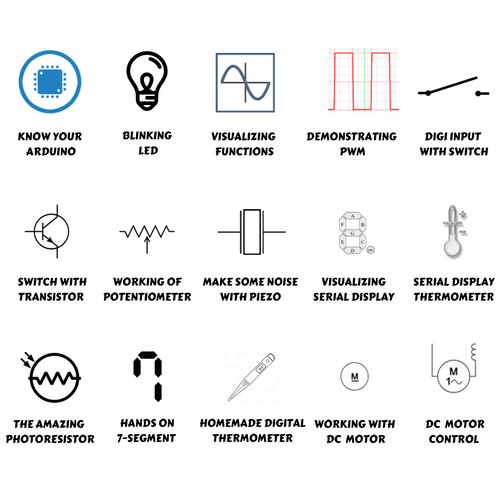
**THE FIRST STEP TO TURN YOUR**

**IMAGINATION INTO REALITY**

**PRESENTING**

**ELECDUINO-UN**

***A PERFECT GUIDE FOR LEARNING ARDUINO***

****

****

****

**WE MAKE ELECTRONICS EASY**

**WHAT ARE YOUR EXPECTATIONS?**

First of all, this is not a novel. It won’t have any dramatic content or a thought provoking notion.

It’s a **STARTER GUIDE** and what it can do very well is provide a clear vision on how you can start working on **ARDUINO** and turn whatever you **IMAGINED INTO REALITY**.

**HOW TO BEGIN?**

**THINK BIG,**

**START SMALL**



E1 Acknowledgments

E2 Setting Up Arduino Board & Software

E3 About Basic Electronics & the Components

E4

Project 1 Know Your Arduino

Project 2 Create a blinking LED Wave

Project 3 Repeating with fuctions and for loops

Project 4 Demonstrating PWM

Project 5 Digital input with switch

Project 6 Digital Switch with Transistor

Project 7 The working of Potentiometer

Project 8 Make some noise with Piezo

Project 9 Loops and Serial Display

Project 10: Feeling Hot/Cold? Create a quick read Serial display Thermometer

Project 11: The amazing photoresistor

Project 12: Hands on 7-Segment Display

Project 13: Create a digital Thermometer

Project 14: Working of DC Motor

Project 15: DC Motor speed Control with potentiometer

E5 The Infinite possibilities

* Who Should Read this Book

Are you feeling like the guy below, confused about what to do with all the components you have?



Ok! You choose “Yes”

Now, do you want to make something useful and innovative out of the Elecduino Kit and components in an amazing way that will help you make the some electronics projects you have never thought to be impossible instead of wasting your precious money and time ?

Your options are:



* If Your Answer is “YES”!!!!

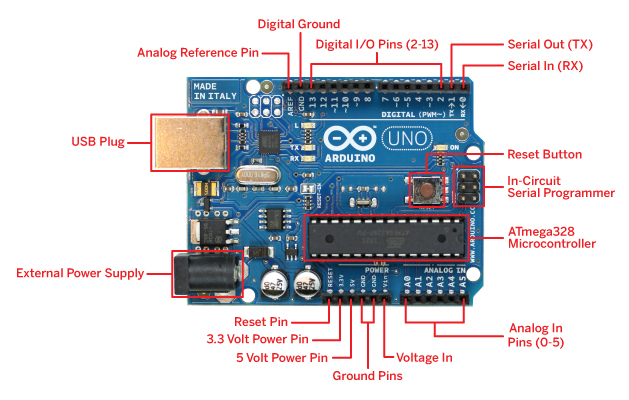


* If Your Answer is “No”, We recommend :



* **Project 1 : Know Your Arduino**

Did you ever look at some gadget or machine and wondered how it really worked? Maybe it was a TV remote control, the automatic fare collection(AFC) gates at the metro stations, an ATM, or an electronic toy? Or did you want to create your own robot or a line follower car, or perhaps you’d like to control the lights and fan of your room with your phone? Where and how to start? Arduino (in the fig. below) to the rescue! The Arduino can help you unravel the mysteries of electronics in a very easy and hands-on way. The original creation of Massimo Banzi and David Cuartielles, the Arduino offers an inexpensive and a very easy way to create interactive and innovative projects, such as remote-controlled robots, GPS tracking systems, and electronic games. The Arduino community has grown exponentially since its introduction in 2005. It’s now a worldwide industry, supported by a community of people united with the common bond of creating something new. You can find both individuals and groups, throughout the globe, all interested in building something ingenious and inventive with the Arduino.



* ***Required Software:***

You'll be able to program your Arduino with almost any computer using a software called an integrated development environment (IDE). To run this IDE, your computer should have one of the following operating systems installed:

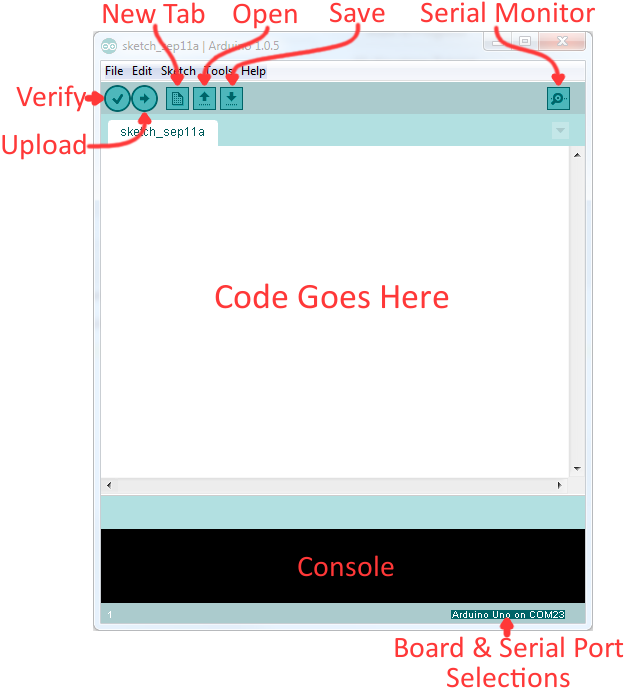
• Mac OS X or higher

• Windows XP 32- or 64-bit, or higher

• Linux 32- or 64-bit (Ubuntu or similar)

* ***Installing The IDE (For Windows 8 and above, see next section):***

***Note:*** If you face difficulty in any section of installation, please refer to the installation video in the “Videos” folder

******

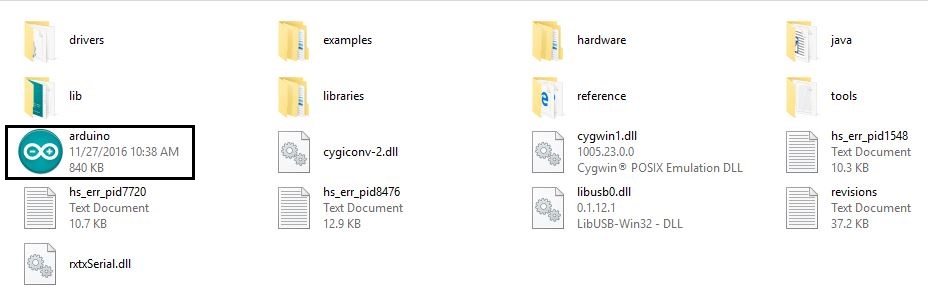
The Arduino Integrated Development Environment - Arduino Software (IDE) – consists of the following- a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs on the Arduino and communicate with it.

In this section, you’ll find the instructions for downloading the IDE, installing the required drivers and setting up the IDE for Windows.

In the software folder of the pen drive provided with this Elecduino UN kit, locate folder Arduino IDE and copy/paste it on your Computer.

Once the copying is finished, locate the folder and open it to reveal the Arduino application icon, as shown in below Figure.

You may wish to copy the icon and place a shortcut on the desktop for easier access in the future.

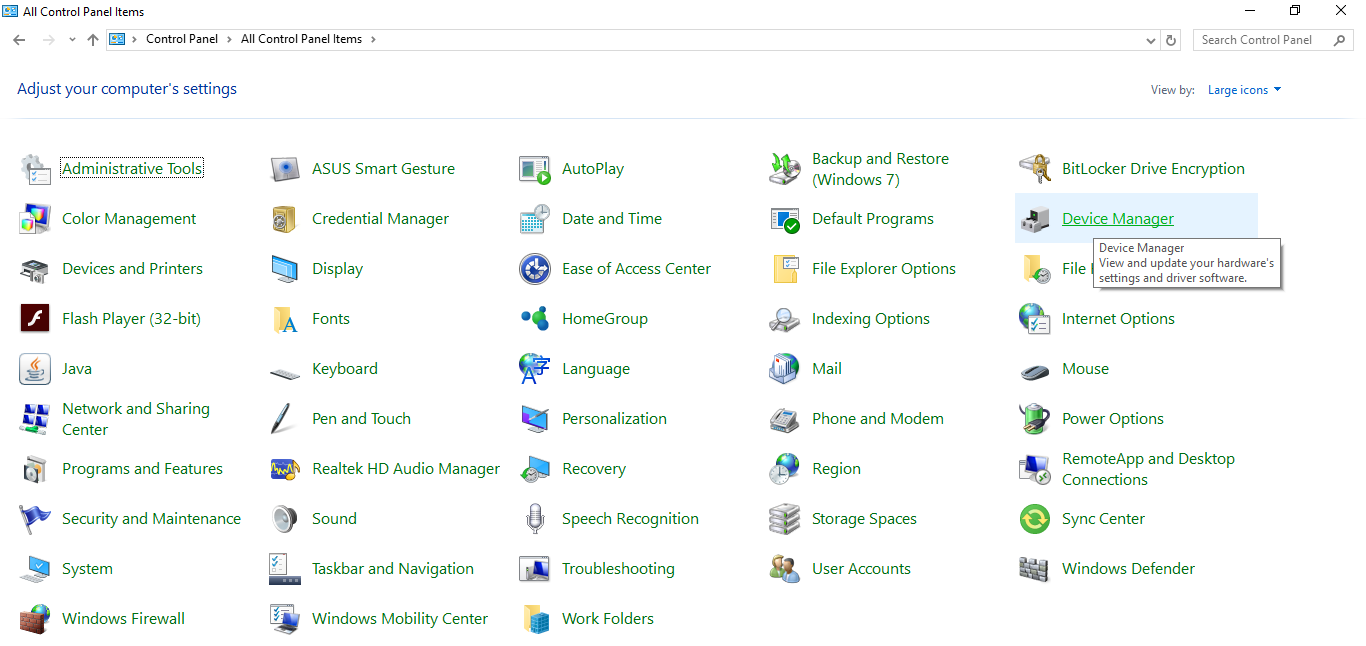


* ***Installing Drivers:***

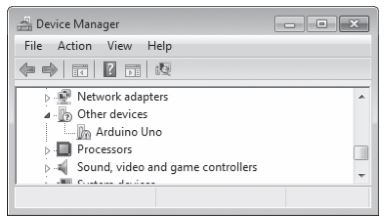
The next task is to install the drivers for your Arduino board’s USB interface.

1. Connect your Arduino to your PC with the USB cable. After some time, an error message will be displayed, which will say something like “Device driver software not successfully installed.” Close or ignore that dialog or balloon.

2. Navigate to the Windows Control Panel. Open the Device Manager and scroll down until you see the Arduino, as shown in Figure



Control panel

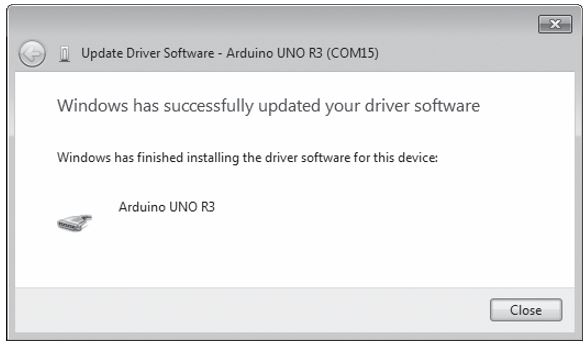


The Device Manager

3. Right-click Arduino Uno under Other Devices and select Update Driver Software. Then, select the Browse my computer for driversoft ware option that appears in the next dialog. Another Browse For Folder dialog will appear; click Browse, and navigate to the drivers folder in the newly installed Arduino software folder (shown in Figure). Click OK



4. Click Next in the dialog . Windows may show a message stating that it “cannot verify the publisher of the driver software.” Click Install this software anyway. After a short wait, Windows will tell you that the driver is successfully installed and the COM port number the Arduino is connected to, as shown in Figure.

