



PHYSICAL COMPUTING

GETTING STARTED GUIDE - PART 2



I'LL MAKE UBIQUITY

I'll make ubiquity is the beginners guide developed and created to assist students to understand and work with technology.

It contains a step wise introduction into one of the technology platforms - Genuino 101.

We urge you to learn & understand how to work with one platform and then go out and explore further.





WHAT'S INSIDE?

1

Tutorials

- Getting Started
- Controlling LED
- Fairy Lights
- Push Button

2

Hands on

- Accelerometer
- Gyroscope

3

Exercise

- Bluetooth works
- Pedometer
- Visual Programming





TUTORIALS



GETTING STARTED



SETTING UP SOFTWARE

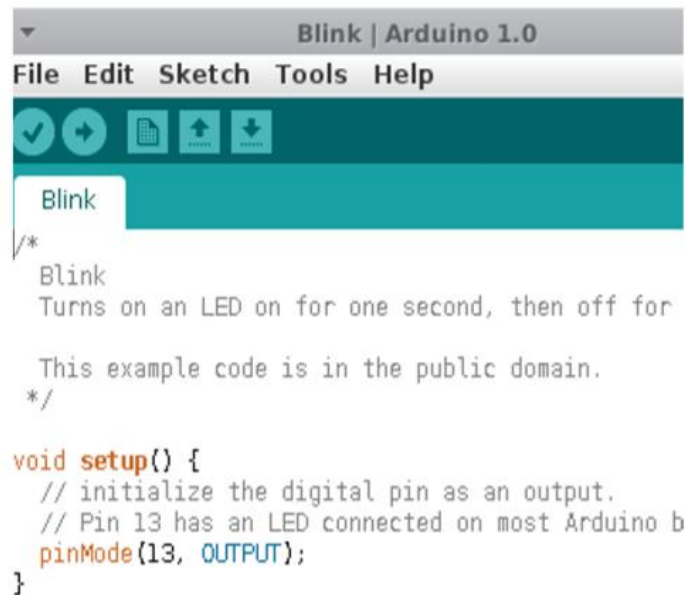
The Genuino 101 board is designed specifically for students. It is easy and simple to use and comes jam-packed with Arduino* sensors and connectivity, making it perfect for young innovators and creators.



Install the latest version of Arduino IDE (Software Package). You can access the latest version here:
<https://www.arduino.cc/en/Main/Software>



Install the Intel® Curie board. To gain access to examples and functions of the Genuino 101 board you will need to install the Intel® Curie board Libraries. [Click on the Tools menu then](#)



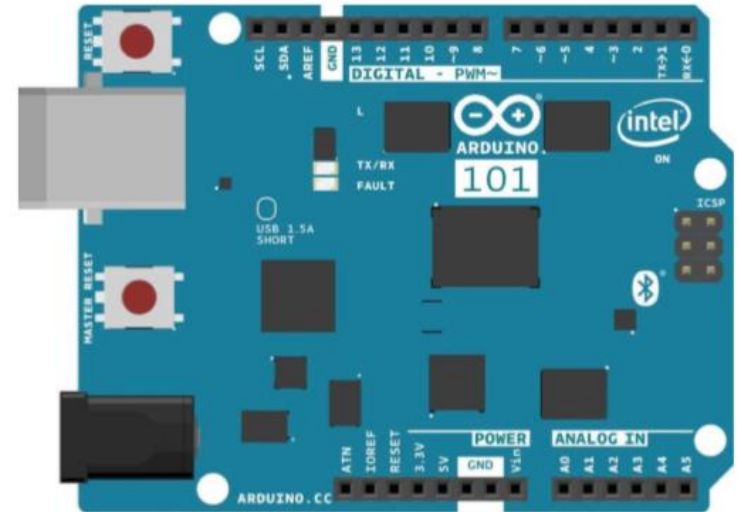
```
Blink | Arduino 1.0
File Edit Sketch Tools Help
[Icons: Checkmark, Arrow, File, Upload, Download]
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for
 *
 * This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino b
  pinMode(13, OUTPUT);
}
```

CONNECTING THE BOARD

Use a standard Arduino USB Cable. Plug one end of the USB cable into the USB port on the Arduino. Plug the other end of the USB cable into a USB port on your computer.

