**E l e c d u i n o-Un**

**W o r k s h o p**

****

**A Hands-on introduction to Arduino with 15 basic projects**

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 this below guy confused what to do with all the components you have?????????



Ok! You choose “Yes”

Now, do you want to make something useful 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”!!!!

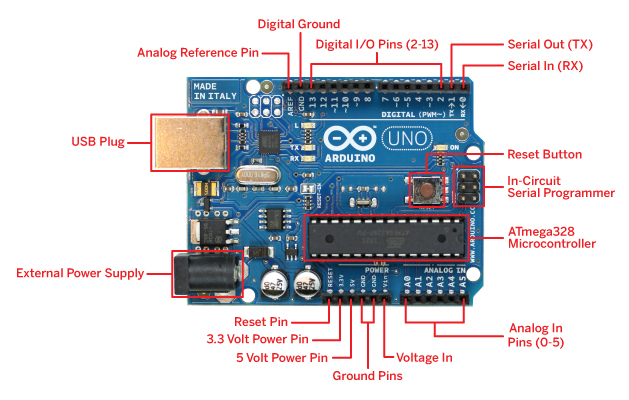


* If Your Answer is “No”, We recommend :



* **Project 1 : Know Your Arduino**

Have you ever looked at some gadget and wondered how it really worked? Maybe it was a remote control boat, the system that controls an elevator, a vending machine, or an electronic toy? Or have you wanted to create your own robot or electronic signals for a model railroad, or perhaps you’d like to capture and analyze weather data over time? Where and how do you start? The Arduino board (shown in Figure) can help you find some of the answers to the mysteries of electronics in a hands-on way. The original creation of Massimo Banzi and David Cuartielles, the Arduino system offers an inexpensive way to build interactive projects, such as remote-controlled robots, GPS tracking systems, and electronic games. The Arduino project has grown exponentially since its introduction in 2005. It’s now a thriving industry, supported by a community of people united with the common bond of creating something new. You’ll find both individuals and groups, ranging from interest groups and clubs to local hackerspaces and educational institutions, all interested in toying with the Arduino



* ***Required Software:***

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

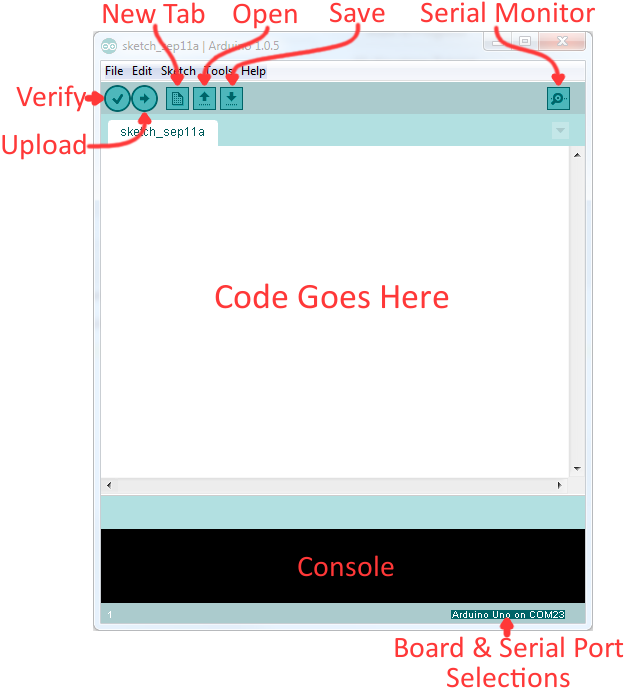
• Mac OS X or higher

• 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 difficultly in any section of installation, please refer to the installation video in the “videos” folder

******

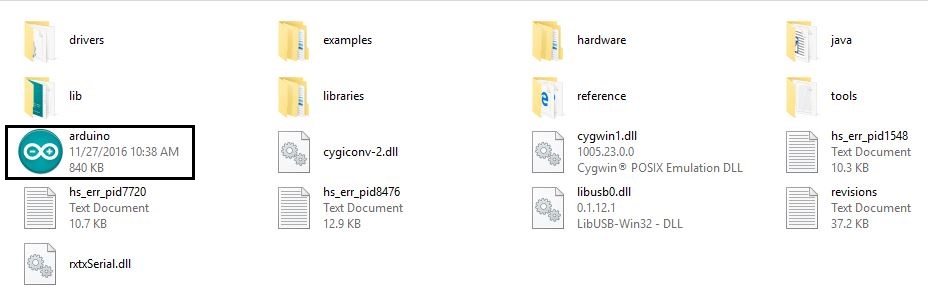
The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with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

In this section, you’ll find instructions for downloading the IDE, installing drivers, and configuring the IDE in 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.