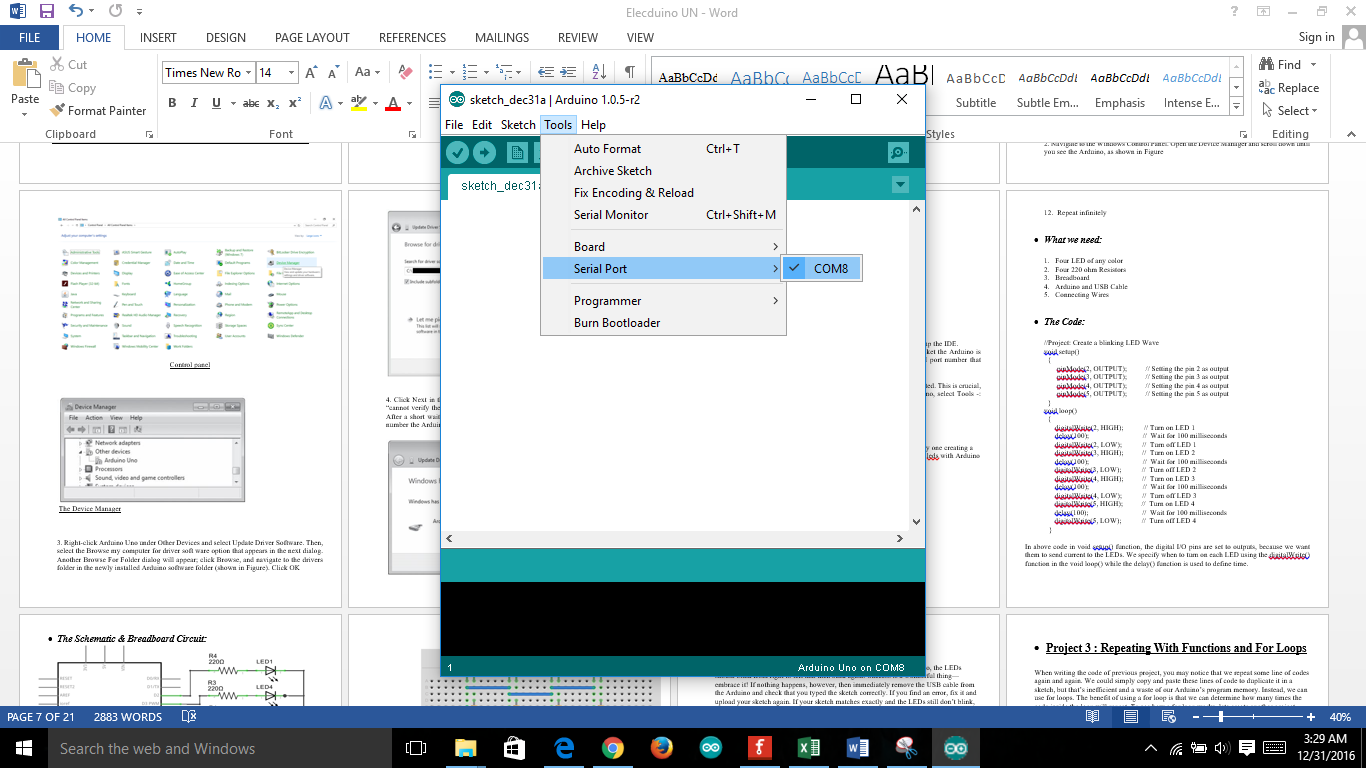
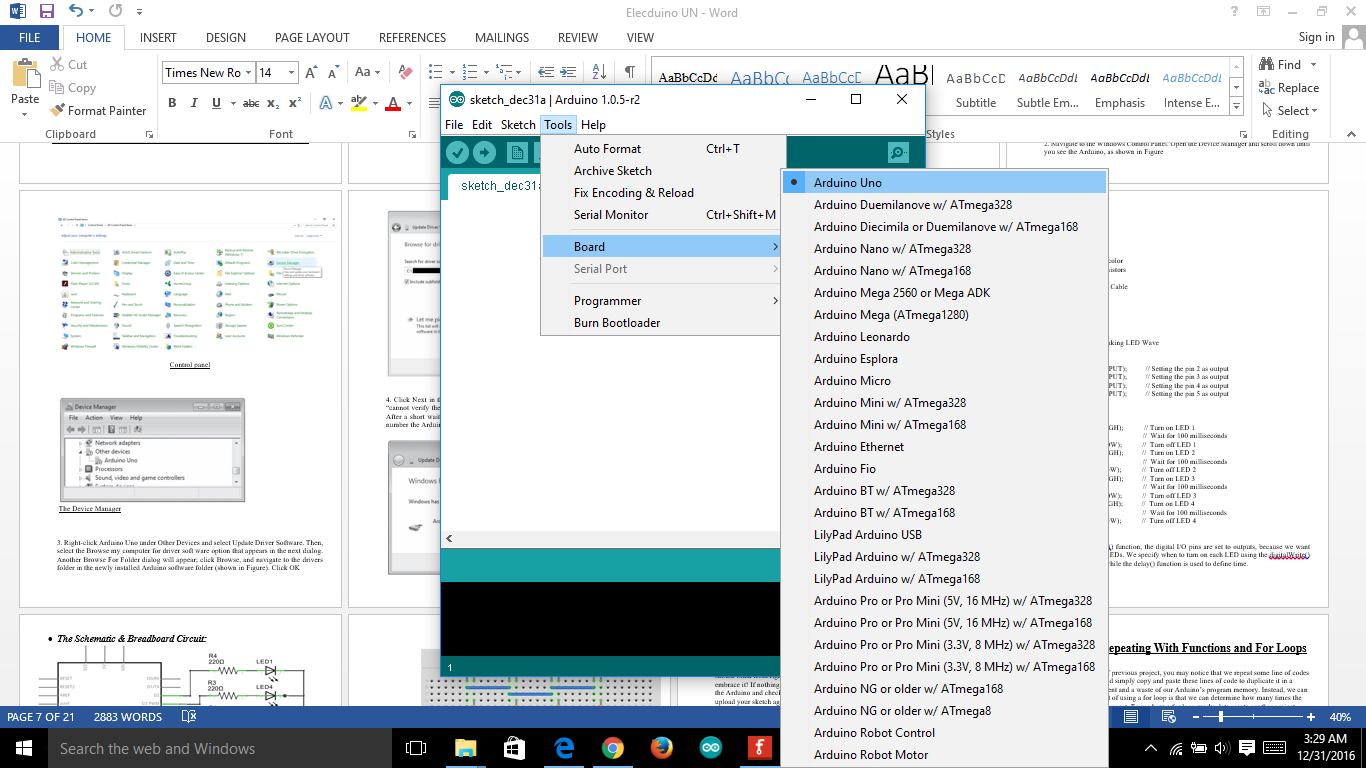


* ***Setting Up The IDE:***

Okay, we’re almost there—just two more things to do to finish setting up the IDE.

1. Open the Arduino IDE. You need to tell the IDE which type of socket or port the Arduino is connected to by selecting Tools -: Serial Port and choosing the COM port number that appeared in the Update Driver Software window.

2. The final step is to tell the IDE which Arduino board we are using. This is important, as the Arduino boards come in different variants. For example, as we are using the Uno, select Tools -: Board -: Arduino Uno.

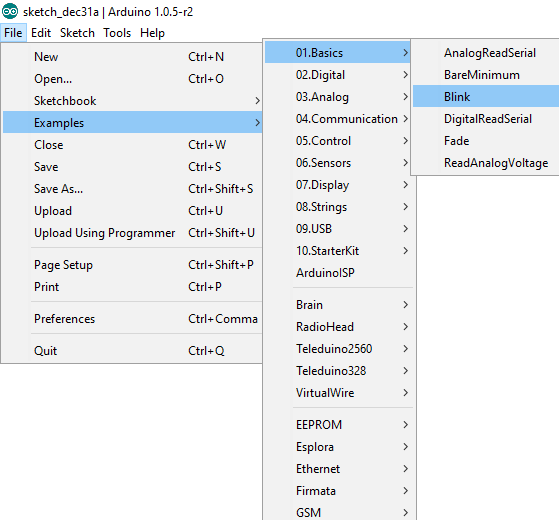
 

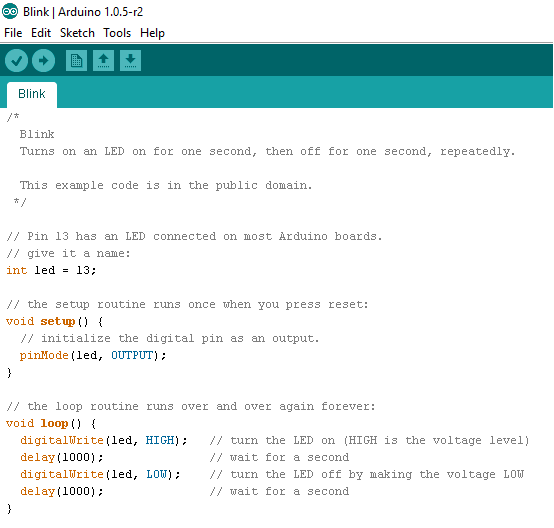


* ***Testing That Everything is Working Fine:***

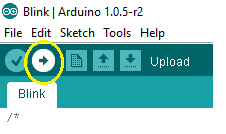
To test that everything is working fine, connect your Arduino board to your computer, Select the Board and COM port.

Go to File – Examples – Basics – Blink and open this sketch

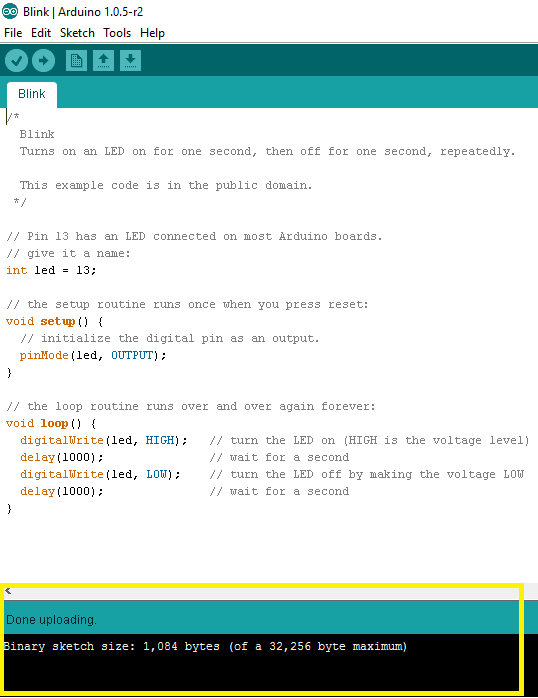




Once the sketch is opened, click the upload button at the top left.



As soon as you click the upload button, the code will start uploading and if everything went well, you will get a “Done Uploading” message on the bottom of the screen.



Now check your Arduino Board, you will see a small blinking LED next to pin number 13 which is turning on and off with 1 second delay.

* ***Explanation of Blinking LED Code:***

The Arduino board already has an LED attached to its pin 13. Writing “int led =13” means we do not have to use number “13” but can use val in its place. It is done because let’s assume you want to attach led to any other pin (pin number 12 for example), so instead of replacing number 13 everywhere we can just rewrite “int led =12”.

After that we initialize led pin as an output pin with the line

pinMode(led, OUTPUT);

In the main loop, you turn the LED on with the following line of code:

digitalWrite(led, HIGH);

This supplies 5 volts to the LED anode. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:

digitalWrite(led, LOW);

That takes the led pin back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time.

You may find that your Arduino board's 'L' LED already blinks when you connect it to a USB plug. This is because Arduino boards are generally made with the 'Blink' sketch pre-installed.



* ***Safety:***

As with any hobby or craft, it’s up to you to take care of yourself and those around you. At no point while making, testing or using your project, should you work with the mains current. Leave that to a licensed electrician who is trained for such work. Remember that contacting the mains current can kill you.



* ***Looking Ahead:***

You’re about to embark on a fun-filled and an interesting journey, and once you complete this book you’ll be creating things you may never have thought were possible. You’ll find 15 basic Arduino projects in this book, ranging from the very simple to the relatively complex. All are designed to help you learn and make something useful. So let’s go!



