By: Francisco Xavier Verdugo

05:45

CONTENTS

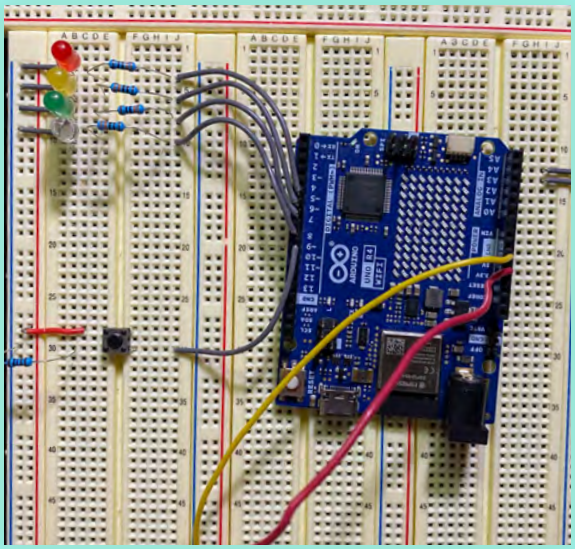
Sections **Pages**

INTRODUCTION TO THE PROJECT

Arduino is a famous Italian company that produces microcontrollers. It is trendy among people who want to dive into the world of electronics because of its price and simplicity of programming.

This instructional set is aimed for teens just starting in the world of programming and electronics. As simple as this project may seem, it introduces fundamental concepts to reader such as good practice when programming and good electronic implementation when working with Arduinos board and electric components.

To be rigurous and promote best practices to the reader, we will first design an online simulation to test out our project. After that, we will assemble this project with real-life components to see the results in a real-world application.



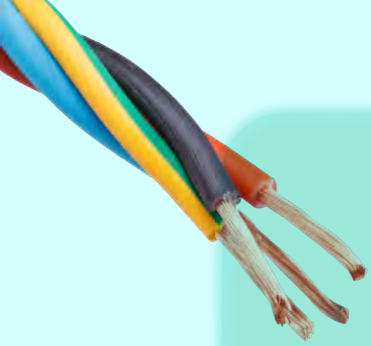
WARNINGS

- Make sure the resistors you have selected are the same value as the one indicated in the materials section. If there is any doubt about the resistors, you can check out this website: **Resistor Color Code Calculator**.

- After connecting any component, double-check it and ensure it is on the correct pin. If it is misplaced, it may cause the circuit not to function at all or may even cause damage to your components.

- Whenever you are using jumper wires, make sure the connection is not loose. If the tip of the wire seems unstable or about to break, replace it immediately.

- Verify our USB cable when connecting to your PC to avoid overpowering your components. If your USB cable is damaged, avoid using it and replace it immediately.

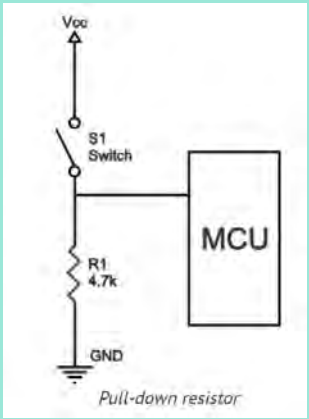


COMMON TERMS USED

Pull-down resistor:
These resistors are connected to the ground, and an appropriate pin is placed on the device.

They ensure a wire is pulled to a low logical state without an input signal.

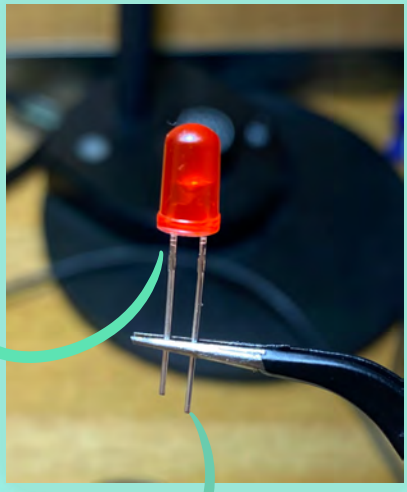
For instance, it will keep an LED turned OFF until it receives an input that changes its state.



Pull-down resistor

LED
An LED (light emitting diode) is a electric component
that will emit light when current passes through it.

Cathode: the negative side of the LED. It is the shorter lead and it connects to ground



Anode: the positive side of the LED and it is the longer lead

NAMING
VARIABLES 101

A significant part of this project is dedicated to programming. Because of this, we will spend this section teaching the reader certain conventions we must follow when naming variables

1. Variable names tend to start with a letter.
2. Digits are allowed, but they cannot be at the beginning of the name.
3. Spaces are not allowed.
4. Special charactes such as "@, !, -, #,..." are not allowed.
5. Most programming languages are case-sensitive, which means variables such as **var, Var, VAR, vAR** are all different.
6. Special words such as **if, else, while, func, sum, int**..., are not allowed.

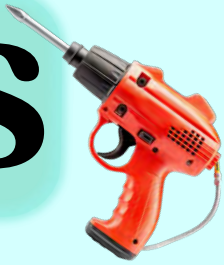
TIPS:

- Avoid making your variable names too long.
- Avoid making your variable names too vague. Be as specific as you can be with your names
- Use camelCase or snake case when naming variables.
  An example of camelCase is: int sumOfTwoNumbers
  An example of snake case: int sum_of_two_numbers;

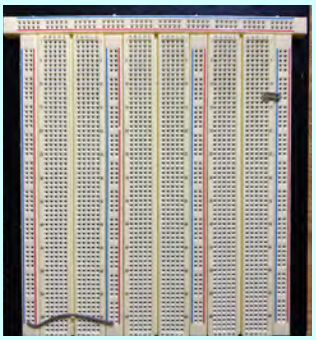
Examples:

DON'T WRITE: ✗	**INSTEAD:** ✓
@sumtwonumbers	sumTwoNumbers
2sumnumbers	sum2numbers
int2numbers	sum_2_numbers
sum numbers	sum2Numbers

MATERIALS



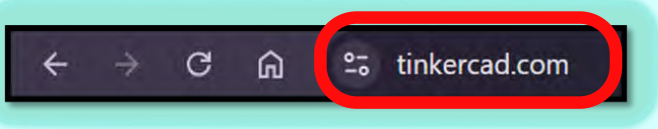
- 1 Laptop or PC
- 4 LEDS: red, green, yellow, white
- 1 USB cable
- Multiple jumper wires
- 4 300Ω resistors
- 1kΩ resistor
- 1 pushbutton
- 1 Arduino Uno R3/R4 board
- 1 Protoboard



CREATING A TINKERCAD ACCOUNT

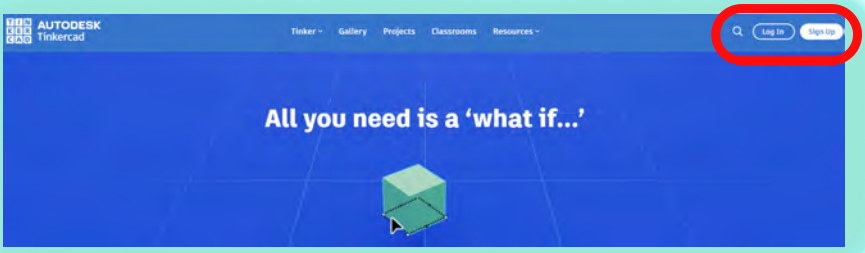
Note: It is recommended that you follow these steps on your PC or Desktop as it is much more comfortable than a table or cellphone

Step 1:
On your browser of preference, go to Tinkercad

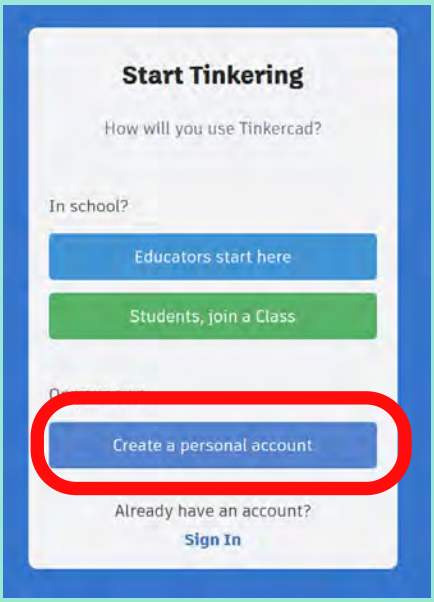


Step 2:
Select the Sign-Up option to create a new account. If you already have an account, you can directly Log In and skip this section.

You should arrive at a new page asking you to create a new account.

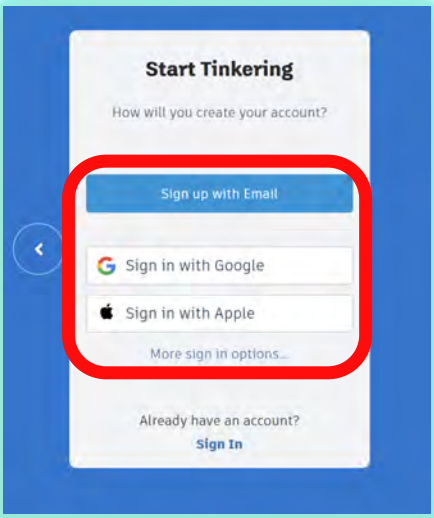


Step 3:
Select Create a personal account.



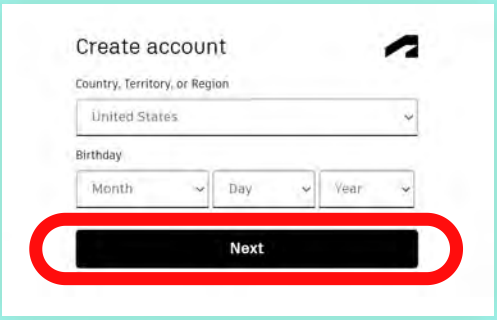
Step 4:
Sign up using your preferred account.

We recommend signing up with a Google account since it can be faster.

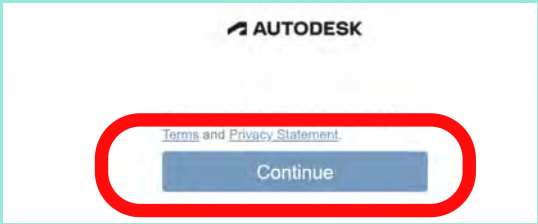


Step 5:
Select your country of residence and your birthday.

After finishing, click Next



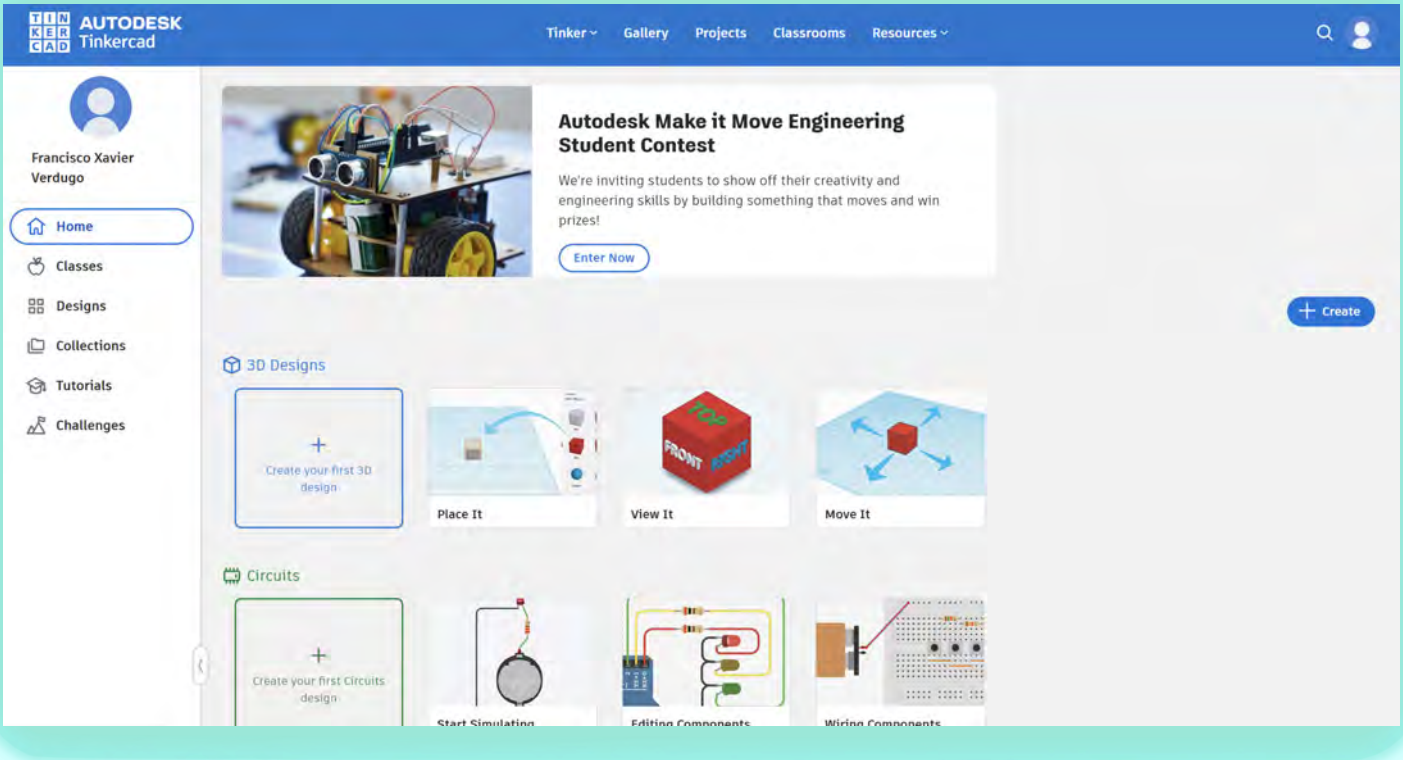
Step 6:
On the screen, click on Continue



Once you have created your account, you should see the Tinkercad home screen.

