

Here are the components that make up an Arduino board and what each of their functions are.

1. **Reset Button** – This will restart any code that is loaded to the Arduino board
2. **AREF** – Stands for "Analog Reference" and is used to set an external reference voltage
3. **Ground Pin** – There are a few ground pins on the Arduino and they all work the same
4. **Digital Input/Output** – Pins 0-13 can be used for digital input or output
5. **PWM** – The pins marked with the (~) symbol can simulate analog output
6. **USB Connection** – Used for powering up your Arduino and uploading sketches
7. **TX/RX** – Transmit and receive data indication LEDs
8. **ATmega Microcontroller** – This is the brains and is where the programs are stored
9. **Power LED Indicator** – This LED lights up anytime the board is plugged in a power source
10. **Voltage Regulator** – This controls the amount of voltage going into the Arduino board
11. **DC Power Barrel Jack** – This is used for powering your Arduino with a power supply
12. **3.3V Pin** – This pin supplies 3.3 volts of power to your projects
13. **5V Pin** – This pin supplies 5 volts of power to your projects
14. **Ground Pins** – There are a few ground pins on the Arduino and they all work the same
15. **Analog Pins** – These pins can read the signal from an analog sensor and convert it to digital

Once the software has been installed on your computer, go ahead and open it up. This is the Arduino IDE and is the place where all the programming will happen.Take some time to look around and get comfortable with it.


Makerspaces_com | Arduino 1.8.1
File Edit Sketch Tools Help
Makerspaces_com §
1
2
3
4
5
6
7
8
9
10
11
12
Done Saving.
11
Arduino/Genuino Uno on COM4

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

1. **Menu Bar:** Gives you access to the tools needed for creating and saving Arduino sketches.
2. **Verify Button:** Compiles your code and checks for errors in spelling or syntax.
3. **Upload Button:** Sends the code to the board that's connected such as Arduino Uno in this case. Lights on the board will blink rapidly when uploading.
4. **New Sketch:** Opens up a new window containing a blank sketch.
5. **Sketch Name:** When the sketch is saved, the name of the sketch is displayed here.
6. **Open Existing Sketch:** Allows you to open a saved sketch or one from the stored examples.
7. **Save Sketch:** This saves the sketch you currently have open.
8. **Serial Monitor:** When the board is connected, this will display the serial information of your Arduino
9. **Code Area:** This area is where you compose the code of the sketch that tells the board what to do.
10. **Message Area:** This area tells you the status on saving, code compiling, errors and more.
11. **Text Console:** Shows the details of an error messages, size of the program that was compiled and additional info.
12. **Board and Serial Port:** Tells you what board is being used and what serial port it's connected to.

Now let’s discuss the basics of Arduino programming.
The structure of Arduino program is pretty simple. Arduino programs have a minimum of 2 blocks,
Preparation & Execution
Each block has a set of statements enclosed in curly braces:

```
void setup( )
{
statements-1;
statement-n;
}
void loop ( )
{
statement-1;
statement-n;
}
```

Here, setup ( ) is the preparation block and loop ( ) is an execution block.

The setup function is the first to execute when the program is executed, and this function is called only once. The setup function is used to initialize the pin modes and start serial communication. This function has to be included even if there are no statements to execute.

```
void setup ( )

{

pinMode (pin-number, OUTPUT); // set the ‘pin-number’ as output

pinMode (pin-number, INPUT); // set the ‘pin-number’ as output

}
```

After the setup ( ) function is executed, the execution block runs next. The execution block hosts statements like reading inputs, triggering outputs, checking conditions etc..

In the above example loop ( ) function is a part of execution block. As the name suggests, the loop( ) function executes the set of statements (enclosed in curly braces) repeatedly.

```
Void loop ( )

{

digitalWrite (pin-number,HIGH); // turns ON the component connected to ‘pin-number’

delay (1000); // wait for 1 sec

digitalWrite (pin-number, LOW); // turns OFF the component connected to ‘pin-number’

delay (1000); //wait for 1sec

}
```

Note: Arduino always measures the time duration in millisecond. Therefore, whenever you mention the delay, keep it in milli seconds.

Now, let’s take a giant leap and do some experiments with Arduino

- Blinking the LED
- Fade-in and fade-out the LED

In the process of experimenting with Arduino, writing the Arduino program is not the only important thing, building the breadboard circuit is equally important.

Let’s take a look at how the breadboard circuit has to be built for both the experiments.