* **Project 2 : Create A Blinking LED Wave**

In this project, we will use FOUR LEDs and will switch them on/off one by one creating a kind of wavelike light pattern which will help us know how to interface LEDs with Arduino

* ***What we do:***

1. Turn on LED 1
2. Wait for 100 milliseconds
3. Turn off LED 1
4. Turn on LED 2
5. Wait for 100 milliseconds
6. Turn off LED 2
7. Turn on LED 3
8. Wait for 100 milliseconds
9. Turn off LED 3
10. Turn on LED 4
11. Wait for 100 milliseconds
12. Repeat infinitely

* ***What we need:***

1. Four LED of any color
2. Four 220 ohm Resistors
3. Breadboard
4. Arduino and USB Cable
5. Connecting Wires

* ***The Code:***

//Project: Create a blinking LED Wave

void setup()

{

pinMode(2, OUTPUT); // Setting the pin 2 as output

pinMode(3, OUTPUT); // Setting the pin 3 as output

pinMode(4, OUTPUT); // Setting the pin 4 as output

pinMode(5, OUTPUT); // Setting the pin 5 as output

}

void loop()

{

digitalWrite(2, HIGH); // Turn on LED 1

delay(100); // Wait for 100 milliseconds

digitalWrite(2, LOW); // Turn off LED 1

digitalWrite(3, HIGH); // Turn on LED 2

delay(100); // Wait for 100 milliseconds

digitalWrite(3, LOW); // Turn off LED 2

digitalWrite(4, HIGH); // Turn on LED 3

delay(100); // Wait for 100 milliseconds

digitalWrite(4, LOW); // Turn off LED 3

digitalWrite(5, HIGH); // Turn on LED 4

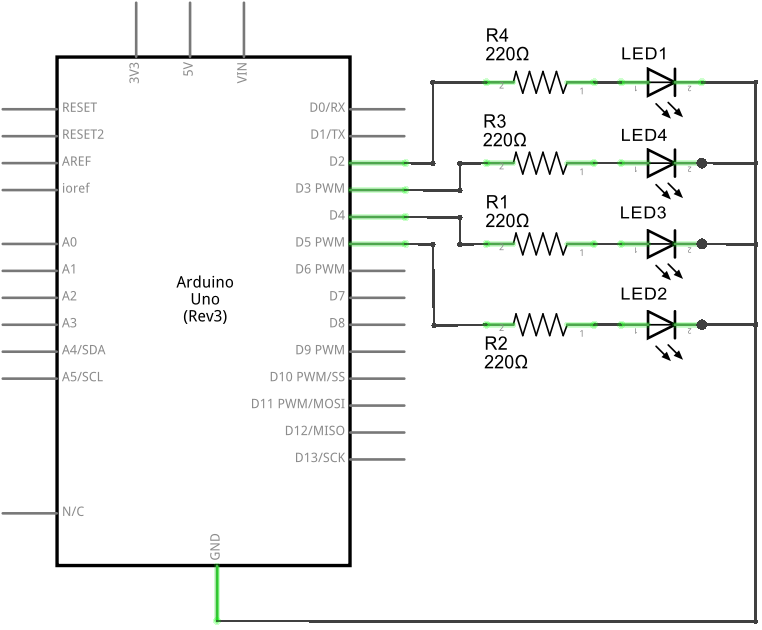
delay(100); // Wait for 100 milliseconds

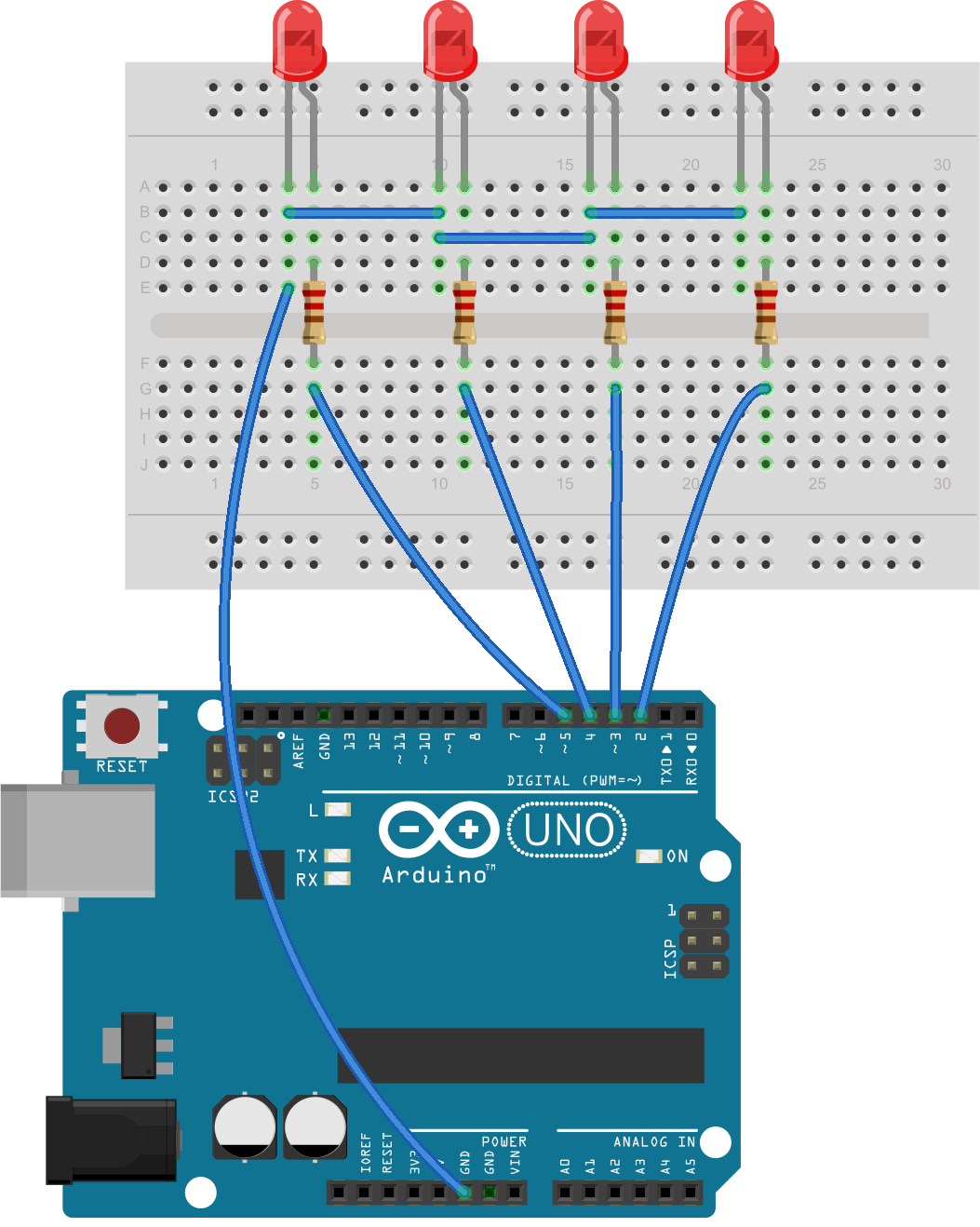
digitalWrite(5, LOW); // Turn off LED 4

}

In the code above, in void setup() function, the digital I/O pins are set to outputs, because we want them to send current to the LEDs. We specify when to turn on each LED using the digitalWrite() function in the void loop() while the delay() function is used to define time.

* ***The Schematic & Breadboard Circuit:***

******



By comparing the wiring diagram to the functions in the sketch, you can begin to get a sense of the circuit. For example, when we use digitalWrite(2, HIGH), a high voltage of 5 V flows from digital pin 2, through current-limiting resistor, through the LED via the anode and the cathode, and then finally back to the Arduino’s GND socket to complete the circuit. Then digitalWrite(2, LOW) stops the current and the LED turns off.

* ***Running the sketch:***

Now connect your Arduino using the USB cable and upload the sketch. After a second or two, the LEDs will blink from right to left and then back again. Success is a wonderful thing and should be embraced! If nothing happens, then immediately remove the USB cable from the Arduino and re-check that you typed the sketch correctly. If you find an error, correct it and upload your sketch again. If your sketch matches exactly and the LEDs still don’t blink, check the wiring on the breadboard. You now know how to make an LED blink with your Arduino, but this sketch is somewhat inefficient. For example, if you wanted to modify this sketch to make the LEDs cycle more quickly, you would need to alter each delay(500). There is a better way which we will discuss in the next project.

* ***WARNING:*** Do not connect LEDs without the use of resistors as the voltage provide by Arduino digital pin is 5V which is maximum than the optimal operating voltage of LEDs.
* ***Using Variables:***

In computer programs, we use variables to store data. For example, in the code for this project, we used the delay(500) function to keep the LEDs turned on. The problem with the code as written is that it’s not very flexible. If we want to make a change to the delay time, then we will have to change each entry manually. To address this problem, we will create a variable to represent the value for the delay() function. Enter the following line in the Project 1 sketch above the void setup() function and just after the initial comment:

int d = 250;

This assigns the number 250 to a variable, d. Next, change every 500 in the sketch to a d. Now when the sketch runs, the Arduino will use the value in d for all delay() functions. When you upload the sketch after making these changes, the LEDs will turn on and turn off at a much faster rate, as the delay value is much smaller at the 250 value. int indicates that the variable stores an integer—a whole number between −32,768 and 32,767. Simply put, any integer value that has no fraction or decimal places. Now, to alter the delay, simply change the variable declaration at the start of the sketch. For example, entering 100 for the delay would speed up things even more:

int d = 100;

Experiment and play with the sketch, perhaps altering the delays and the sequence of HIGH and LOW. Have some fun with it. Don’t disassemble the circuit yet, though; we’ll use the same in the next project.



* **Project 3 : Repeating With Functions and For Loops**

While writing the code of previous project, you may notice that we repeat some lines of codes again and again. We could simply copy and paste these lines of code to duplicate it in a sketch, but that’s can be highly inefficient and a waste of our Arduino’s program memory. Instead, we can use ‘for’ loops. The benefit of using a ‘for’ loop is that we can controlhow many times the code inside the loop will repeat. To see how a for loop works, lets create another project having the same function as the previous one but this time using for loops:

* ***What we do:***

1. Turn on LED 1
2. Wait for 100 milliseconds
3. Turn off LED 1
4. Turn on LED 2
5. Wait for 100 milliseconds
6. Turn off LED 2
7. Turn on LED 3
8. Wait for 100 milliseconds
9. Turn off LED 3
10. Turn on LED 4
11. Wait for 100 milliseconds
12. Repeat infinitely

* ***What we need:***

1. Four LED of any color
2. Four 220ohm Resistors
3. Breadboard
4. Arduino and USB Cable
5. Connecting Wires

* ***The Code:***

//Project: Create a blinking LED Wave

int val = 100; // Creating a integer variable d with value equal to 100

void setup()

{

pinMode(2, OUTPUT); // Setting the pin 2 as output

pinMode(3, OUTPUT); // Setting the pin 3 as output

pinMode(4, OUTPUT); // Setting the pin 4 as output

pinMode(5, OUTPUT); // Setting the pin 5 as output

}

void loop()

{

for (int i =2; i<6; i++)

{

digitalWrite(i, HIGH);

delay(val);

digitalWrite(i, LOW);

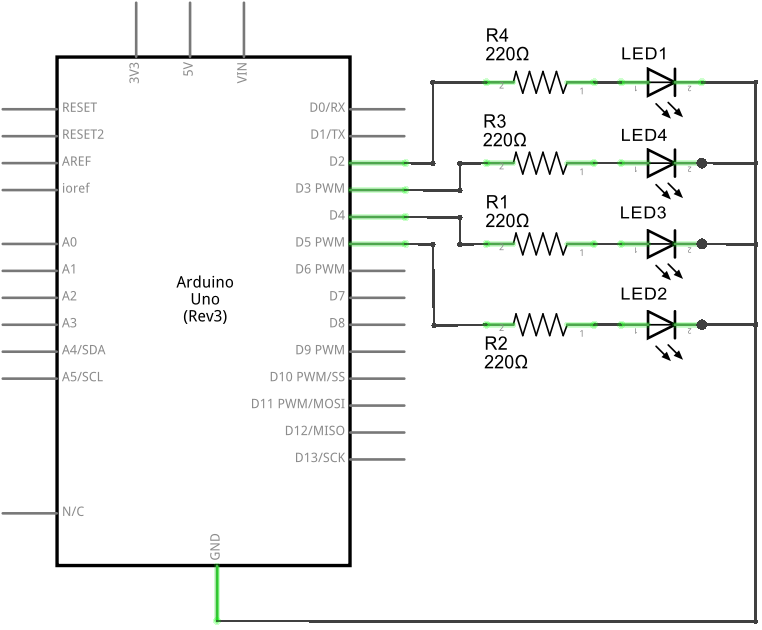
delay(val);