Here, you will be able to create new projects and access existing projects

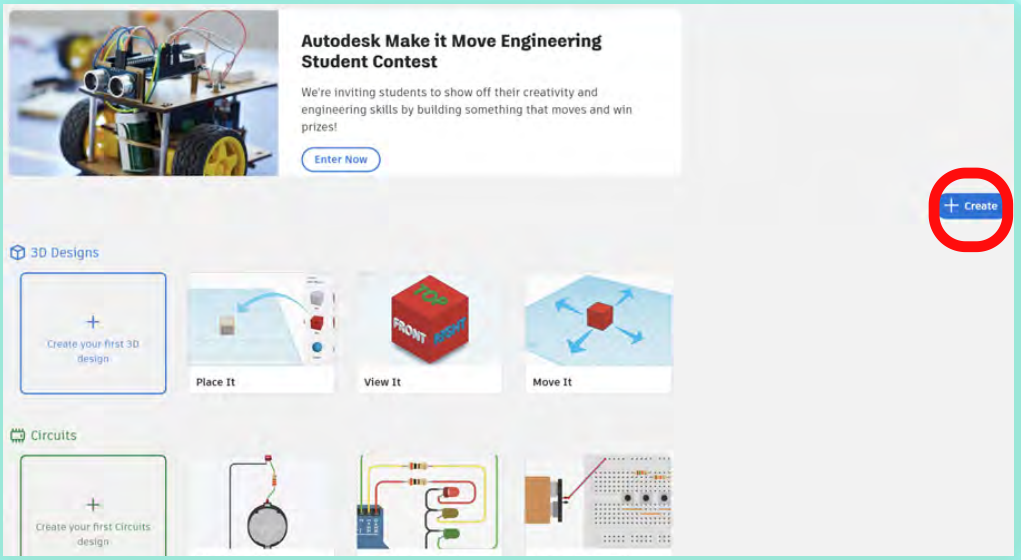
Having successfully created an account, we will create our traffic light design using the Tinkercad Software.



CREATING THE TRAFFIC LIGHT USING TINKERCAD

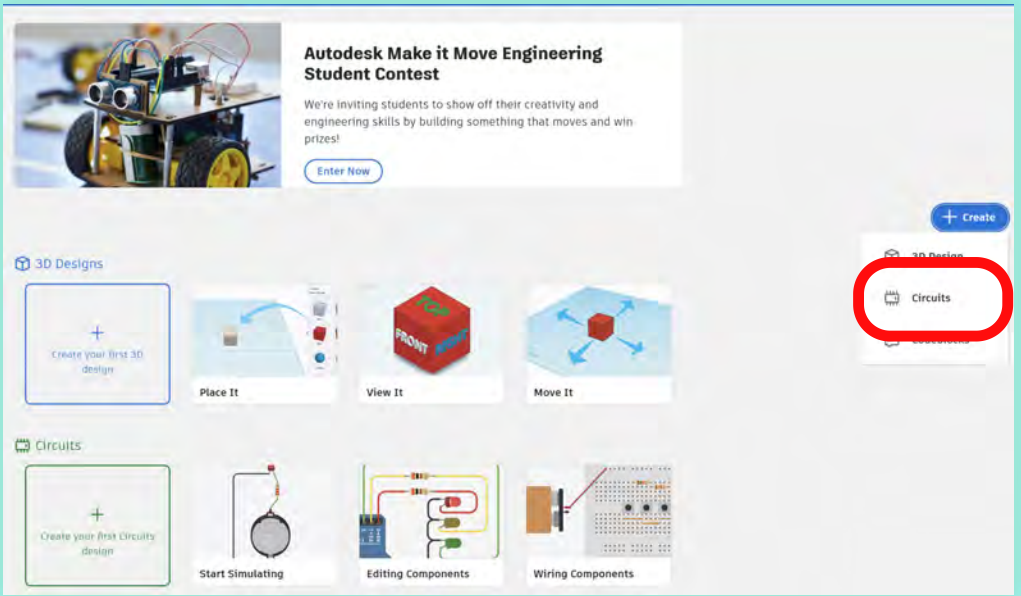
Step 1:
On the Tinkerca Home Screen, click on Create

You should be able to see a drop-down menu

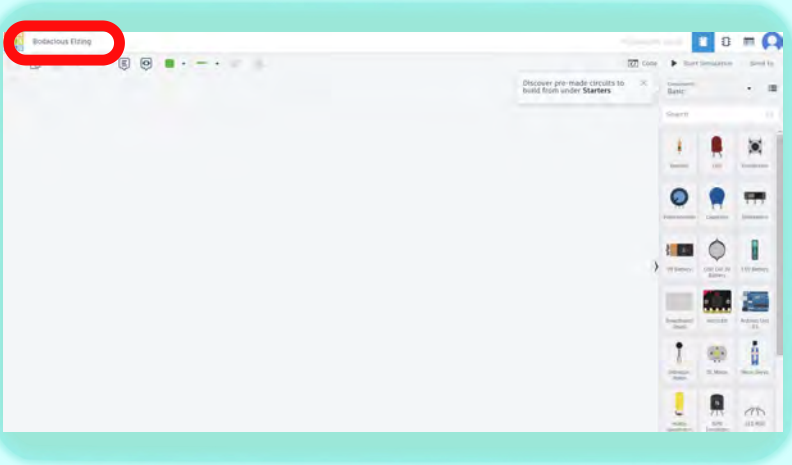


Step 2:
On the drop menu, click on Circuits

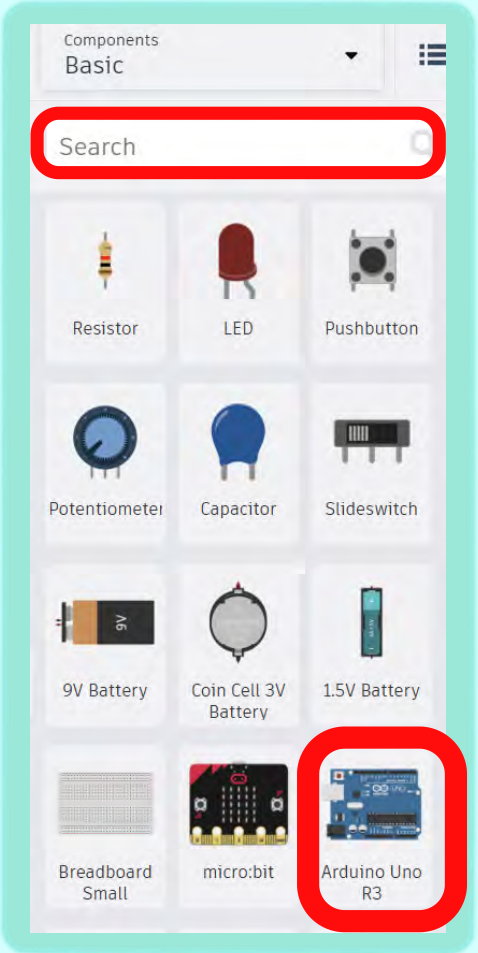
This will create a blank project



Step 3:
On the top-left corner of your screen, double-click on the default name and change it to "Traffic Light"

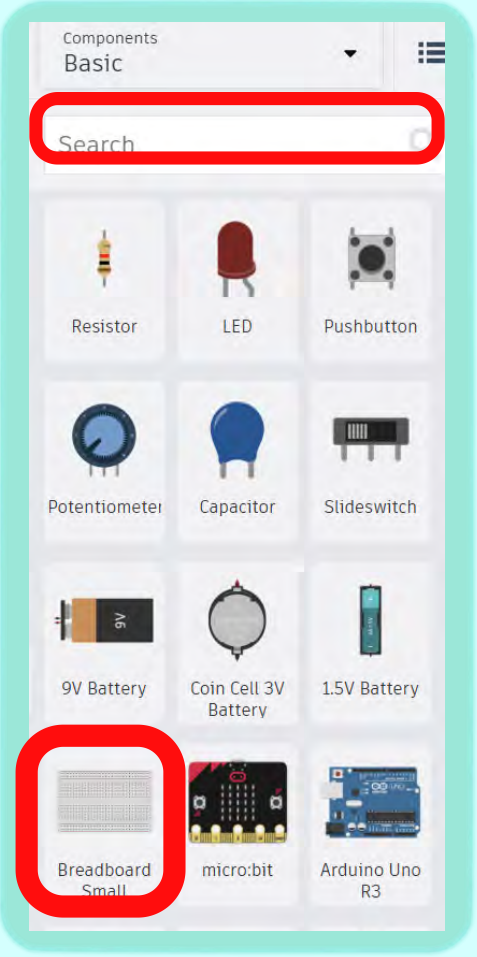


Step 4:
Click on the Arduino Uno R3 icon that is on the menu on the right-hand side of your screen
You should see the board on your screen. If it is not static, click the screen, and the board should stay in
position.

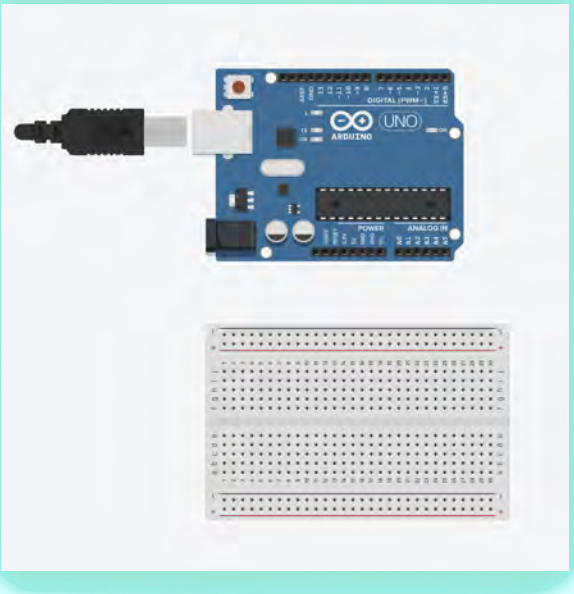


TIP: If you cannot find the component you are looking for, try typing its name on the search bar

Step 5:
Click on the Breadboard small icon that is on the menu on the right-hand side of your screen



Your screen should look like this

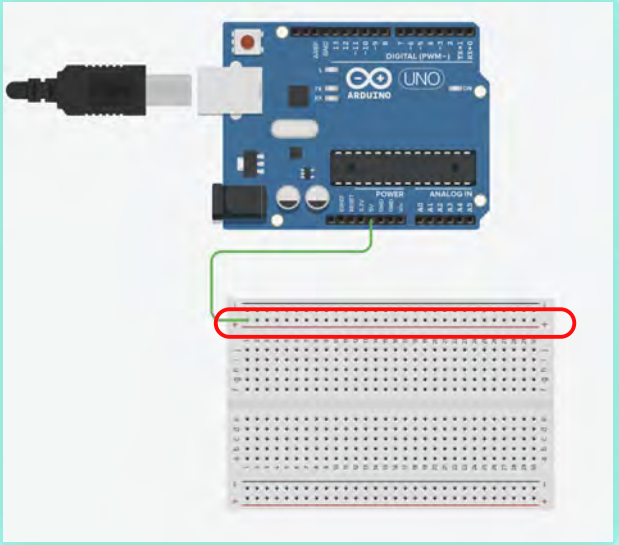


TIP: If the size of the object is too big or too small, you can use “Ctrl” + “+” to zoom in, and “Ctrl” + “-” to zoom out

Step 6:
Locate pin 5V on the Arduino board and connect it to the **red line (+)** on the breadboard.

To connect it, click on **pin 5V** on your board and then click on the indicated line on the breadboard.