Once connected, the green power LED labelled 'on' should glow. Make sure you have selected Tools >> Boards >> Genuino 101 and that a COM port is selected Tools >> Port (select the port corresponding to your Genuino 101 board - it should look like "COM* (Genuino 101)")



BLINK TEST

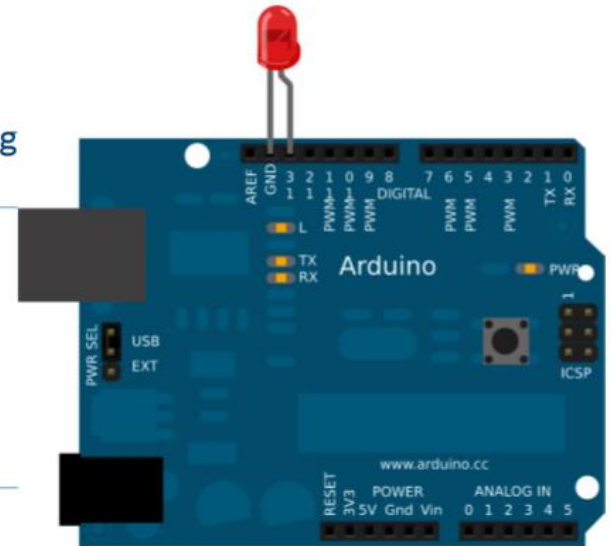
The code we write in the Arduino IDE is called a sketch.

We now need to upload a basic sketch to make sure everything is working

1. In the IDE, go to **File > Examples > 01.Basics > Blink**
2. A new sketch will open with **a pre installed code.**
3. **Click the upload button** on the tool-bar

You should notice a small LED is blinking on your board!

Now give yourself a cheer!





CONTROLLING LED



THINGS YOU'LL NEED

Here are the things you'll need for this adventure!



