OLD DOMINION
UNIVERSITY

Virtual Activities with TinkerCAD Software using Arduino

Dr. Murat Kuzlu
Jorge Ramirez III
Department of Engineering Technology

Outline

- Arduino Introduction
- Component Overview
- Virtual Activities (1-5)

https://medium.com/@otavioguastamacchia/creating-a-simple-iot-case-8102f22908a7

Why Arduino ?

- It is an **open-source project**, software/hardware is extremely **accessible** and very flexible to be customized and extended
- It is **flexible**, offers a variety of digital and analog inputs, *SPI* and serial interface and digital and *PWM* outputs
- It is **easy to use**, connects to computer via USB and communicates using standard serial protocol, runs in standalone mode and as interface connected to PC/Macintosh computers
- It is **inexpensive**, and comes with free authoring software
- Arduino is backed up by a growing **online community**, lots of source code is already available and we can share and post our examples for others to use, too.

Autodesk TinkerCAD

- Free online collection of software tools that help people all over the world think, create and make. A leader in 3D design, engineering and entertainment software.
- Arduino Simulation environment

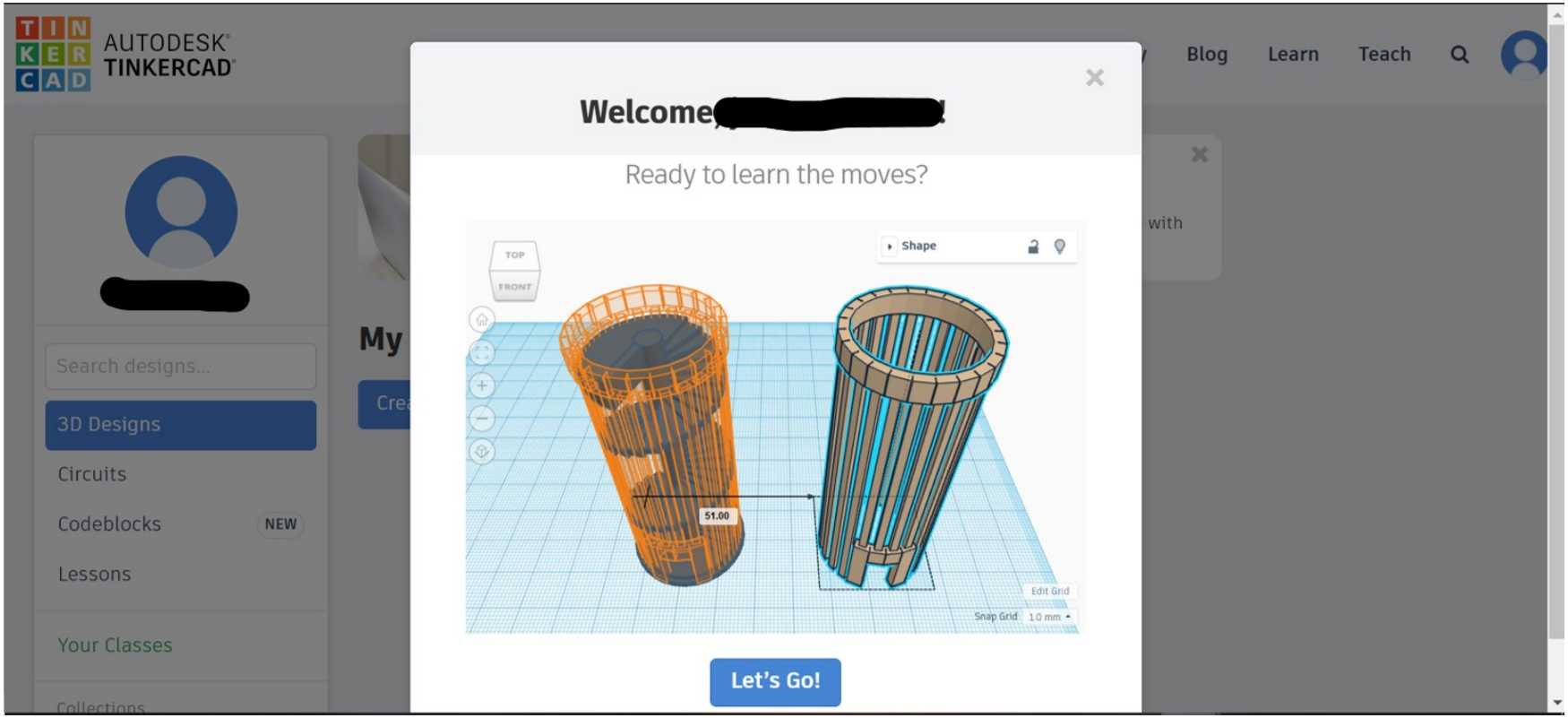
- https://all3dp.com/2/tinkercad-arduino-how-to-design-simulate-circuits/

Creating an account with TinkerCAD

1. Go to https://www.tinkercad.com/
2. Top right corner, click on "Join Now"
3. At the bottom, click on "Create a personal account"
4. Select an email option that suits you
   - You may be ask to fill in extra information ex: Country, birthdate.
5. Enter email address and a password you wish to use
6. Click "Create account"

Creating an account with TinkerCAD

1. You should now see this screen, go ahead and close the welcome window and you'll be at your home profile page.

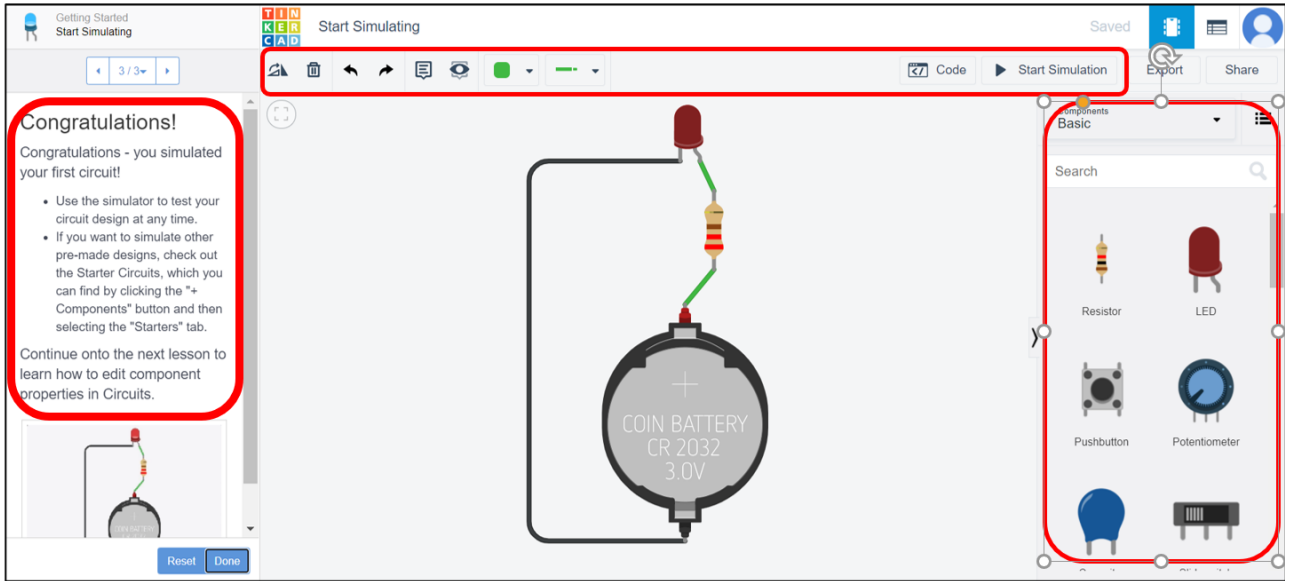


Getting Started with TinkerCAD

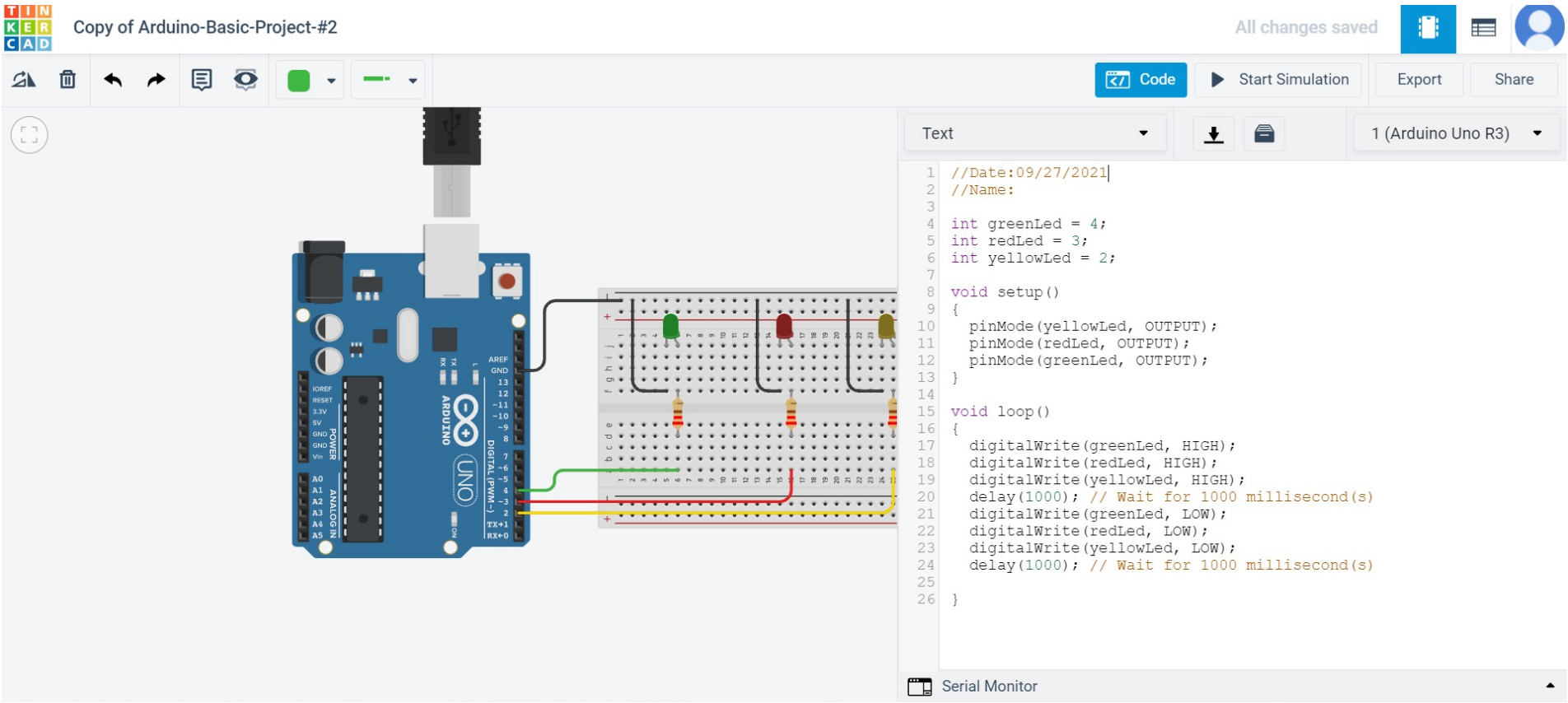
1. Top right corner click on "Learn"
2. On the left, click on the drop down menu "3D"
   - From the drop down menu select Circuits
   - Follow this link directly if needed: https://www.tinkercad.com/learn/circuits/learning

Getting Started with TinkerCAD

- Review four introduction lessons on TinkerCAD
  1. Start Simulating
  2. Editing components
  3. Wiring components
  4. Adding components
- Be sure to read through each step carefully on the left and explore as much as you want to get comfortable!