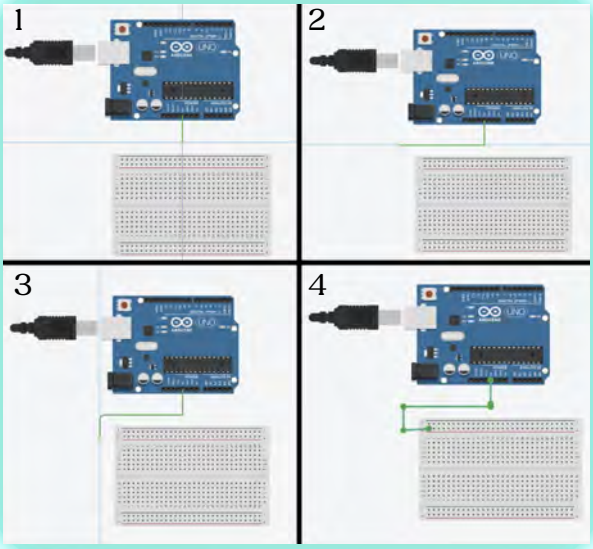
Any of the pins indicated in the picture are fine. This wire will give 5V to the entire line shown in the red box.



TIP: You can control the way you connect your wires.

To bend the wire in different directions, click on the white space on your screen before making the connections

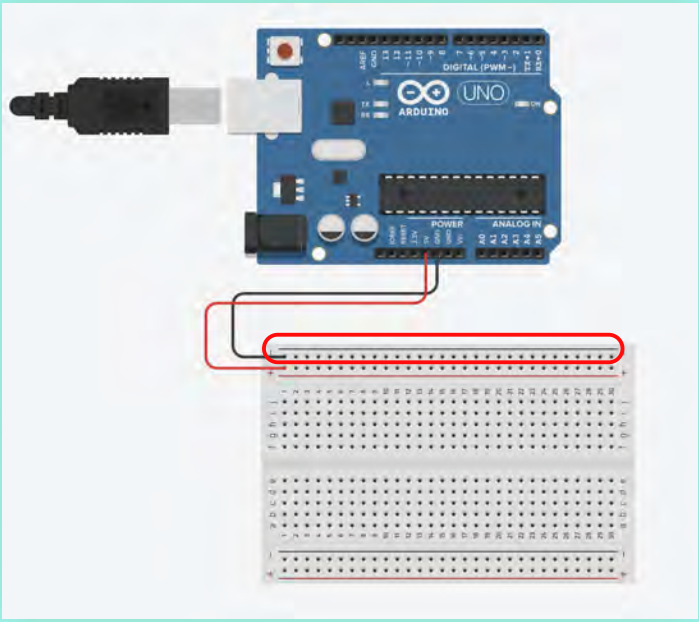
To adjust the angles, you can use the node in the wire to move it as you wish and the blue lines as reference



Step 7:
Locate the GND (ground) pin on the Arduino board and connect it to the breadboard's **black line (-)**.

To connect it, click on the **pin GND** on your board and then click on the indicated line on the breadboard.

Any of the pins indicated in the picture are fine. This wire will provide 0V to the line shown in the red box.

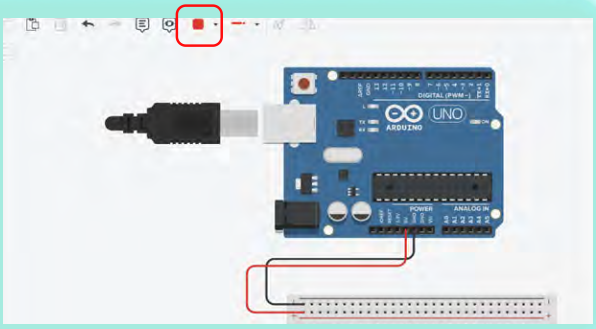


Using different color wires to distinguish the connections whenever possible is best practice.

A Red Wire for voltage connections and a Black Wire for ground connections is considered convention.

TIP: To change the color of the wires, click on the color icon on the menu at the top right.

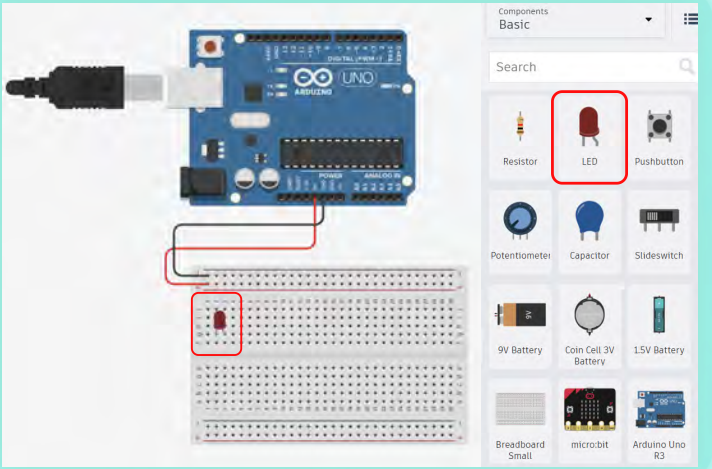
On the drop-down menu, choose the new color you want for your wire



Step 8:
On the menu on the right-hand side of your screen, select the LED icon and place it on the breadboard

Connect the anode (the bend/curved leg) on **row g, column 3**

Connect the cathode on **row g, column 2**

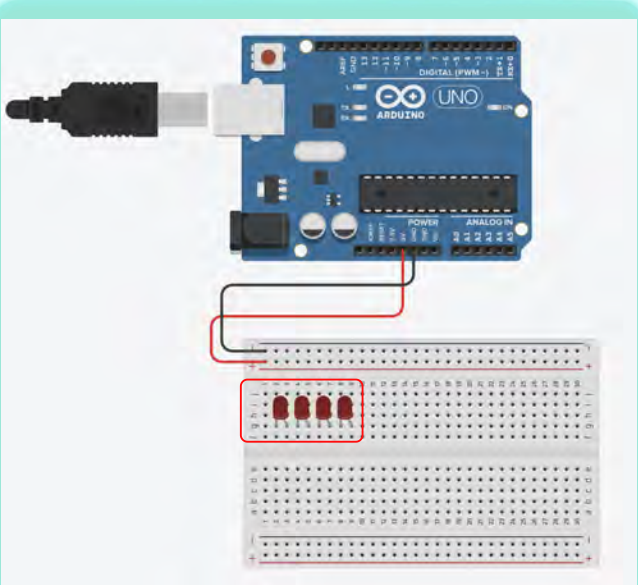


Step 9:
Repeat step 8 for the remaining three LEDs

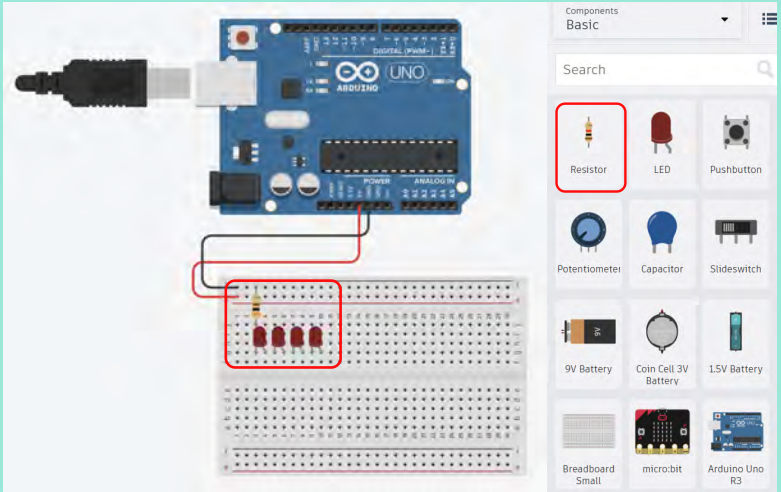
For the 2nd LED, connect the anode on **row g, column 5**, and the cathode on **row g, column 4**

For the 3rd LED, connect the anode on **row g, column 7**, and the cathode on **row g, column 6**

For the 4th LED, connect the anode on **row g, column 9**, and the cathode on **row g, column 8**



Step 9:
On the menu on the right-hand side, click on the resistor icon, place it on **row i, column 3,** and connect it to the ground

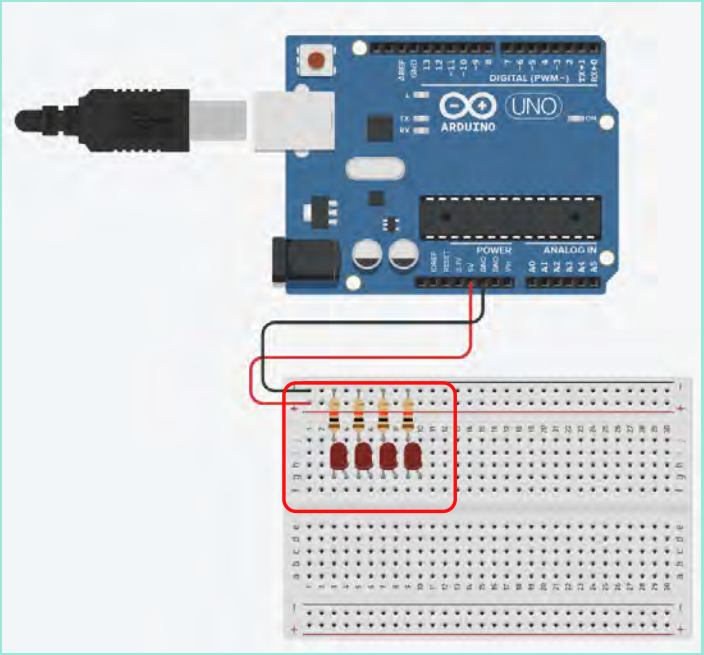


Step 10:
Repeat the previous step with the remaining three resistors

Connect the 2nd resistor from **row i, column 5** to ground

Connect the 3rd resistor from **row i, column 7** to ground

Connect the 4th resistor from **row i, column 9** to ground

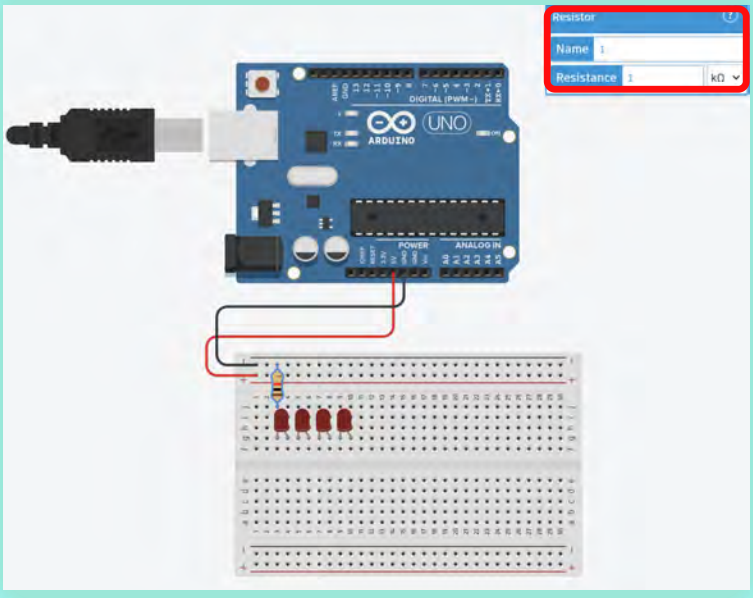


Step 11:
Change the value of the resistors from the default value to 300Ω

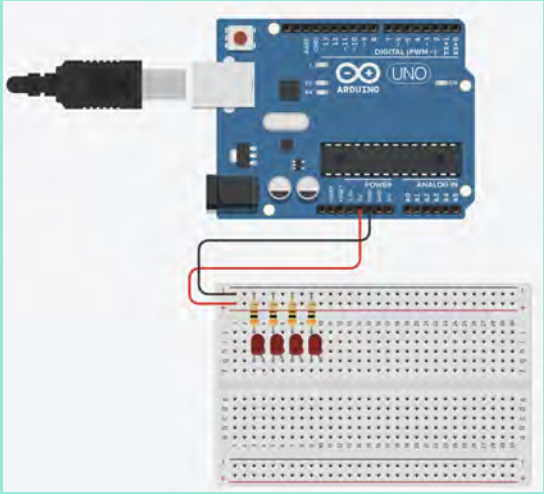
To do so, click on the resistor and change the value to 300

Also, change the unit from kΩ to Ω

Repeat this step for the remaining three resistors

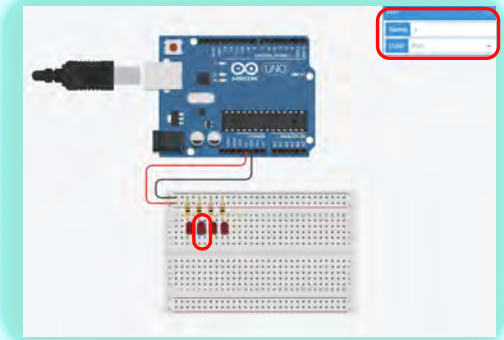


Your project should look like this picture

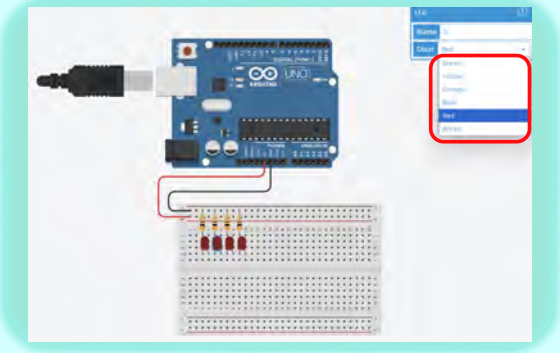


Step 12:
Change the color of the LEDs to match a traffic light
From left to right

