Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

INSTITUTION'S INNOVATION COUNCIL
(Ministry of Education Initiative)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE
ON

SKETCH WITH ARDUINO

STARTS ON January 2, 2022

In association with TLC

Registration : Free
Course Duration : 4 Week
Weekend Course (Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 30 Participants/Slot
Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Mr. E.Parasuramu	Prof.k.Ashok Babu	Dr.G.Suresh
9989575859		

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

Sketch with Arduino

Date: From 02.01.2022 (4 Week Course, Only on Saturday)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4
1	18D41A0447	Chilkur Teja Vardhan Reddy	Teja Vardhan	Teja Vardhan	(A)	Teja Vardhan
2	18D45A0403	Etikyala Arunkumar				
3	18D41A0490	Kamisetty Vinay				
4	17D41A0402	Jatavath Subhash Naik	Subhash	Subhash	Subhash	Subhash
5	18D41A04J7	Saluvala Bharath Kumar	Bharath kumar	Bharath kumar	Bharath kumar	Bharath kumar
6	19D41A0427	Bheemreddy Sanjana	B. Sanjana	B. Sanjana	B. Sanjana	B. Sanjana
7	19D41A04C9	Namani Sindhu	N. Sindhu	N. Sindhu	N. Sindhu	N. Sindhu
8	18D41A0429	Sowmya Boddu		(A)		
9	17D41A0496	Kalavakuri Mahesh Babu	K. Mahesh babu	K. Mahesh Babu	K. Mahesh babu	K. Mahesh babu
10	17D41A04A4	Kathi Kartheek Reddy	Kartheek	Kartheek	Kartheek	Kartheek
11	17D41A04L8	T.Sanjana				
12	17D41A04E9	Muppidi Pramod	Pramod	Pramod	Pramod	Pramod
13	18D45A0432	Jangili Raghavendra				
14	17D41A04H9	R Sai Abhinav Goud	Sai Abhinav Goud	Sai Abhinav Goud	Sai Abhinav Goud	Sai Abhinav Goud
15	17D41A04H7	Quazi Mohammad Abdul Raheem Siddique	AbdR	AbdR	AbdR	AbdR
16	17D41A04N2	Vemuri Moses Abhishek	VABS	VABS	VABS	(A)
17	19D41A0412	Anemoni Shirisha Mudhiraj	A. Shirisha	A. Shirisha	A. Shirisha	A. Shirisha

18	17D41A04L7	T Kavya	T. Kavya	T. Kavya	T. Kavya	T. Kavya
19	17D41A04J5	Rangaraju Rahul	Rangaraju	Rangaraju.	Rangaraju	Rangaraju.
20	19D41A0441	Dachepally Saisri	Sai sri	Sai sri	Sai sri	(A)
21	19D41A04A0	Lunavath Rahul Naik				
22	19D41A0435	B.Sainath	Sainath	Sainath.	Sainath	Sainath.
23	17D41A04G9	Pasunuri Shiva	shiva	shiva	shiva	shiva.
24	19D41A0436	Buduru Gayatri	Gayatri	Gayatri	Gayathri	Gayatri.
25	19D41A0417	Ashwala Madhuri	madhuri			
26	19D41A04A3	Mahesh Bhukya	Mahesh	Mahesh	Mahesh	Mahesh.
27	18D41A0451	Rudhinika	Rudhinika	Rudhinika	Rudhinika	Rudhinika,
28	17D41A04G1	Nimisha Reddy	Nimisha Reddy	Nimisha Reddy	Nimisha Reddy	Nimish Reddy,
29	19D41A0424	Beemagani Dedeepya	Dedeepya	Dedeepya	Dedeepya	Dedeepya
30	17D41A04C5	L.Pruthvi Kiran	Prathvi Kiran	Prathvi Kiran	Prathvi Kiran	Prathvi Kiran
31	17D41A04B2	Kontham Lasya Reddy	Lasya Reddy	Lasya Reddy	Lasya Reddy	Lasya Reddy,
			(31)	(30)	(30)	(29)

Coordinator **Convener** **HOD/ECE**

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

INSTITUTION'S INNOVATION COUNCIL
(Ministry of Education Initiative)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE
ON

SKETCH WITH ARDUINO

STARTS ON January 2, 2022

In association with TLC

Registration : Free
Course Duration : 4 Week
Weekend Course (Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 30 Participants/Slot
Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Mr. E.Parasuramu	Prof.k.Ashok Babu	Dr.G.Suresh
9989575859		

The easiest way for beginners to get started with Arduino is by creating circuits using a solderless breadboard. These simple projects will teach you the basics of Arduino Uno, electronics and programming. In this tutorial, you will be creating circuits using the following electronic components:

- LED
- RGB LED
- Temp Sensor
- Pushbutton
- Potentiometer
- Photoresistor
- Servo
- Motor
- Buzzer
- LCD screen

This tutorial is going to allow you to jump right in and start building circuits. If you need some background on the Arduino Uno board or the tools that are needed, please check out post – Arduino Uno For Beginners.

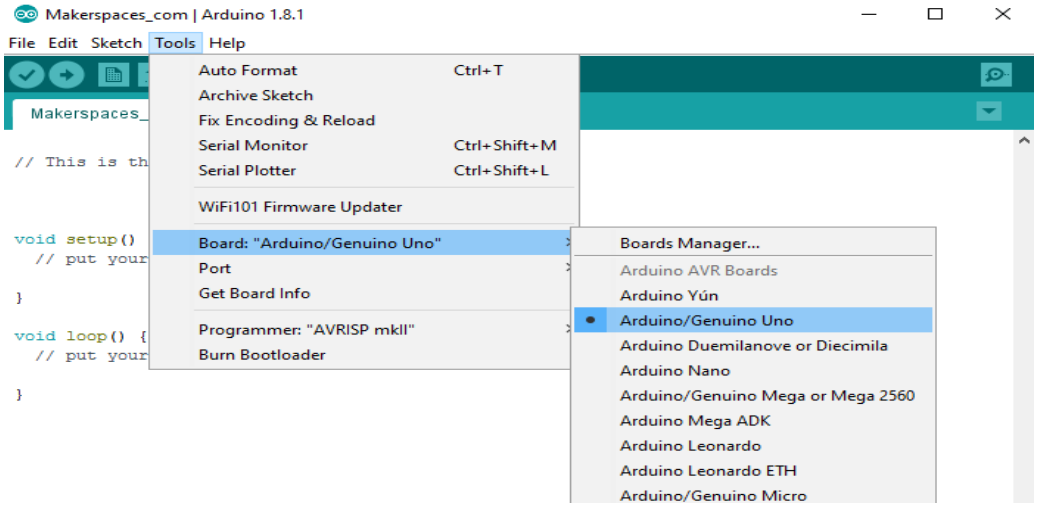
Getting Started

Before you can start working with Arduino, you need to make sure you have the IDE software installed on your computer. This program allows you to write, view and upload the code to your Arduino Uno board. You can download the IDE for free on Arduino's website.