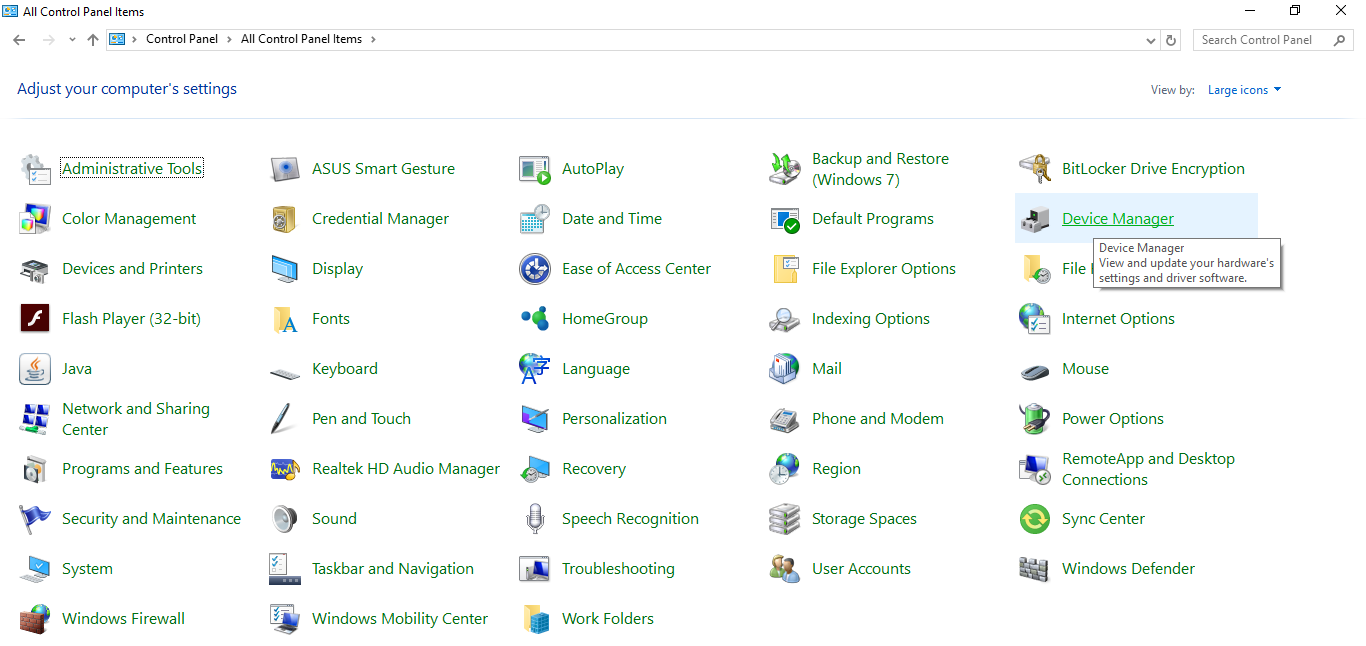


* ***Installing Drivers:***

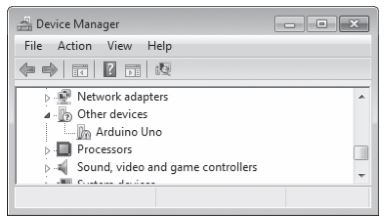
The next task is to install the drivers for your Arduino board’s USB interface.

1. Connect your Arduino to your PC with the USB cable. After a few moments an error message will be displayed, which will say something like “Device driver software not successfully installed.” Just close 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

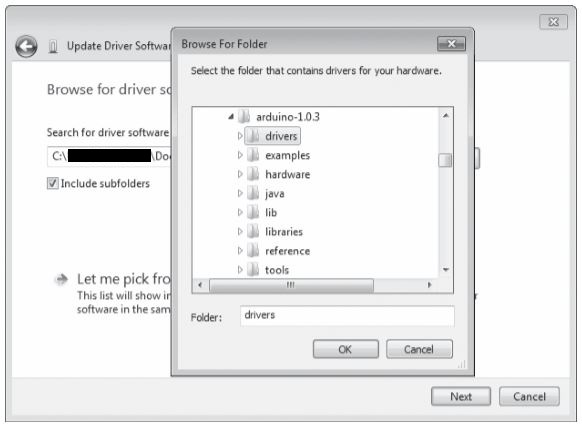


Control panel

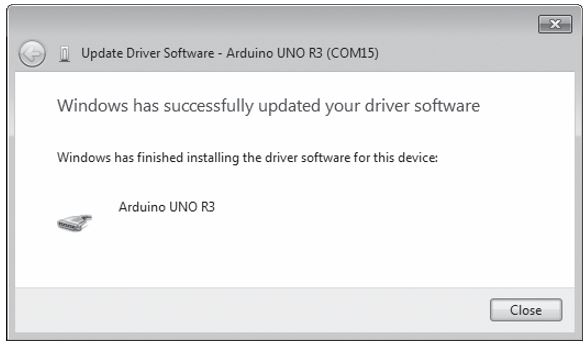


The Device Manager

3. Right-click Arduino Uno under Other Devices and select Update Driver Software. Then, select the Browse my computer for driver soft 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 that follows. Windows may present 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 installed and the COM port number the Arduino is connected to, as shown in Figure.