2x Jumper Wires



1x LED



1x Genuino Board



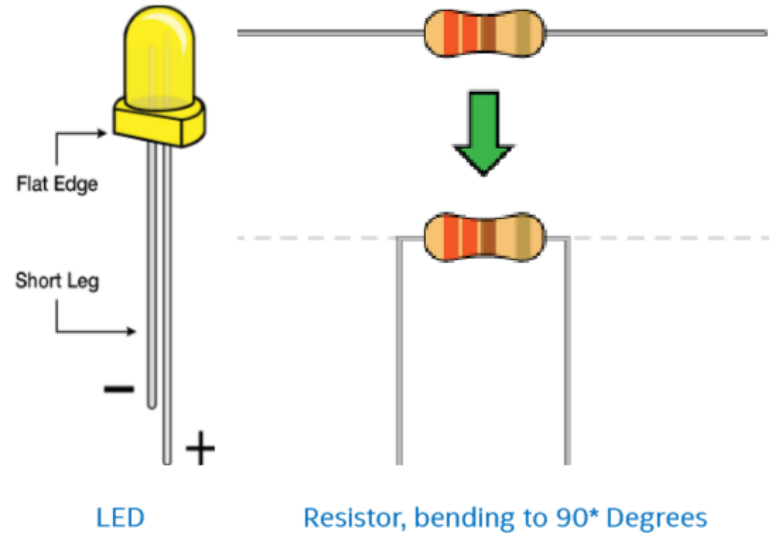
1x Breadboard



1x Resistor 330 ohm

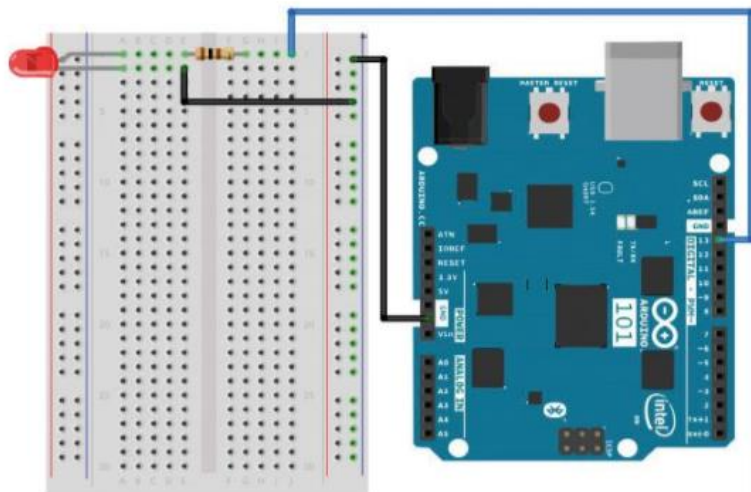
ATTACHING STRINGS

1. Connecting things is really simple. All you have to do first is to insert the components into the bread board.
2. Inserting LED will come easy, however; inserting the resistor will need a few tweaks.
3. To insert a resistor into the bread board, you'll need to bend its legs, 90* degrees from its current position.





The diagram shows how everything is connected.



The table below might come handy if the diagram is hard to grasp.

Component	Genuino	Breadboard	Breadboard
LED*		c2 (+)	c3 (-)
330 Ω Resistor		a3	(-)
Jumper Wire	GND	(-)	
Jumper Wire	PIN 13	e2	

* The LED is a polarized component, meaning it can be connected in the circuit in one direction only.

Well... That's all the setup you need for your first project!





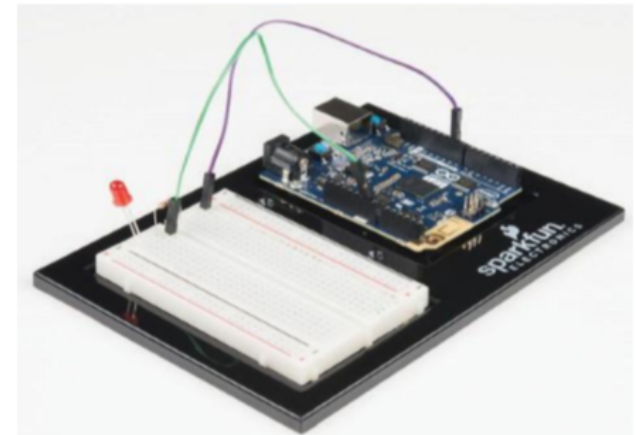
Most Arduinos already have an LED and resistor connected to pin 13, so you may not need any additional circuitry.

But if you'd like to connect a second LED to pin 13, or use a different pin, follow these steps:

- Connect the positive side of your LED (longer leg) to Genuino digital pin 13 (or another digital pin, don't forget to change the code to match).
- Connect the negative side of your LED (shorter leg) to a 330 Ohm resistor (orange-orange-brown). Connect the other side of the resistor to ground.

pin 13 ____ + LED - ____ 330 Ohm ____ GND ____

(We always use resistors between the Genuino and LEDs to keep the LEDs from burning out due to too much current.)



This is how your board should look like!



WRITING THE CODE

Open up the Arduino IDE and get ready to code this project!

To make use of Genuino's pins, we need to first tell the Genuino, whether the pin we wish to use, will be an INPUT or an OUTPUT. In order to do so, we'll use the inbuilt command.

```
pinMode()
```

Now, to define pin number 13 as output, we write the following.

```
pinMode(13, OUTPUT);
```

While using a pin as an output, we need to define it the output will be high (5v) or Low (0v). To do this, we write the following code.

```
digitalWrite(13, HIGH);
```

Now refer to the code on the next page and upload the same on your Genuino board. You should then see the LED blink on and off. If that doesn't happen, make sure you have assembled the circuit correctly.