Once the IDE is installed, you will need to connect your Arduino to your computer. To do this, plug one end of the USB cable to the Arduino Uno and then the other end of the USB to your computer's USB port.

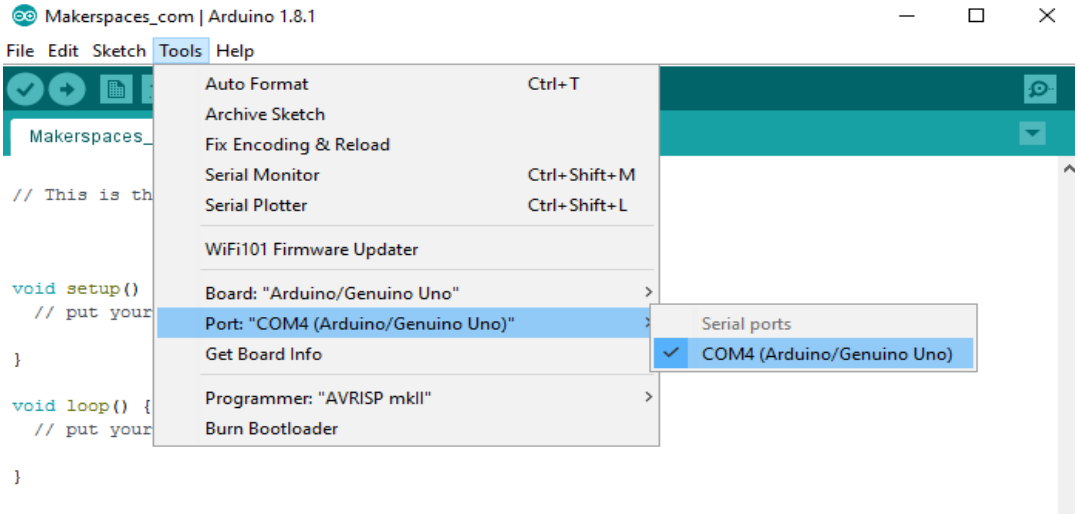
Select The Board

Once the board is plugged in, you will need to open the IDE and click on **Tools** > **Board** > **Arduino Uno** to select the board.



Select Serial Port

Next, you have to tell the Arduino which port you are using on your computer. To select the port, go to **Tools > Port** and then select the port that says **Arduino.**



Project Code

To complete the projects in this tutorial, you will need to download the project code which are known as sketches. A sketch is simply a set of instructions that tells the board what functions it needs to perform. For some of these projects, we are using open-source code that was released by the good people at Sparkfun and Arduino. Use the link below to download the zip folder containing the code.

Download Project Code – (ZIP File)

Once the file has been downloaded, you will need to unzip/extract the folder in order to use it.

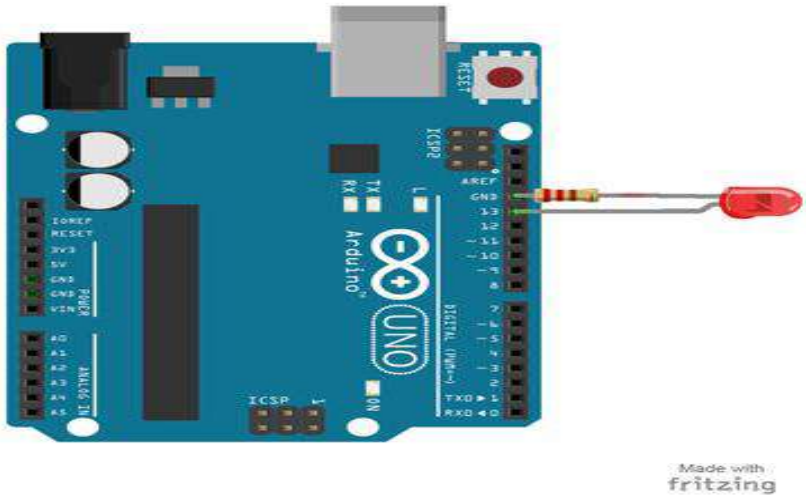
#1 – Test Arduino

The first project is one of the most basic and simple circuits you can create with Arduino. This project will test your Arduino by blinking an LED that is connected directly to the board.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) LED 5mm
- (1) 220 Ω Resistor

Project Diagram



Project Steps

1. Twist a 220 Ω resistor to the long leg (+) of the LED.
2. Push the short leg of the LED into the ground (GND) pin on the board.
3. Push the resistor leg that's connected to the LED into the #13 pin.

Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_01_TestArduino**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

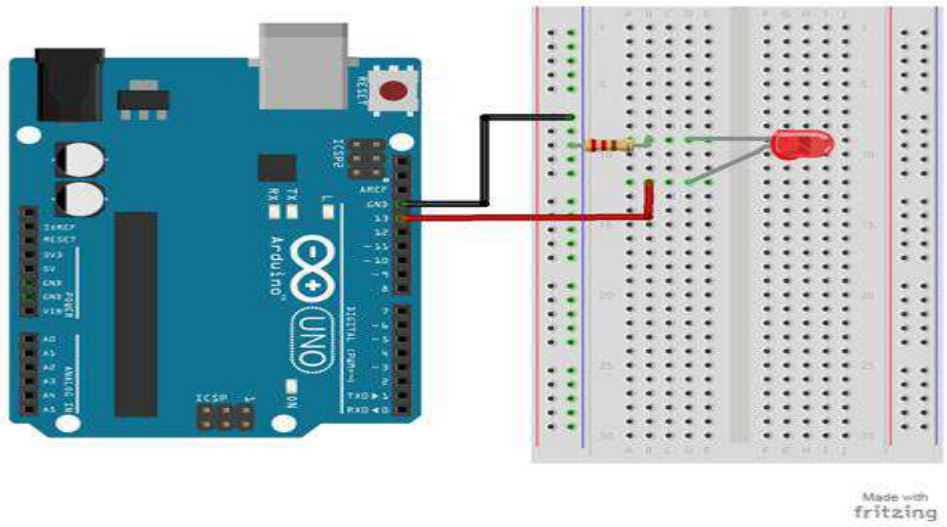
#2 – Blink an LED

This project is identical to project #1 except that we will be building it on a breadboard. Once complete, the LED should turn on for a second and then off for a second in a loop.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (2) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_02_Blink**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