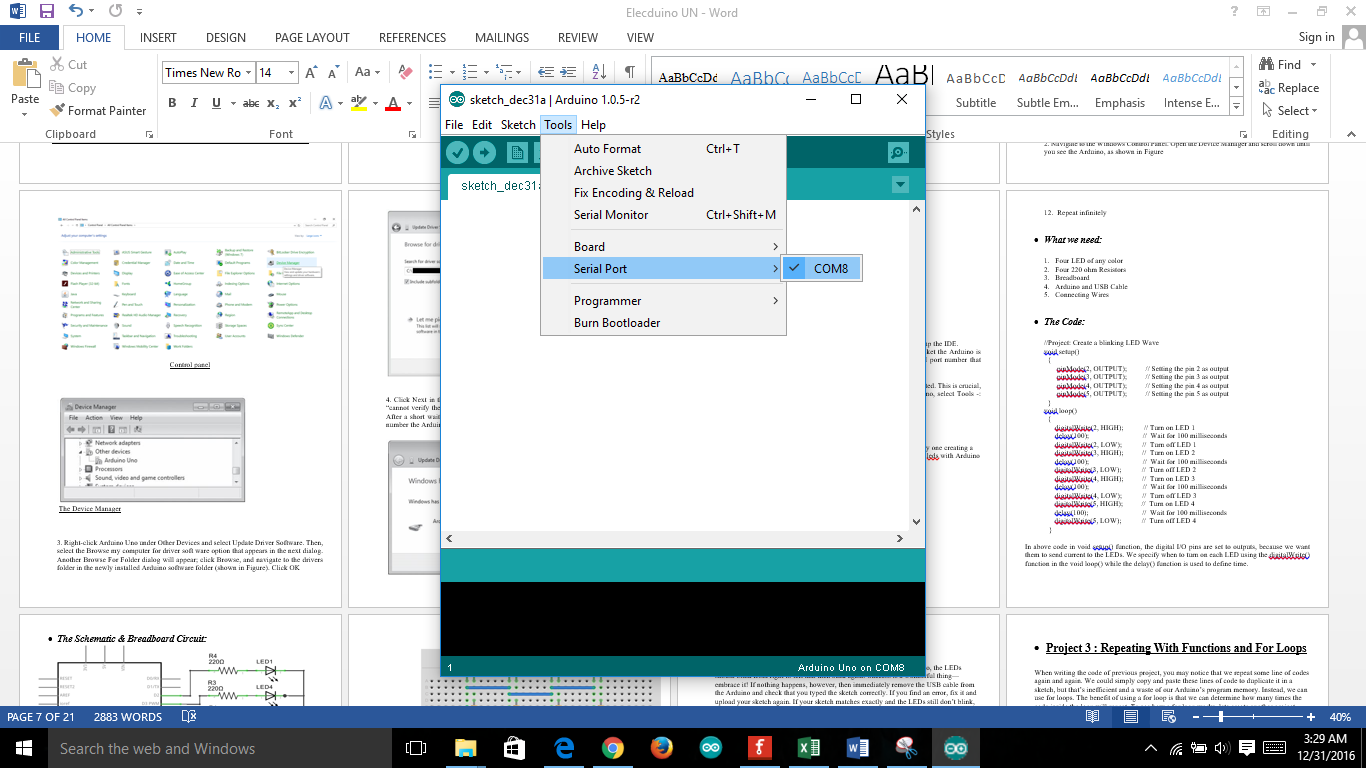
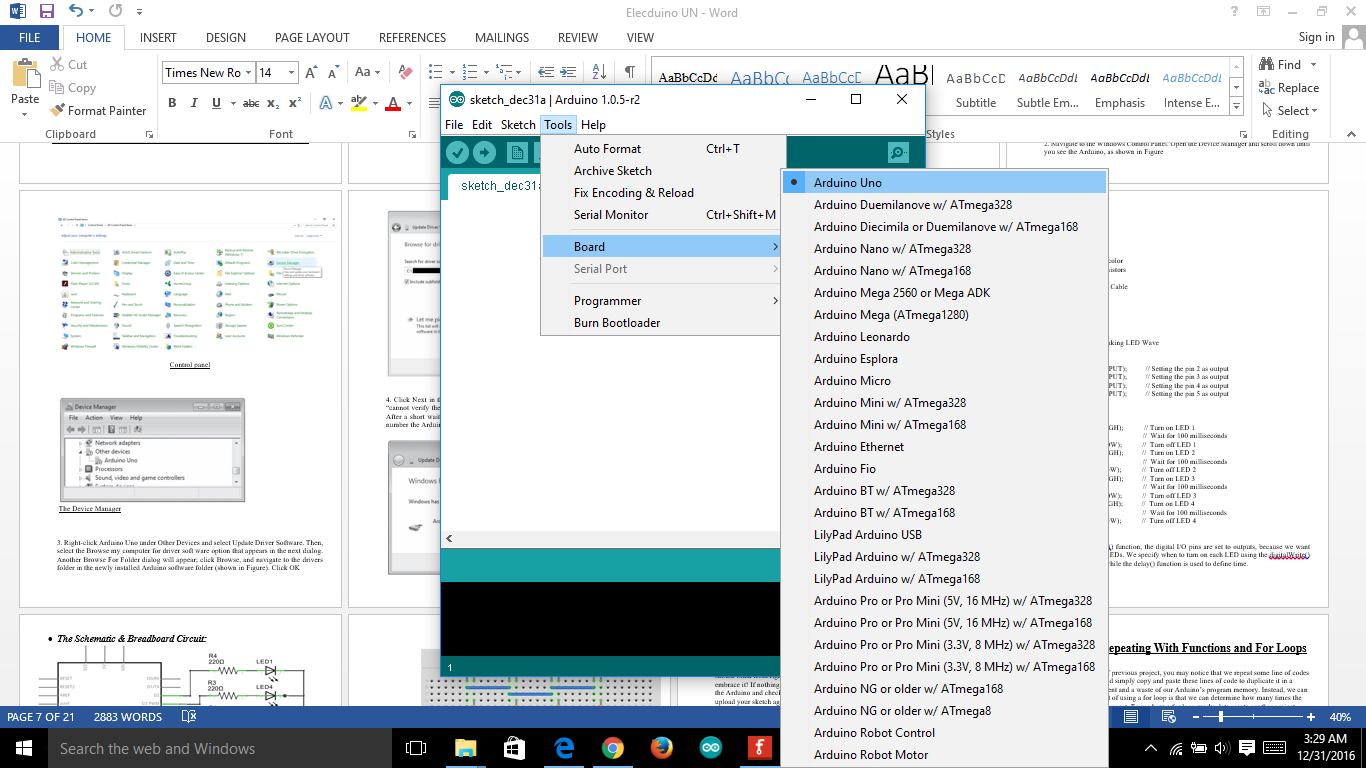


* ***Setting Up The IDE:***

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 the Arduino is connected to by selecting Tools -: Serial Port and selecting 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 have connected. This is crucial, as the Arduino boards do differ. For example, as we are using the Uno, select Tools -: Board -: Arduino Uno.

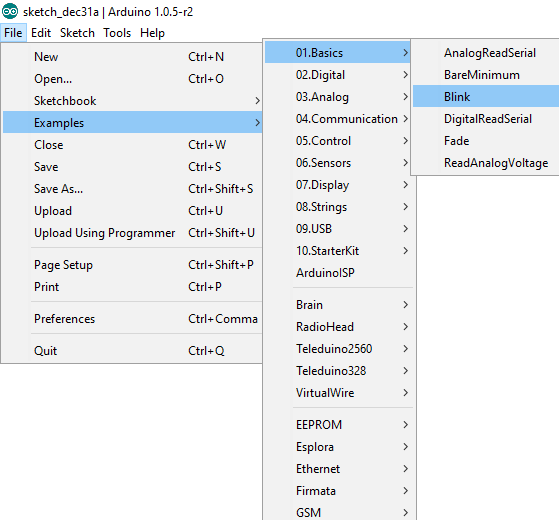
 

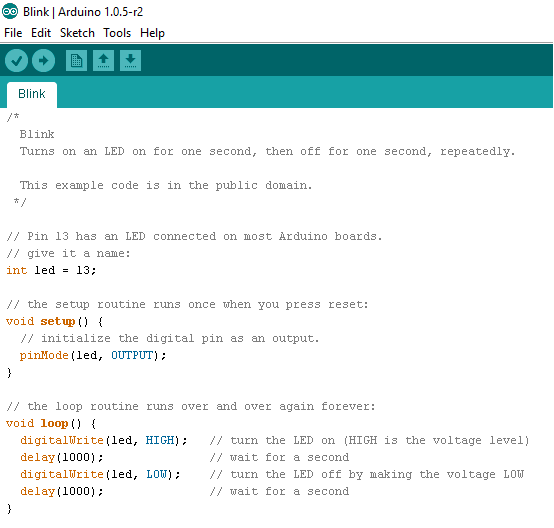


* ***Testing That Everything is Working Fine:***

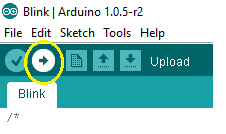
To test that everything is working fine, connect your Arduino board to your computer, Select the Board and COM port.