TinkerCAD Environment



Overview of components

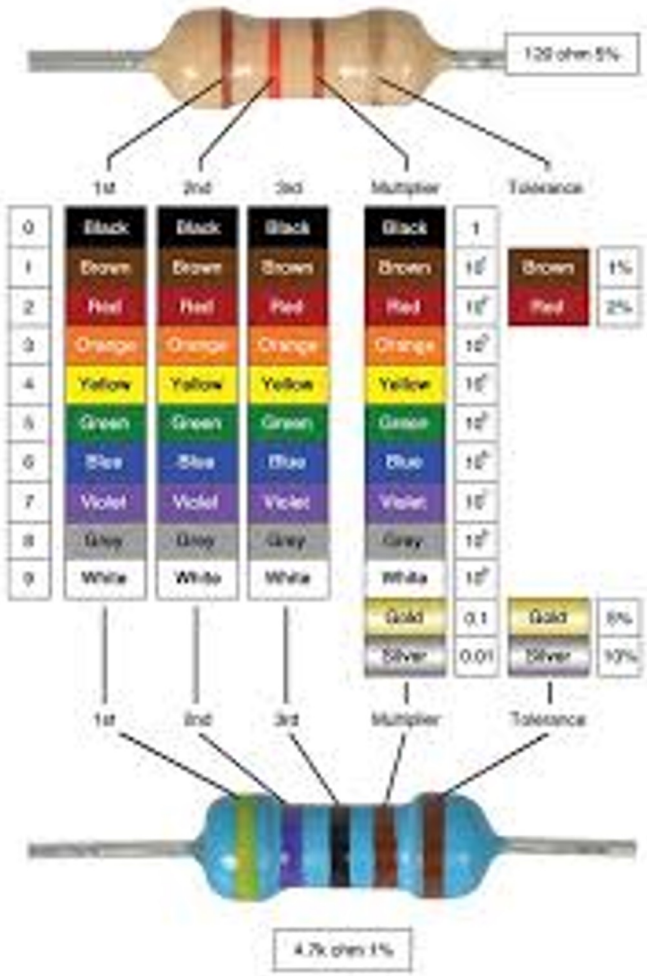
- The resistor
- The LED
- Arduino
- Breadboard

The resistor

Resistors are the **workhorse of electronics**

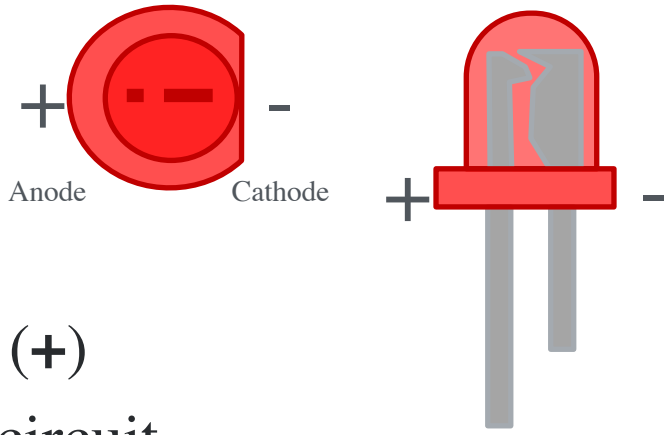
Resistance is **measured in Ω** (Ohm) and adds up in series; a resistors orientation doesn't matter

A resistors Ω value is **color-coded** right on it



The LED

The **LED** (**L**ight **E**mitting **D**iode)
is a simple, digital **actuator**

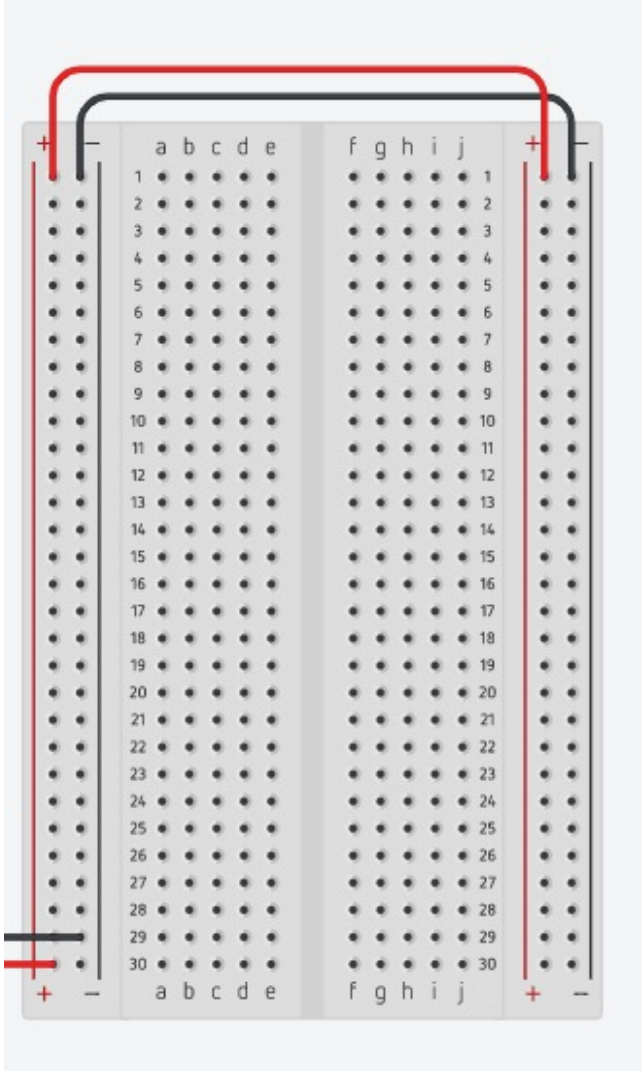


LEDs have a **short leg** (**-**) and a **long leg** (**+**)
and it matters how they are oriented in a circuit

To prevent damage, LEDs are used together with a 1KΩ **resistor** (or anything from 300Ω to 2KΩ)

Breadboard

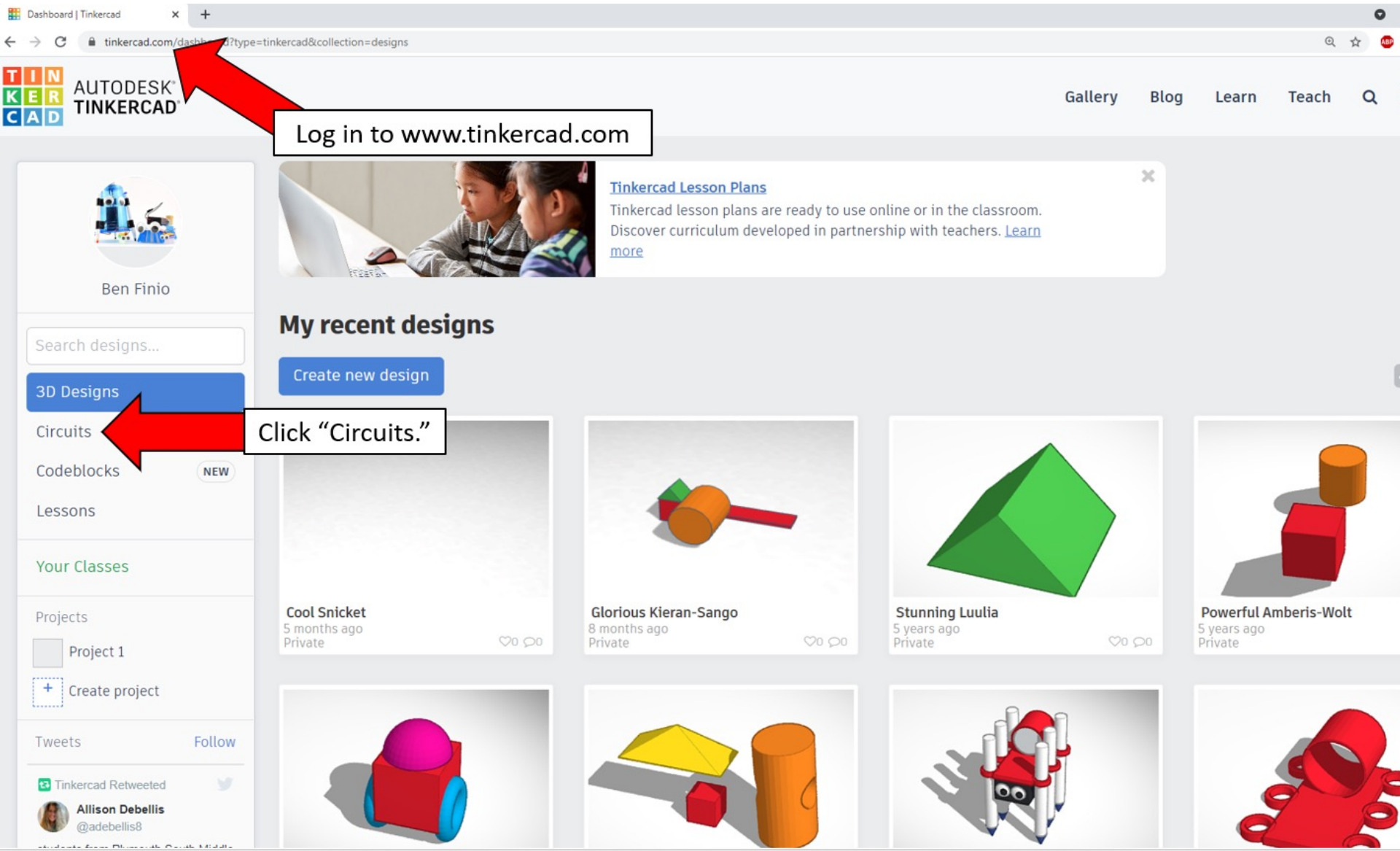
- The power bus
- Horizontal connections
- Vertical connections