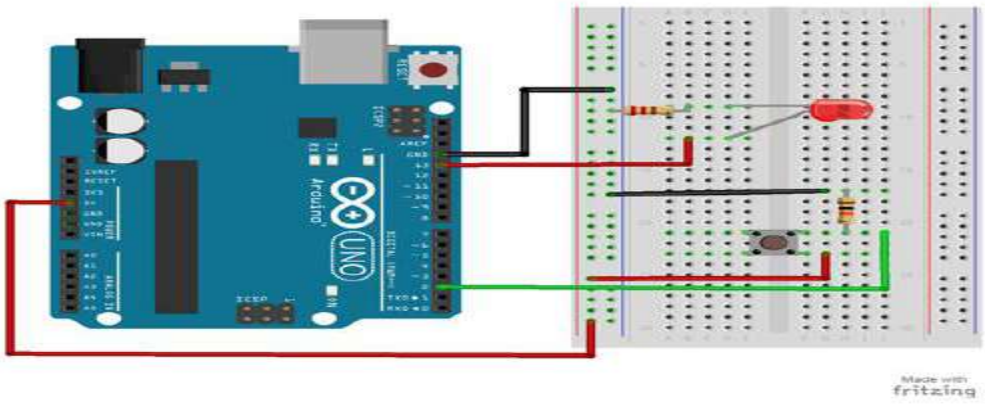
#3 – Push Button

Using a push button switch, you will be able to turn on and off an LED.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (1) 10K Ω Resistor
- (1) Push Button Switch
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_03_Pushbutton**

3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

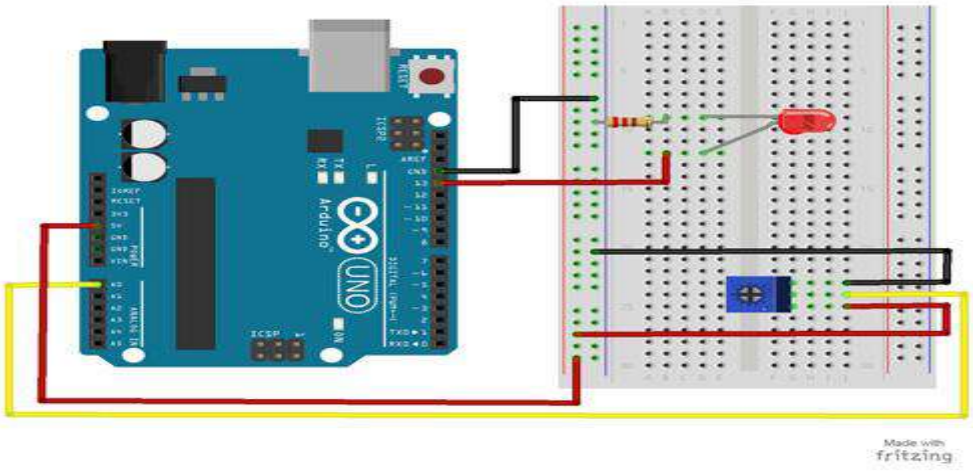
#4 – Potentiometer

Using a potentiometer, you will be able to control the resistance of an LED. Turning the knob will increase and decrease the frequency the LED blinks.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (1) Potentiometer (10k Trimpot)
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_04_Potentiometer**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

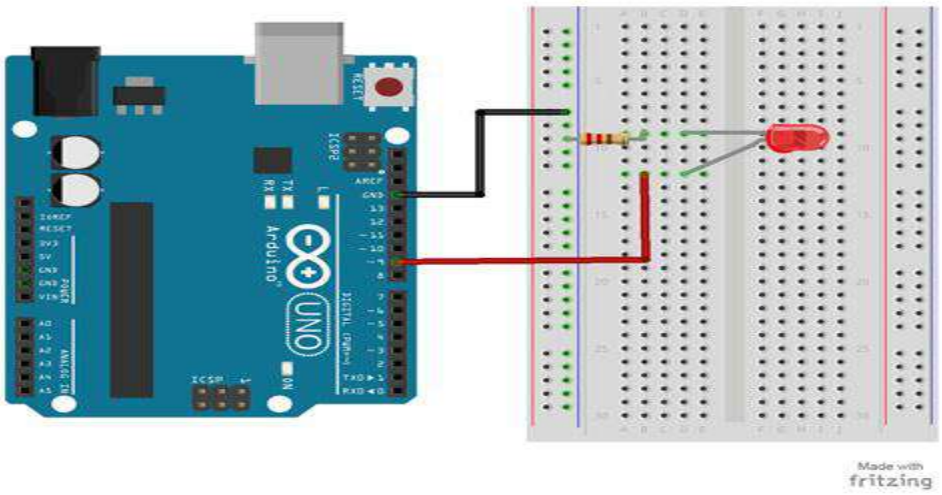
#5 – Fade an LED

By using a PWM pin on the Arduino, you will be able to increase and decrease the intensity of brightness of an LED.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (2) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_05_Fade**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

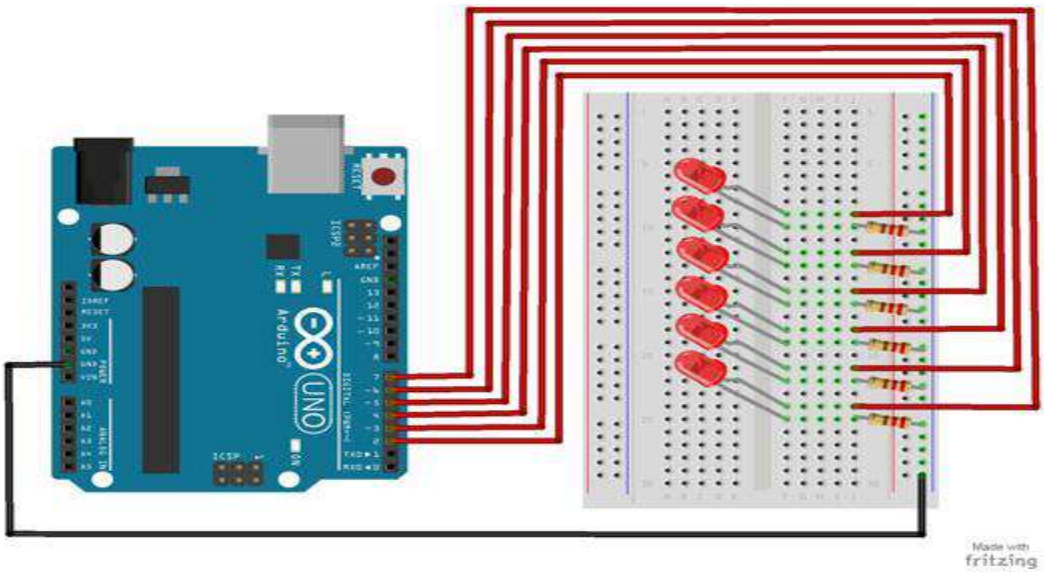
#6 – Scrolling LED

This project will blink 6 LEDs, one at a time, in a back and forth formation. This type of circuit was made famous by the show Knight Rider which featured a car with looping LEDs.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (6) LED 5mm
- (6) 220 Ω Resistor
- (7) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_06_Scrolling**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