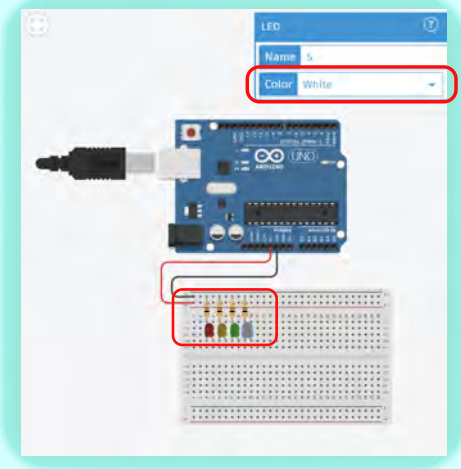
Step 12.1:
Leave the first LED as it is



Step 12.2:
Click on the 2nd LED
You should see a drop-down menu on the right-hand side of your screen



Step 12.3:
On the drop-down menu, change the color to yellow



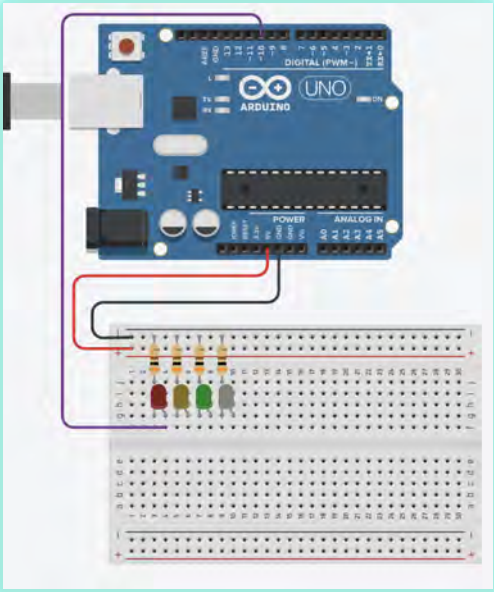
Step 12.4:
Repeat steps 12.2, 12.3, and 12.4 for the 3rd and 4th LED, and change the colors to green and white, respectively

Step 13:
Connect each resistor to a digital pin in the Arduino board

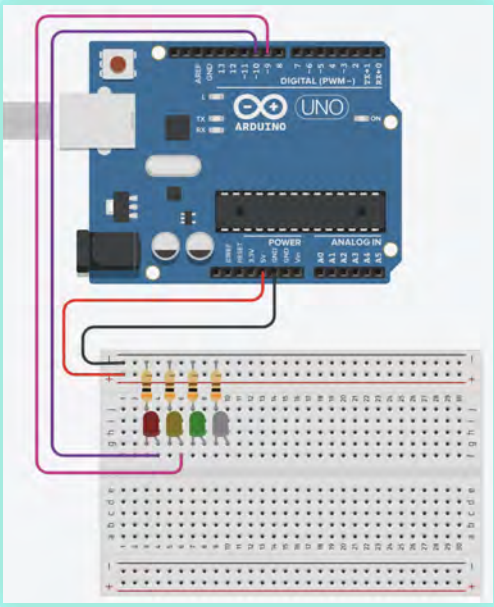
From left to right

Step 13.1:
Connect the 1st LED anode to digital pin 10

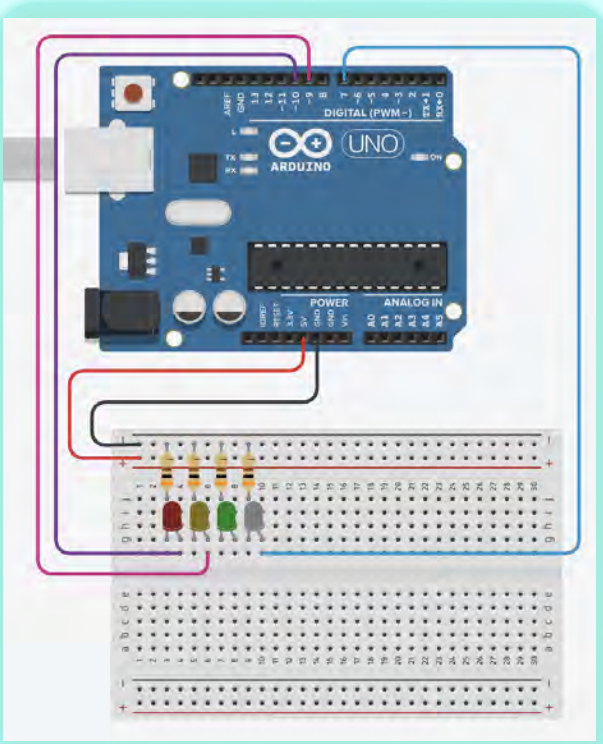
TIP: Check the legs if you have trouble identifying the LED's cathode or anode. If it is longer or bent, it is the anode, otherwise, is the cathode



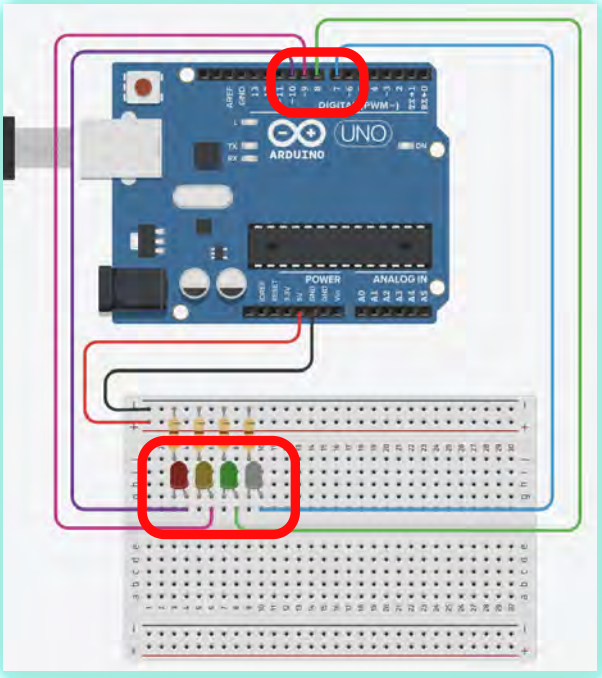
Step 13.2:
Connect the 2nd LED anode to digital pin 9



Step 13.3:
Connect the 3rd LED anode to digital pin 8

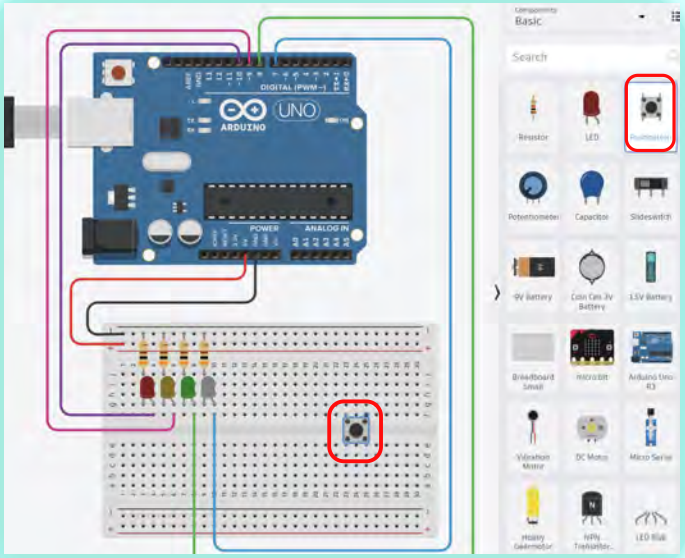


Step 13.4:
Connect the 4th LED anode to digital pin 7

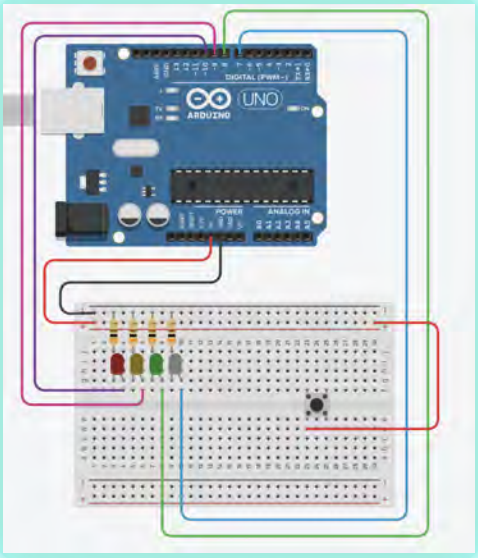


TIP: Try using different color wires whenever possible to tell the connections apart

Step 14:
On the menu on the right-hand side of your screen, click on the pushbutton icon and place it on the protoboard as shown in the picture

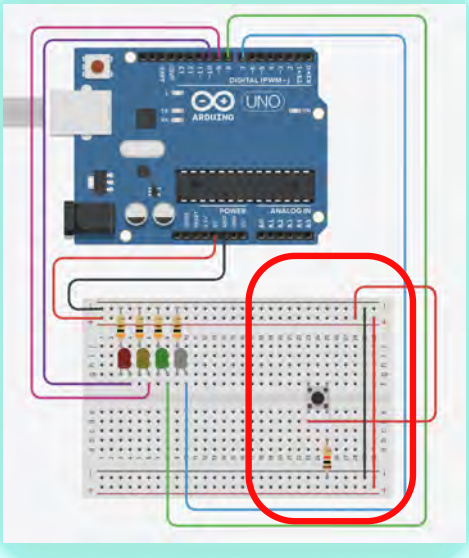


Step 15:
Connect the pushbutton to 5V as shown in the picture

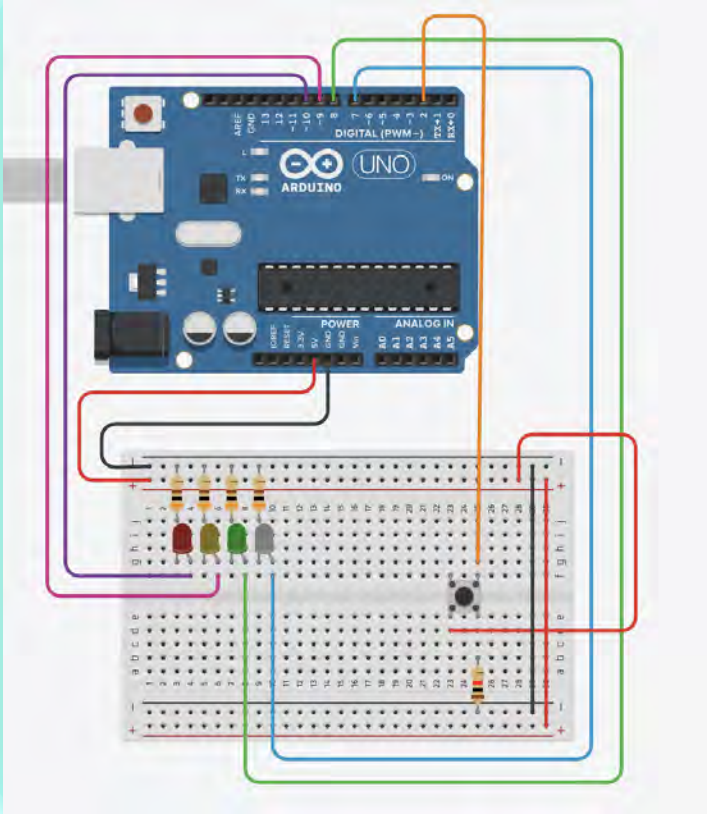


Step 16:
Place a resistor from the push button to the **ground.** Make sure the value of the resistor is **1kΩ**

Use a wire to connect the resistor to the ground



Step 17:
Connect the pushbutton to pin 2 on the Arduino board



This is the last step concerning making physical connections.

The next step is to write the code that will implement the functionality of the traffic light.

WRITING THE SCRIPT

Having completed the last section, we can focus on writing the necessary code for the traffic light.