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

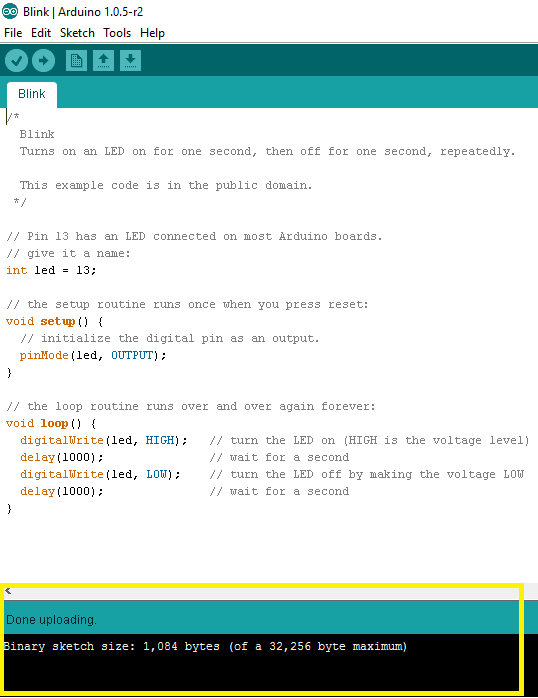




Once the sketch is opened, click on the upload button at the top left.



As soon as you do this, 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 Blink Code:***

The Arduino board has already a led attach with its pin 13. By 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 (example pin number 12) 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 line:

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 in your projects 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 will kill you.



* ***Looking Ahead:***

You’re about to embark on a fun and interesting journey, and once you complete this book you’ll be creating things you may never have thought 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 to 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

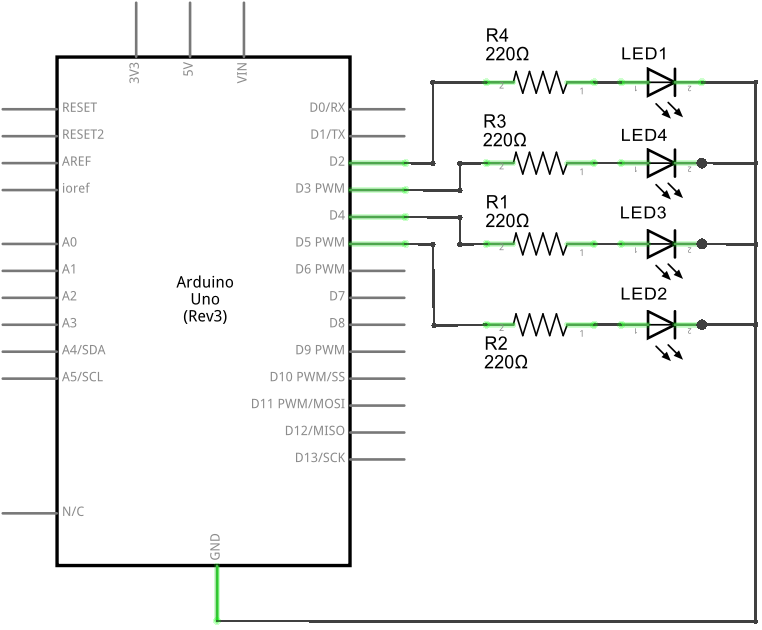
delay(100); // Wait for 100 milliseconds

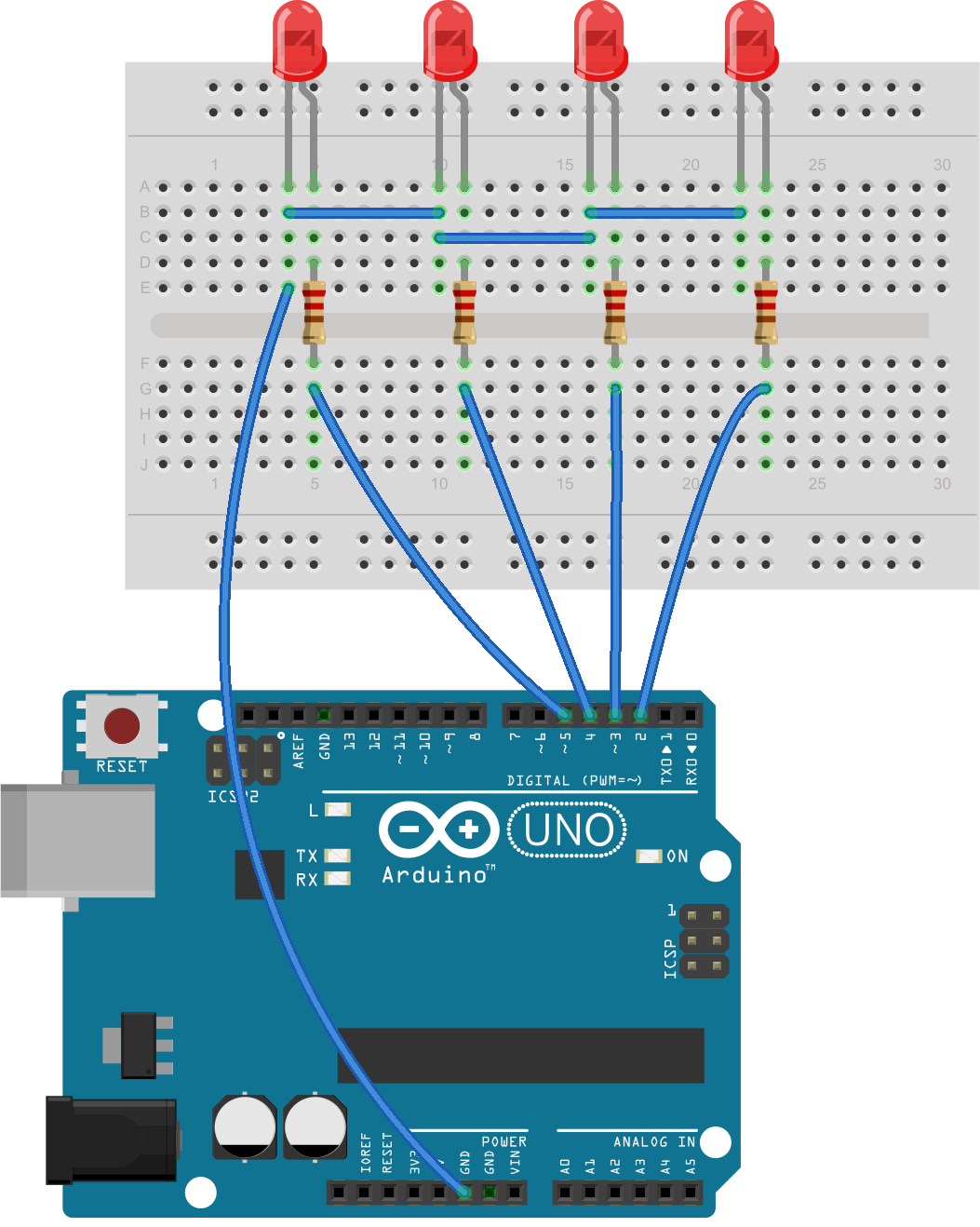
digitalWrite(5, LOW); // Turn off LED 4

}

In above code 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 make sense of the circuit. For example, when we use digitalWrite(2, HIGH), a high voltage of 5 V flows from digital pin 2, through the current-limiting resistor, through the LED via the anode and then the cathode, and finally back to the Arduino’s GND socket to complete the circuit. Then, when we use digitalWrite(2, LOW), the current stops and the LED turns off.