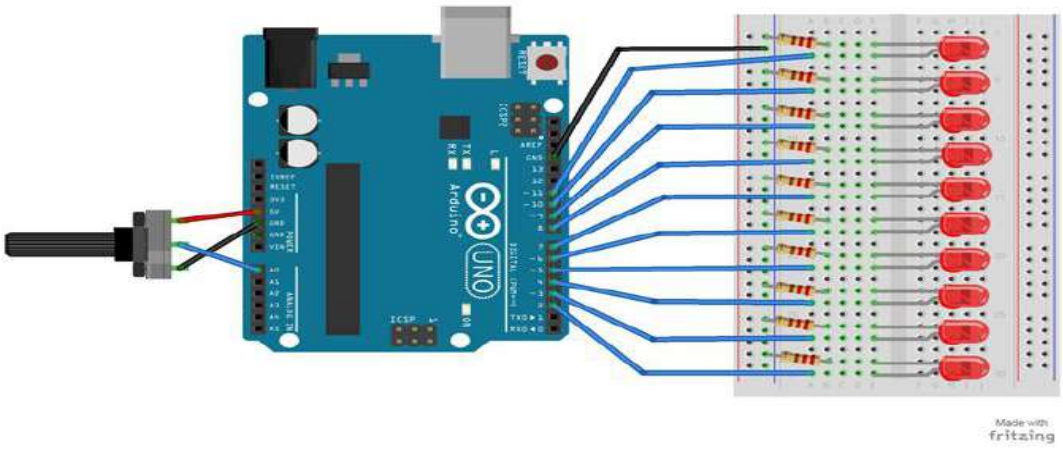
#7 – Bar Graph

Using a potentiometer, you can control a series of LEDs in a row. Turning the potentiometer knob will turn on or off more of the LEDs.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Potentiometer – Rotary
- (10) LED 5mm
- (10) 220 Ω Resistor
- (11) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_07_BarGraph**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

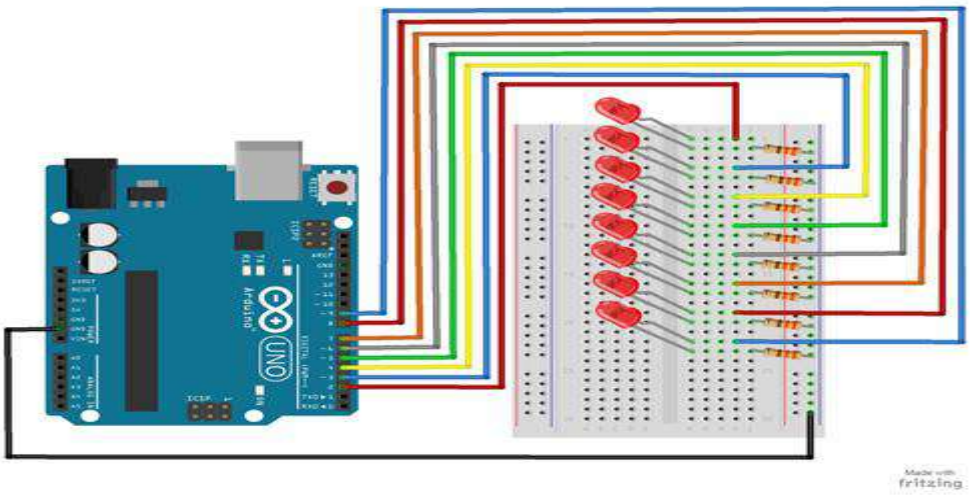
#8 – Multiple LEDs

This project will use 8 pins on the Arduino board to blink 8 LEDs at the same time.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (8) LED 5mm
- (8) 330 Ω Resistor
- (9) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_08_MultipleLEDs**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

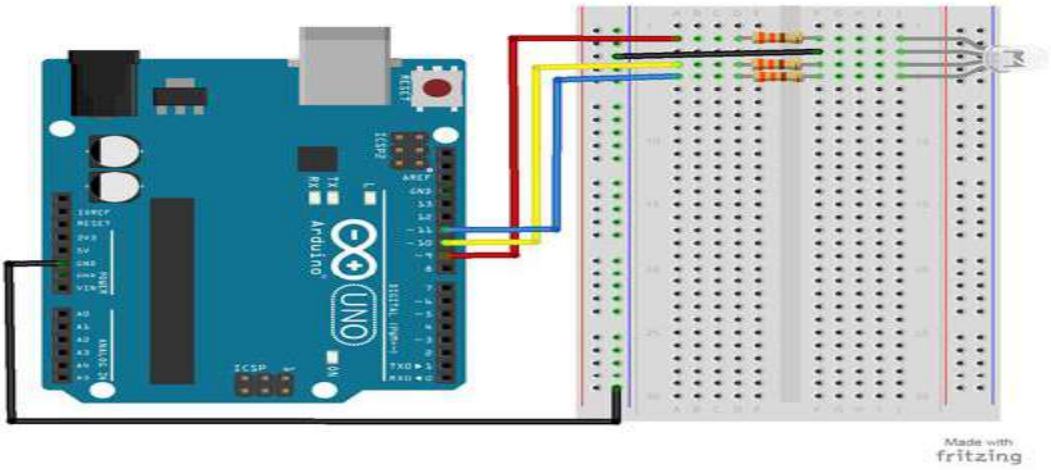
#9 – RGB LED

This project will be using an RGB LED to scroll through a variety of colors. RGB stands for Red, Green and Blue and this LED has the ability to create nearly unlimited color combinations.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) RGB LED
- (3) 330 Ω Resistor
- (5) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_09_RGBLED**