Read the code on the next page completely. This will be helpful.



/ Project 1*

CONTROLLING LED

Turn an LED on for one second, off for one second, and repeat forever.

This code is written by Nirman Dave, This code is completely free for use. */

```
// The Genuino is a computer that runs programs called  
// "sketches". These are text files written using instructions// the computer understands. This is a sketch.  
// Sketches have code in them, and also  
// "comments" which explain what the code does.  
// This is a comment - anything on a line after "/" is ignored  
// by the computer.  
/* This is also a comment - this one can be multi-line, but it must start and end with a star and a slash. */  
// A "function" is a block of code that performs specific task.  
// All Genuino sketches MUST have two specific functions named  
// "setup()" and "loop()". The Genuino runs these functions  
// automatically when it starts.  
// The setup() function runs only when the sketch starts.
```

```
Void setup()  
{
```

code continue on next page





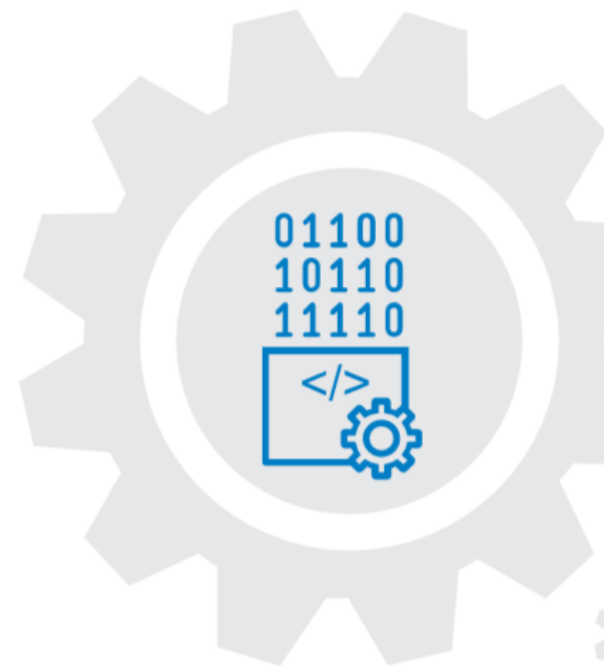
```
// The Genuino has 13 digital input/output pins.  
// You can modify it using the pinMode() function.  
// The pinMode() function takes two values, which you type in  
// the parenthesis after the function name. The first value is  
// a pin number, the second value is the word INPUT or OUTPUT.  
// Therefore, we have : pinMode(13, OUTPUT) in our code.
```

```
pinMode(13, OUTPUT);  
}
```

```
// After setup() finishes, the loop() function runs over and over again, forever
```

```
Void loop()  
{
```

```
  // Because we have an LED connected to pin 13, if we make that  
  // output HIGH, the LED will get voltage and light up. If we make  
  // that output LOW, the LED will have no voltage and turn off.  
  // digitalWrite() is the built-in function we use to make an  
  // output pin HIGH or LOW. It takes two values; a pin number,  
  // and the word HIGH or LOW:  
  digitalWrite(13, HIGH); // Turn on the LED  
  // delay() is a function that pauses for a given amount of time.  
  // It takes one value, the amount of time to wait, measured in milliseconds.
```



code continue on next page



```
Delay(1000);           // Wait for one second
digitalWrite(13,LOW);  // Turn off the LED
Delay(1000);           // Wait for one second

// And hence, the above code turns the LED on and off. Forever, and ever, and ever, and ever.
}
```



**TV, Phones and many
other devices today have
LED indicators...**



FAIRY LIGHTS



THINGS YOU'LL NEED

Here are the things you'll need for this adventure!