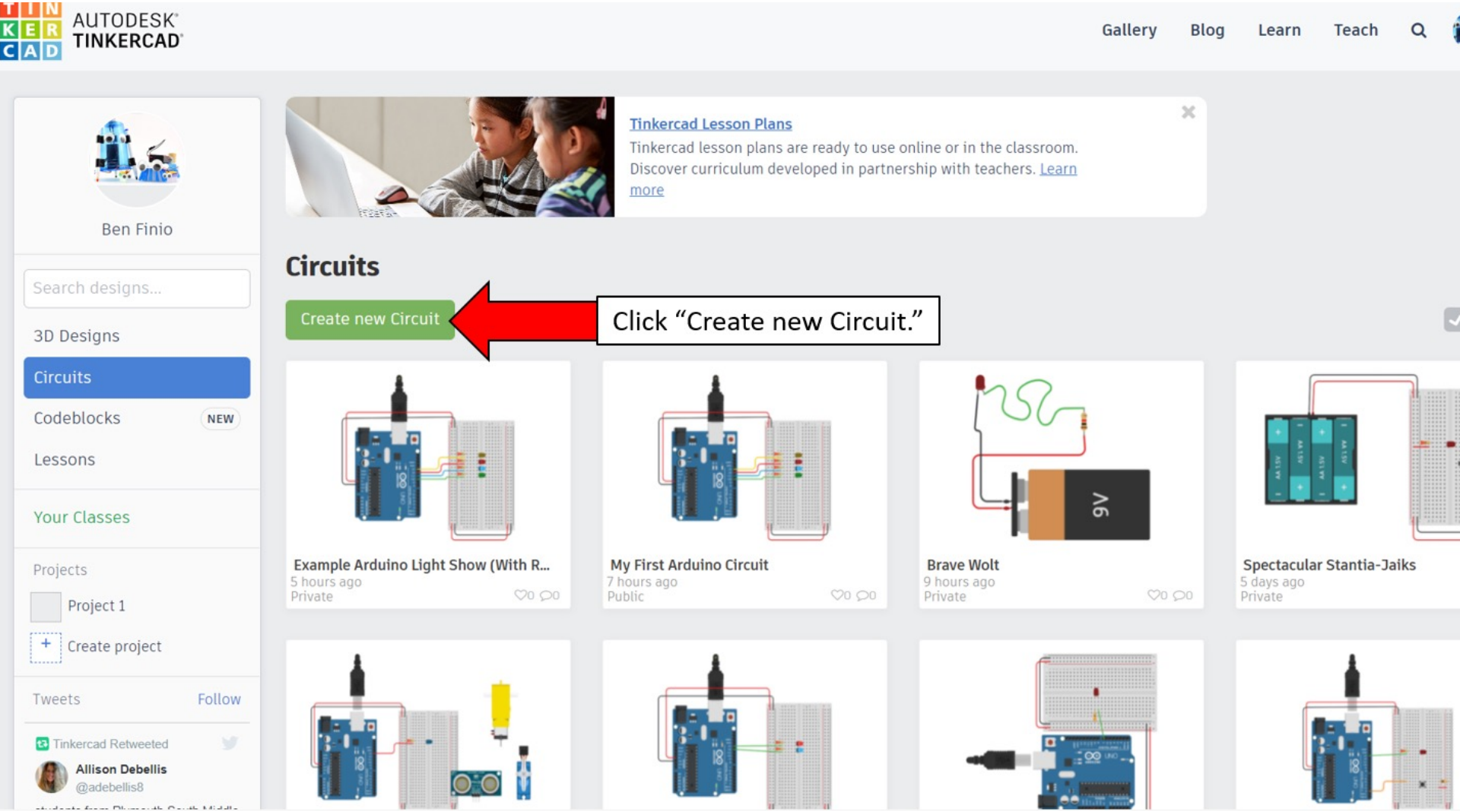
Virtual Activity - 1

Blinking LED

Starting a new circuit

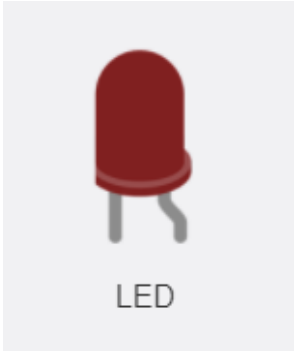


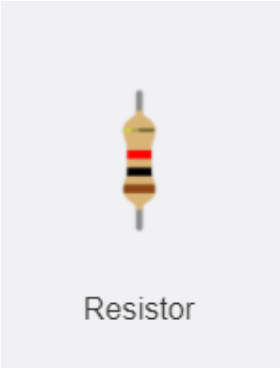
Starting a new circuit

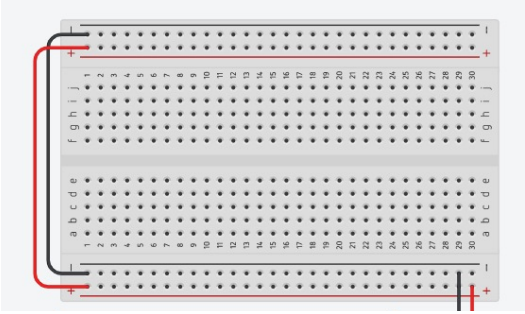


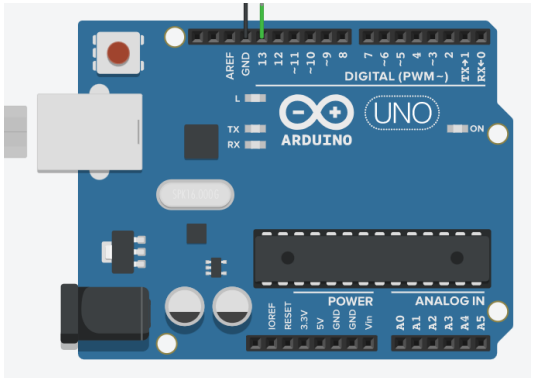
Components to be used

- LED
- Resistor
- Arduino
- Breadboard



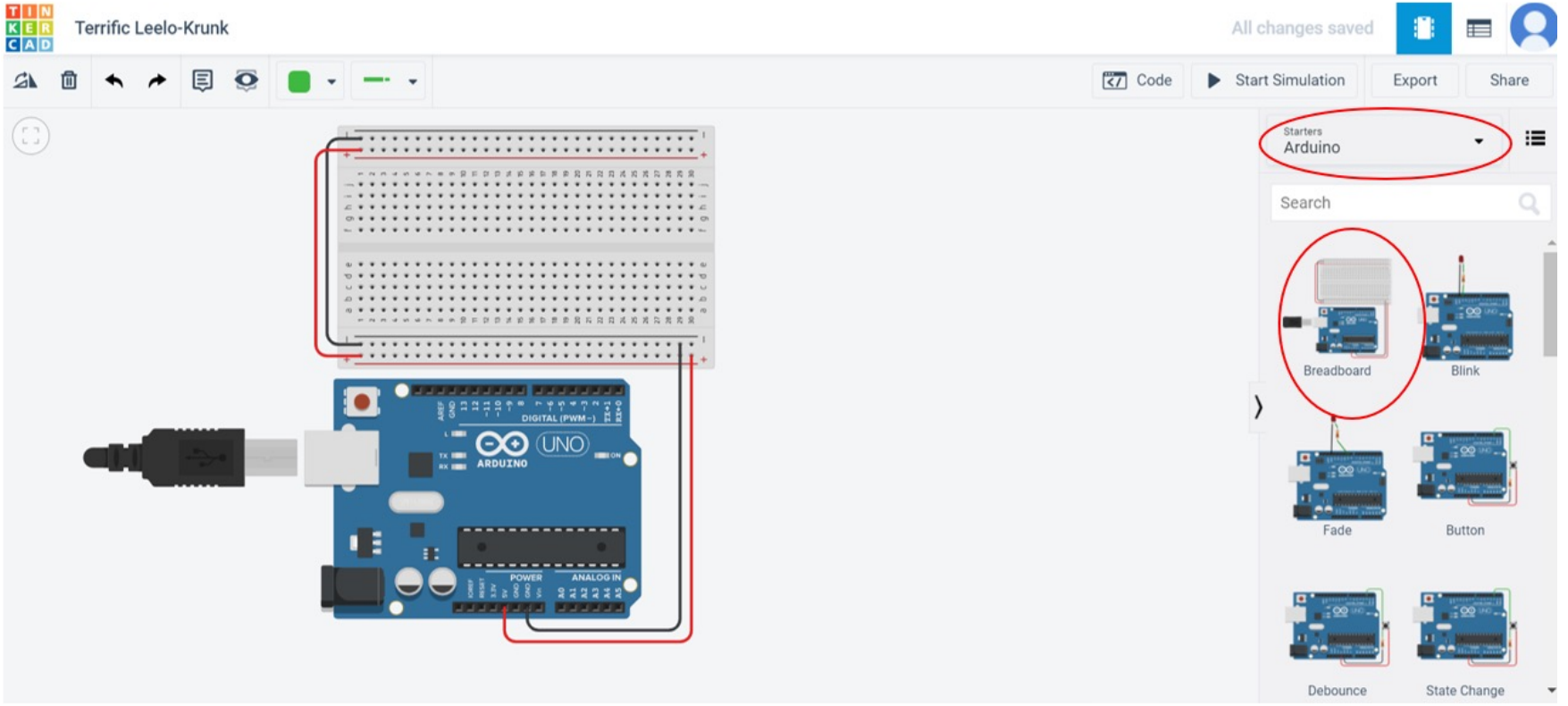






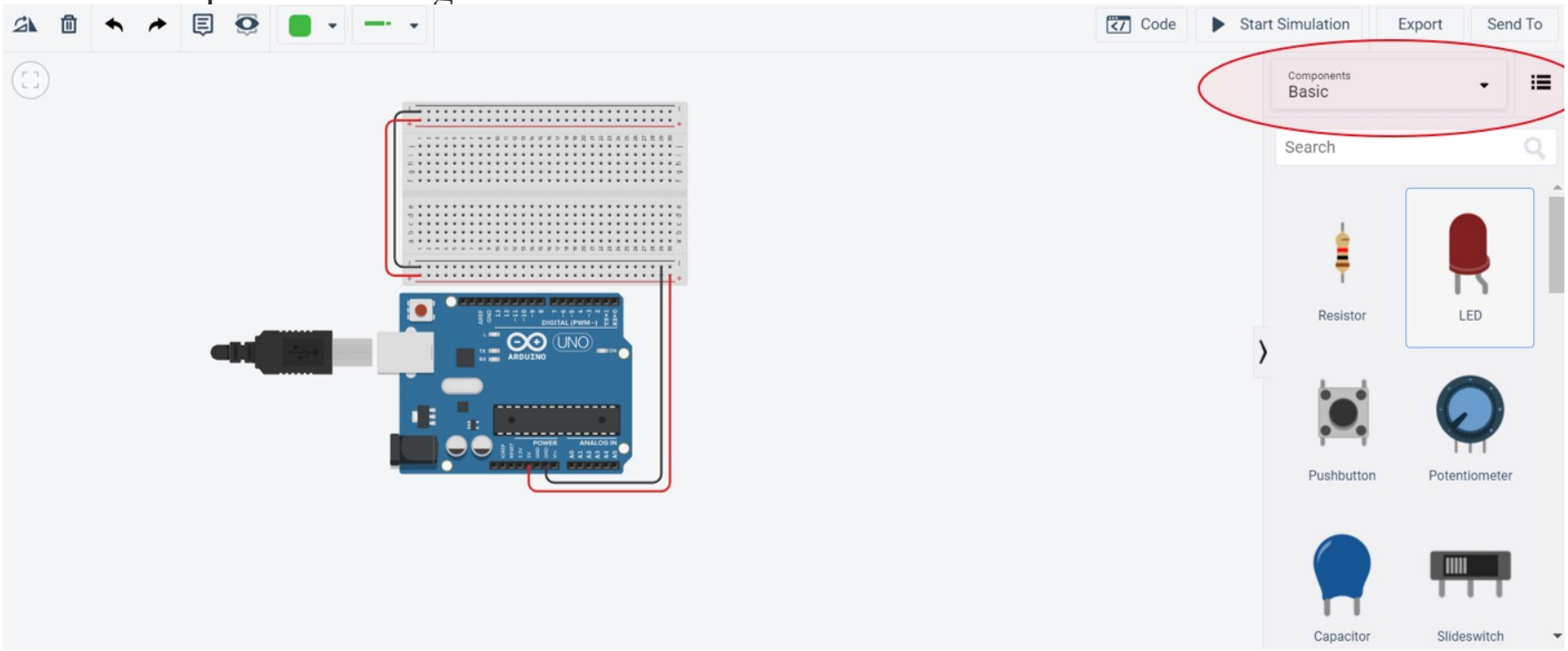
Setting up our Workplace

1. Under Starters/Components: click Arduino.
2. Click and drag on the first component, place it on the workplace and let go.



Setting up our Workplace

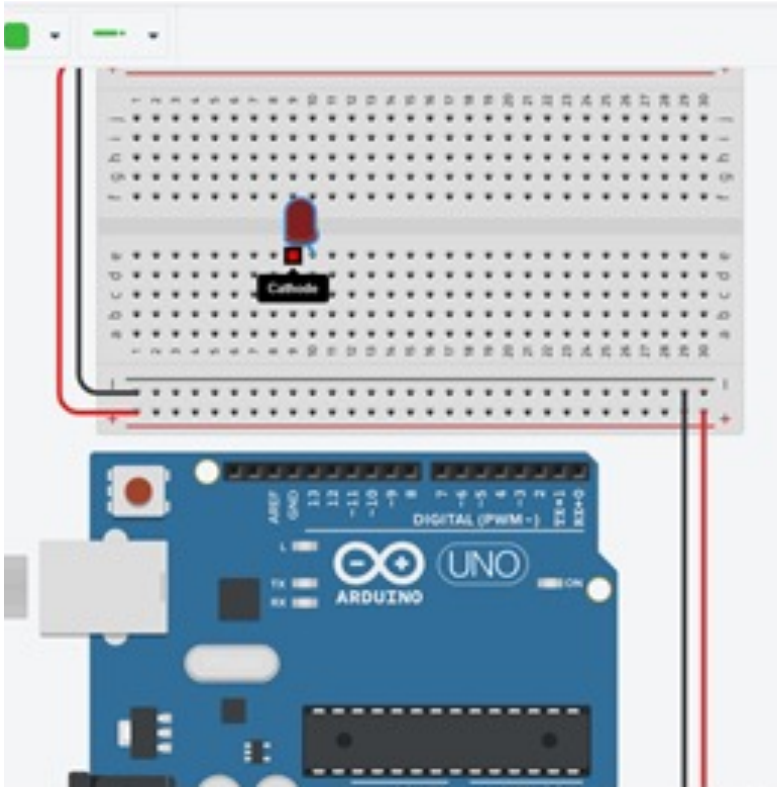