3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

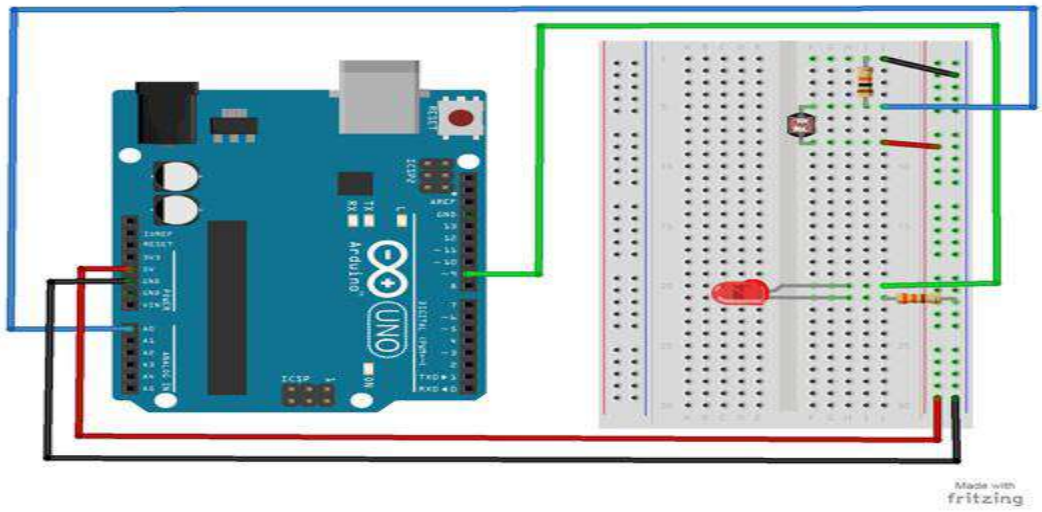
#10 – Photoresistor

A photoresistor changes the resistance a circuit gets based on the amount of light that hits the sensor. In this project, the brightness of the LED will increase and decrease based on the amount of light present.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 330 Ω Resistor
- (1) 10K Ω Resistor
- (1) Photoresistor
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_10_Photoresistor**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

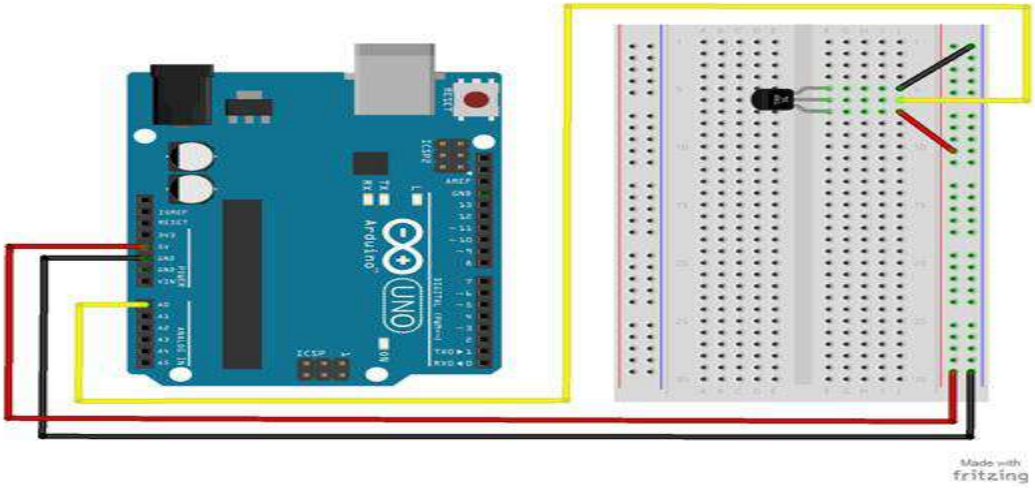
#11 – Temp. Sensor

A temperature sensor measures ambient temperatures of the world around it. In this project, we will be displaying the temperature in the serial monitor of the Arduino IDE.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Temperature Sensor – TMP36
- (5) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_11_TempSensor**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

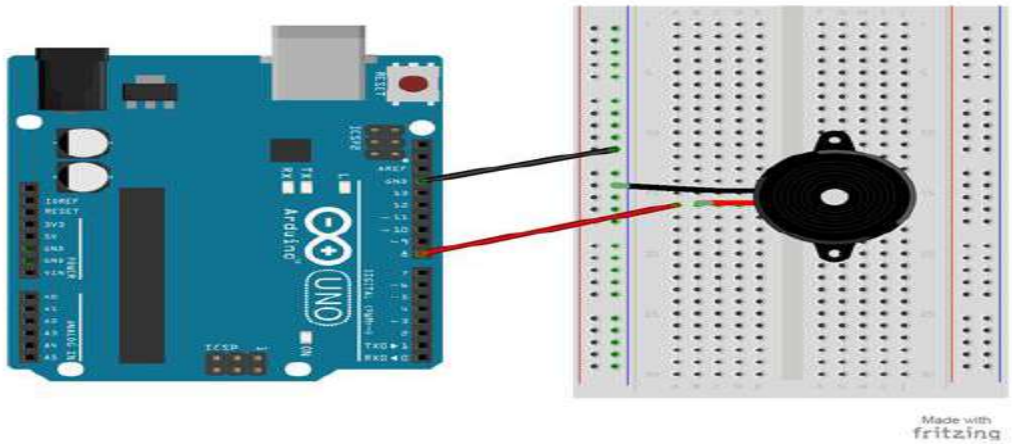
#12 – Tone Melody

The project will use a piezo buzzer/speaker to play a little melody.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Piezo Buzzer/Speaker
- (2) Jumper Wires

Project Diagram



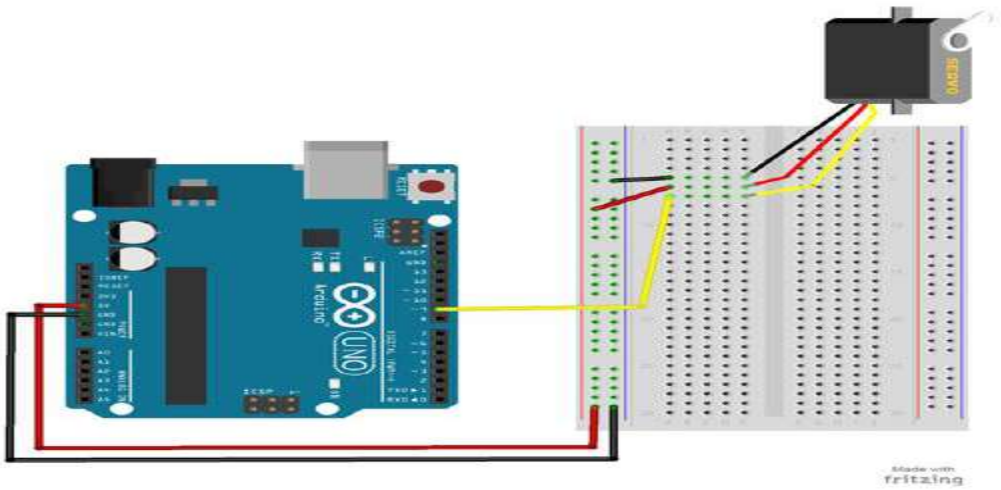
Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_12_ToneMelody**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

#13 – Servo
In this project, you will be able to sweep a servo back and forth through its full range of motion.