2x Jumper Wires



1x LED



1x Genuino Board



1x Breadboard



1x Resistor 330 ohm

ATTACHING STRINGS

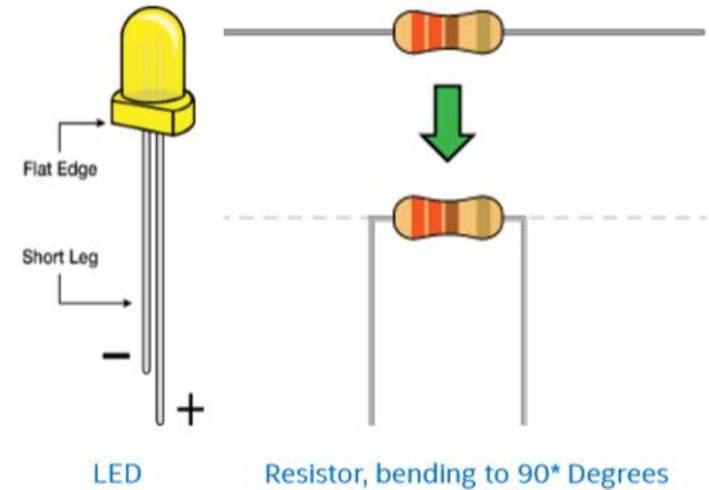
It's time to spice up things a little bit. Now that you have worked with one LED, we will try and work with 8 LEDs. When you do so, please note that **all resistors are connected to the ground**. This is a great project to prompt you to start writing your own programs.

Functions like

```
for() loops and arrays[ ]
```

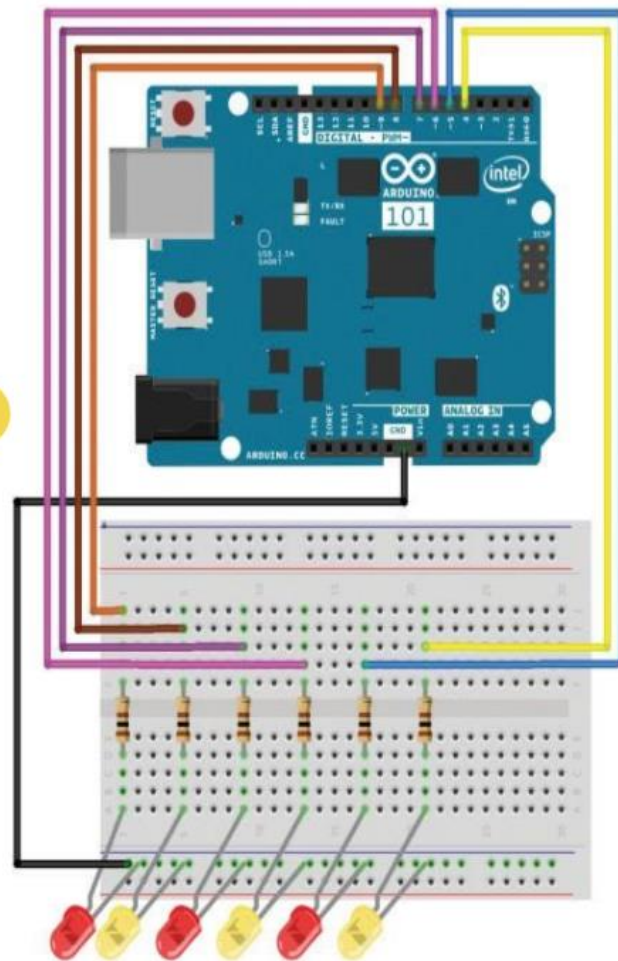
might come handy. For loops will run a code several times, while array will make managing variables easy by grouping them.

Again, you'll have to adjust the LED and the resistor to insert them into the bread board.





The diagram shows
how everything is
connected.