1. Under Basic/Components: click on the LED.
2. Click and drag on the first component, place it on the workplace and let go.

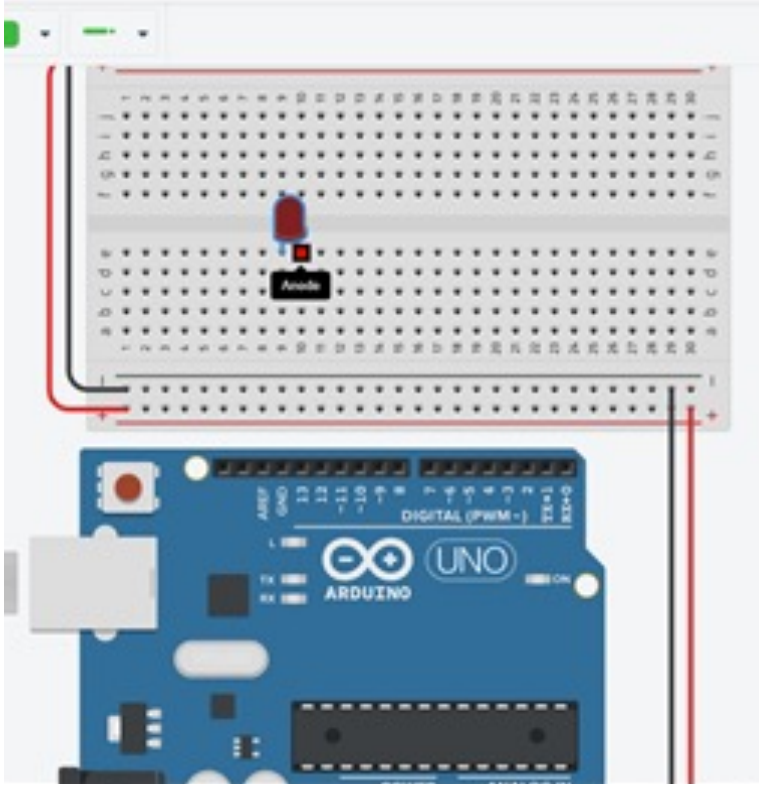


Setting up our Workplace

1. Under Basic/Components: click on the LED.
2. Click and drag on the first component, place it on the workplace and let go.

*Notice the Cathode is on Row 9, Column e. And the Anode is on Row 10, Column e.

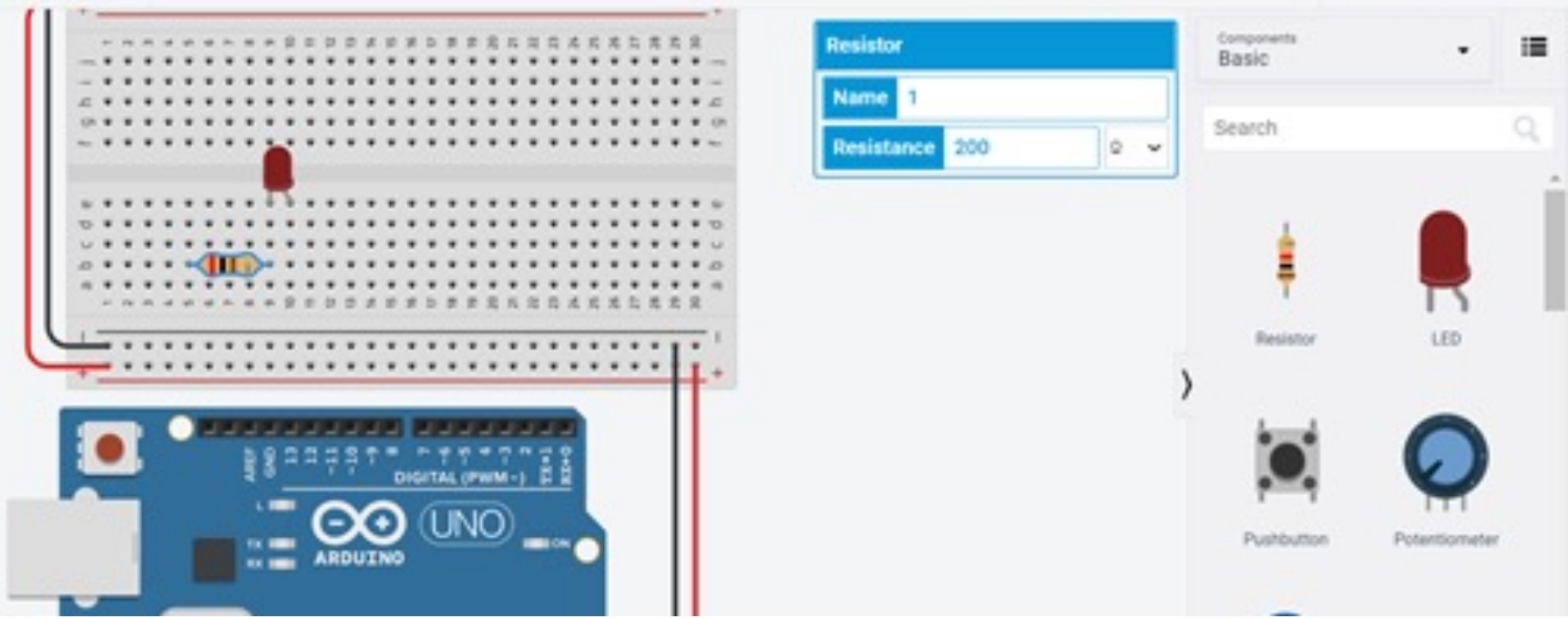




Setting up our Workplace

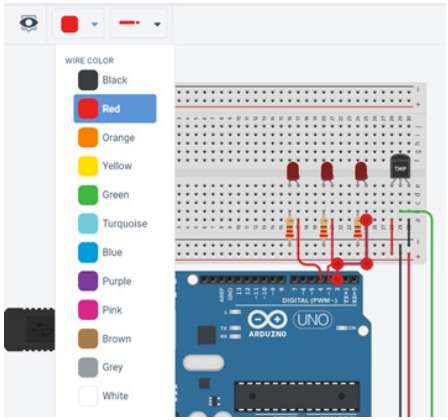
1. Under Basic/Components: click on the Resistor.
2. Click and drag on the component, place it on the workplace and let go.