Parts Needed
- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Servo
- (6) Jumper Wires

Project Diagram

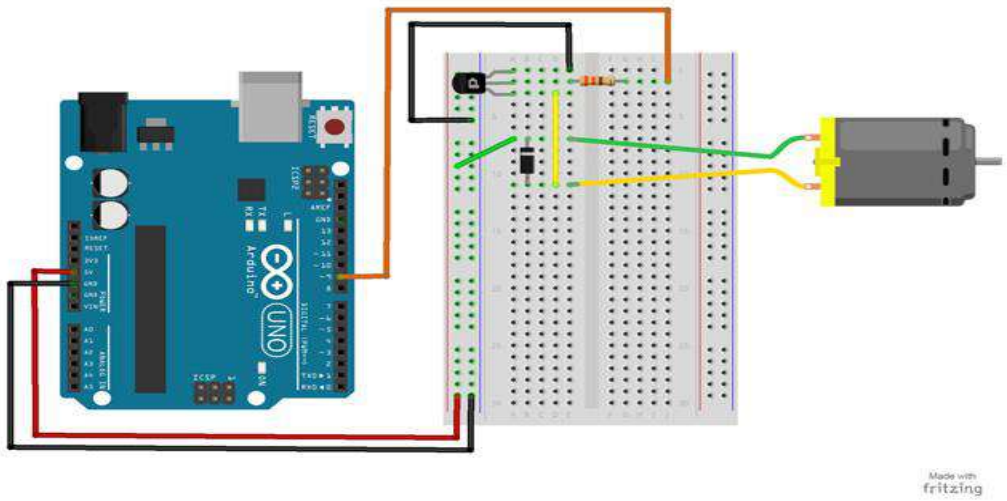


Project Code
1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_13_Servo**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

#14 – Motor
Using a switching transistor, we will be able to control a DC motor. If everything is connected correctly, you should see the motor spinning.

Parts Needed
- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) DC Motor
- (1) 330 Ω Resistor
- (1) Diode 1N4148
- (1) NPN Transistor
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_14_Motor**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

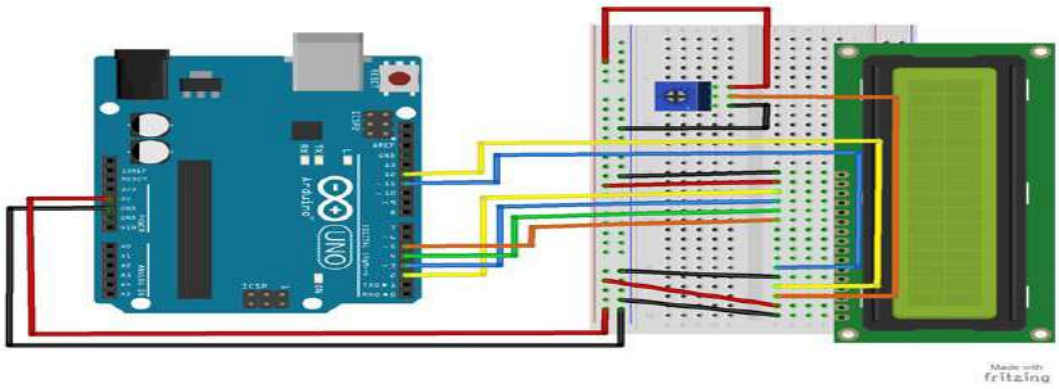
#15 – LCD Screen

An LCD is a liquid crystal display that is able to display text on its screen. In this project, you should see the words "hello,world!" displayed on the screen. The potentiometer is used to adjust the contrast of the display.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LCD Screen
- (1) Potentiometer
- (16) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_15_LCD**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

Troubleshooting

- Make sure your board and serial port is selected in the IDE. To do this, plug your board in and go to **Tools > Board >Arduino** to select your board. Next, go to **Tools > Port >Com (Arduino)** to select your serial port.
- The long leg of the LED is the (+) positive and the short leg is the (-) negative. Make sure the correct leg of the LED is in the proper pin of the Arduino or breadboard as directed.
- It can be easy to put a component or jumper into the wrong pin on the Arduino or the breadboard. Double check the correct pin is being used.

Experiment 1: Turn an LED

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

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);   // Turn on the LED
  delay(1000);              // Wait for one second
  digitalWrite(13, LOW);    // Turn off the LED
  delay(1000);              // Wait for one second
}
/*
```

Experiment 2: Turns on and off LED

Turns on and off a light emitting diode(LED) connected to digital
pin 13, when pressing a pushbutton attached to pin 2.

The circuit:
* LED attached from pin 13 to ground
* pushbutton attached to pin 2 from +5V
* 10K resistor attached to pin 2 from ground

* Note: on most Arduinos there is already an LED on the board
attached to pin 13.