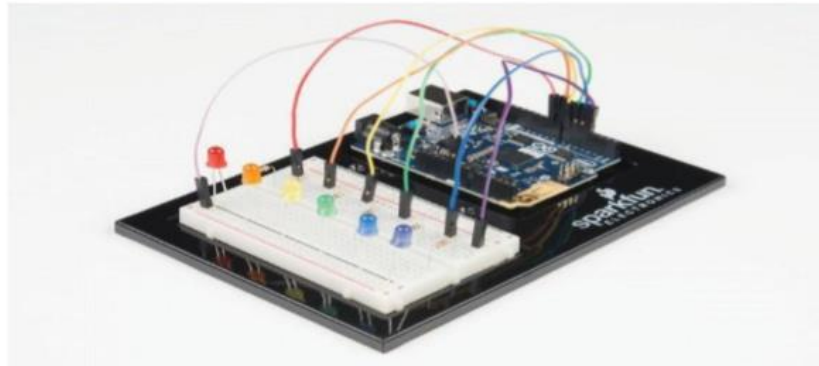


You'll need eight LEDs, and eight 330 Ohm resistors (orange-orange-brown).

For each LED, connect the negative side (shorter leg) to a 330 Ohm resistor.

Connect the other side of the resistors to GND.

Connect the positive side (longer leg) of the LEDs to Arduino digital pins 2 through 9.



This is how your board should look like!



WRITING THE CODE

Open up the Genuino Arduino IDE and get ready to code this project!

To manage a lot of elements, you need an array. This allows you to group your elements. We will group our variables using an array and will name this group as: ledPins.

```
int ledPins[] = {2,3,4,5,6,7,8,9};
```

We refer to an element by its position. Starting from 0. So, position of '2' = 0, position of '3' = 1 and so on. Here, x in ledPins[x] would refer to an element position. Lets set the pin '2' as HIGH.

```
digitalWrite(ledPins[0], HIGH);
```

To ignore monotonous activity, we can program Genuino for conducting random activity.

```
index = random(8);
```



Read the code on the next page completely. This will be helpful.



/*Project 02

MORE LED WORK!

Spice up the LEDs

This code is written by Nirman Dave, This code is completely free for use.*/

```
// Creating an array to manage our elements (pins) in a group  
// named ledPins
```

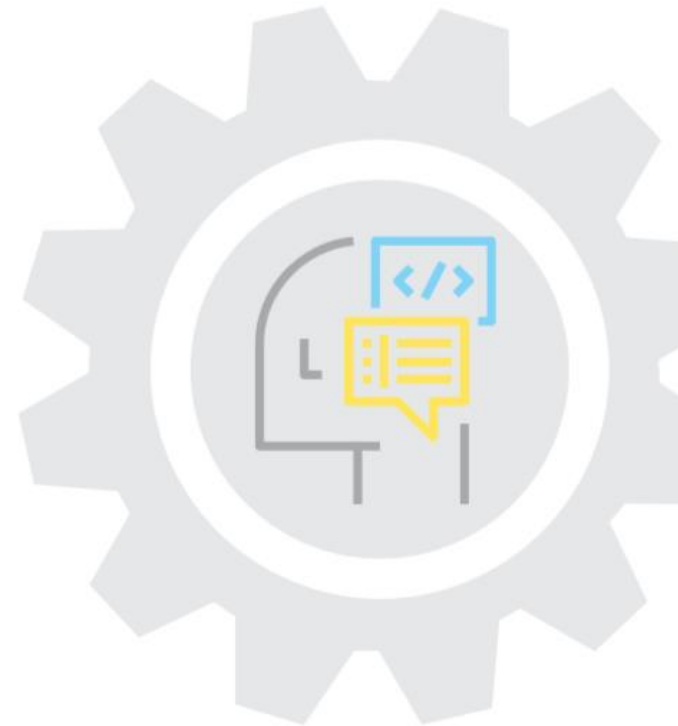
```
int ledPins[] = {2,3,4,5,6,7,8,9};
```

```
// Defining the for loop void setup()
```

```
{  
  int index;  
  // The for loop below runs a code forever, if it holds true.  
  // Here, the for loop will run until and the integer value  
  // of index will be less than or equal to 7.  
  // The ++ sign indicate the addition of a number to the  
  // integer value of index.  
  for(index = 0; index <= 7; index++)  
  {  
    pinMode(ledPins[index],OUTPUT);  
  }  
}
```

```
// Defining the oneAfterAnotherLoop for later uses. This will  
// switch the led on and off in a pattern.
```

```
void loop()  
{  
  oneAfterAnotherLoop(); // Same as oneAfterAnotherNoLoop,  
                          // but with much less typing  
}
```