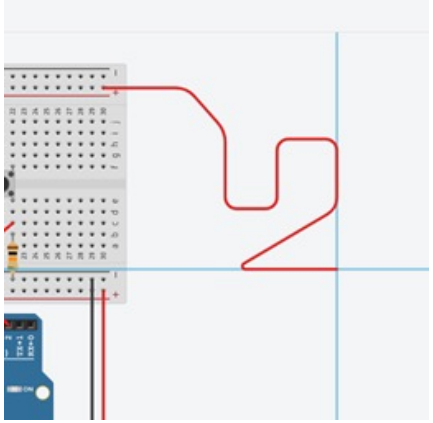
-Notice one end is on Row 5, Column b. And the other end is on Row 9, Column b.
-You can rotate the resistor by press "R" on your keyboard.
-Set your resistor to 200 Ohms

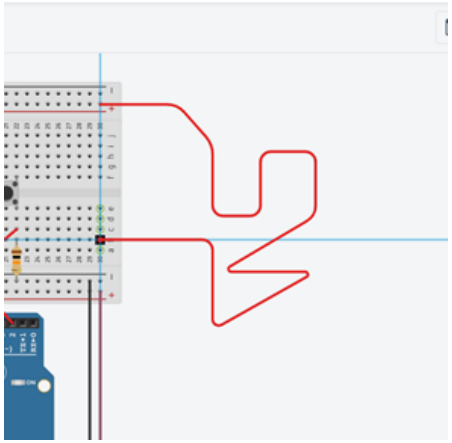


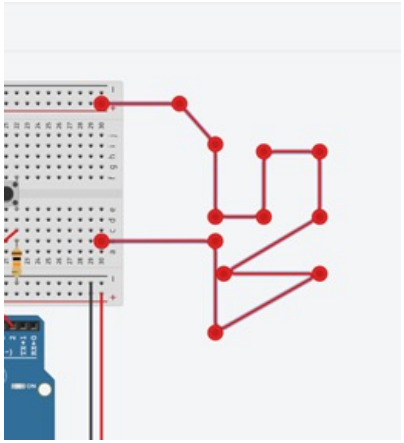
Wiring Orientation

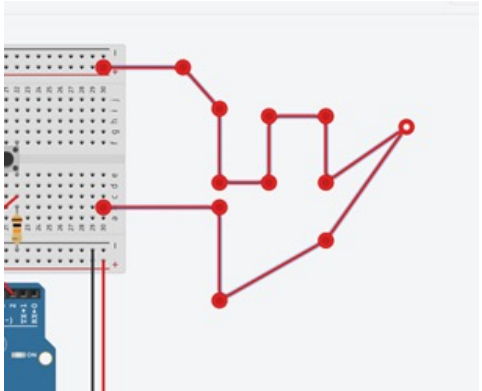
1. Choose a color for the wire. Commonly, your ground will be black, and power will be red.
2. Next choose a start point and click once, you must choose a port or a location on the bread board. You will now have a wire that can be stretched almost at any angle.
3. In order to make a your wire "curve" at different points, you can right click, as long as you do not click when hoovering over a port or else it'll assumingly connect to that port.
4. You add as many turning points as you want, and they will appear after you connect to a port.
5. After which you can stretch the wire through its' turning points.











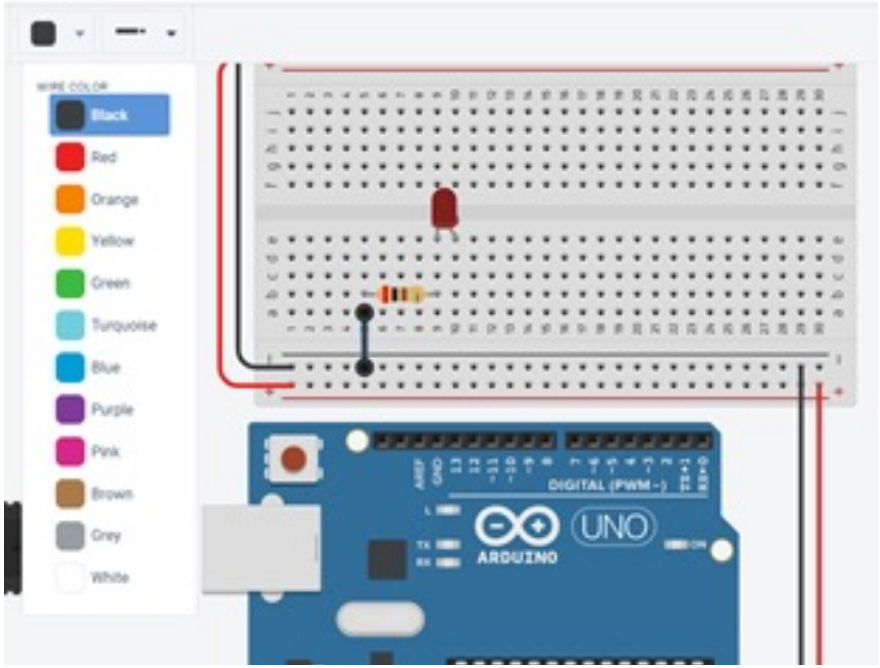
Wiring a LED with Arduino

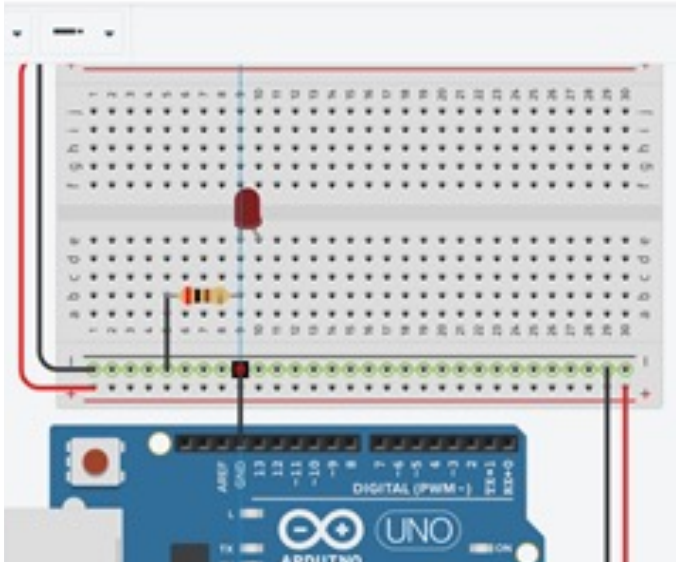
Set up

1. Select the color black for the following wires as seen in the image.

2. One end starts at the Ground bus (row 5). The other end connects to Row 5, Column a.

3. The second wire connects from the GND port on the Arduino to the Ground bus, aligned with row 9. This provide a return path to ground while using a pull down resistor on the Cathode side of the LED.





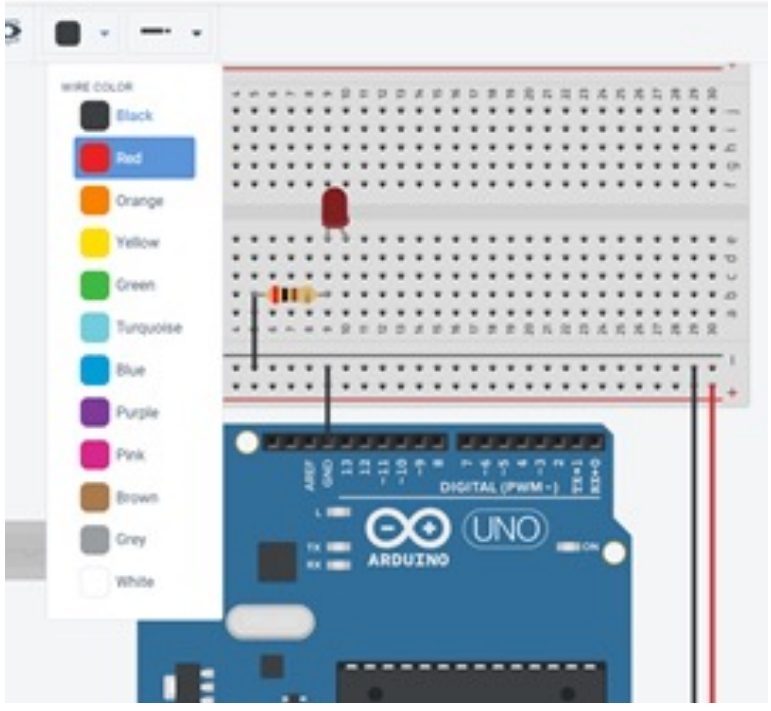
Wiring a LED with Arduino

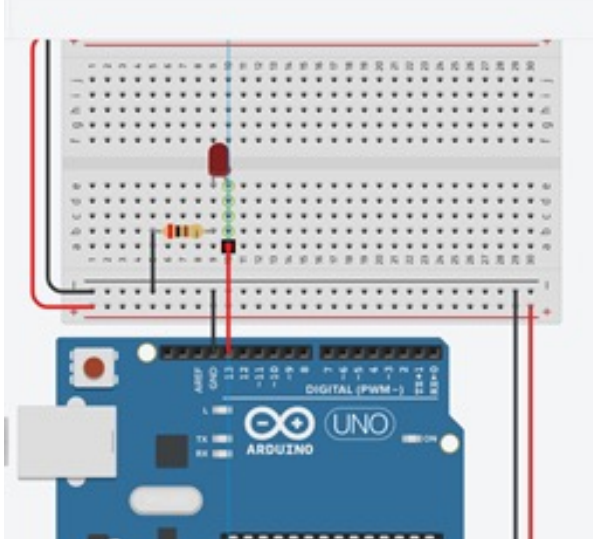
Set up

1. Select the color red for the following wire as seen in the image.

2. One end starts at the port on the Arduino, Port 13.

3. The other end of the wire connects to Row 10, Column a. This provides power to the Anode side of the LED.

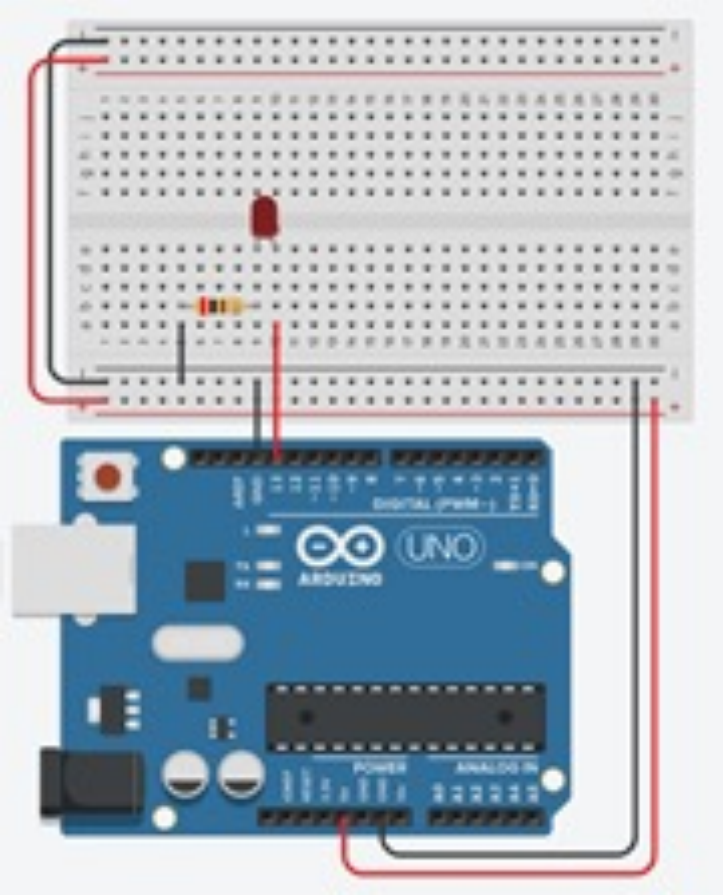




Wiring a LED with Arduino Cont.