```
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
```

```
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Experiment 3: Display RGB LED

```
const int RED_PIN = 9;
const int GREEN_PIN = 10;
const int BLUE_PIN = 11;

const int DISPLAY_TIME = 1000;  // used in mainColors() to determine the
// length of time each color is displayed.

void setup()       //Configure the Arduino pins to be outputs to drive the LEDs
{
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
}

void loop()
{
  mainColors();        // Red, Green, Blue, Yellow, Cyan, Purple, White
  //  showSpectrum();    // Gradual fade from Red to Green to Blue to Red
}

/**********************************************************************
 * void mainColors()
 * This function displays the eight "main" colors that the RGB LED
 * can produce. If you'd like to use one of these colors in your
 * own sketch, you can copy and paste that section into your code.
/**********************************************************************/
void mainColors()
{
  // all LEDs off
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
```

```
 digitalWrite(BLUE_PIN, LOW);
 delay(DISPLAY_TIME);

 // Red
 digitalWrite(RED_PIN, HIGH);
 digitalWrite(GREEN_PIN, LOW);
 digitalWrite(BLUE_PIN, LOW);
 delay(DISPLAY_TIME);

 // Green
 digitalWrite(RED_PIN, LOW);
 digitalWrite(GREEN_PIN, HIGH);
 digitalWrite(BLUE_PIN, LOW);
 delay(DISPLAY_TIME);

 // Blue
 digitalWrite(RED_PIN, LOW);
 digitalWrite(GREEN_PIN, LOW);
 digitalWrite(BLUE_PIN, HIGH);
 delay(DISPLAY_TIME);

 // Yellow (Red and Green)
 digitalWrite(RED_PIN, HIGH);
 digitalWrite(GREEN_PIN, HIGH);
 digitalWrite(BLUE_PIN, LOW);
 delay(DISPLAY_TIME);

 // Cyan (Green and Blue)
 digitalWrite(RED_PIN, LOW);
 digitalWrite(GREEN_PIN, HIGH);
 digitalWrite(BLUE_PIN, HIGH);
 delay(DISPLAY_TIME);

 // Purple (Red and Blue)
 digitalWrite(RED_PIN, HIGH);
 digitalWrite(GREEN_PIN, LOW);
 digitalWrite(BLUE_PIN, HIGH);
 delay(DISPLAY_TIME);

 // White (turn all the LEDs on)
 digitalWrite(RED_PIN, HIGH);
 digitalWrite(GREEN_PIN, HIGH);
 digitalWrite(BLUE_PIN, HIGH);
 delay(DISPLAY_TIME);
}

/*****************************************************************
 * void showSpectrum()
 *
 * Steps through all the colors of the RGB LED, displaying a rainbow.
 * showSpectrum() calls a function RGB(int color) that translates a number
 * from 0 to 767 where 0 = all RED, 767 = all RED
 *
```

```
 * Breaking down tasks down into individual functions like this
 * makes your code easier to follow, and it allows.
 * parts of your code to be re-used.
/*****************************************************************/

void showSpectrum()
{
  for (int x = 0; x <= 767; x++)
  {
    RGB(x);      // Increment x and call RGB() to progress through colors.
    delay(10);   // Delay for 10 ms (1/100th of a second) - to help the "smoothing"
  }
}

/*****************************************************************
 * void RGB(int color)
 *
 * RGB(###) displays a single color on the RGB LED.
 * Call RGB(###) with the number of a color you want
 * to display. For example, RGB(0) displays pure RED, RGB(255)
 * displays pure green.
 *
 * This function translates a number between 0 and 767 into a
 * specific color on the RGB LED. If you have this number count
 * through the whole range (0 to 767), the LED will smoothly
 * change color through the entire spectrum.
 *
 * The "base" numbers are:
 * 0   = pure red
 * 255 = pure green
 * 511 = pure blue
 * 767 = pure red (again)
 *
 * Numbers between the above colors will create blends. For
 * example, 640 is midway between 512 (pure blue) and 767
 * (pure red). It will give you a 50/50 mix of blue and red,
 * resulting in purple.
/*****************************************************************/
void RGB(int color)
{
  int redIntensity;
  int greenIntensity;
  int blueIntensity;

  color = constrain(color, 0, 767); // constrain the input value to a range of values from 0 to 767

    // if statement breaks down the "color" into three ranges:
  if (color <= 255)       // RANGE 1 (0 - 255) - red to green
  {
    redIntensity = 255 - color;    // red goes from on to off
    greenIntensity = color;        // green goes from off to on
    blueIntensity = 0;             // blue is always off
  }
```

```
  else if (color <= 511)  // RANGE 2 (256 - 511) - green to blue
  {
    redIntensity = 0;                   // red is always off
    greenIntensity = 511 - color;       // green on to off
    blueIntensity = color - 256;        // blue off to on
  }
  else                  // RANGE 3 ( >= 512)- blue to red
  {
    redIntensity = color - 512;         // red off to on
    greenIntensity = 0;                 // green is always off
    blueIntensity = 767 - color;        // blue on to off
  }

  // "send" intensity values to the Red, Green, Blue Pins using analogWrite()
  analogWrite(RED_PIN, redIntensity);
  analogWrite(GREEN_PIN, greenIntensity);
  analogWrite(BLUE_PIN, blueIntensity);
}
```

Experiment 4: Dancing LED

```
int ledPins[] = {2,3,4,5,6,7,8,9};   // Defines an array to store the pin numbers of the 8 LEDs.
// An array is like a list variable that can store multiple numbers.
// Arrays are referenced or "indexed" with a number in the brackets [ ]. See the examples in
// the pinMode() functions below.

void setup()
{
  // setup all 8 pins as OUTPUT - notice that the list is "indexed" with a base of 0.
  pinMode(ledPins[0],OUTPUT);  // ledPins[0] = 2
  pinMode(ledPins[1],OUTPUT);  // ledPins[1] = 3
  pinMode(ledPins[2],OUTPUT);  // ledPins[2] = 4
  pinMode(ledPins[3],OUTPUT);  // ledPins[3] = 5
  pinMode(ledPins[4],OUTPUT);  // ledPins[4] = 6
  pinMode(ledPins[5],OUTPUT);  // ledPins[5] = 7
  pinMode(ledPins[6],OUTPUT);  // ledPins[6] = 8
  pinMode(ledPins[7],OUTPUT);  // ledPins[7] = 9
}

void loop()
{

  // This loop() calls functions that we've written further below.
  // We've disabled some of these by commenting them out (putting
  // "//" in front of them). To try different LED displays, remove
  // the "//" in front of the ones you'd like to run, and add "//"
  // in front of those you don't to comment out (and disable) those
  // lines.

  oneAfterAnother();  // Light up all the LEDs in turn

  //oneOnAtATime();         // Turn on one LED at a time
```

```
  //pingPong();           // Same as oneOnAtATime() but change direction once LED reaches edge

  //marquee();           // Chase lights like you see on theater signs

  //randomLED();         // Blink LEDs randomly
}



/*****************************************************************
 * oneAfterAnother()
 *
 * This function turns all the LEDs on, pauses, and then turns all
 * the LEDS off. The function takes advantage of for() loops and
 * the array to do this with minimal typing.
/*****************************************************************/
void oneAfterAnother()
{
  int index;
  int delayTime = 100; // milliseconds to pause between LEDs
                       // make this smaller for faster switching

  // Turn all the LEDs on:
  for(index = 0; index <= 7; index = ++index)  // step through index from 0 to 7
  {
    digitalWrite(ledPins[index], HIGH);
    delay(delayTime);
  }

  // Turn all the LEDs off:
  for(index = 7; index >= 0; index = --index)  // step through index from 7 to 0
  {
    digitalWrite(ledPins[index], LOW);
    delay(delayTime);
  }
}

/*****************************************************************
 * oneOnAtATime()
 *
 * This function will step through the LEDs, lighting only one at
 * a time. It turns each LED ON and then OFF before going to the
 * next LED.
/*****************************************************************/

void oneOnAtATime()
{
  int index;
  int delayTime = 100; // milliseconds to pause between LEDs
                       // make this smaller for faster switching

  for(index = 0; index <= 7; index = ++index)   // step through the LEDs, from 0 to 7
  {
```

```
   digitalWrite(ledPins[index], HIGH);  // turn LED on
   delay(delayTime);                    // pause to slow down
   digitalWrite(ledPins[index], LOW);   // turn LED off
  }
}

/**********************************************************************
 * pingPong()
 *
 * This function will step through the LEDs, lighting one at at
 * time in both directions. There is no delay between the LED off
 * and turning on the next LED. This creates a smooth pattern for
 * the LED pattern.
/**********************************************************************/
void pingPong()
{
  int index;
  int delayTime = 100; // milliseconds to pause between LEDs

  for(index = 0; index <= 7; index = ++index)   // step through the LEDs, from 0 to 7
  {
    digitalWrite(ledPins[index], HIGH);  // turn LED on
    delay(delayTime);                    // pause to slow down
    digitalWrite(ledPins[index], LOW);   // turn LED off
  }

  for(index = 7; index >= 0; index = --index)   // step through the LEDs, from 7 to 0
  {
    digitalWrite(ledPins[index], HIGH);  // turn LED on
    delay(delayTime);                    // pause to slow down
    digitalWrite(ledPins[index], LOW);   // turn LED off
  }
}

/**********************************************************************
 * marquee()
 *
 * This function will mimic "chase lights" like those around
 * theater signs.
/**********************************************************************/
void marquee()
{
  int index;
  int delayTime = 200; // milliseconds to pause between LEDs

  // Step through the first four LEDs
  // (We'll light up one in the lower 4 and one in the upper 4)

  for(index = 0; index <= 3; index++) // Step from 0 to 3
  {
    digitalWrite(ledPins[index], HIGH);    // Turn a LED on
    digitalWrite(ledPins[index+4], HIGH);  // Skip four, and turn that LED on
    delay(delayTime);                      // Pause to slow down the sequence
```

```
    digitalWrite(ledPins[index], LOW);     // Turn the LED off
    digitalWrite(ledPins[index+4], LOW);   // Skip four, and turn that LED off
  }
}

/**********************************************************************
 * randomLED()
 *
 * This function will turn on random LEDs. Can you modify it so it
 * also lights them for random times?
/**********************************************************************/
void randomLED()
{
  int index;
  int delayTime;

  index = random(8);  // pick a random number between 0 and 7
  delayTime = 100;

  digitalWrite(ledPins[index], HIGH);  // turn LED on
  delay(delayTime);                    // pause to slow down
  digitalWrite(ledPins[index], LOW);   // turn LED off
}
```