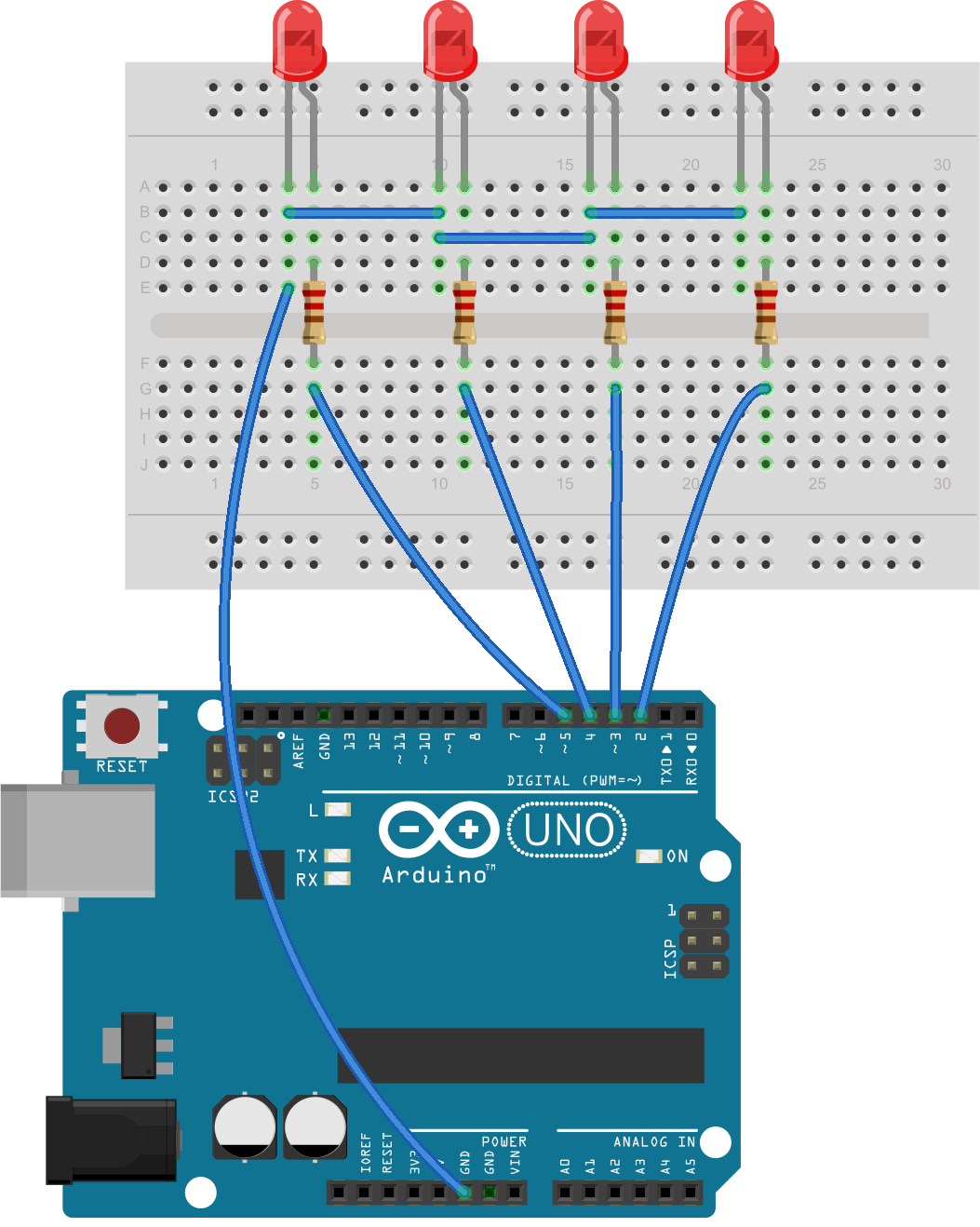
}

}

The for loop will repeat the code inside the curly brackets beneath it as long as some condition is true. Here, we have used a new integer variable, i, which starts with the value 2. Every time the code is executed, the ‘i++’ will add 1 to the value of i. The loop will continue in this way while the value of i is less than 6 (the condition). Once it is equal to or greater than 6, the control comes out of the loop and moves on and continues with whatever code comes after the for loop.

* ***The Schematic & Breadboard Circuit:***

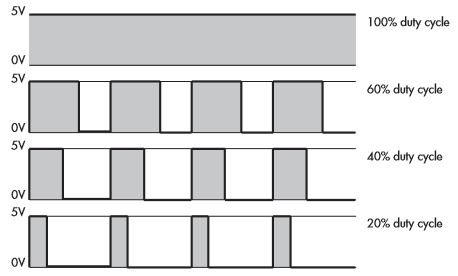




* ***WARNING:*** Do not connect LEDs without the use of resistors as the voltage provide by Arduino digital pin is 5V which is maximum than the optimal operating voltage of LEDs.
* **Project 4 : Demonstrating PWM**

PWM can be used to create the illusion of an LED being on at different levels of brightness by turning the LED on and off quickly, at around 500 cycles per second. The brightness we see is determined by the amount of time the digital output pin is on versus the amount of time it is off—that is, every time the LED is lit or unlit. Because our eyes can’t see flickering faster than 50 cycles per second, the LED appears to have a constant brightness.

The greater the duty cycle (the longer the pin is on compared to the time its off in each cycle), the greater the perceived brightness of the LED connected to the digital output pin. The figure below shows various PWM duty cycles. The filled-in gray areas represent the amount of time that light is on. As you can see, the amount of time per cycle that the light is on increases with the duty cycle.

****

To create a PWM signal, we use the following function: analogWrite(x, y), where x is the digital pin and y is a value for the duty cycle, between 0 and 255, where 0 indicates a 0 percent duty cycle and 255 indicates a 100 percent duty cycle.

* ***What we do:***

Rather than just turning LEDs on and off rapidly using digitalWrite(), we can define the level of brightness of an LED by adjusting the amount of time between each LED’s on and off states using PWM.

* ***What we need:***

1. Arduino UNO & USB Cable
2. Connecting Wires
3. Led
4. 220 ohm resistor

* ***The Code:***

// Project: Demonstrating PWM

int d = 5;

void setup()

{

pinMode(3, OUTPUT); // LED control pin is 3, a PWM capable pin

}

void loop()

{

for ( int a = 0 ; a < 256 ; a++ )

{

analogWrite(3, a);

delay(d);

}

for ( int a = 255 ; a >= 0 ; a-- )

{

analogWrite(3, a);

delay(d);

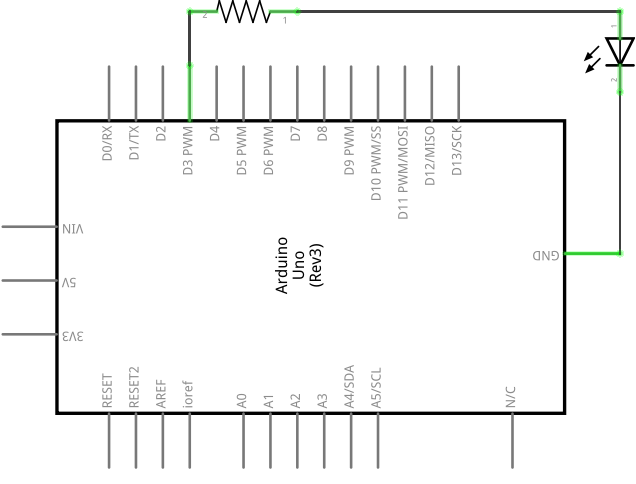
}

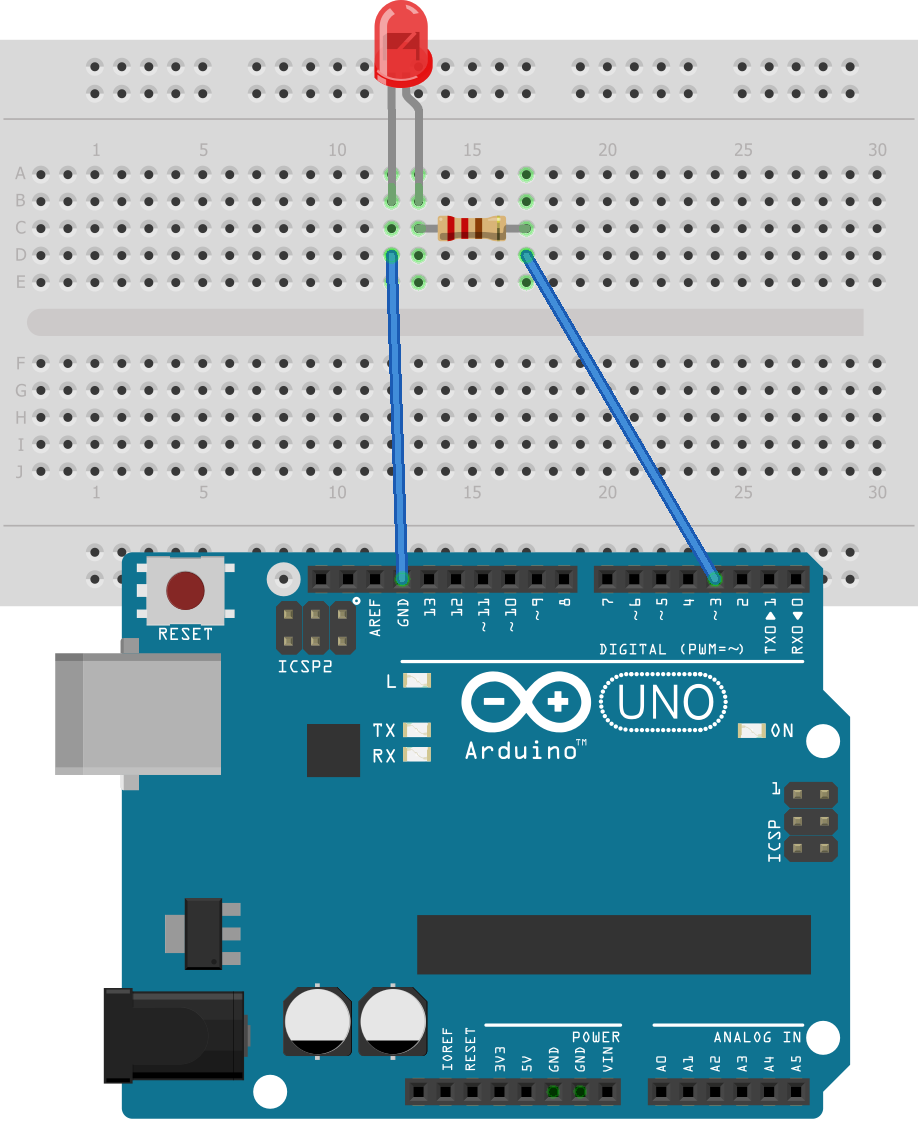
delay(200);

}

The LED at digital pin 3 will exhibit a “breathing effect” as the duty cycle increases and decreases. In other words, the LED will turn on, increasing its brightness until its fully lit, and then reverse. Experiment with the sketch and circuit. For example, make four LEDs breathe at once, or have them do so sequentially (you can use the previous project for this).

* ***The Schematic & Breadboard Circuit:***

******



* ***Remember:*** Only digital pins 3, 5, 6, 9, 10, and 11 on an Arduino board can be used for PWM. They are marked on the Arduino board with a tilde sign (~).
* **Project 5 : Digital Input With Switch**

In previous projects, we used digital I/O pins as outputs to turn LEDs on and off. We can use these same pins to accept input from users—such as detecting whether a push button has been pressed by a user. Like digital outputs, digital inputs have two states: high and low. The simplest form of digital input is a push button. We can insert these directly into our solderless breadboard. A push button allows a voltage or current to pass when the button is pressed, and digital input pins are used to detect the presence of the voltage and to determine whether a button is pressed.

* ***What we do:***

We will create a button that turns on an LED for half a second when pressed.

1. Test to see if the button has been pressed.
2. If the button has been pressed, then turn on the LED for half a second, and then turn it off.
3. If the button has not been pressed, then do nothing.
4. Repeat indefinitely

* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. One push button
5. One LED
6. One 220ohm resistor
7. One 10 kohm resistor
8. One 22 pF capacitor

* ***The Code:***

// Project 4 - Demonstrating a Digital Input

#define LED 12

#define BUTTON 7

void setup()

{

pinMode(LED, OUTPUT); // output for the LED

pinMode(BUTTON, INPUT); // input for the button

}

void loop()

{

if ( digitalRead(BUTTON) == HIGH ) // Check if the button has been pressed

{

digitalWrite(LED, HIGH); // turn on the LED

delay(500); // wait for 0.5 seconds

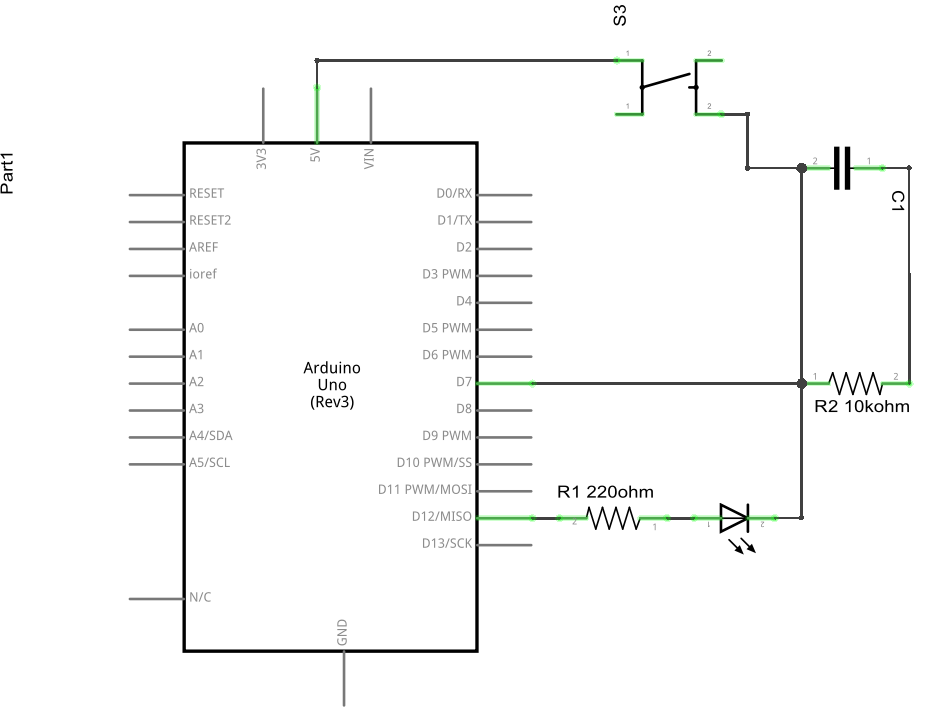
digitalWrite(LED, LOW); // turn off the LED