Components required:

- Arduino UNO R3 -1
- Breadboard -1
- Breadboard connectors -3
- LED -1
- 1K resistor -1

Blinking LED

Steps in building a breadboard connection:

Step-1: Connect the Arduino to the Windows / Mac / Linux system via a USB cable

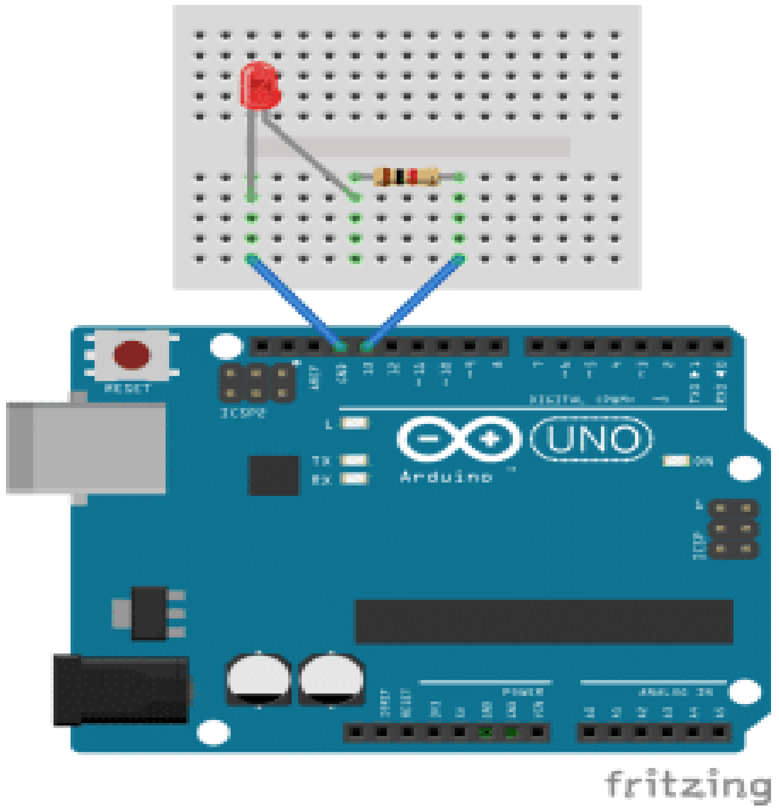
Step-2: Connect the 13th digital pin of Arduino to the positive power rail of the breadboard and GND to the negative

Step-3: Connect the positive power rail to the terminal strip via a 1K ohm resistor

Step-4: Fix the LED to the ports below the resistor connection in the terminal strip

Step-5: Close the circuit by connecting the cathode (the short chord) of the LED to the negative power strip of the breadboard

Arduino program for LED blink (Version-1)



```
int LED =13; // The digital pin to which the LED is connected


void setup ( )
{


pinMode (LED, OUTPUT); //Declaring pin 13 as output pin


}


void loop( ) // The loop function runs again and again
{

digitalWrite (LED, HIGH); //Turn ON the LED
delay(1000); //Wait for 1sec
digitalRead (LED, LOW); // Turn off the LED
delay(1000); // Wait for 1sec

}
```

Arduino program for LED blink (Version-2)

```
void setup ( )
{

pinMode (13, OUTPUT); //pin 13 is set as output pin

}

void loop( ) // The loop function runs again and again
{

digitalWrite (13,HIGH); // Turn ON the LED on pin 13
delay (1000); //Wait for 1sec
digitalWrite (13, LOW); //Turn OFF the LED on pin 13


}
```

In version-1 of the LED blink program LED is declared globally and is set to pin number 13. This will reduce the number of iterations required to update the pin number in the program when you connect the LED to the other digital pin. Whereas, the pin number has to be changed in 3 different statements in version-2.

The magic happens when you click the upload icon in the Arduino IDE. The program will be uploaded into the microcontroller of Arduino board and LED in the circuit starts blinking.

Fade-in and fade-out the LED

The step for the LED bling experiment are the same as those followed for building the breadboard cir cuit except that in step-2, you connect the 9th digital pin to the positive power rail of the breadboard.