* ***Running the sketch:***

Now connect your Arduino and upload the sketch. After a second or two, the LEDs should blink from right to left and then back again. Success is a wonderful thing—embrace it! If nothing happens, however, then immediately remove the USB cable from the Arduino and check that you typed the sketch correctly. If you find an error, fix it and upload your sketch again. If your sketch matches exactly and the LEDs still don’t blink, check your 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 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 sketch for this project, we used the function delay(500) to keep the LEDs turned on. The problem with the sketch as written is that it’s not very flexible. If we want to make a change to the delay time, then we have to change each entry manually. To address this problem, we’ll 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 called 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 the delay() functions. When you upload the sketch after making these changes, the LEDs will turn on and off at a much faster rate, as the delay value is much smaller at the 250 value. int indicates that the variable contains an integer—a whole number between −32,768 and 32,767. Simply put, any integer value 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 things up even more:

int d = 100;

Experiment 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 continue to use it with the next project.



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

When writing the code of previous project, you may notice that we repeat some line of codes again and again. We could simply copy and paste these lines of code to duplicate it in a sketch, but that’s 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 determine how 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 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: Repeating with function and for loops

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);

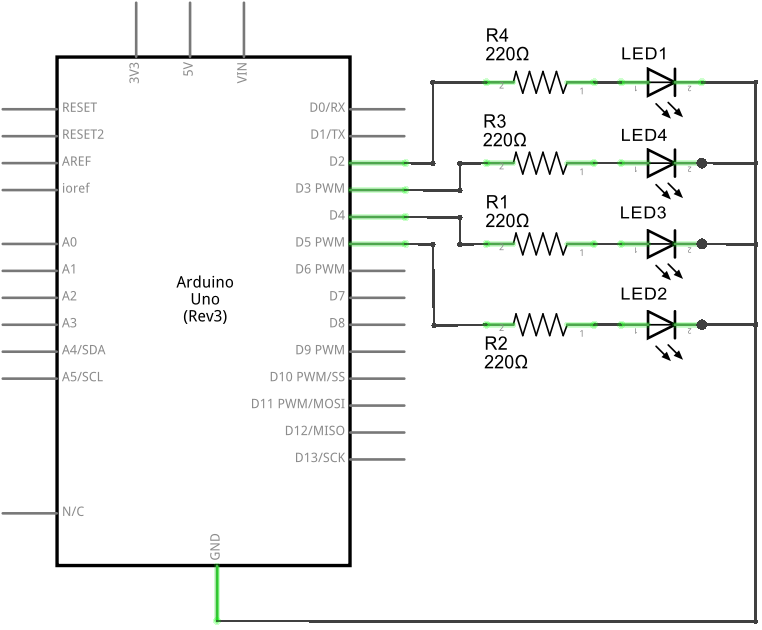
delay(val);

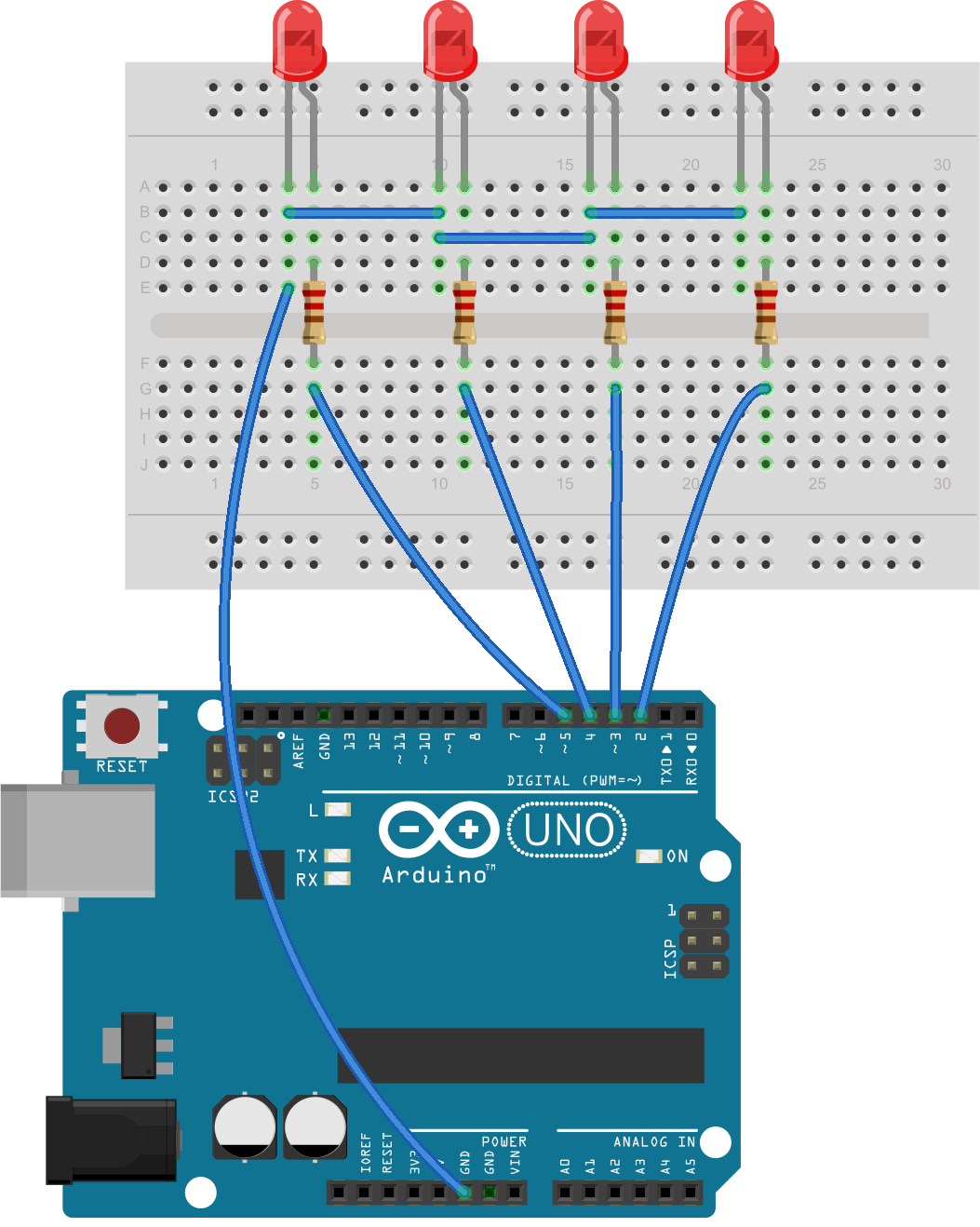
}

}

The for loop will repeat the code within 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 fashion while the value of i is less than 6 (the condition). Once it is equal to or greater than 6, the Arduino moves on and continues with whatever code comes after the for loop.