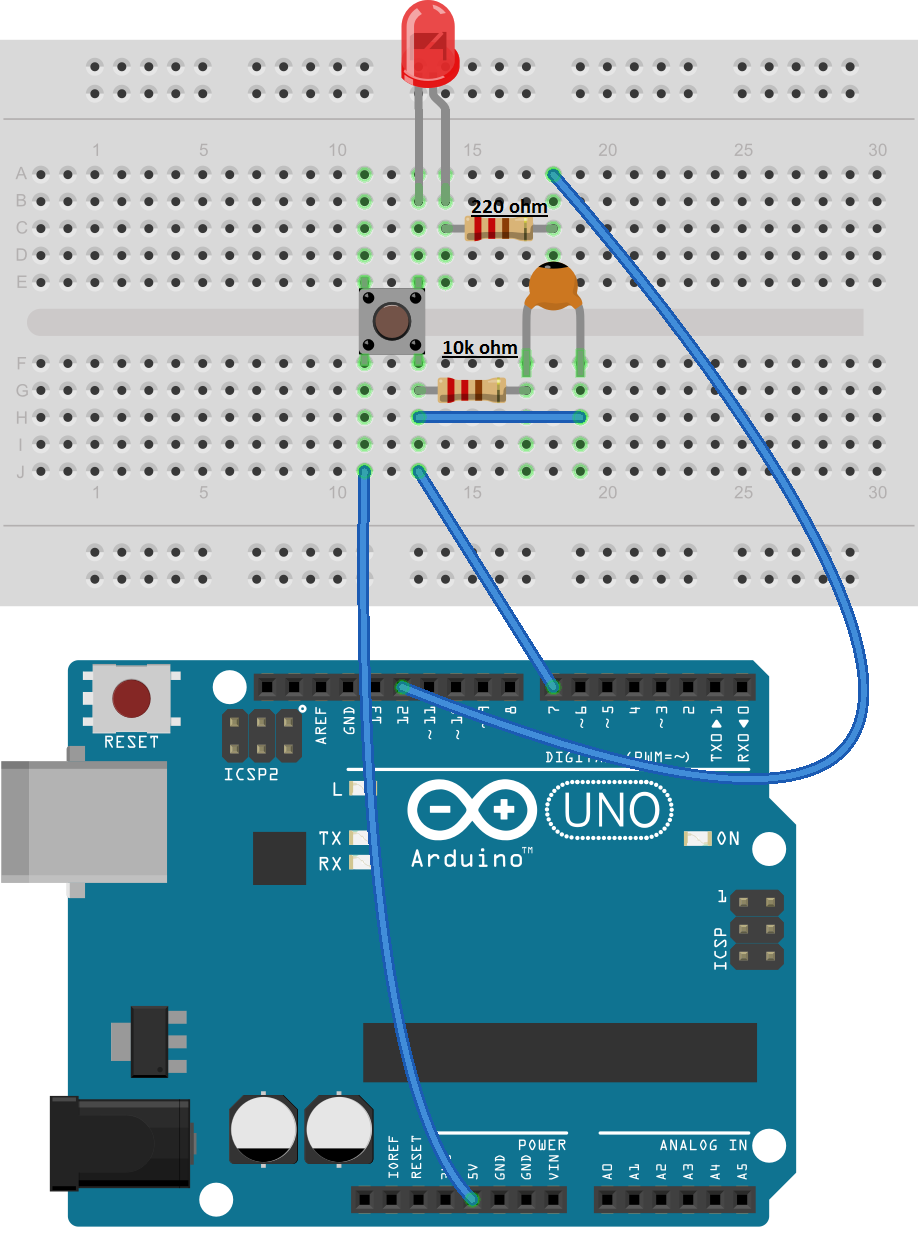
}

}

* ***The Schematic & Breadboard Circuit:***

Notice how the 10 kohm resistor is connected between GND and digital pin seven. We call this a pull-down resistor, because it pulls the voltage at the digital pin almost to zero. Furthermore, by adding a 100 pF capacitor across the 10 kohm resistor, we create a simple **debounce** circuit to help filter out the switch bounce. When the button is pressed, the digital pin goes immediately to high. But when the button is released, digital pin seven is pulled down to GND via the 10 kW resistor, and the 100 nF capacitor creates a small delay. This effectively covers up the bouncing pulses by slowing down the voltage falling to GND, thereby eliminating most of the false readings due to floating voltage and erratic button behavior.





* ***Remember:*** Read the switch Bouncing article to understand schematic and the project completely.
* **Project 7 : The Working Of Potentiometer**
* ***What we do:***

A potentiometer is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value. In this project, that value controls the rate at which an LED blinks.

* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. One LED
5. Two 220ohm resistor
6. One Potentiometer

* ***The Code:***

// Project 7 - Demonstrating the working of potentiometer

int potPin = 0; // select the input pin for the potentiometer

int ledPin = 3; // select the pin for the LED

int val = 0; // variable to store the value coming from the sensor

void setup()

{

pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT

}

void loop()

{

val = analogRead(potPin); // read the value from the sensor

digitalWrite(ledPin, HIGH); // turn the ledPin on

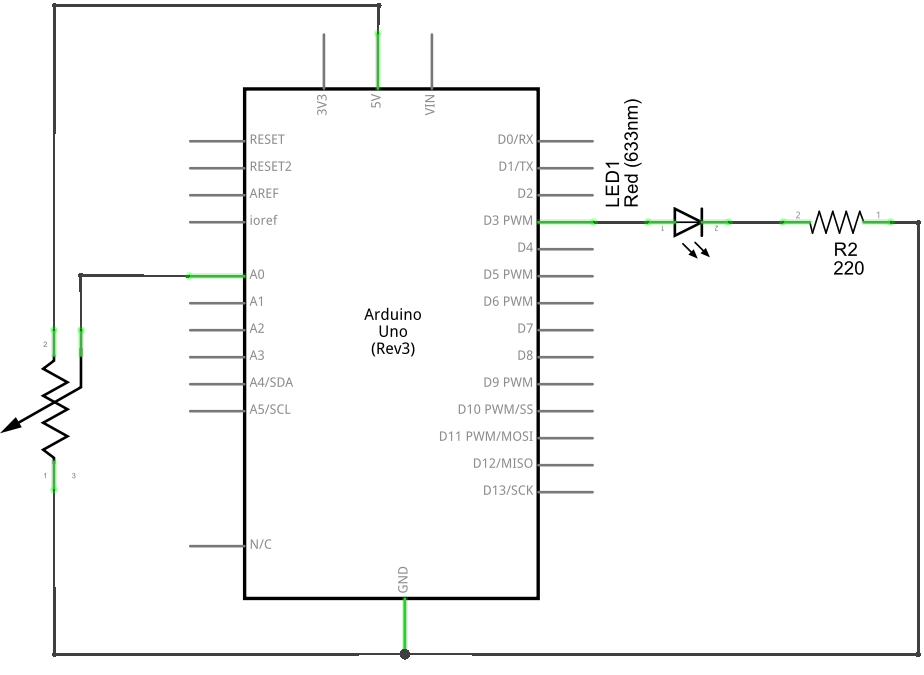
delay(val); // stop the program for some time

digitalWrite(ledPin, LOW); // turn the ledPin off

delay(val); // stop the program for some time

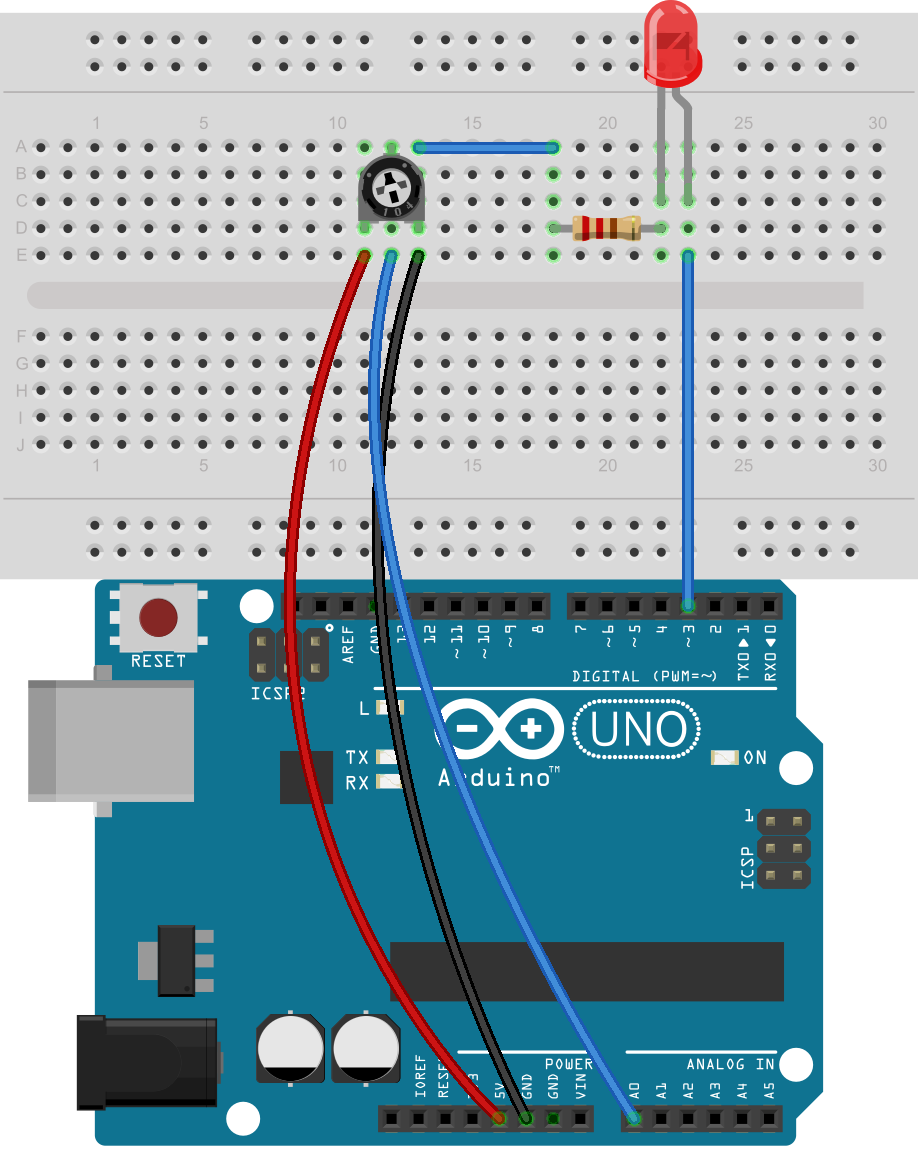
}

* ***The Schematic & Breadboard Circuit:***

******

We connect three wires to the Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from 5 volts to the other outer pin of the potentiometer. The third goes from analog input 2 to the middle pin of the potentiometer.

By turning the shaft of the potentiometer, we change the amount of resistence on either side of the wiper which is connected to the center pin of the potentiometer. This changes the relative "closeness" of that pin to 5 volts and ground, giving us a different analog input. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and we read 0. When the shaft is turned all the way in the other direction, there are 5 volts going to the pin and we read 1023. In between, analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

******

* ***Remember:*** The middle leg of the potentiometer will connect to analog pin.
* **Project 8 : The Working Of Photo-resistor**
* ***What we do:***

In this experiment we are going to use a element called photoresistor. This sensor allows us to interact with the external environment, through intensity of light. The photoresistor is based on light resistance, it will sense the light and will allow the microcontroller in this case Arduino to react and change the intensity of Led Diode.

The photoresistor creates a different resistance based on the intensity or the light. Changing the resistance through intensity changes the voltage too. The microntroller reads different values and will light up the Led with more or less intensity. A low resistance value will occur when the sensor is well lighted and a high resistance value will occur when it is in darkness.

* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. One LED
5. Two 220ohm resistor
6. One Photo-resistor

* ***The Code:***

// Project 8 - Demonstrating the working of photo-resistor

int ledpin = 3; // Define led pin

int photoresistor = 0; // Variable to store data from analog pin for photoresistor

void setup()

{

pinMode(ledpin, OUTPUT); // output for the led pin

}

void loop()

{

photoresistor = (analogRead(0)/4); // Divides input 0-1023 to resemble 0-255

analogWrite(ledpin,photoresistor); // Getting the desired dimming effect

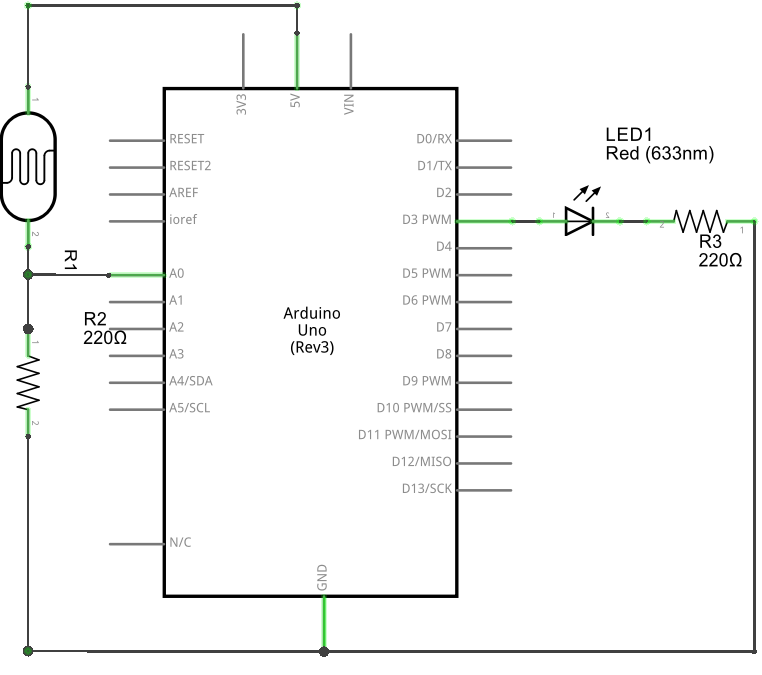
delay(20);

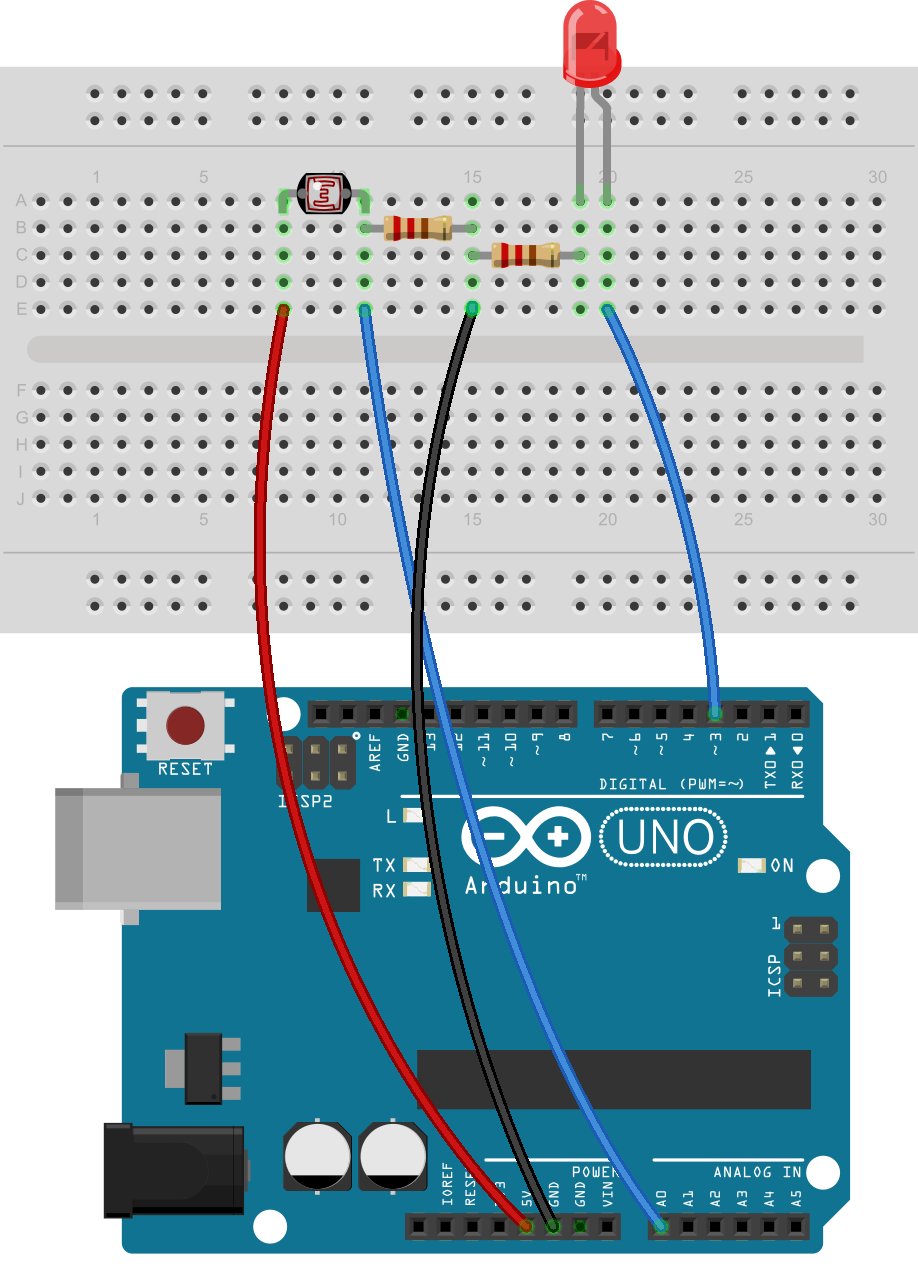
}

We declare the pins that we are going to use for the photoresistor and led diode. We then declare the function of the components that we have connected if is an input or an output . Line 8 is the starting of the cycle that is going to perform while Arduino is powered on. Line 10 is used for reading analog values from photoresistor and storing the values to a variable called “int readAnalogValue”.

Line 11 is a function that stores the value analog read divided by 4 to a variable called “int photoresistor”. Line 13 is the function that makes the Led lighting to change and line 14 is the delay that Arduino performs between different values.

* ***The Schematic & Breadboard Circuit:***

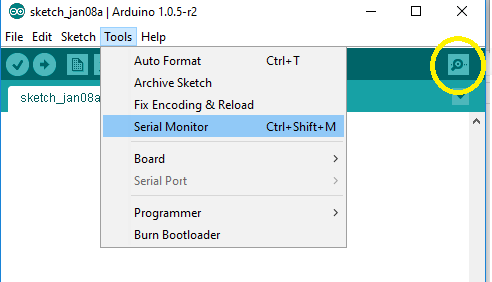




* ***Remember:*** Do not connect led or photo-resistor without the use of resistors as it can harm the components
* **Project 9 : Arduino Serial Monitor Display**
* ***What we do:***

So far, we have sent sketches to the Arduino and used the LEDs to show us output. Blinking LEDs make it easy to get feedback from the Arduino, but blinking lights can tell us only so much. In this project you’ll learn how to use the Arduino’s cable connection and the IDE’s Serial Monitor window to display data from the Arduino.

**The Serial Monitor** is a separate pop-up window that acts as a separate terminal that communicates by receiving and sending Serial Data. See the icon on the far right of the image below.

******

We will us the same schematic as the previous project

* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. One LED
5. Two 220ohm resistor
6. One Photo-resistor

* ***The Code:***

// Project 9 – Arduino Serial Monitor Display

int ledpin = 3; // Define led pin

int photoresistor = 0; // Variable to store data from analog pin for photoresistor

void setup()

{

pinMode(ledpin, OUTPUT); // output for the led pin

Serial.begin (9600); // Serial communication begins

}

void loop()

{

Serial.println(analogRead(0); //Write the value of the photoresistor to the serial

// monitor

photoresistor = (analogRead(0)/4); // Divides input 0-1023 to resemble 0-255

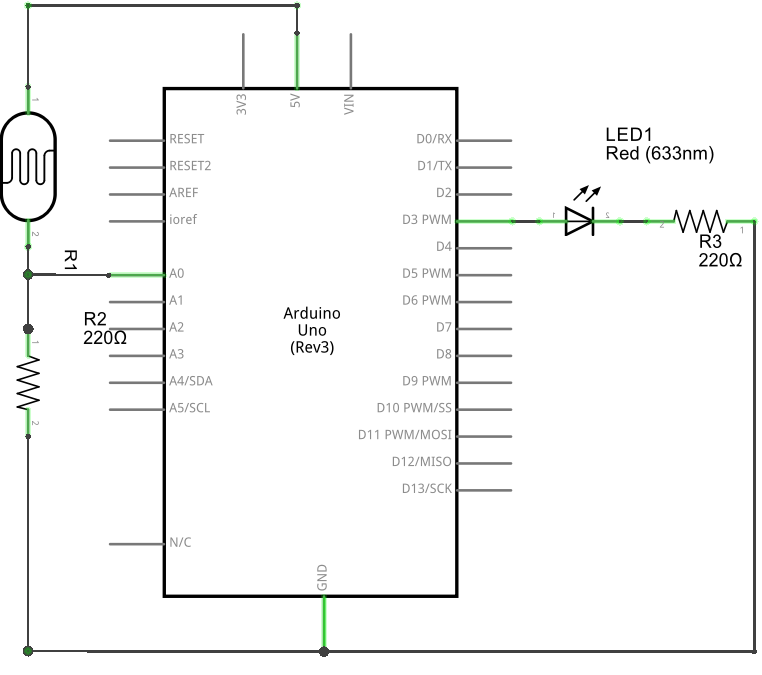
analogWrite(ledpin,photoresistor); // Getting the desired dimming effect

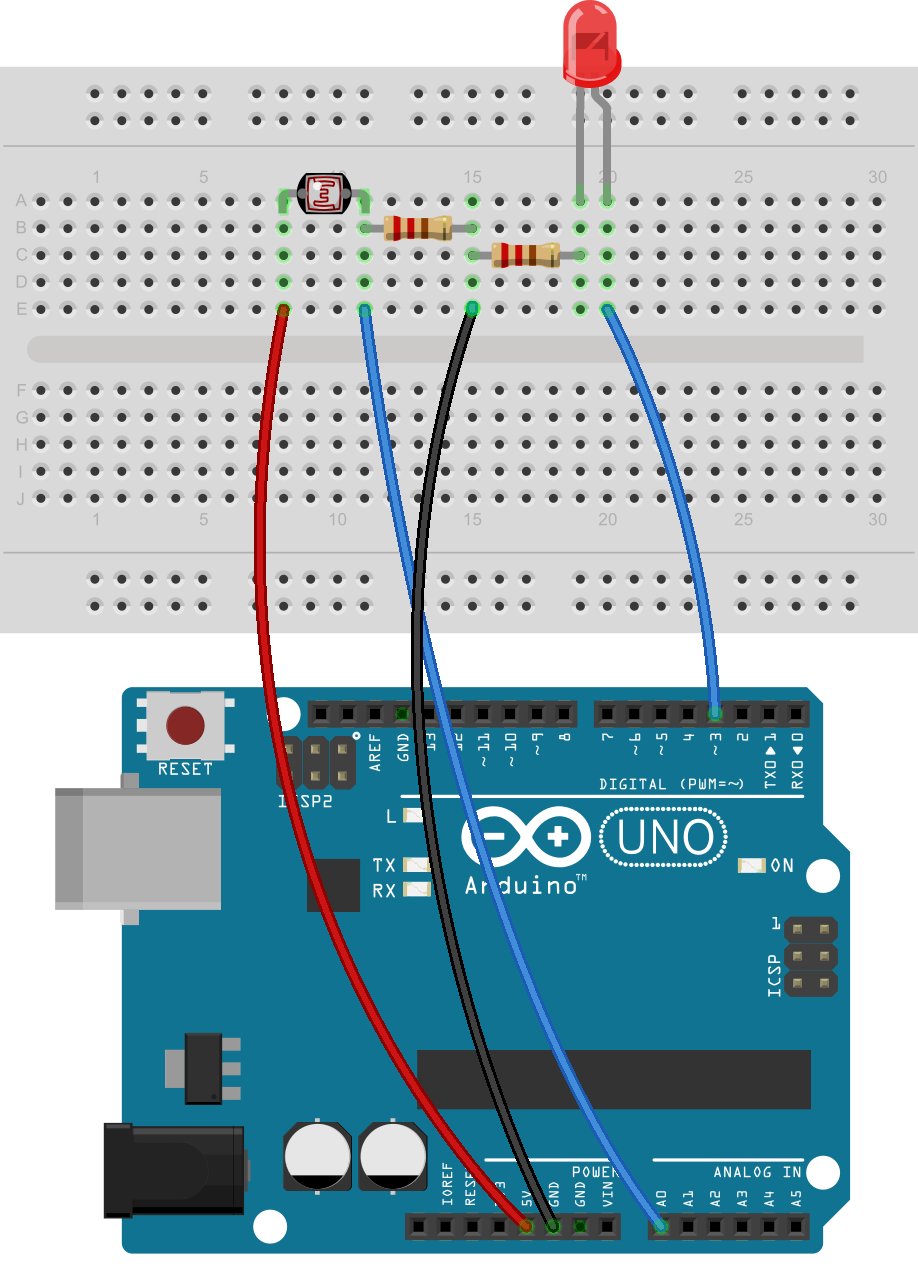
delay(20);

}

Once the code is uploaded, open the serial Monitor and you will be able to monitor the values from the photoresistor on the serial monitor.

* ***The Schematic & Breadboard Circuit:***





* ***Remember:*** Do not connect led or photo-resistor without the use of resistors as it can harm the components
* **Project 10 : Make Some Noise With Piezo Buzzer**
* ***What we do:***

A Piezo is nothing but an electronic device that can both be used to play tones and to detect tones. In our project we are plugging the Piezo on the pin number 9, that supports the functionality of writing a PWM signal to it, and not just a plain HIGH or LOW value. We are taking advantage of the processors capability to produde PWM signals in order to play music.

* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. One Piezo
5. One 1kohm resistor

* ***The Code:***

// Project 10 – Make Some Noise With Piezo Buzzer

#define piezo\_pin 3 // pin 3 is capable of PWM output to drive tones

int del = 500;

void setup()

{

pinMode (piezo\_pin, OUTPUT);

}

void loop()

{

analogWrite (piezo\_pin, 128); // 50 percent duty cycle tone to the piezo

delay(del);

digitalWrite (piezo\_pin, LOW); // turn the piezo off

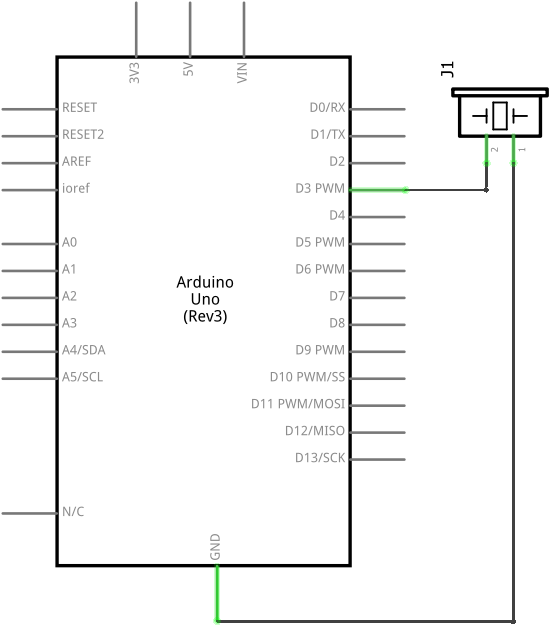
delay(del);

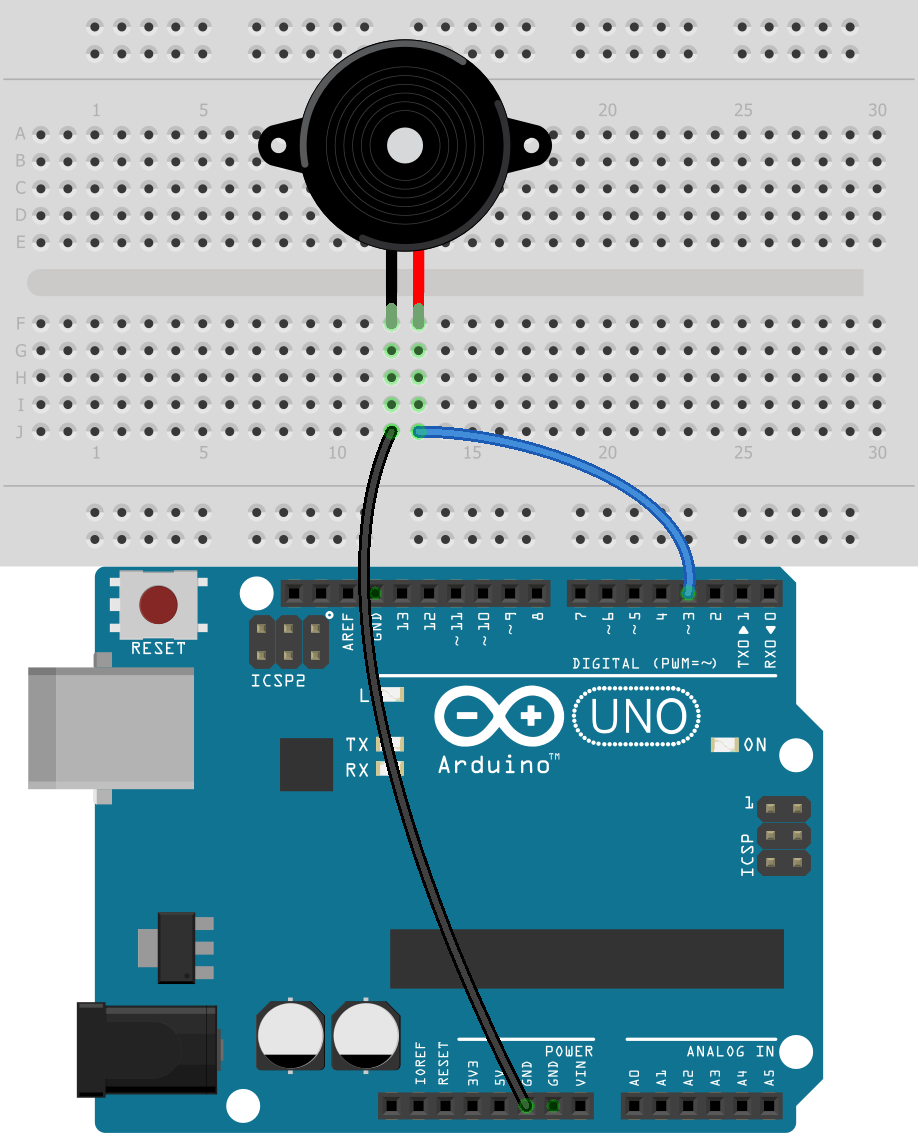
}

This sketch uses pulse-width modulation on digital pin three. If we change the duty cycle in the analogWrite() function (currently it’s 128, which is 50 percent on), then you can alter the volume of the buzzer.

The other thing to remember is that Piezos have polarity, commercial devices have one long and one short pin. We connect the short one to ground and the long one to the output.

* ***The Schematic & Breadboard Circuit:***

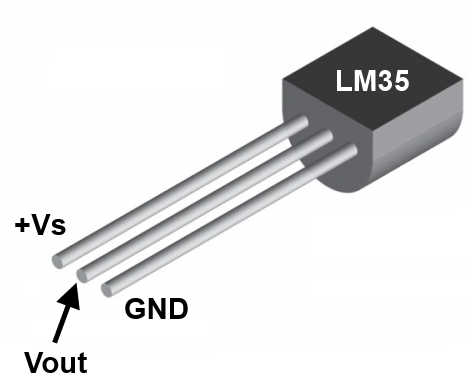
******

******

* ***Note:*** Resistor is not needed if you have 5V Piezo***.*** Also remember to connect the short pin to Ground.
* **Project 11 : Create A Quick Read Thermometer**
* ***What we do:***

In this project, we will make a simple temperature sensor using one LM35 Precision Temperature Sensor and Arduino. The circuit will send serial information about the temperature that you can see on your computer with the help of Serial monitor.

**LM35** is an analog, linear temperature sensor whose output voltage varies linearly with change in temperature. LM35 is three terminal linear temperature sensor from National semiconductors. The pin out of  LM35 is shown in the figure below.



* ***What we need:***

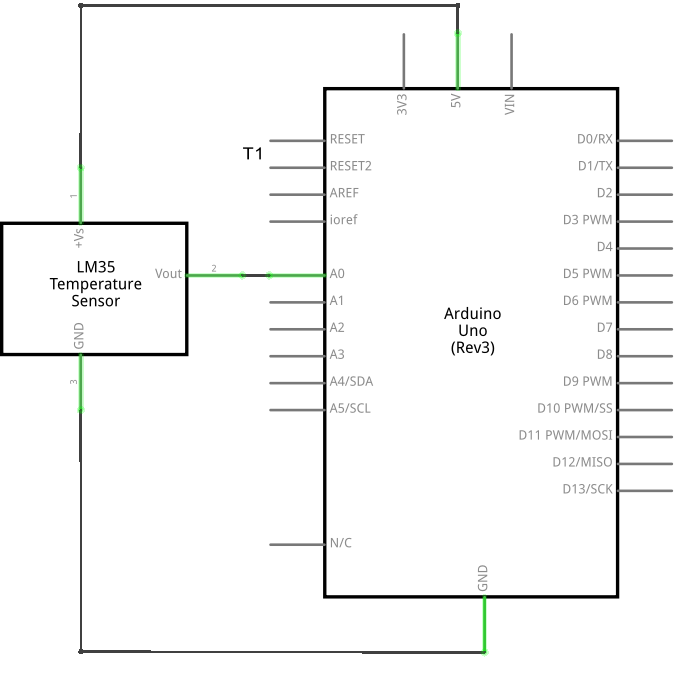
1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. LM35 Temperature Sensor

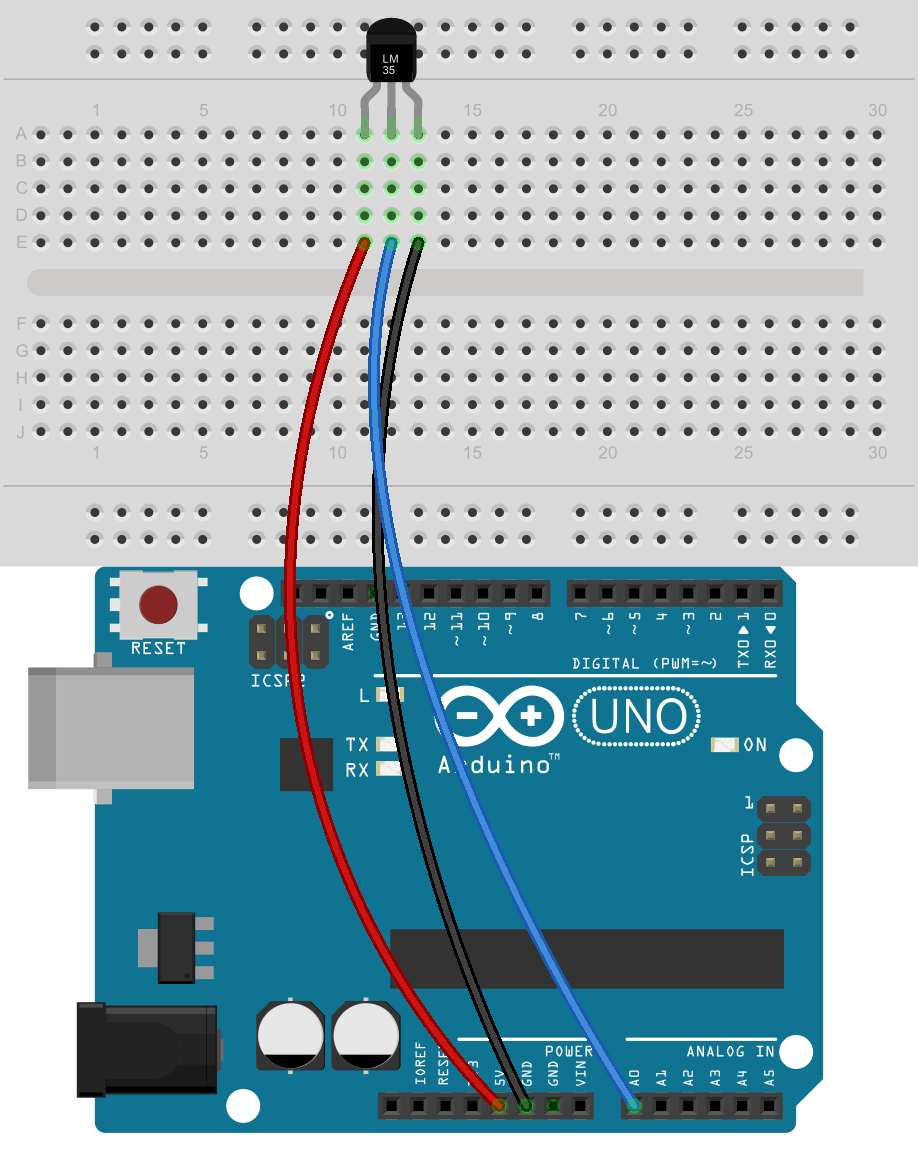
* ***The Code:***

// Project 11 – Create A Quick Read Thermometer

int val; // to store value from LM35  
int tempPin = 0; // analog pin to connect LM35  
  
void setup()  
{  
 Serial.begin(9600);   
}  
void loop()  
{  
 val = analogRead(tempPin);  
 float mv = ( val/1024.0)\*5000;   
 float cel = mv/10;  
  
 Serial.print("TEMPRATURE = ");  
 Serial.print(cel);   
 Serial.print("\*C");  
 Serial.println();  
 delay(1000);  
}

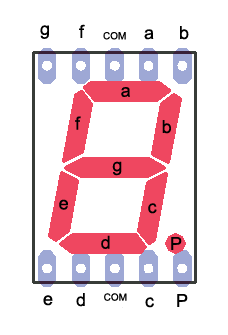
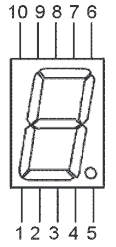
* ***The Schematic & Breadboard Circuit:***

******

******

* ***Note:*** Check twice the connection of LM53 before switching on the circuit or else the temperature sensor will burn up.
* **Project 12 : Hands On Seven Segment Display**
* ***What we do:***

LEDs are fun, but there are limits to the kinds of data that can be displayed with individual lights. In this section we’ll begin working with numeric digits in the form of seven-segment LED displays.The severn-segment display has seven LEDs arranged in the shape of number eight

* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. 7 Segment display
5. 220 ohm resistor

* ***The Code:***

// Project 11 – Hands on Seven Segment Display

// we will display number ‘1’ and ‘2’ on the 7-Segment and will leave rest for you to

// explore

int pin\_1 = 6;

int pin\_2 = 5;

int pin\_4 = 4;

int pin\_6 = 9;

int pin\_7 = 8;

int pin\_9 = 2;

int pin\_10 = 3;

void setup()

{

pinMode (pin\_1, OUTPUT);

pinMode (pin\_2, OUTPUT);

pinMode (pin\_4, OUTPUT);

pinMode (pin\_6, OUTPUT);

pinMode (pin\_7, OUTPUT);

pinMode (pin\_9, OUTPUT);

pinMode (pin\_10, OUTPUT);

}

void loop()

{

// first we will display number ‘1’ :

pinMode (pin\_6, LOW);

pinMode (pin\_4, LOW);

delay(5000);

pinMode (pin\_6, HIGH);

pinMode (pin\_4, HIGH);