Arduino program for LED fade-in and fade-out (Version-1)

```
int led = 9; // The digital pin to which the LED is connected
int brightness = 0; // Brightness of LED is initially set to 0
int fade = 5; // By how many points the LED should fade
 void setup()
{ pinMode(led, OUTPUT); //pin 10 is set as output pin
}
 void loop() // The loop function runs again and again
{ analogWrite(led, brightness); // set the brightness of LED
 brightness = brightness + fade; //Increase the brightness of LED by 5 points
 if (brightness <= 0 || brightness >= 255) // check the level of brightness
 {   fade = -fade;   }
delay(30); // Wait for 30 milliseconds
}
```

The same output could be generated by modifying the above program.

Arduino program for LED fade-in and fade-out (Version-2)

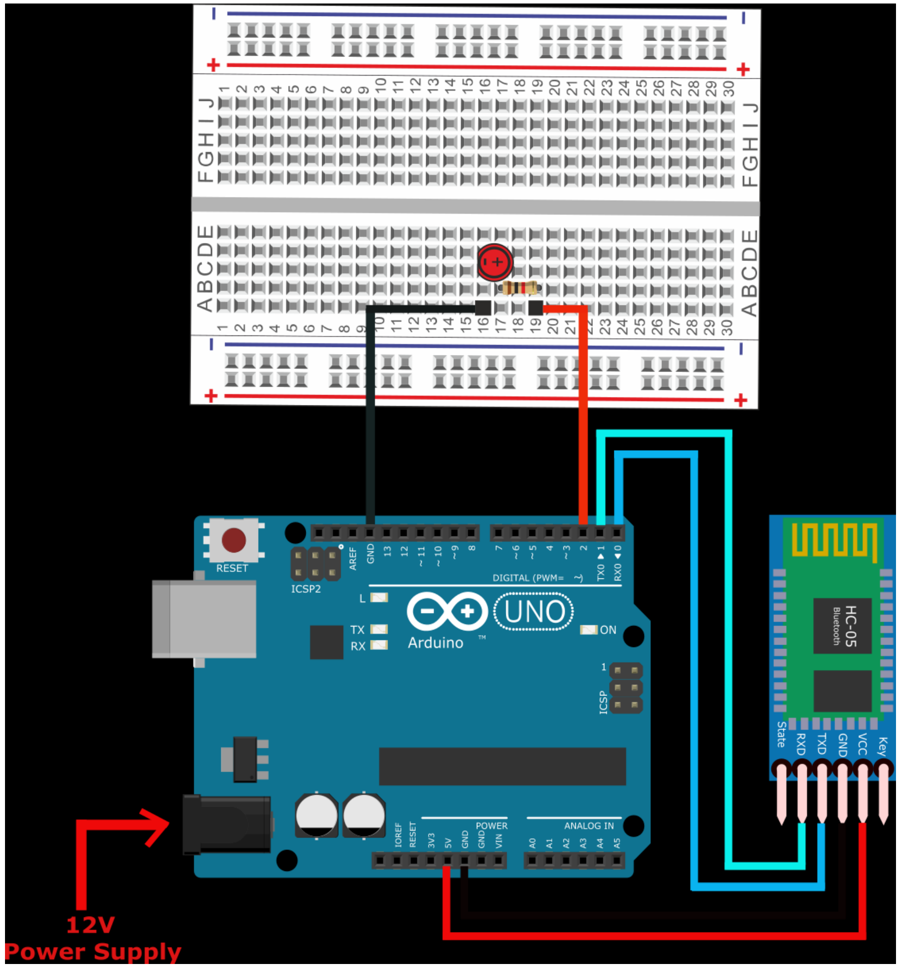
```

int led=19; // The digital pin to which the LED is connected
 void setup()
{
 pinMode(led, OUTPUT); //pin 10 is set as output pin
 }
 void loop() // The loop function runs again and again
{
 for (int fade=0; fade<=255; fade=fade+5)
{
 analogWrite (led, fade); // Change the brightness of LED by 5 points
delay (30);
 }
}

```

In both the versions of the LED fade-in and fade-out programs, the analogWrite statement is used for a led connected to a digital pin. The reason for this is that, digital pin 10 is a PWM pin. As discussed in the previous article, A tour of the Arduino UNO board, a PWM pin is capable of generating Analog output. In both the programs pin 10 is used as analog output pin.