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

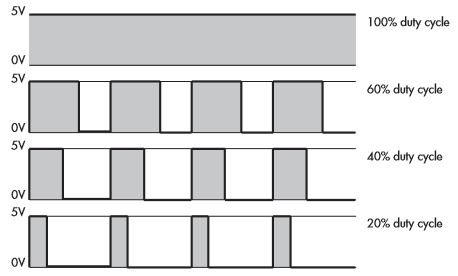




* ***WARNING:*** Do not connect leds without the use of resistors as the voltage provide by Arduino digital pin is 5V which is maximum than the optimal operating voltage of leds.
* **Project 4 : Demonstrating PWM**

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 rapidly, at around 500 cycles per second. The brightness we perceive 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 flickers 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 off in each cycle), the greater the perceived brightness of the LED connected to the digital output pin. Below figure shows various PWM duty cycles. The filled-in gray areas represent the amount of time that the 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 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 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 pulse-width modulation

* ***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);

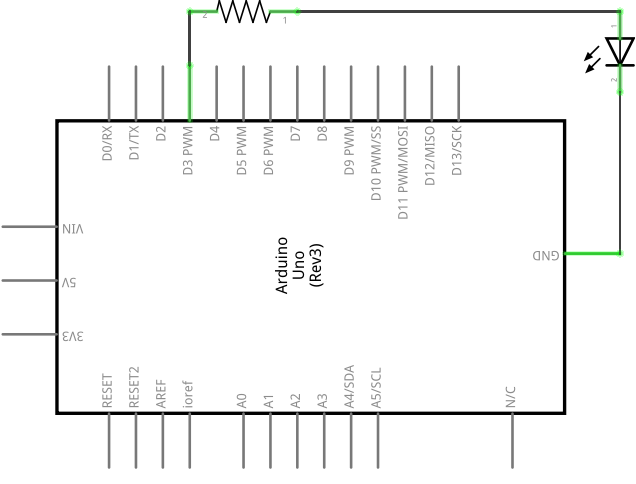
}

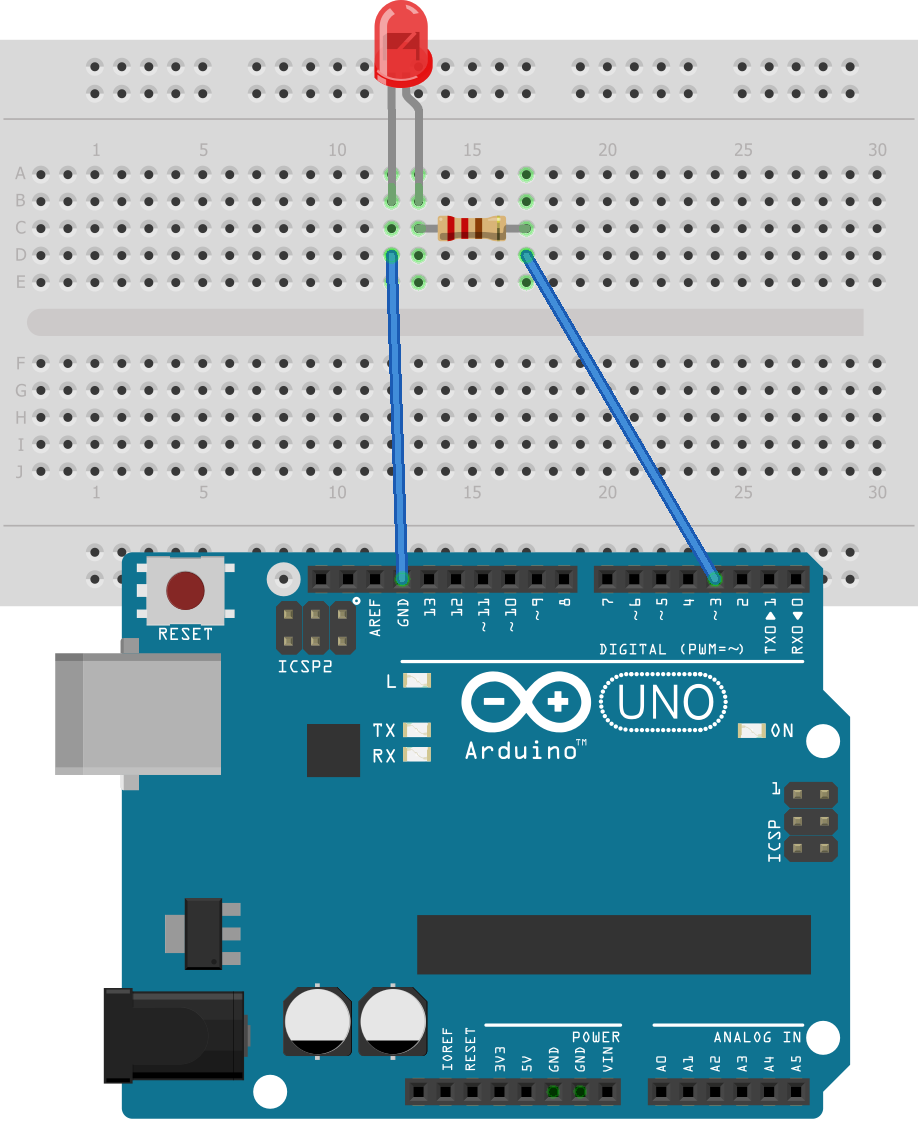
delay(200);

}

The LED on digital pin 3 will exhibit a “breathing effect” as the duty cycle increases and decreases. In other words, the LED will turn on, increasing in brightness until fully lit, and then reverse. Experiment with the sketch and circuit. For example, make five 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 a regular Arduino board can be used for PWM. They are marked on the Arduino board with a tilde (~),
* **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