Before going any further, we will define three functions that are crucial for our implementation

pinMode(pin, mode)

This function takes a **pin number** as the first parameter

The second parameter is mode, where we can define it as **INPUT**, **OUTPUT**, or **INPUT_PULLUP**. We will only use INPUT and OUTPUT

digitalWrite(pin, value)

This function takes a **pin number** as the first parameter

The second parameter is mode, which we can define as **HIGH (ON)** or **LOW (OFF)**

digitalRead(pin)

This function takes a pin number and reads it as **HIGH(ON)** or **LOW(OFF)**

delay(ms)

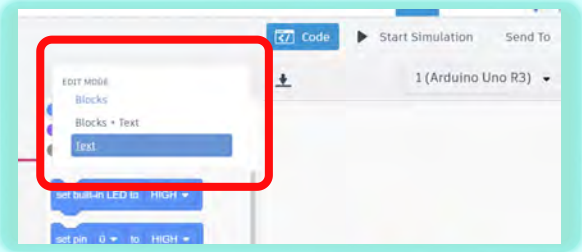
This function takes an integer and will pause the program for a time specified in milliseconds. There are 1000 milliseconds in a second

To make it easier to understand, this section is subdivided into six sections

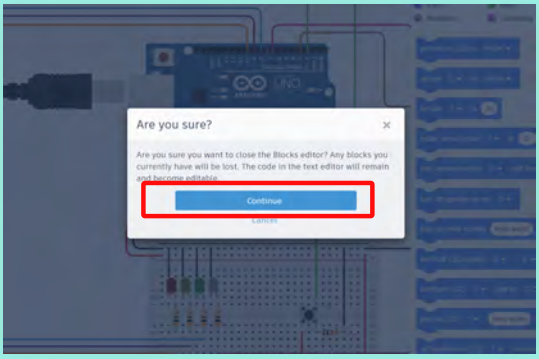
1. CHANGING
FROM BLOCK TO TEXT

Step 1:
On your screen, click on the Code button located at the top-right side of your screen

Step 2:
Click on the Text button and select Text on the drop-down menu.



Step 3:
On the pop-up screen, click on Continue



2. SETTING UP THE CONSTANTS

Step 1:
Set up a const int for each LED of the traffic LED using the pins values of the Arduino board.

```
// Traffic light set up
const int LED_red = 10;
const int LED_yellow = 9;
const int LED_green = 8;
```

For **LED_red**, use pin **10.**
For **LED_yellow**, use pin **9.**
For **LED_green**, use **pin 8.**

Step 2:
Set up a const int for each element of the crossing signal, the white LED, and the pushbutton.

```
//Pedestrian cross set up
const int LED_white = 7;
const int button = 2;
```

For **LED_white**, use pin **7.**
For **button**, use pin **2.**

Step 3:
Set up a boolean variable for the state of the button.

This will help us determine whether the button has been pressed or not.

```
/*
    Setting up the constants
*/

// Traffic light set up
const int LED_red = 10;
const int LED_yellow = 9;
const int LED_green = 8;

//Pedestrian cross set up
const int LED_white = 7;
const int button = 2;

//Boolean variable
bool button_state;
```

Note: All the pin values must match with the values of the Arduino. If you have used any other pin number, change your values accordingly.

TIP: It is best practice to set up constants at the beginning of the program instead of writing each value every time you need it. This is time and space-efficient.

3. SETUP FUNCTION

Step 1:
Erase the current contents of void setup()

```
24  void setup()
25  {
26  |
27  }
```

Step 2:
Inside setup, use the **pinMode(pin, mode)** to set your LEDs and button

Setup all your LEDs as pinMode(LED_color, OUTPUT) Change LED_color for each of the name of the LEDs

```
24  void setup()
25  {
26    //Outputs Traffic Light
27
28    //pinMode(pin: pin number,mode: Input/Output)
29    pinMode(LED_red, OUTPUT);
30    pinMode(LED_yellow, OUTPUT);
31    pinMode(LED_green, OUTPUT);
32    pinMode(LED_white, OUTPUT);
33
34  }
35
```

Step 3:
Right below your LEDs configuration, use **pinMode(button, INPUT)** to set up your pushbutton

```
24  void setup()
25  {
26    //Outputs Traffic Light
27
28    //pinMode(pin: pin number,mode: Input/Output)
29    pinMode(LED_red, OUTPUT);
30    pinMode(LED_yellow, OUTPUT);
31    pinMode(LED_green, OUTPUT);
32    pinMode(LED_white, OUTPUT);
33
34    //Inputs of the traffic light
35    pinMode(button, INPUT);
36  }
37
38  //-------------------------------------------------
```

TIP: It is best practice to add comments to your program. This way, you can keep track of what every piece of code does. Use // to do so. Try to be as brief and precise as possible

4. TRAFFIC LIGHT FUNCTION

Step 1:
Create a new function called trafficLight()

```
36  void trafficLight()
37  {
38
39  }
```

Step 2:
Inside the trafficLight() function, set up the functionality of the

Step 2.1:
Write **digitalWrite(LED_green, HIGH);**
This will turn ON the green LED

Step 2.2:
Write **delay(3000);**
This will keep the LED ON for 3 seconds

```
38    //green light set-up
39    digitalWrite(LED_green, HIGH);
40    delay(4000);
41    digitalWrite(LED_green, LOW);
```

Step 2.3:
Write **digitalWrite(LED_green, LOW);**
This will turn OFF the green LED after the delay

Step 3:
Repeat step 2 for LED_red with a delay of 4000ms, and LED_yellow with a delay of 1000ms

```
36  void trafficLight()
37  {
38    //green light set-up
39    digitalWrite(LED_green, HIGH);
40    delay(4000);
41    digitalWrite(LED_green, LOW);
42
43    //yellow light set-up
44    digitalWrite(LED_yellow, HIGH);
45    delay(1000);
46    digitalWrite(LED_yellow, LOW);
47
48    //red light set-up
49    digitalWrite(LED_red, HIGH);
50    delay(3000);
51    digitalWrite(LED_red, LOW);
52  }
```

5. WALKINNG MODE FUNCTION

Step 1:
Create a new function called walkinMode()

```
void walkingMode()
{


}

```

Step 2:
Copy the code of trafficLight() function and paste it here

```
void walkingMode()
{

  //digitalWrite(parameter1= pin number, parameter2= state HIGH/LOW
  digitalWrite(LED_green, HIGH);//turn on the green light
  delay(4000); //give it 4 seconds to be on
  digitalWrite(LED_green, LOW); //turn it off

  digitalWrite(LED_yellow, HIGH); //turn on the yellow light
  delay(1000);//much faster yellow light, only 1 second
  digitalWrite(LED_yellow, LOW);//turn it off


  digitalWrite(LED_red, HIGH);//turn on the red ligh
  delay(3000);//make this last 3 seconds
  digitalWrite(LED_red, LOW);
}

//--------------------------------------------------------------------
```

Step 3:
Modify the Red light set up

Step 3.1:
Add digitalWrite(LED_white, HIGH) in digitalWrite(LED_red, HIGH) and delay(3000)

Step 3.2:
Leave the delay(3000) as it is

Step 3.3:
Add digitalWrite(LED_white, LOW) in digitalWrite(LED_red, LOW) and delay(3000)

```
void walkingMode()
{

  //digitalWrite(parameter1= pin number, parameter2= state HIGH/LOW
  digitalWrite(LED_green, HIGH);//turn on the green light
  delay(4000); //give it 4 seconds to be on
  digitalWrite(LED_green, LOW); //turn it off

  digitalWrite(LED_yellow, HIGH); //turn on the yellow light
  delay(1000);//much faster yellow light, only 1 second
  digitalWrite(LED_yellow, LOW);//turn it off


  digitalWrite(LED_red, HIGH);//turn on the red ligh
  digitalWrite(LED_white, HIGH);// turn on crossing signal

  delay(3000);//make this last 3 seconds

  //turn off all elements for crossing
  digitalWrite(LED_red, LOW);
  digitalWrite(LED_white, LOW);

}
```

6. LOOP FUNCTION

Step 1:
Erase the content of the function loop()

```
81  //Configuring the loop for the entire program
82
83  void loop()
84  {
85
86  }
```

Step 2:
Set up button_state with digitalRead(button)

```
83  void loop()
84  {
85    //getting the input from the button
86    button_state = digitalRead(button);
87  }
88
```

Step 3:
Create an if statement.

If the button is pressed, then we call the function walkingMode()

Else, we call the function trafficLight()

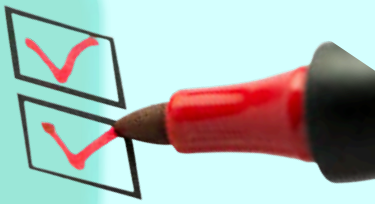
```
83  void loop()
84  {
85    //getting the input from the button
86    button_state = digitalRead(button);
87
88    //setting up delay to double check the state of  the button
89    if(button_state)
90    {
91      delay(15);
92
93      //if we receive an input, call walking function
94      if(button_state){walkingMode();}
95    }
96
97    //if the button is not pressed, call traffic light function
98    else{trafficLight();}
99  }
```

TIP: We use an outside if statement to read the button properly and avoid any issues

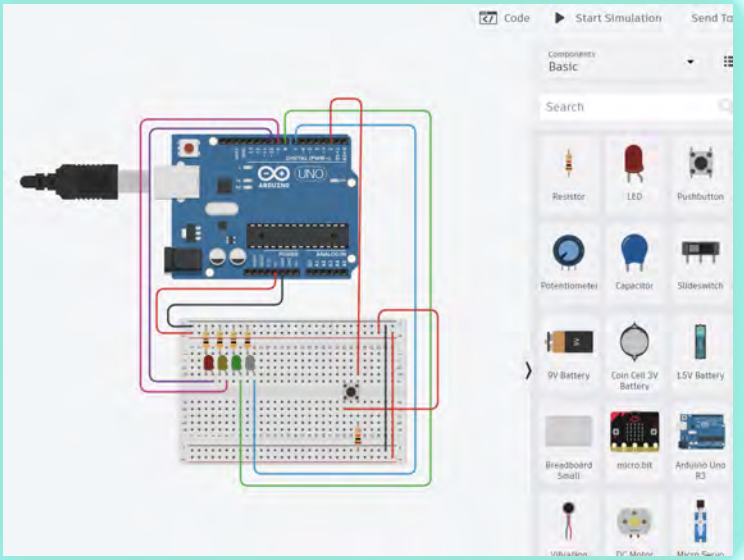
Note: The entire code is shown on the next page. You may copy the code directly if your code is not working correctly.

```
//————————————————————————————————————————————————————————————————————————————
//Setting up the constants

// Traffic light set up
const int LED_red = 10;
const int LED_yellow = 9;
const int LED_green = 8;

//Pedestrian cross set up
const int LED_white = 7;
const int button = 2;

//Boolean variable
bool button_state;

//————————————————————————————————————————————————————————————————————————————
//Configuring setup module

void setup()
{
  //Outputs Traffic Light

  //pinMode(pin: pin number,mode: Input/Output)
  pinMode(LED_red, OUTPUT);
  pinMode(LED_yellow, OUTPUT);
  pinMode(LED_green, OUTPUT);
  pinMode(LED_white, OUTPUT);

  //Inputs of the traffic light
  pinMode(button, INPUT);
}

//————————————————————————————————————————————————————————————————————————————
//Configuring the traffic light mode function

void trafficLight()
{
  //green light set-up
  digitalWrite(LED_green, HIGH);
  delay(4000);
  digitalWrite(LED_green, LOW);

  //yellow light set-up
  digitalWrite(LED_yellow, HIGH);
  delay(1000);
  digitalWrite(LED_yellow, LOW);

  //red light set-up
  digitalWrite(LED_red, HIGH);
  delay(3000);
  digitalWrite(LED_red, LOW);
}
//————————————————————————————————————————————————————————————————————————————
//Configuring the walking mode function

void walkingMode()
{

  //digitalWrite(parameter1= pin number, parameter2= state HIGH/LOW
  digitalWrite(LED_green, HIGH);//turn on the green light
  delay(4000); //give it 4 seconds to be on
  digitalWrite(LED_green, LOW); //turn it off

  digitalWrite(LED_yellow, HIGH); //turn on the yellow light
  delay(1000);//much faster yellow light, only 1 second
  digitalWrite(LED_yellow, LOW);//turn it off


  digitalWrite(LED_red, HIGH);//turn on the red ligh
  digitalWrite(LED_white, HIGH);// turn on crossing signal

  delay(3000);//make this last 3 seconds

  //turn off all elements for crossing
  digitalWrite(LED_red, LOW);
  digitalWrite(LED_white, LOW);

}

//————————————————————————————————————————————————————————————————————————————
//Configuring the loop for the entire program

void loop()
{
  //getting the input from the button
  button_state = digitalRead(button);

  //setting up delay to double check the state of  the button
  if(button_state)
  {
         delay(15);

    //if we receive an input, call walking function
    if(button_state){walkingMode();}
  }

  //if the button is not pressed, call traffic light function
  else{trafficLight();}
}
//————————————————————————————————————————————————————————————————————————————
```

TESTING THE PROJECT

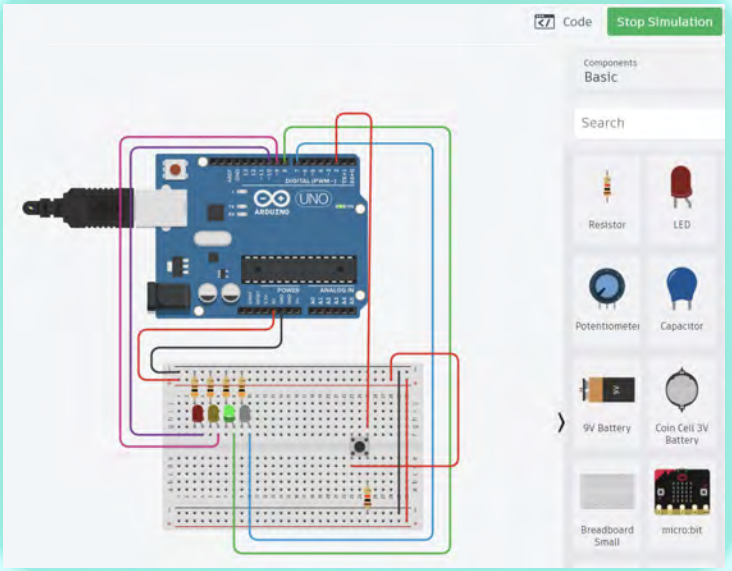


Step 1:
On the top-right corner of your screen, click on Start Simulation



To test your crossing lights, press the button until you see a new cycle starting on the green LED being ON

After that, when the light turns red, you should be able to see the white LED being ON

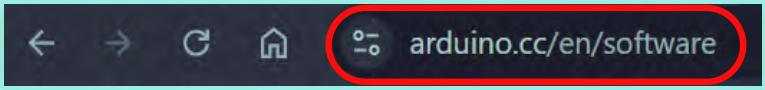


DOWNLOADING ARDUINO IDE

Now that we have created a virtual version of our project, the next step is to take it to real life.