Set up

- Ensure hardware is wired as shown on the image.
- Next will be the coding process.

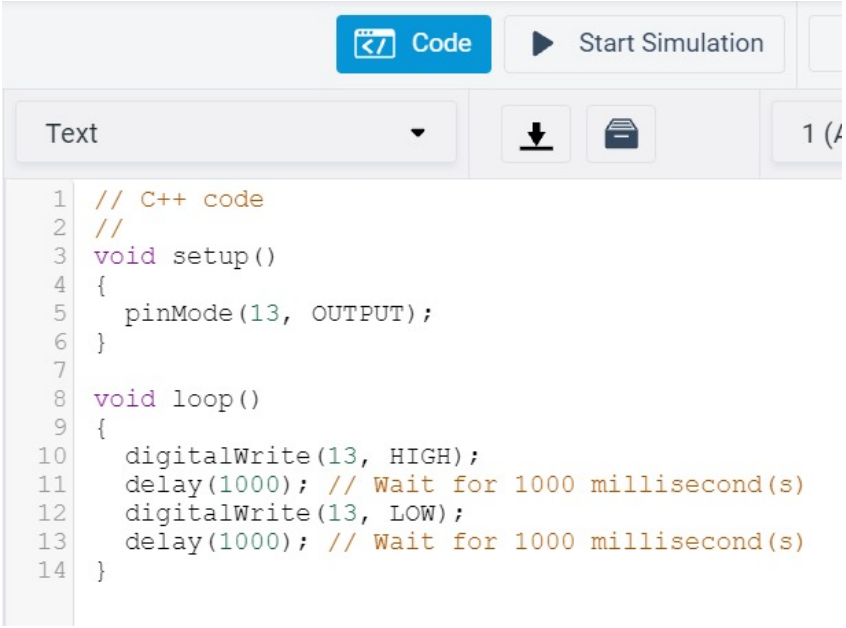


C++ Review

Code

1. Click on "Code" button to open the coding window.
2. Review code format:

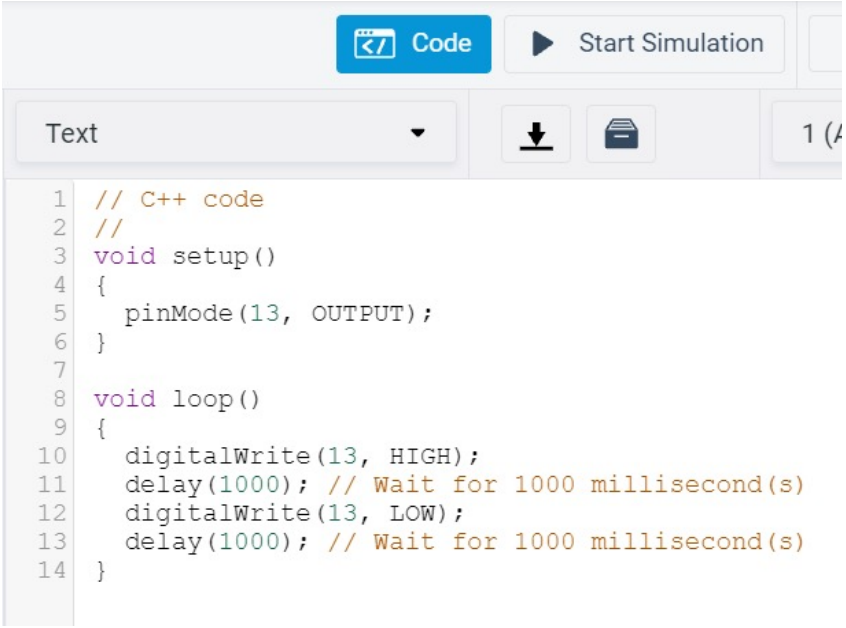
```
// Comments for the user

void setup()
{
  variables; // here we setup our variables
}

void loop()
{
  predefined variables; // here we setup our loop
  delay(1000); // wait for 1 second
}
```



Copy code C++

Code

1. When code is similar to the image on the right then go ahead and Click "Start Simulation".



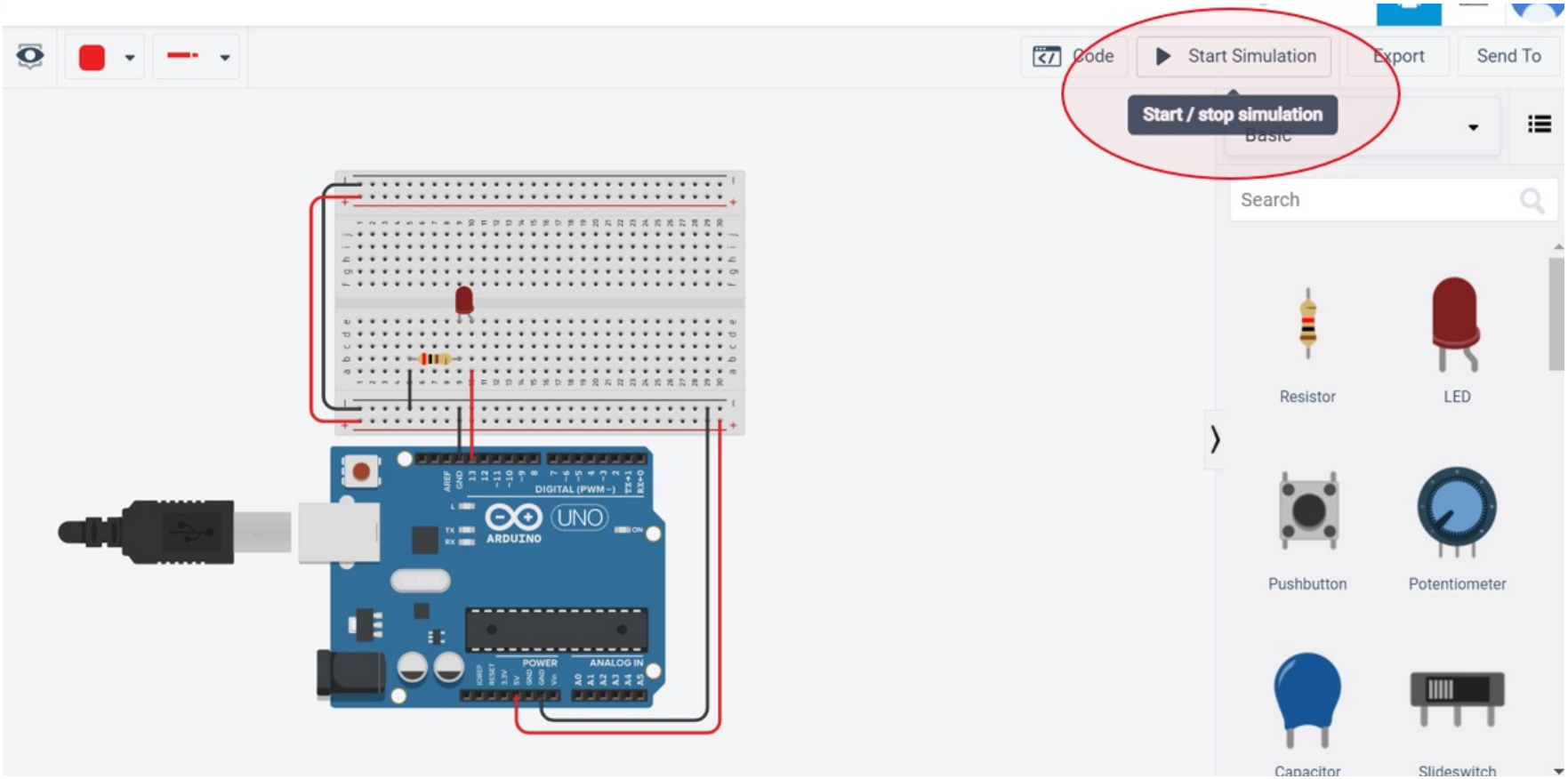
2. Copy Code:

```
// C++ code
//
void setup()
{
 pinMode(13, OUTPUT);
}

void loop()
{
 digitalWrite(13, HIGH);
 delay(1000); // Wait for 1000 millisecond(s)
 digitalWrite(13, LOW);
 delay(1000); // Wait for 1000 millisecond(s)
}
```