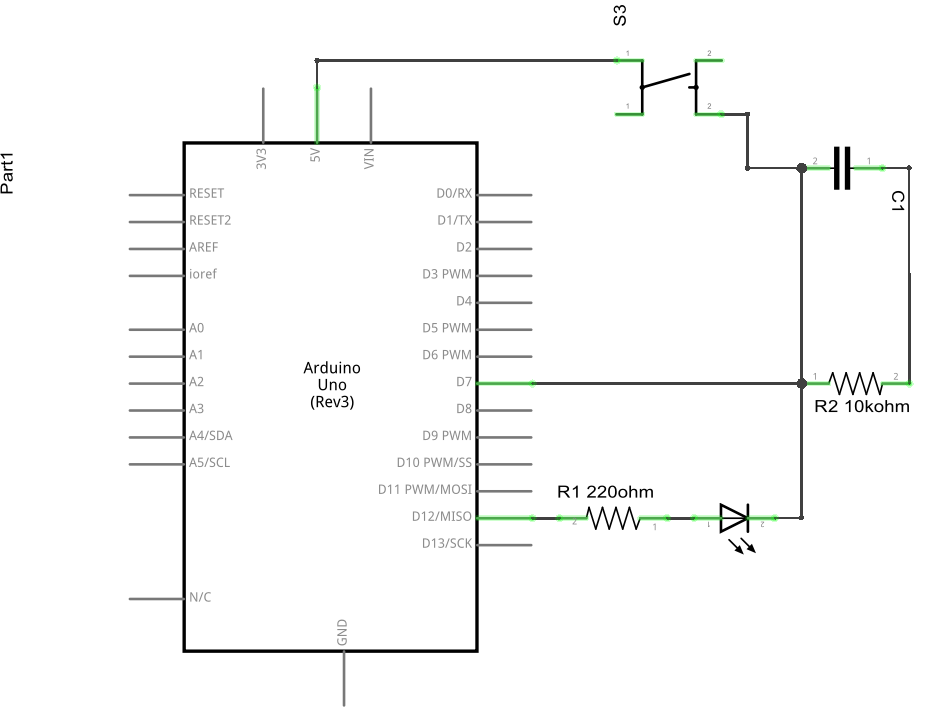
digitalWrite(LED, LOW); // turn off the LED

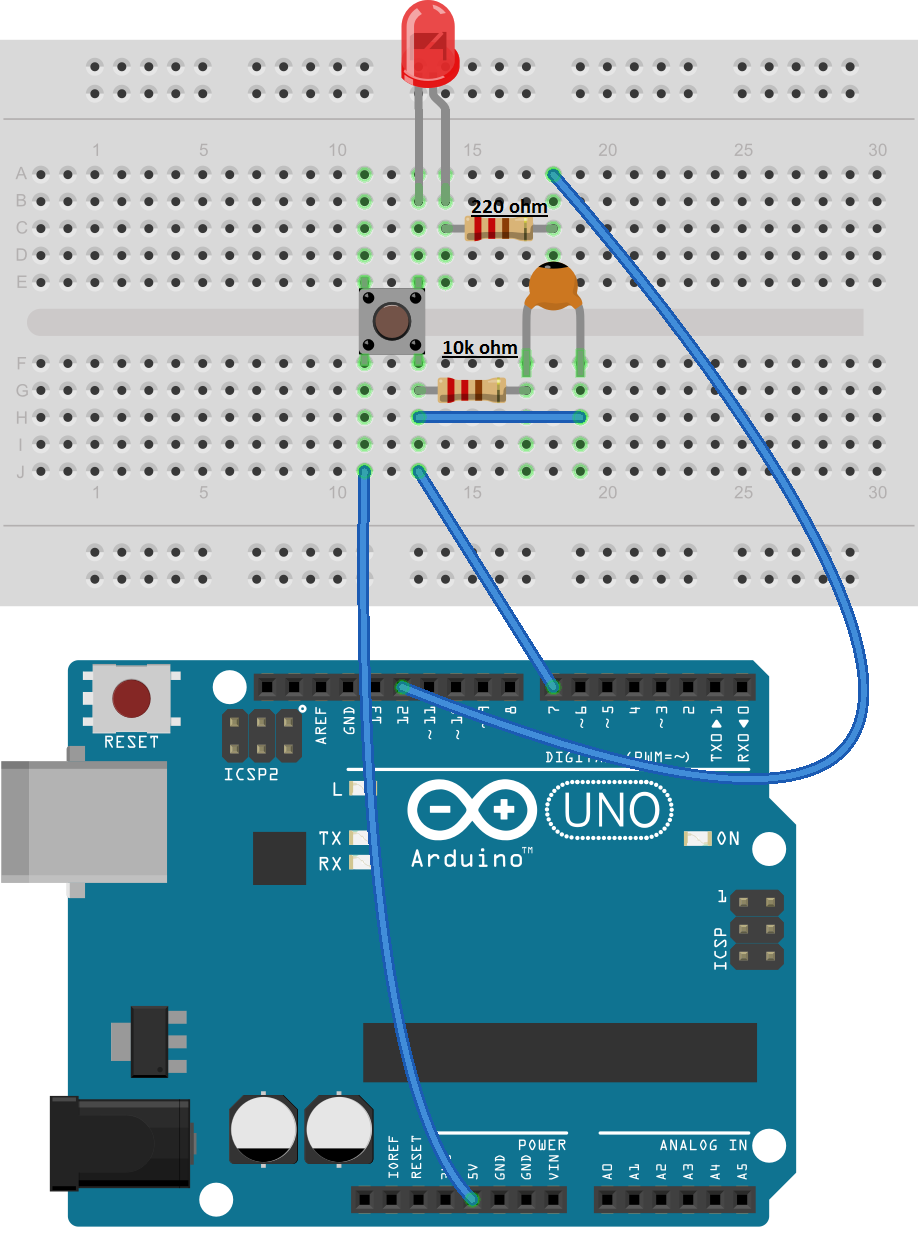
}

}

* ***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.



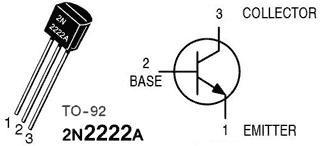


* ***Remember:*** Read the switch Bouncing article to understand schematic and the project completely.
* **Project 6 : Digital Switch With Transistor**
* ***What we do:***

We will create a digital button using the switch property of a transistor that will turns on and off a LED with 5 seconds delay.