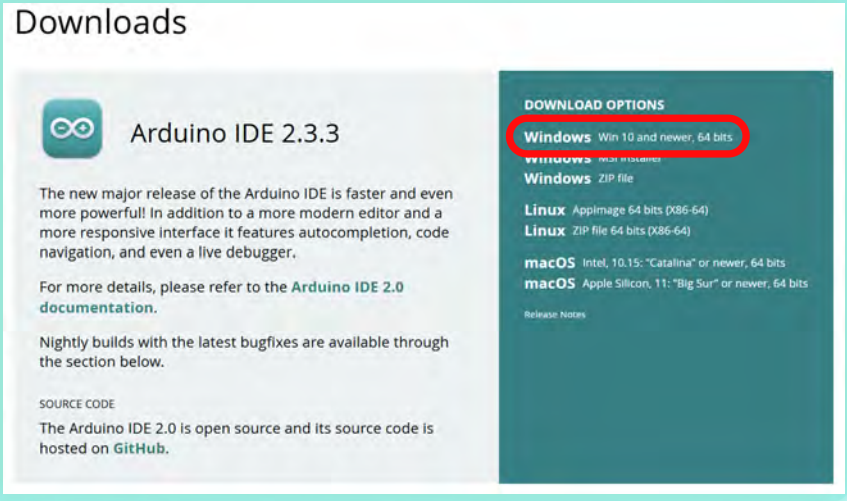
We first want to download and install the software requirements.

Step 1:
Go to Arduino Software website



Step 2:
Download the correct version of the software based on your Operating System

In this instructional set, the Windows version is chosen

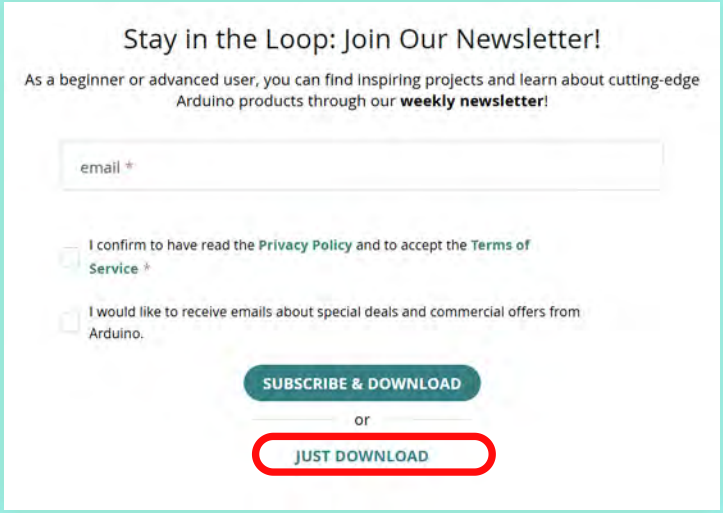


Step 3:
Click on Just Download

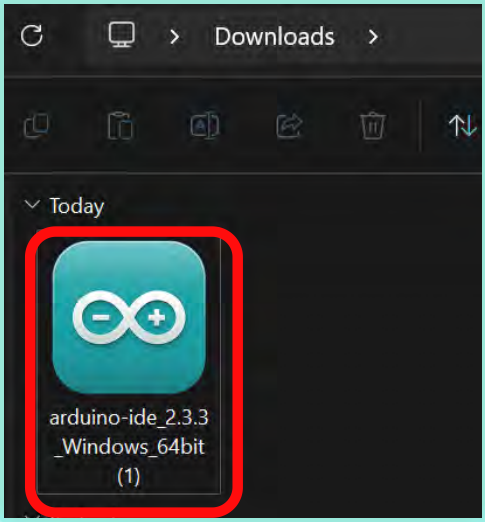
Step 4:
Click on Just Download

Your download should start now. If not, go to the main Arduino page and retry these steps.

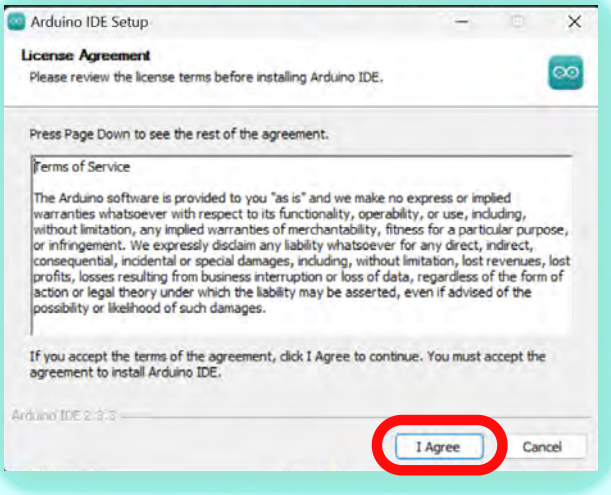
Once it is finished, you should be able to see the file in your download folder.



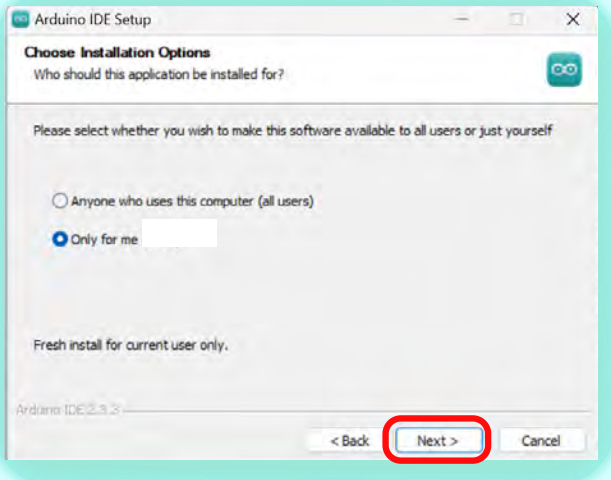
Step 5:
On your download folder, double click on the Arduino file just downloaded



Step 6:
Click on I agree on the pop-up screen



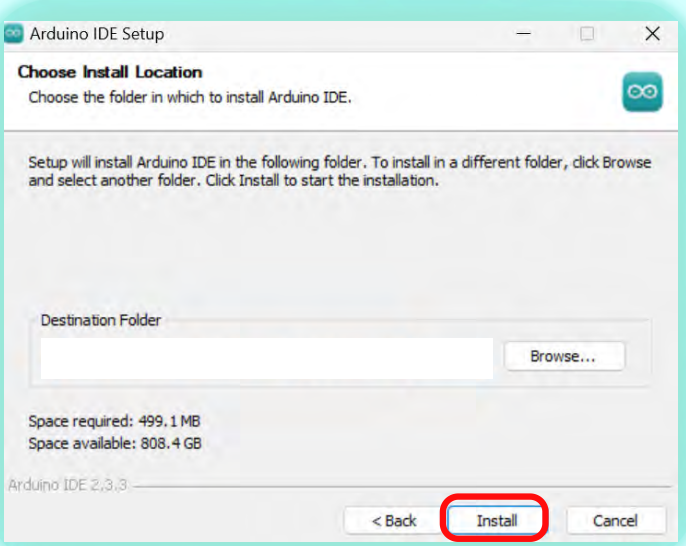
Step 7:
Click on Next. Do not change anything.



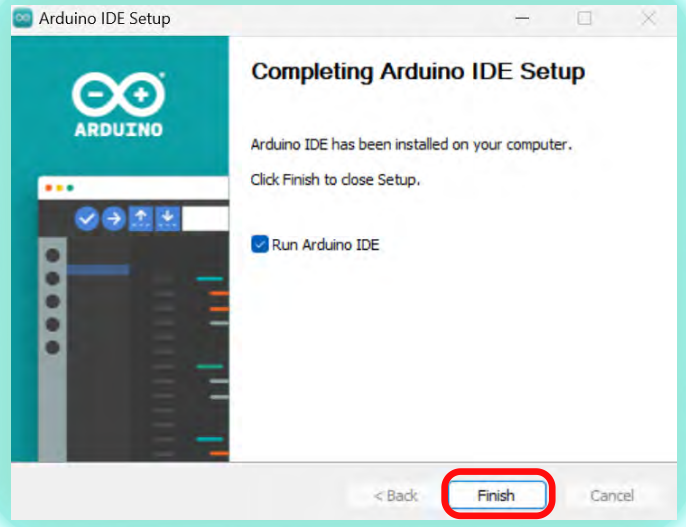
Step 8:
Choose your preferred install location and click Next

You may leave as it is or change it as you see fit

This process may take a couple of minutes



Step 9:
After the installation process is complete, click on Finish



ASSEMBLING THE PROJECT

We will now assemble the project with real components. The steps are the same as those used in the simulation process.

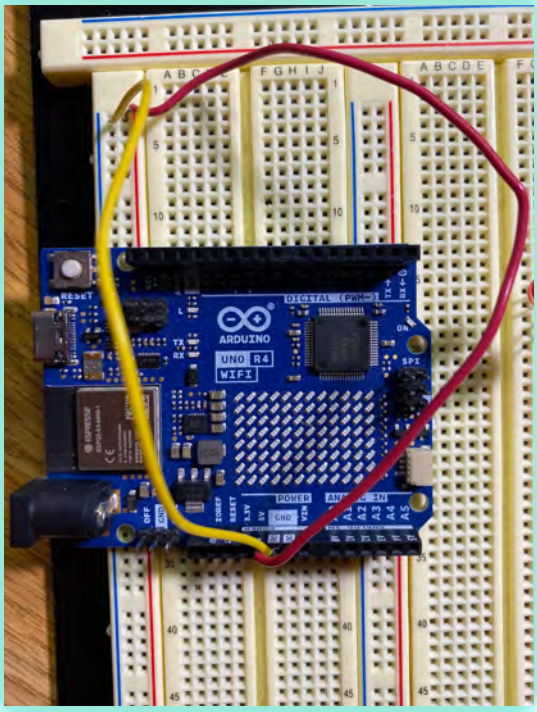
Note: the components shown here may differ from yours. As long as the value is correct, everything should be fine.

Disclaimer: the Arduino board shown here is an Arduino UNO R4. The result will be the same regardless of the use of an Arduino UNO R4 or Arduino UNO R3.