Begin Simulation

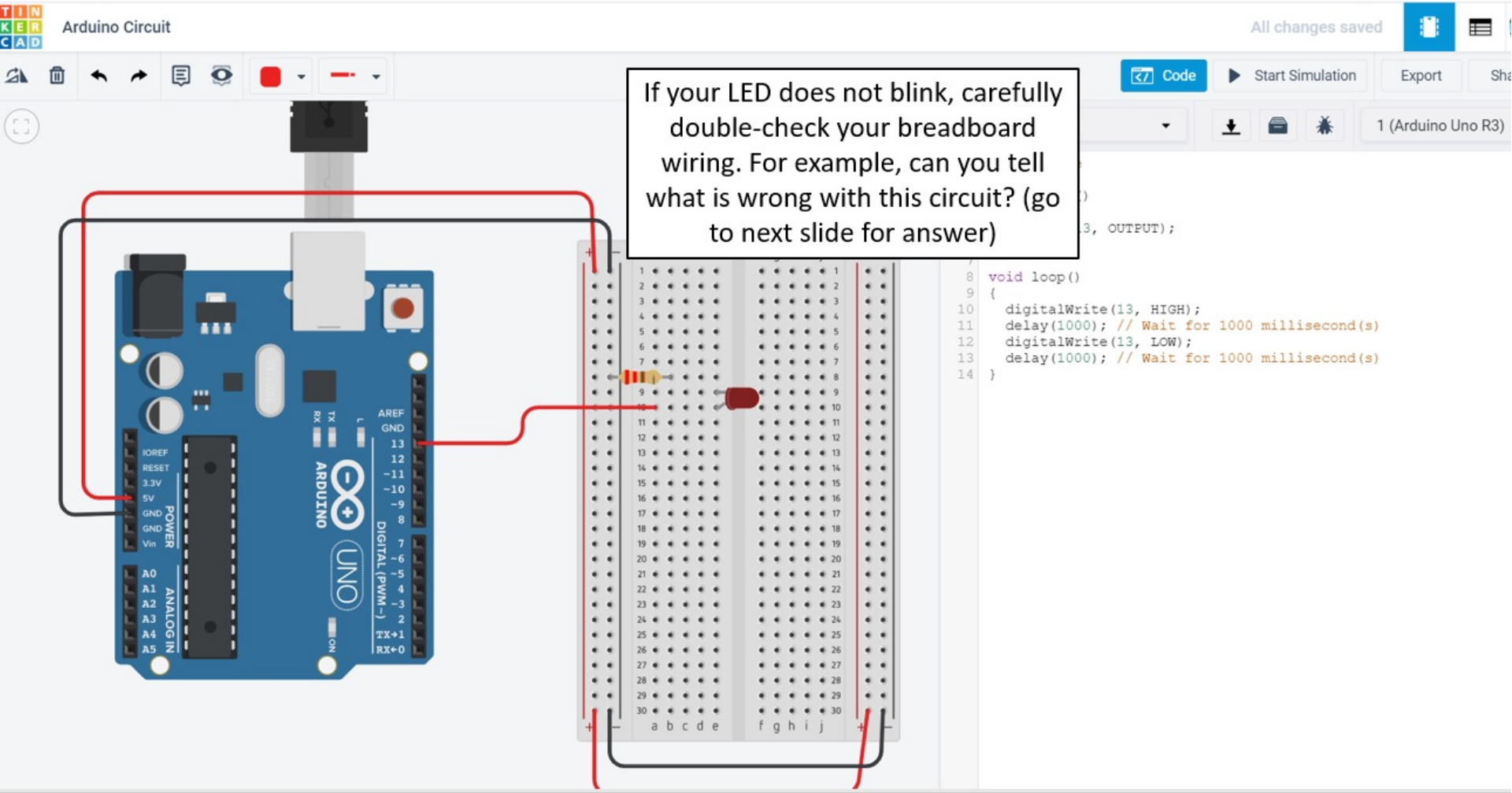
Blink LED

Click on Start Simulation to begin simulation and you'll see the LED start to blink.