Almost everyone has heard of a transistor, but don’t really know how it works. A transistor can turn on or off the flow of a much larger current than the Arduino Uno can handle. We can, however, safely control a transistor using an Arduino digital output pin

* Control a transistor through Arduino
* The transistor in turn will control a much high load which Arduino isn’t capable of



Similar to the LED, the transistor’s pins have a unique function and needs to be connected in the proper orientation. With the flat front of the transistor facing you, the pins on the 2N2222A are called (from left to right) emitter, base, and collector. (Note that this pin order, or pinout, is for the 2N2222A transistor; other transistors may be oriented differently.) When a small current is applied to the base, such as from an Arduino digital I/O pin, the larger current we want to switch enters through the collector; then it is combined with the small current from the base, and then it flows out via the emitter. When the small control current at the base is turned off, no current can flow through the transistor and thus it acts as a switch.

* ***What we need:***

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

* ***The Code:***

// Project 4 - Demonstrating a Digital Switch with Transistor

#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

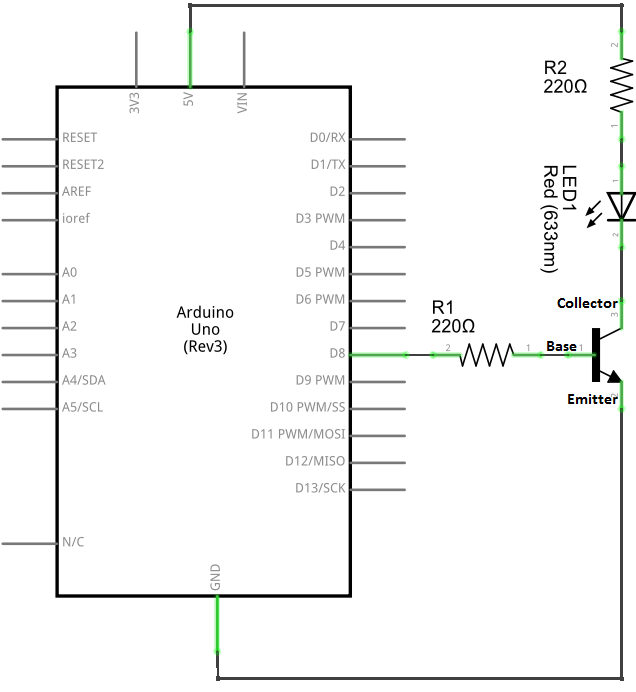
delay(5000); // wait for 5 seconds

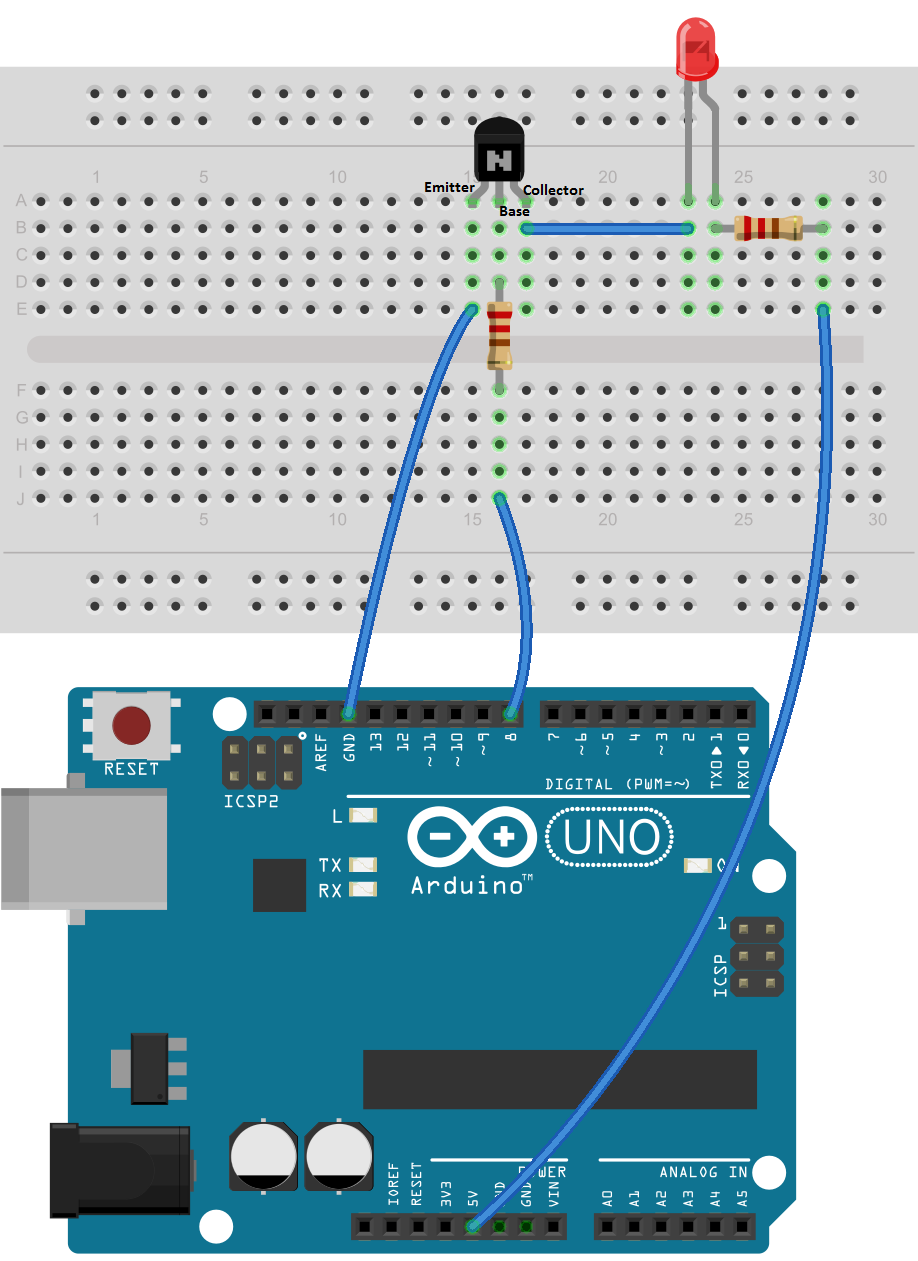
digitalWrite(base\_pin, LOW); // stop the current flow to the transistor base pin

delay(5000); // wait for 5 seconds

}

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





* ***Remember:*** Make sure to double check which pin is base, collector or emitter on a transistor.
* ***Note:*** We will use this configuration in a later project to control the motor with Arduino using the transistor