Step 1:
Connect the 5V and GND pins of your Arduino Board to your breadboard

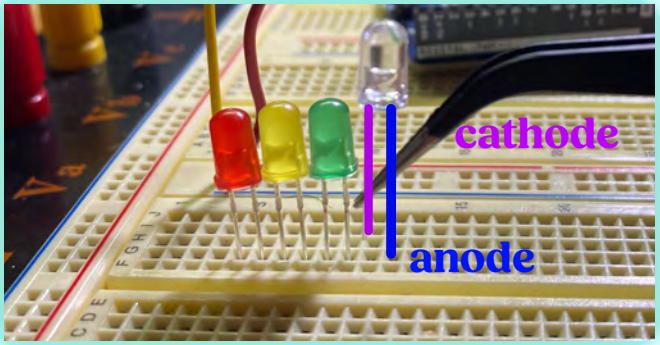
The yellow wire goes from GND pin to the blue line in the breadboard

The red wire goes from 5V pin to the red line in the breadboard



Step 2:
Place your LEDs as shown in the figure following a traffic light pattern

Place the cathode (shorter leg) of the LED to your left, and the anode (longer leg) to your right

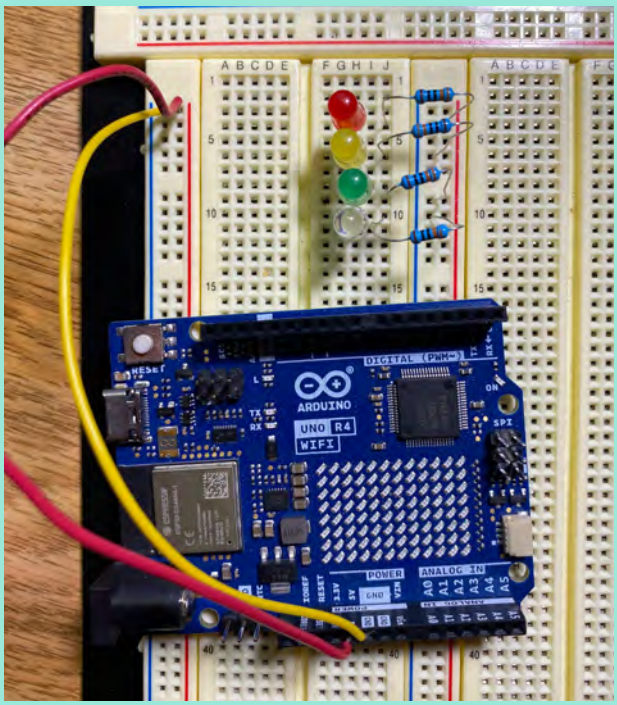


CAUTION

Make sure each LEDs' lead has its own row. Make sure there is no overlapping with the leads. Failure to do so, may result is the LEDs not working properly and may even overload them resulting in permanent damage

Step 3:
Connect each LED's cathode to ground using a 300Ω resistor

In the breadboard, the ground line is the blue line



Step 4:
Connect the LEDs' anode to its respective pin

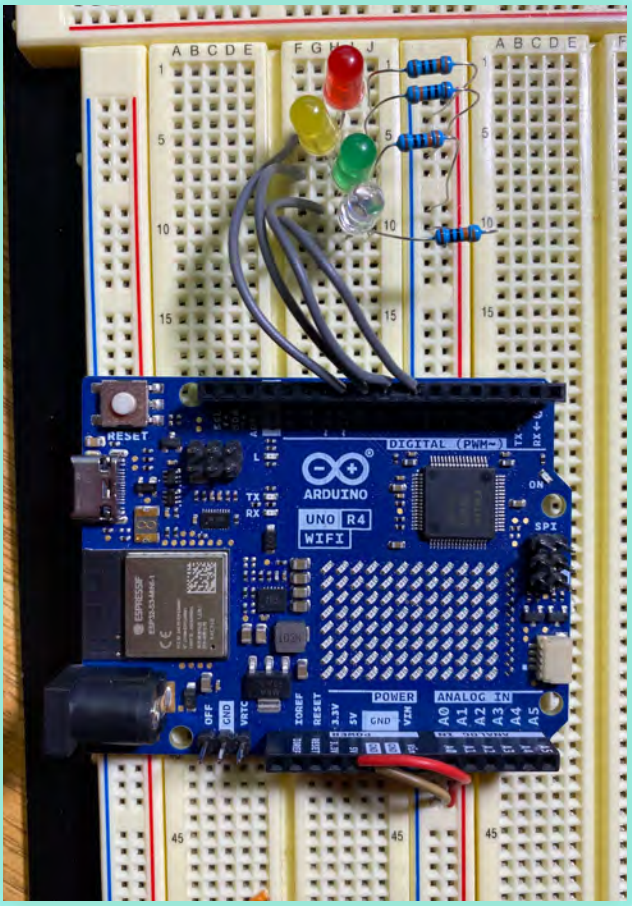
Step 4.1:
Connect the red LED's anode to digital pin 10

Step 4.2:
Connect the yellow LED's anode to digital pin 9

Step 4.3:
Connect the green LED's anode to digital pin 8

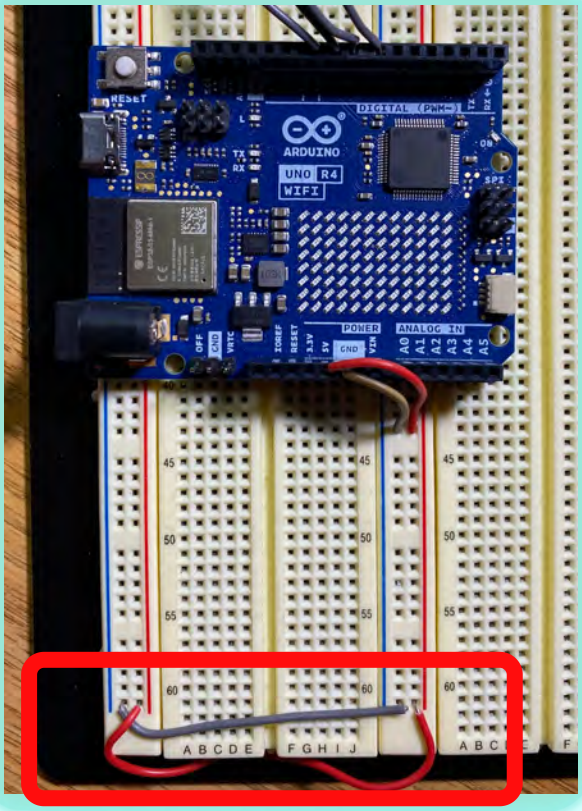
Step 4.4:
Connect the white LED's anode to digital pin 7

TIP: You may move the connections for ground and 5V as you see fit. Keep the 5V going to a red line and GND to a blue line

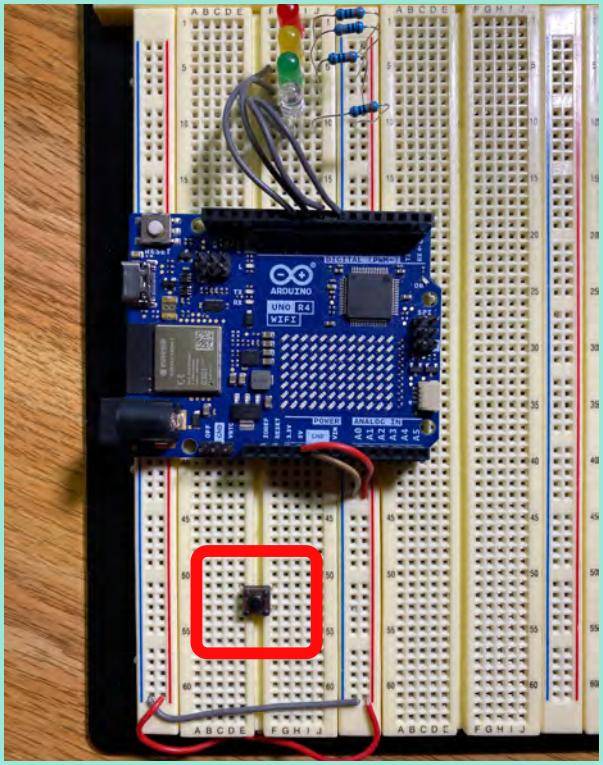


Step 5:
Connect 5V and GND line to the column on your left side

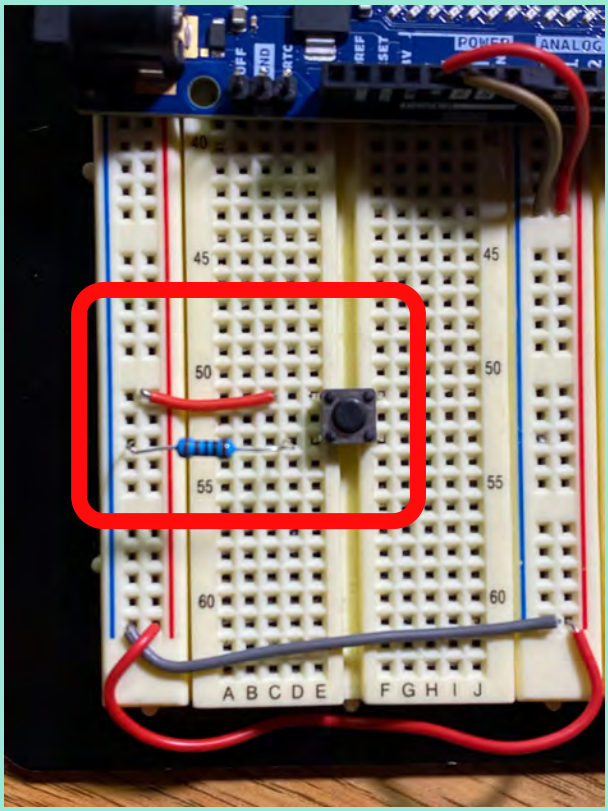
This will provide a new line of power and ground for your design, and it will be more convenient for future connections



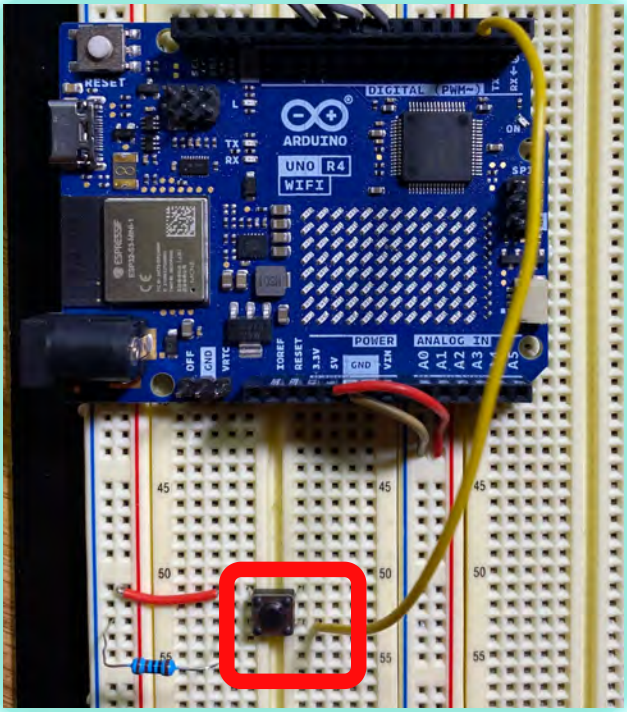
Step 6:
Place your push button on the breadboard



Step 7:
Wire 5V to your push button using jumper wires and connect your button to ground using a 1kΩ resistor as shown in the picture

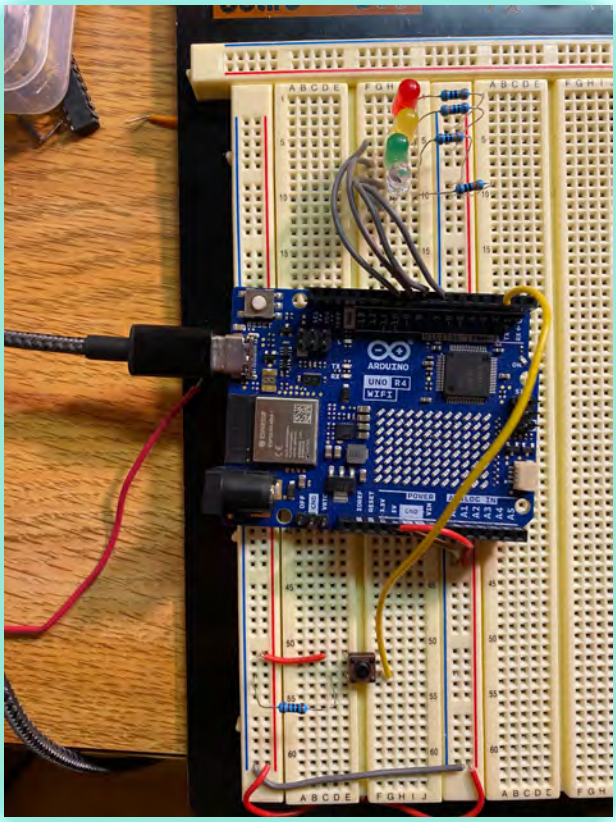


Step 8:
Connect the push button to digital pin 2 on your Arduino board



Step 9:
Connect your Arduino board to your PC using a USB cable

This is the last step to setting up the hardware



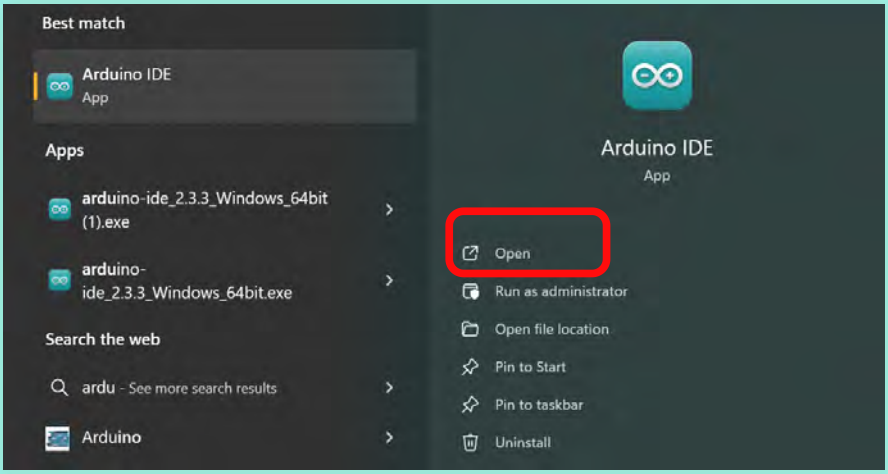
CAUTION

To avoid any damge to your board and electric circuits, make sure your metal components are not in contact with each other. Failure to do so may result in the components overloading.