code continue on next page



```
// The oneAfterAnotherLoop comes in handy as it requires lesser
// typing as compared to the for loop.

void oneAfterAnotherLoop()
{
  int index;
  int delayTime = 100;
  // Remember delay time is in microseconds for(index = 0; index <= 7; index++)
  {
    // HIGH indicates the LED on as voltage of 5v passes digitalWrite(ledPins[index], HIGH);
    delay(delayTime);
  }
  for(index = 7; index >= 0; index--)
  {
    // LOW indicates the LED off as the voltage of 0v passes digitalWrite(ledPins[index], LOW);
    delay(delayTime);
  }

  // That was the code for spicing up your 8 LEDs!
}
```



**Scrolling marquee displays
are generally made of
multiple LEDs!**



PUSH BUTTON



THINGS YOU'LL NEED

Here are the things you'll need for this adventure!



7x Jumper
Wires



1x LED



1x Genuino
Board



1x
Breadboard



1x Resistor
330 ohm



2x Resistor
10k ohm



2x
Push buttons



ATTACHING STRINGS

Until now we have only talked about projects that deal with out- puts. Here is a project that deals with inputs, believe me, it's cool!

The way a push button works with Genuino is that when the button is pushed, the voltage goes LOW. The Genuino reads this and reacts accordingly.

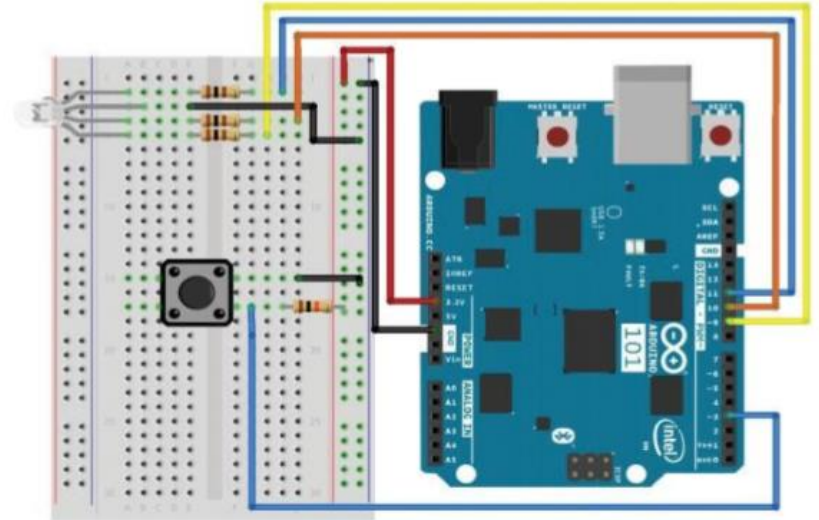
A pull up resistor will be used too, to keep the voltage HIGH when the button is not pressed.

Pushbuttons

The easiest way to hook up

the pushbutton is to connect two wires to any opposite corners. Connect any pin on pushbutton 1 to ground (GND).

Connect the opposite diagonal pin of the pushbutton to digital pin 2.





Connect any pin on pushbutton 2 to ground (GND). Connect the opposite diagonal pin of the pushbutton to digital pin 3.