Project 2.HC-05 Bluetooth controlled LED

Download Arduino Bluetooth app//

https://play.google.com/store/apps/details?id=com.satech.arduinocontroller

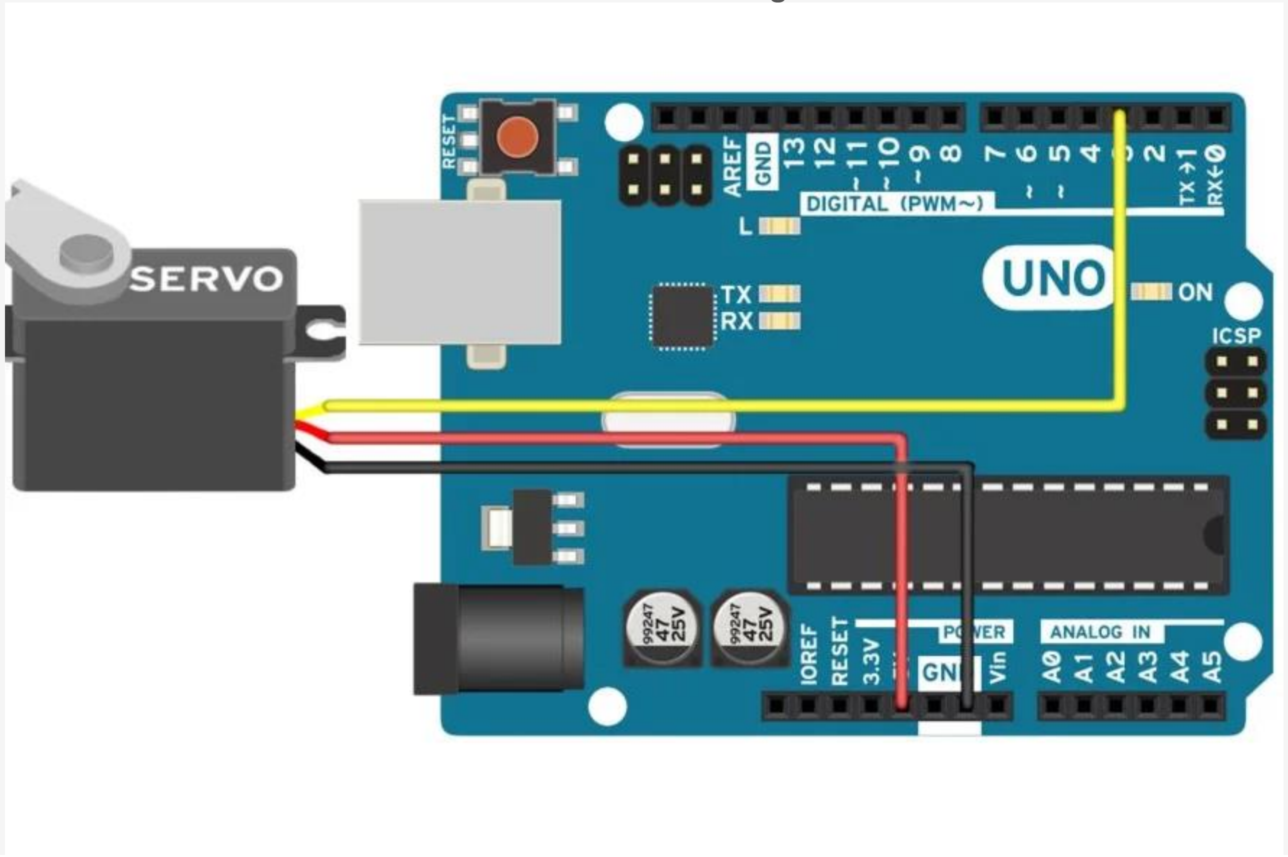


```
String BT_input;                              // to store input character received via BT.
int LED = 2;                                  // device to control
void setup()
{
  Serial.begin(9600);                         //default baud rate of module
  pinMode(LED, OUTPUT);
  while (!Serial)
      {      // wait for serial port to connect. Needed for native USB port only
         }
  }
void loop()
 {   if (Serial.available())
    {
        BT_input = Serial.readString();   // read input string from bluetooth
                if (BT_input=="A")
            {
          digitalWrite(LED, HIGH);
          Serial.println(BT_input);
          Serial.println("LED is ON");
            }
        else if (BT_input=="B")
        {
          digitalWrite(LED, LOW);
          Serial.println(BT_input);
          Serial.println("LED is OFF");
        }
        else
        {
          Serial.println(BT_input);
          Serial.println("Send 'A' to get LED ON");
          Serial.println("Send 'B' to get LED OFF");
        }
    }
 }
```