// now we will display number ‘2’ :

pinMode (pin\_7, HIGH);

pinMode (pin\_6, HIGH);

pinMode (pin\_10, HIGH);

pinMode (pin\_1, HIGH);

pinMode (pin\_2, HIGH);

delay(5000);

}

The above code will display number

Connect the pins described below:

1. Arduino Pin 2 to Pin 9

2. Arduino Pin 3 to Pin 10

3. Arduino Pin 4 to Pin 4.

4. Arduino Pin 5 to Pin 2

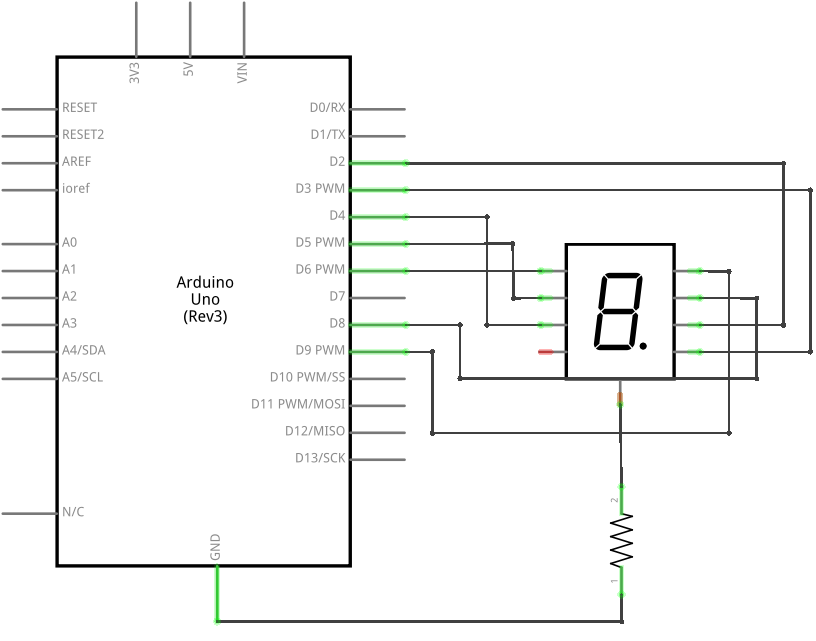
5. Arduino Pin 6 to Pin 1

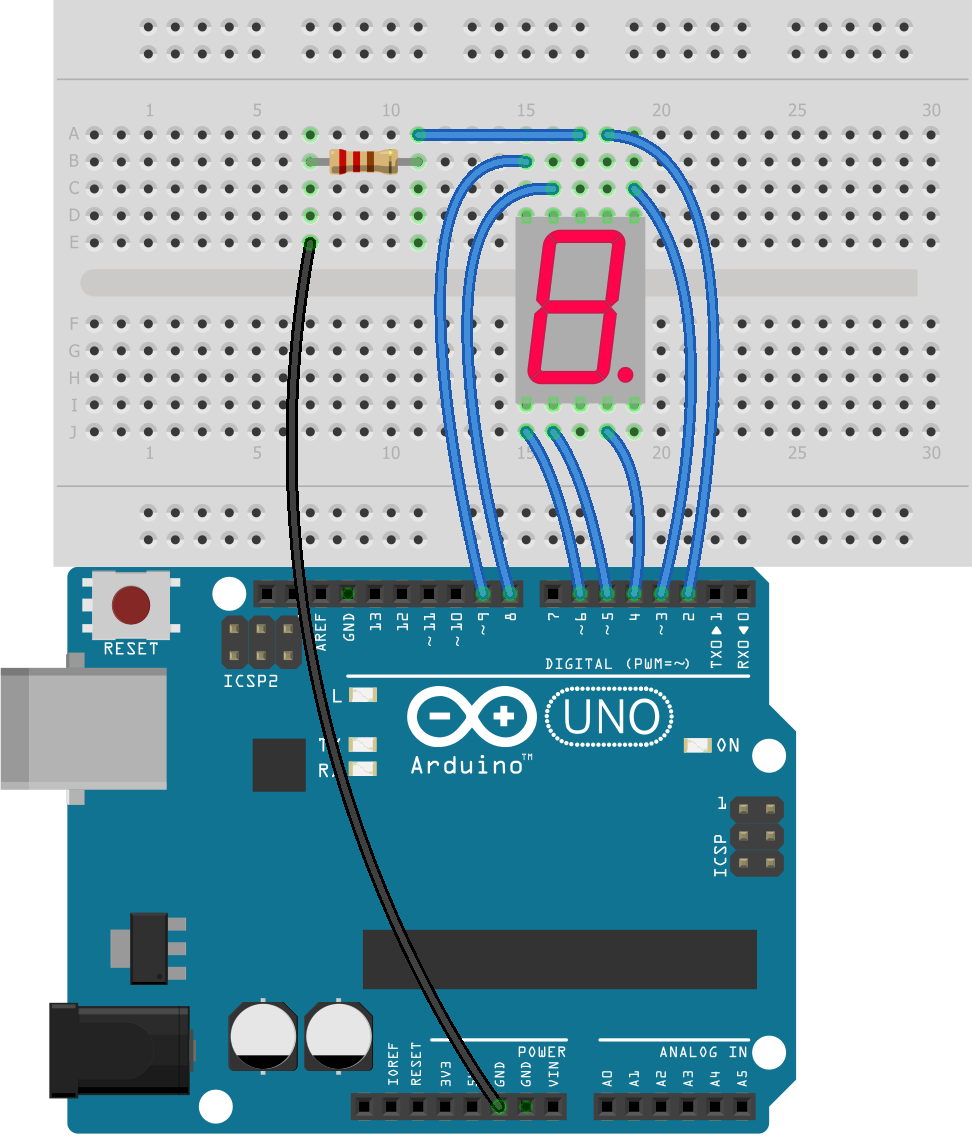
6. Arduino Pin 8 to Pin 7

7. Arduino Pin 9 to Pin 6

8. GND to Pin 3 or Pin 8 connected with 220 ohm resistor

* ***The Schematic & Breadboard Circuit:***

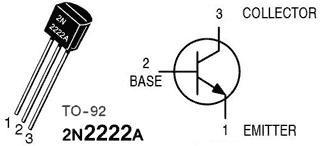
******

******

* ***Note:*** Do connect the resistor before switching on the circuit or the leds in the 7 Segment will break down
* **Project 13 : Working of DC Motor**
* ***What we do:***

In this lesson, you will learn how to control a small DC motor using an Arduino and a transistor.

We will use of the circuit and code, we make in project 6, the only difference is that we will use a DC Motor instead of led and will power it up using the 9V Battery



* ***What we need:***

1. Breadboard
2. Arduino and USB Cable
3. Connecting Wires
4. DC Motor
5. 220 ohm resistor
6. 1N4007 Diode
7. 22pf Capacitor
8. 2N2222A Transistor

* ***The Code:***

// Project 13: Working of DC Motor

#define base\_pin 8

void setup()

{

pinMode(base\_pin, OUTPUT); // output for the transistor base pin

}

void loop()

{

digitalWrite(base\_pin, HIGH); // provide current to base pin of transistor

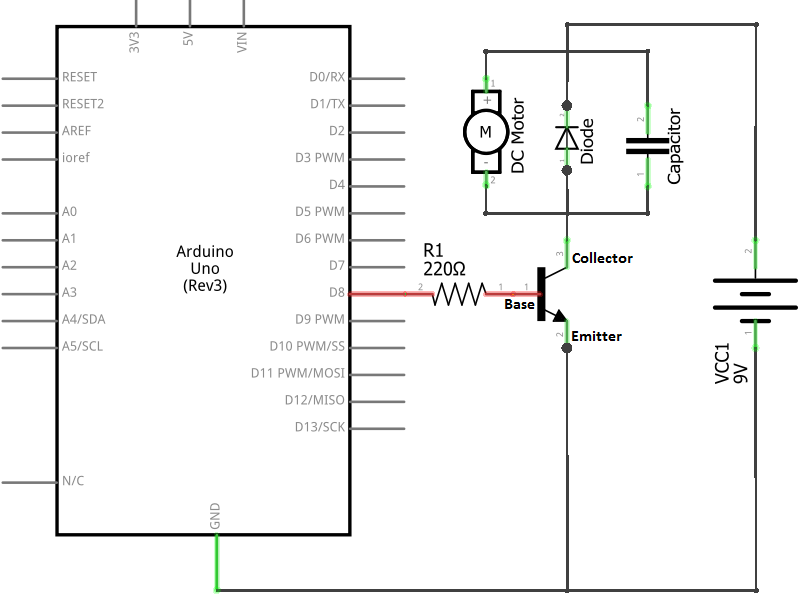
delay(5000); // wait for 5 seconds

digitalWrite(base\_pin, LOW); // stop the current flow to the transistor base pin

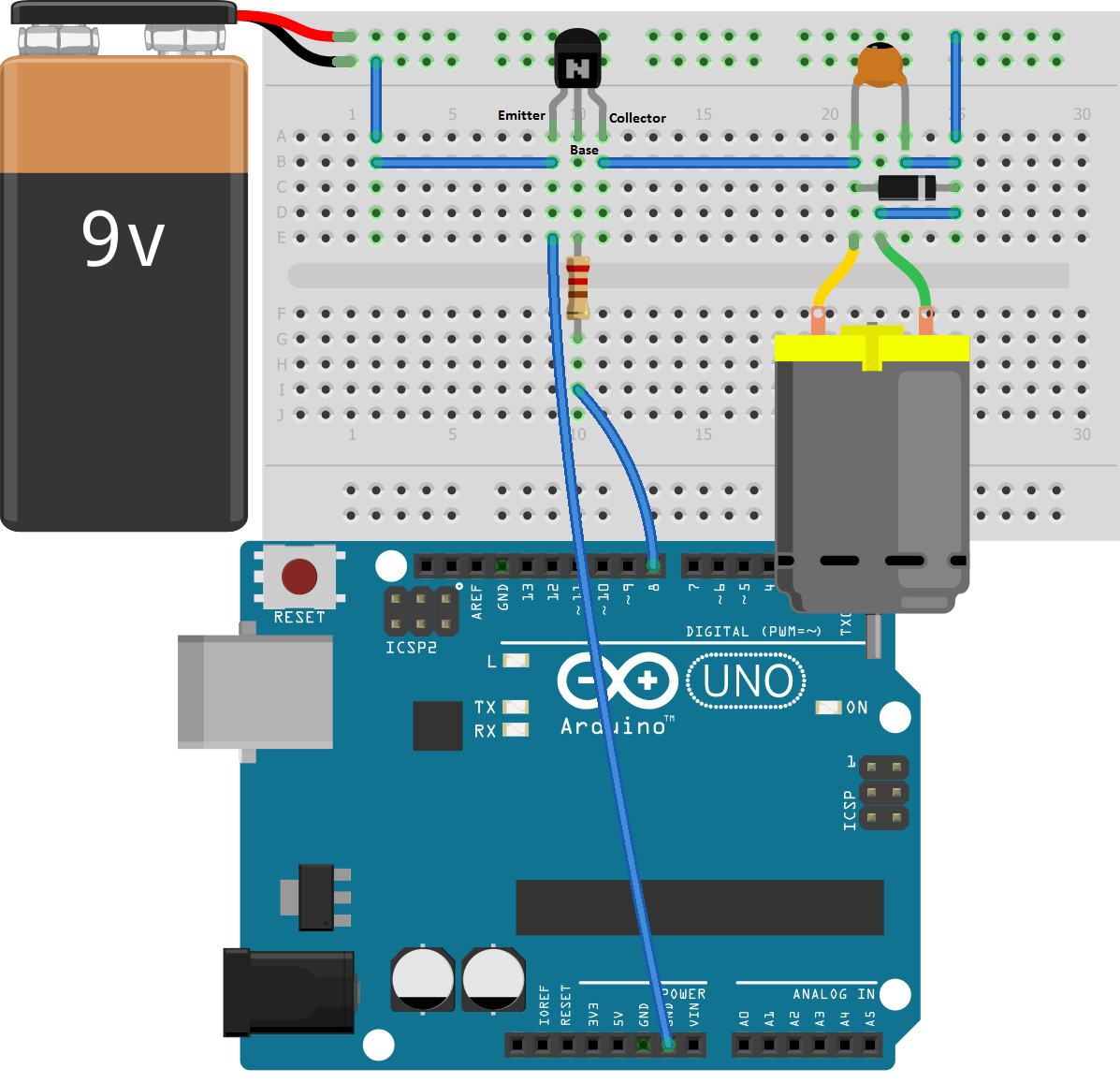
delay(5000); // wait for 5 seconds

}

* ***The Schematic & Breadboard Circuit:***

******

When the current is switched off from the motor, stray current exists for a brief amount of time inside the motor’s coil and has to go somewhere. The diode allows the stray current to loop around through the coil until it dissipates as a tiny amount of heat.

******

* ***Note :*** Please complete the project 6 before attempting this one
* ***Warning:*** Do not try to run the motor using the Arduino digital Pin or 5V as Arduino digital pin is not capable of providing the amount of current required by DC Motor.