Also connect 10K resistors (brown/black/red) between digital pins 2 and 3 and GND. These are called “pull-up” resistors. They ensure that the input pin will be either

5V (unpushed) or GND (pushed), and not somewhere in between.

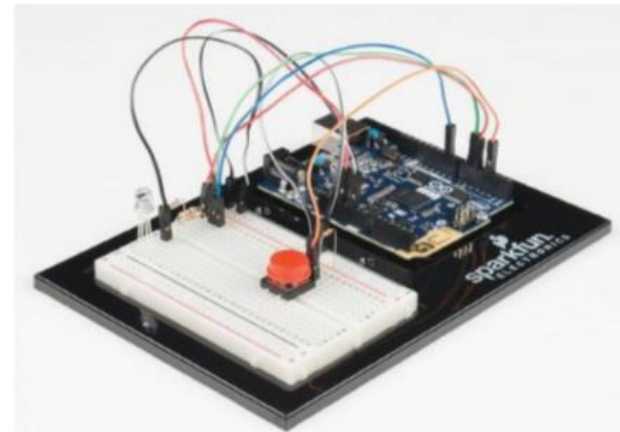
(Remember that unlike analog inputs, digital inputs are only HIGH or LOW.)

LED

Most Arduinos, including the Uno, already have an LED and resistor connected to pin 13, so you don't need any additional circuitry.

But if you'd like to connect a second LED to pin 13, Connect the positive side of your LED to Arduino digital pin 13 Connect the negative side of your LED to a 330 Ohm resistor

Connect the other side of the resistor to ground



This is how your board should look like!



WRITING THE CODE

Open up the Genuino Arduino IDE and get ready to code this project! We need to define to Genuino if the digital input we are using, is used as an input or an output. The code below, does that.

```
pinMode(button2Pin, INPUT);
```

To read the digital input, we use the following code with the digital.

```
button1State = digitalRead(button1Pin);
```

Since we have connected button to GND, it has to read LOW when being pressed. The code below, does that.

```
if (button1State == LOW)
```



Read the code on the next page completely. This will be helpful.



*/*Project 03*

Push Button!

Buttons to the rescue!

*This code is written by Nirman Dave, This code is completely free for use. */*

// First we'll set up constants for the pin numbers.

// This will make it easier to follow the code below.

const int button1Pin = 2; // pushbutton 1 pin const int button2Pin = 3; // pushbutton 2 pin const int ledPin = 13; // LED pin

void setup()

{

// Command pushbutton pins as input pinMode(button1Pin, INPUT); pinMode(button2Pin, INPUT);

// Command LED pin to be an output: pinMode(ledPin, OUTPUT);

}

void loop()

{

int button1State, button2State; // variables to hold the pushbutton states

// Since a pushbutton has only two states, we can use it as a digitalRead

// function. And we'll read the current pushbutton states into

// two variables.

button1State = digitalRead(button1Pin); button2State = digitalRead(button2Pin);

// If the button is being pressed, it will be

// connected to GND. If the button is not being pressed,

// the pullup resistor will connect it to 5 Volts.

// So the state will be LOW when it is being pressed,

// and HIGH when it is not being pressed.



code continue on next page



```
if (((button1State == LOW) || (button2State == LOW))

// if we're pushing button 1 OR button 2
  && !                // AND we're NOT
  ((button1State == LOW) && (button2State == LOW))) // pushing button 1 AND button 2
  // then...

{
  digitalWrite(ledPin, HIGH); // turn the LED on
}
else
{
  digitalWrite(ledPin, LOW); // turn the LED off

  // Well, that was the code for Push button.
}
}
```



**Well, I am not
telling you the uses
of a push button...**

A cluster of five interlocking gears is positioned in the top left corner. The gears are in various shades of blue, yellow, and orange. They are of different sizes and are partially cut off by the top and left edges of the frame.

THANK YOU