Voice controlled led on off

```
String readString;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(6, OUTPUT);
}

void loop() {
  // put your main code here, tao run repeatedly:
while(Serial.available()){
  delay(3);
  char c = Serial.read();
  readString+=c;
}

if(readString.length() >0)
{
  Serial.println(readString);
  if(readString == "hello")
  {
  digitalWrite(6, HIGH);
  }
 else if(readString == "not")
  {
  digitalWrite(6, LOW);
  }
  readString = "";
}
}
```

Servo Controlling

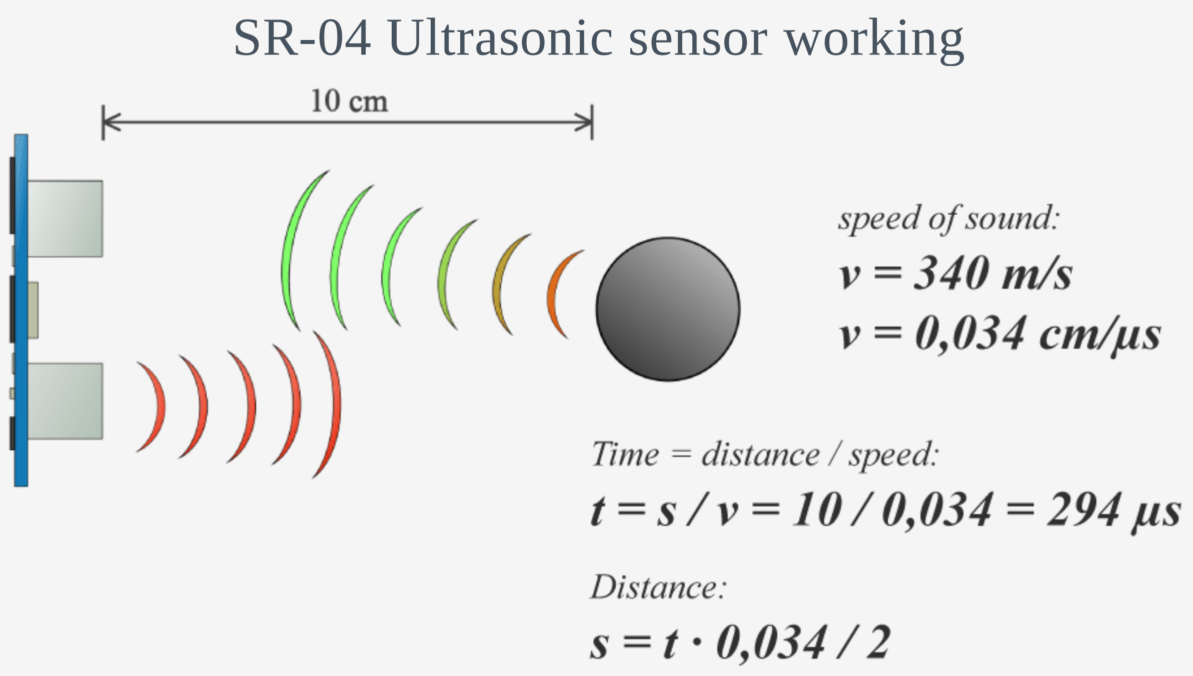


A servo motor has everything built in: a motor, a feedback circuit, and most important, a motor driver. It just needs one power line, one ground, and one control pin.

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.