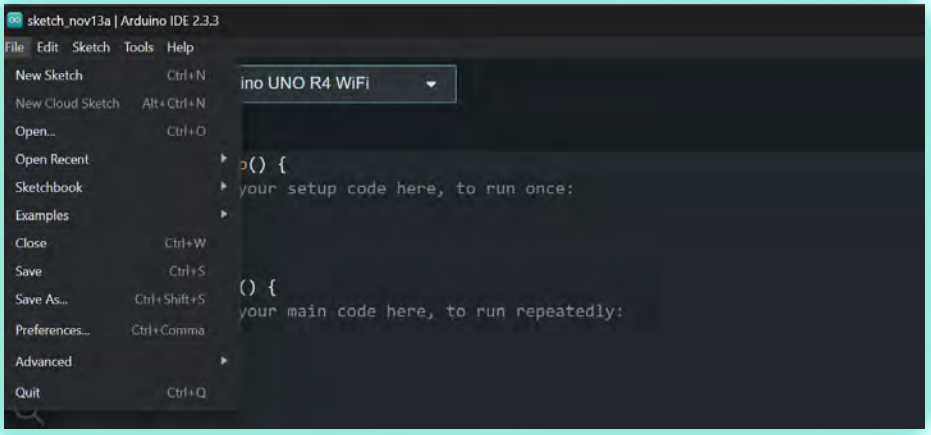
TESTING THE PHYSICAL PROJECT

The final step in this project is to test the assembled project using Arduino software and observe the results in our electronic circuit

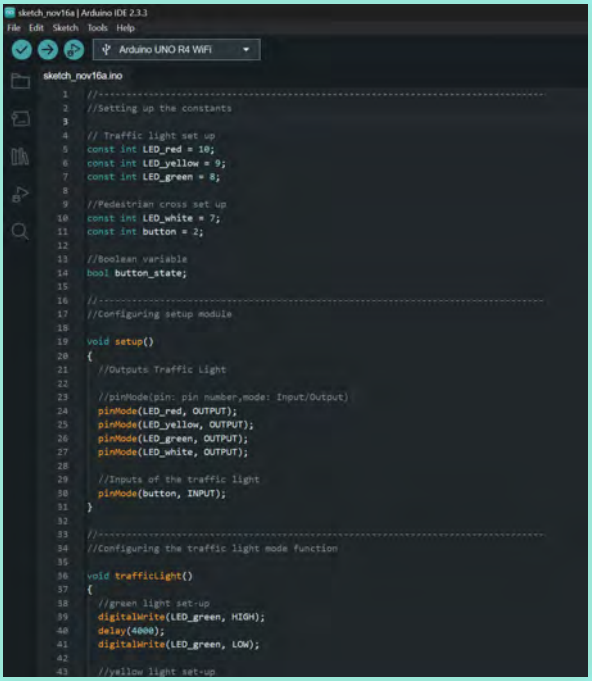
Step 1:
Open the Arduino IDE application



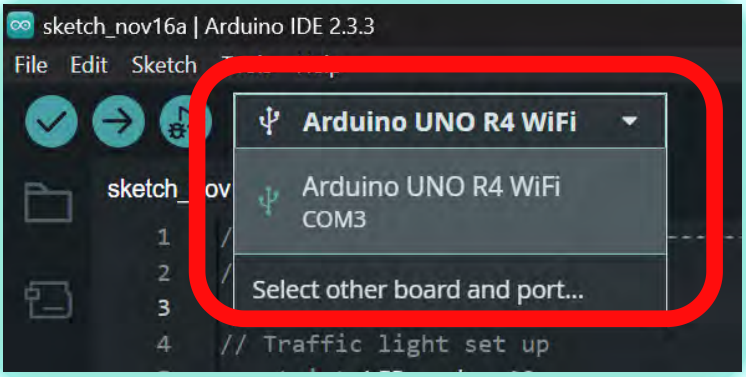
Step 2:
On the left corner of your screen, click on File and New Sketch



Step 3:
Go to your Tinkercad project, copy the code, and paste it into the Arduino IDE new project



Step 4:
Make sure your board is selected. If not, use the drop-down menu to search for your board



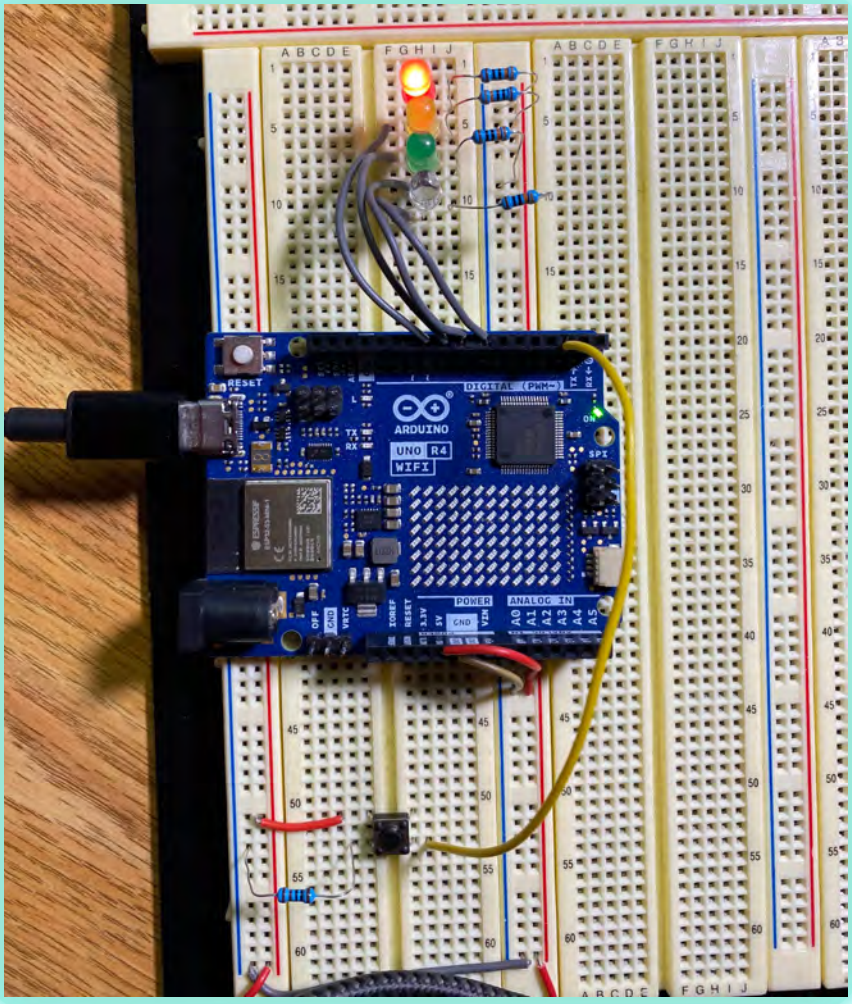
Step 5:
On the left-hand side of the screen, click on Verify

Step 6:
On the left-hand side of the screen, click on Upload

Now, the project should be uploaded to your Arduino board, and the project should be functional

To test the crossing signal, push the button until a new cycle begins

You can identify a new cycle whenever you see the green LED turn ON



FAQ

Creating a Tinkercad account:

- **Do I need to pay for a Tinkercad account?**
  - **A:** No, Tinkercad offers a free version with all the features needed for this project.

- **Do I need to verify my account to start using Tinkercad?**
  - **A:** No, generally, Tinkercad does not require you to verify your account to start using its software. You will receive a confirmation email after signing up.

- **Can I use a phone or tablet?**
  - **A:** You may create your account on these devices. However, to get the full experience, it is recommended that you use a laptop or desktop.

- **Can I use Tinkercad without creating an account?**
  - **A:** No. Tinkercad requires you to create an account mainly to save projects and have access to all its features.

- **Can I sign up with a school or organization account?**
  - **A:** Yes, you may use this option. Check your institution's website for further information.

Setting up the Components in Tinkercad:

- **How can I find a specific component faster?**
  - **A:** You can use the search bar on the panel on your screen's right-hand side.

- **I still need help finding my component; what can I do?**
  - **A:** You can go to the panel on the right side of the screen and change from Basic Components to All Components.

- **My component list is missing; what can I do?**
  - **A:** It is likely that it is just minimized. On the right-hand side of your screen, you should see an arrow. Click on it, and the component list should be back. If this does not work or you cannot find this arrow, try refreshing the page.

- **I closed my tab/browser. Is my project gone?**
  - **A:** No. Tinkercad automatically saves your project. To find it, go back to the Tinkercad official website, log in if necessary, and then you should see it in your projects list.

- **How do I organize my wires?**
  - **A:** Tinkercad allows you to click on the screen, creating multiple nodes you can drag and organize. You can also change the colors of the wires

.

- **Does my layout have to be the same as the one shown in this instruction set?**
  - **A:** No. You will likely find slight differences between your project and the one shown in this instruction set as long as the components are correctly connected. Instead, pay more attention to matching the connections.

Setting up the LEDs, resistors, and breadboard:

- **How do I check if my LEDs are working or not?**
  - **A:** You may write a small piece of code like the one shown in this instruction set. Use pinMode (pinNumber, OUTPUT) on the setup () function. Use digitalWrite (pinNumber, HIGH) on the loop () function. This code will allow you to troubleshoot your LED connections. This will allow you to test the LEDs step by step.

- **My LEDs are still not working; what can I do?**
  - **A:** Check your connections. Remember that the longer/bent leg (anode or positive side) should be connected to a resistor connected to a digital pin on the Arduino board, and the shorter leg (cathode or negative side) directly to the ground. Check the LEDs are not overlapping and make sure each one of them has a resistor.

- **How do you know if overload is a component?**
  - **A:** In specific components, you may see a flame coming out, indicating this component has been overloaded. For LEDs, Tinkercad will warn you that an excess of current exists. In real life, you may see smoke coming out of the component.

- **Why is my LED blinking?**
  - **A:** This may be caused by the lack of a current limiting resistor. Check that you are connecting resistors to each of the LEDs and buttons.

- **Why is my LED dim?**
  - **A:** This may be caused by a wrong value for the resistors. Double-check that the values for the resistors are 300 ohms.

- **My button is not working; what can I do?**
  - **A:** Double-check the connections. Ensure you are connected to the correct ports and the resistor goes from the button to the ground.

Programming the Traffic Light:

- **What programming language is used in Arduinos?**
  - **A:** Arduinos uses a version of C/C++.

- **What can I do when getting an error message when running the code?**
  - **A:** The vast majority of errors are caused by missing semicolons or missing closing brackets. Carefully check the lines in your code. Whenever an error occurs, the compiler will tell you which line to go to and make sure everything is correct and the same as shown in this instruction set.

- **Why do we write 4000 instead of 4 when setting up a 4-second delay?**
  - **A:** The program reads the value in milliseconds; thus, 4000 milliseconds is 4 seconds. You can do the same calculations with any other value.

- **Why do we need the delay in the if statement?**
  - **A:** This will prevent the button from accidentally triggering multiple times and check if it is being pressed for more extended periods.

- **What programming language is used in Arduinos?**
  - **A:** Arduinos uses a version of C/C++.

- **My code is not working and still generates erros. What can I do?**
  - **A:** If your code is not working correctly, and you still have doubts about the project, you can consult this Github repository: Traffic Light Arduino

REFERENCES:

Athought this project was fully developed by the author of this instructional set, external sources were used to be as accurate as possible when describing the steps shown here.

Additionaly, a GitHub repository has been created where you can find the source code for this project alongside images of the Tinkercad implementation.

Lastly, in case you wish you get a video tutorial, an external source from another author has been included as well.

Pull-up and Pull-down resistors theory:
Van Oorschot, P. F., & Pustjens, J. W. (2022). EE Power Resistor Guide. Boise, ID; EETECH MEDIA.

Pull-up and Pull-down resistors Arduino:
Instructables. (2022, April 26). Understanding the Pull-up/Pull-down resistors with Arduino. Instructables.https://www.instructables.com/Understanding-the-Pull-up-Resistor-With-Arduino/

Arduino Documentation:
Language reference | Arduino documentation. (n.d.). https://docs.arduino.cc/language-reference/#functions

How to wire a button:
How to wire and program a button | Arduino Documentation. (n.d.). https://docs.arduino.cc/built-in-examples/digital/Button/

External Tutorial with a video:
Hackster.io. (2018, March 3). Traffic lights and push button. https://www.hackster.io/BitteristSquash/traffic-lights-and-push-button-9ac870

GitHub Repository: https://github.com/fxvr2307/Traffic-Light-using-Arduino