Experiment 5: Running Motor

```
const int motorPin = 9;  // Connect the base of the transistor to pin 9.
                         // Even though it's not directly connected to the motor,
                         // we'll call it the 'motorPin'

void setup()
{
  pinMode(motorPin, OUTPUT);  // set up the pin as an OUTPUT
  Serial.begin(9600);         // initialize Serial communications
}


void loop()
{ // This example basically replicates a blink, but with the motorPin instead.
  int onTime = 3000;  // milliseconds to turn the motor on
  int offTime = 3000; // milliseconds to turn the motor off

  analogWrite(motorPin, 255); // turn the motor on (full speed)
  delay(onTime);              // delay for onTime milliseconds
  analogWrite(motorPin, 0);  // turn the motor off
  delay(offTime);             // delay for offTime milliseconds

  // Uncomment the functions below by taking out the //. Look below for the
  // code examples or documentation.

  // speedUpandDown();
  // serialSpeed();
}
```

```
// This function accelerates the motor to full speed,
// then decelerates back down to a stop.
void speedUpandDown()
{
  int speed;
  int delayTime = 20; // milliseconds between each speed step

  // accelerate the motor
  for(speed = 0; speed <= 255; speed++)
  {
    analogWrite(motorPin,speed);            // set the new speed
    delay(delayTime);                  // delay between speed steps
  }
  // decelerate the motor
  for(speed = 255; speed >= 0; speed--)
  {
    analogWrite(motorPin,speed);            // set the new speed
    delay(delayTime);                  // delay between speed steps
  }
}

// Input a speed from 0-255 over the Serial port
void serialSpeed()
{
  int speed;

  Serial.println("Type a speed (0-255) into the box above,");
  Serial.println("then click [send] or press [return]");
  Serial.println();  // Print a blank line

  // In order to type out the above message only once,
  // we'll run the rest of this function in an infinite loop:

  while(true)  // "true" is always true, so this will loop forever.
  {
    // Check to see if incoming data is available:
    while (Serial.available() > 0)
    {
      speed = Serial.parseInt();  // parseInt() reads in the first integer value from the Serial Monitor.
      speed = constrain(speed, 0, 255); // constrains the speed between 0 and 255
                                        // because analogWrite() only works in this range.
      Serial.print("Setting speed to ");  // feedback and prints out the speed that you entered.
      Serial.println(speed);

      analogWrite(motorPin, speed);  // sets the speed of the motor.
    }
  }
}
```

Experiment 6: Potentiometer

```
int sensorPin = A0;    // select the input pin for the potentiometer
```

```
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

Experiment 7: Scrolling LED

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

void setup() {
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }

  // loop from the highest pin to the lowest:
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }
}
```

Experiment 8: Potentiometer

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

Experiment 9: LED with PWM

```
int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Experiment 10: To measure the temperature sensor's
// signal pin.

```
const int temperaturePin = A0;

void setup()
{

        Serial.begin(9600); //Initialize serial port & set baud rate to 9600 bits per second (bps)
}

void loop()
{

        float voltage, degreesC, degreesF; //Declare 3 floating point variables

        voltage = getVoltage(temperaturePin); //Measure the voltage at the analog pin

        degreesC = (voltage - 0.5) * 100.0; // Convert the voltage to degrees Celsius

        degreesF = degreesC * (9.0 / 5.0) + 32.0; //Convert degrees Celsius to Fahrenheit

        //Now print to the Serial monitor. Remember the baud must be 9600 on your monitor!
        // These statements will print lines of data like this:
        // "voltage: 0.73 deg C: 22.75 deg F: 72.96"

        Serial.print("voltage: ");
        Serial.print(voltage);
        Serial.print("  deg C: ");
        Serial.print(degreesC);
        Serial.print("  deg F: ");
        Serial.println(degreesF);

        delay(1000); // repeat once per second (change as you wish!)
}
float getVoltage(int pin)          //Function to read and return
                                                        //floating-point value (true voltage)
                                                        //on analog pin
{

        return (analogRead(pin) * 0.004882814);
        // This equation converts the 0 to 1023 value that analogRead()
        // returns, into a 0.0 to 5.0 value that is the true voltage
        // being read at that pin.
}

// Other things to try with this code:

//   Turn on an LED if the temperature is above or below a value.

//   Read that threshold value from a potentiometer - now you've
//   created a thermostat!
```