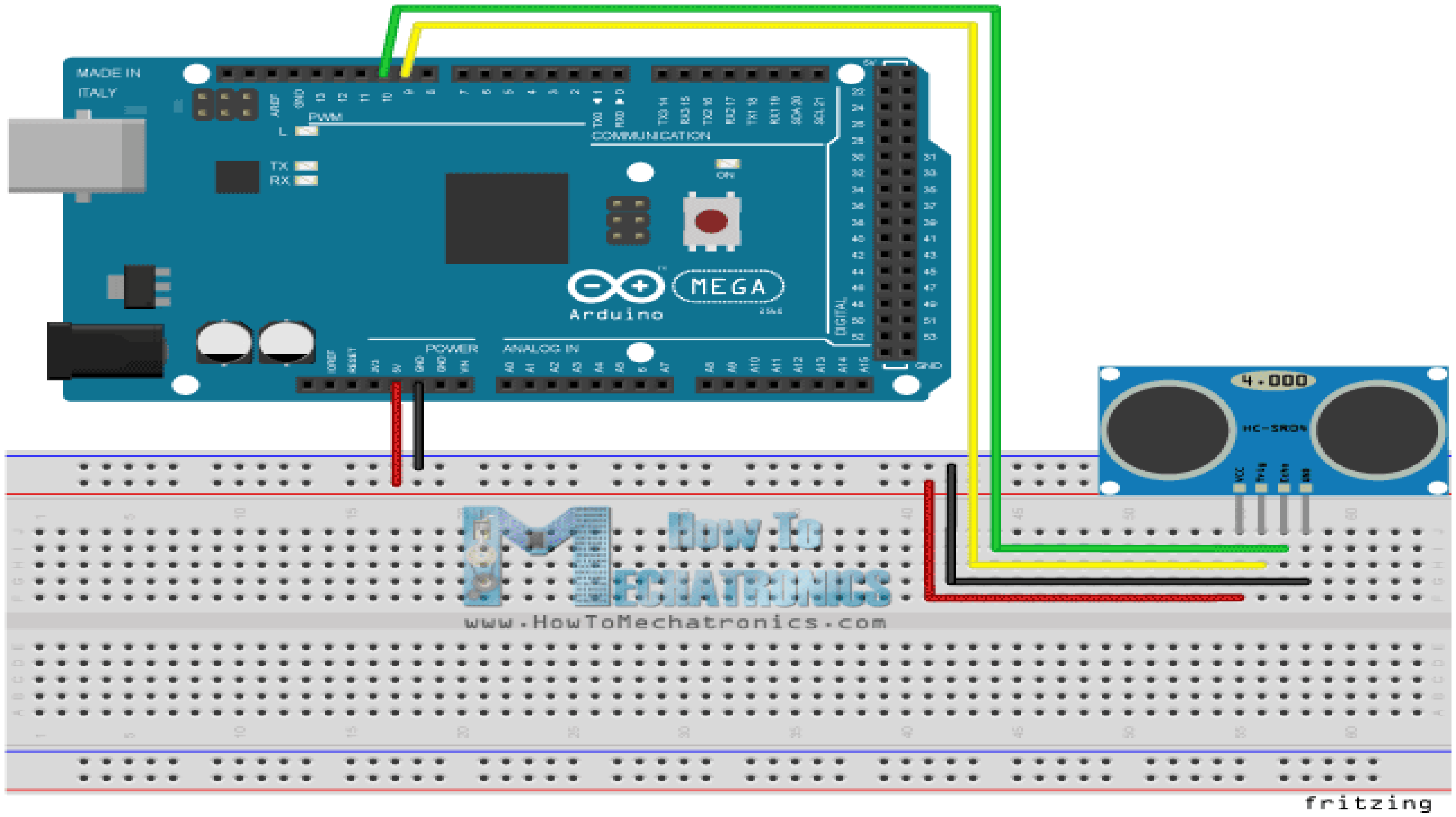
```
// Include the Servo library
#include <Servo.h>
// Declare the Servo pin
int servoPin = 3;
// Create a servo object
Servo Servo1;
void setup() {
   // We need to attach the servo to the used pin number
   Servo1.attach(servoPin);
}
void loop(){
   // Make servo go to 0 degrees
   Servo1.write(0);
   delay(1000);
   // Make servo go to 90 degrees
   Servo1.write(90);
   delay(1000);
   // Make servo go to 180 degrees
   Servo1.write(180);
   delay(1000);
}
```



1.
2. It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

3.
4. In order to generate the ultrasound you need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.



MADE IN ITALY
PWM
COMMUNICATION
MEGA
Arduino
POWER
ANALOG IN
DIGITAL
GND
HC-SR04
Vcc
Trig
Echo
Gnd
How To Mechatronics
www.HowToMechatronics.com
fritzing

```
const int trigPin = 9;
const int echoPin = 10;

// defines variables
        long duration;
        int distance;

        void setup() {
        pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
        pinMode(echoPin, INPUT); // Sets the echoPin as an Input
        Serial.begin(9600); // Starts the serial communication
        }

        void loop() {
        // Clears the trigPin
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);

        // Sets the trigPin on HIGH state for 10 micro seconds
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);

        // Reads the echoPin, returns the sound wave travel time in microseconds
        duration = pulseIn(echoPin, HIGH);

        // Calculating the distance
        distance= duration*0.034/2;

        // Prints the distance on the Serial Monitor
        Serial.print("Distance: ");
        Serial.println(distance);
        }
```