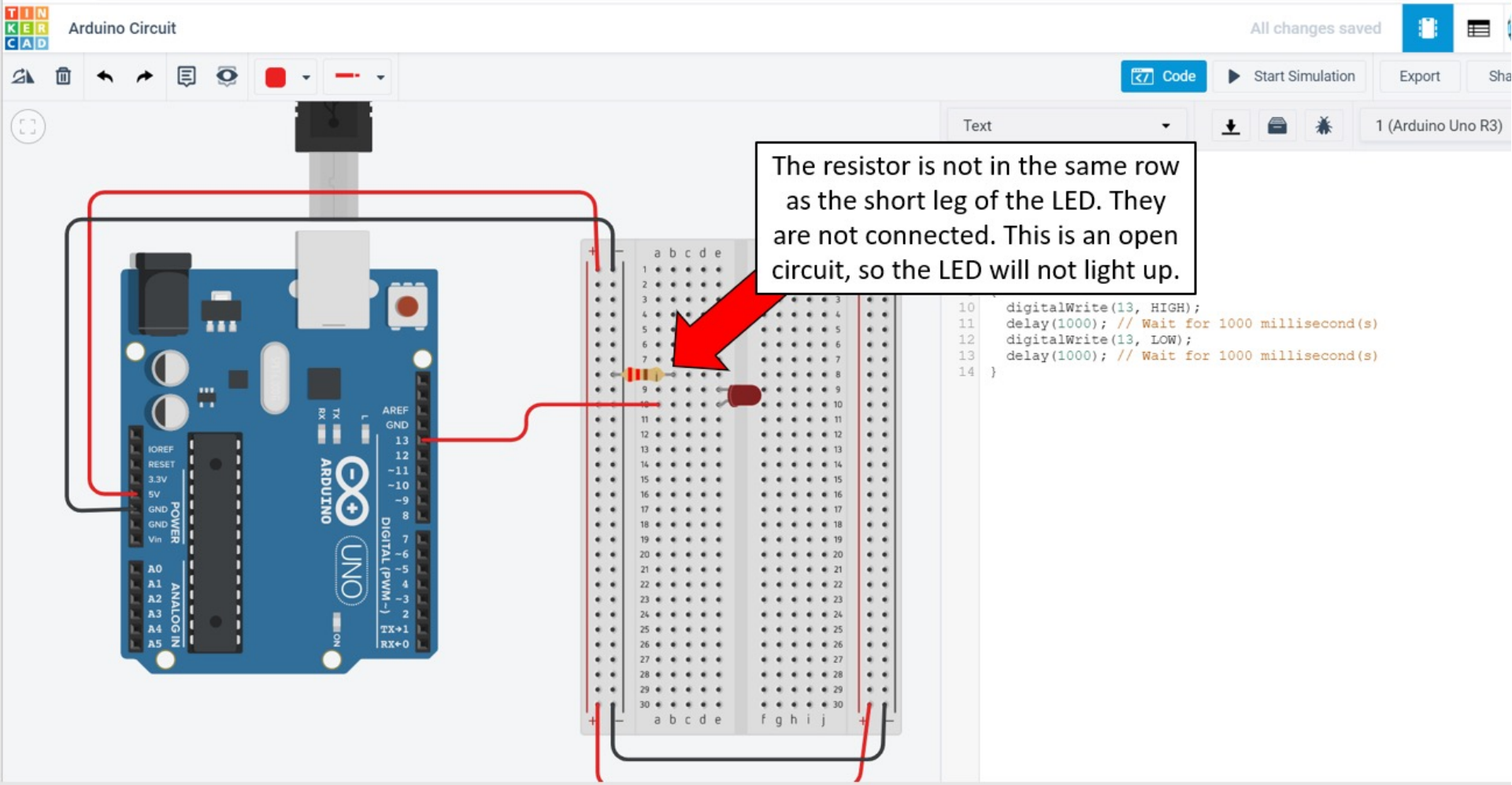
Simulation

Blink LED



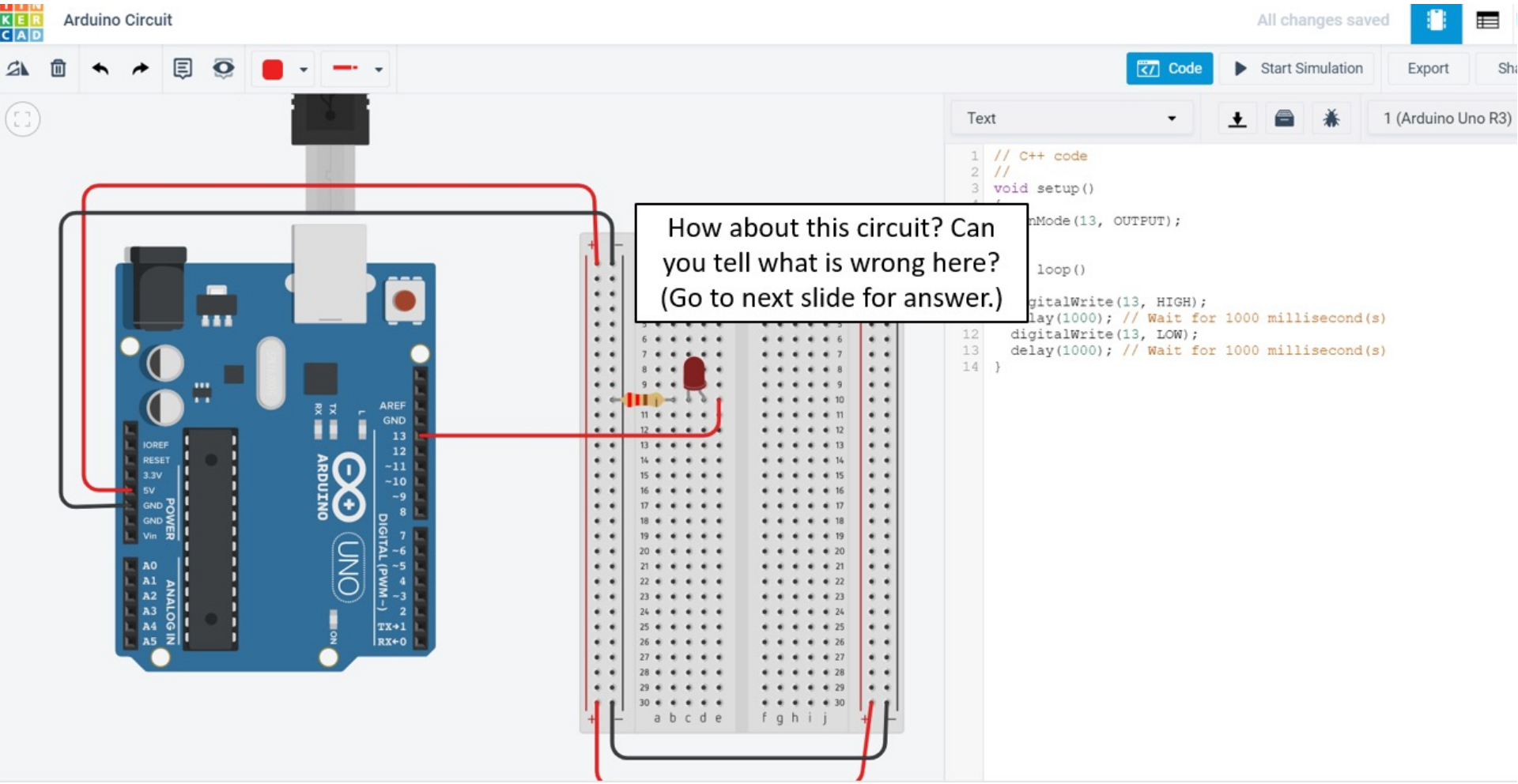
Simulation

Blink LED



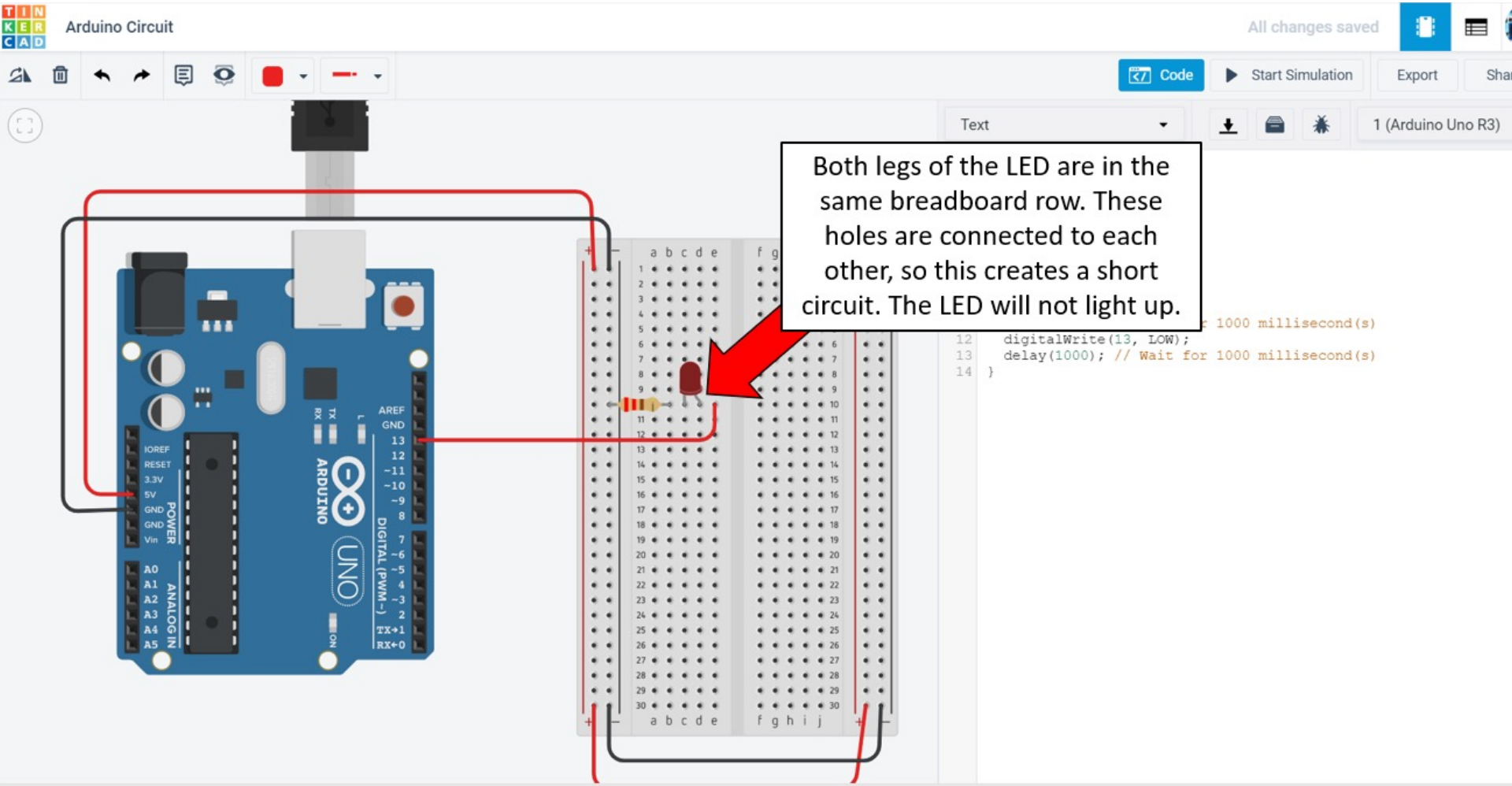
Simulation

Blink LED



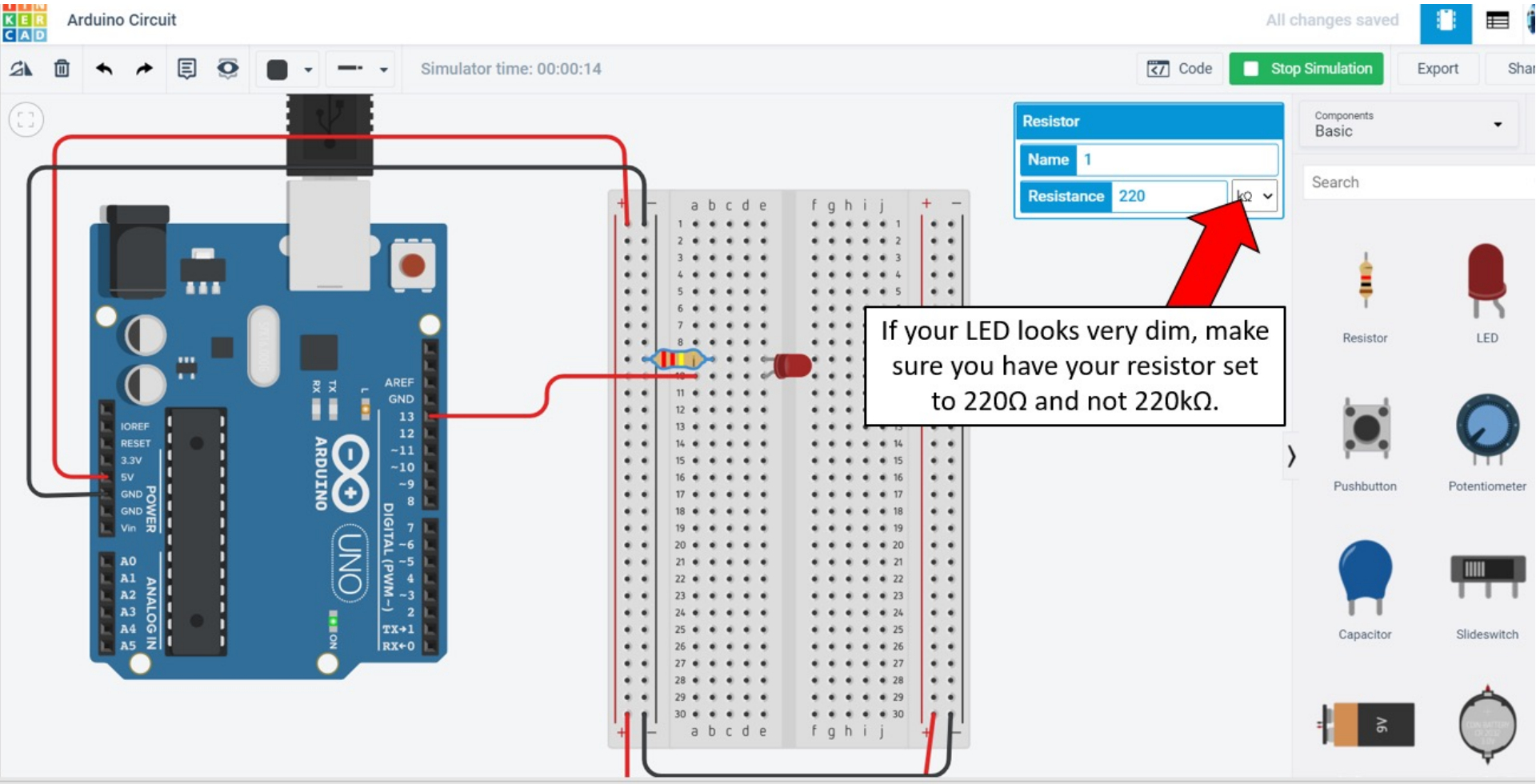
Simulation

Blink LED



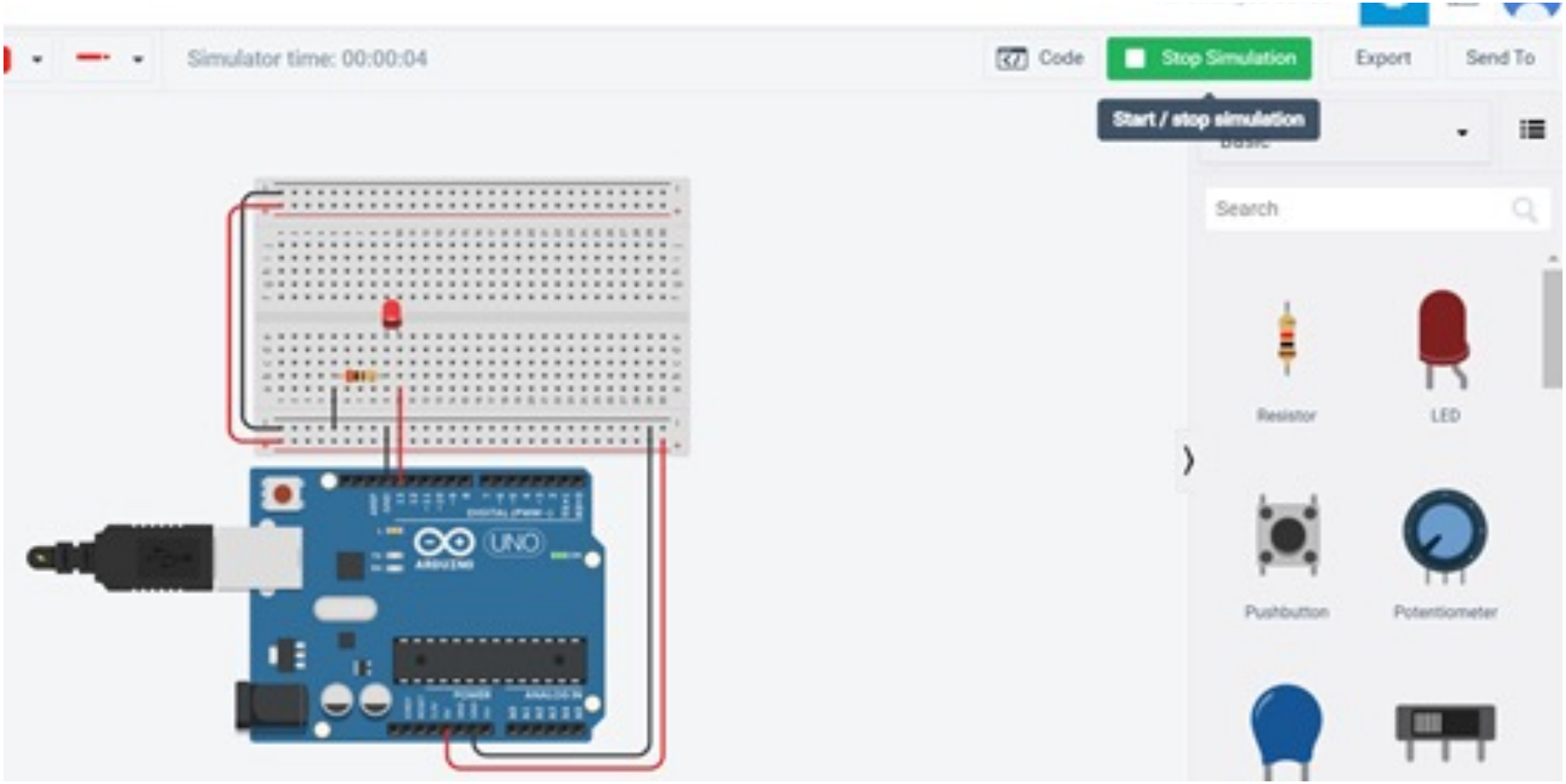
Simulation

Blink LED



Stop simulation

Click on Stop Simulation to end simulation and you'll see the LED stop blinking.



What did we witness?

- Did you notice the small LED flashing on the board itself? This built in LED is also connected to pin 13, and is meant to be used for testing purposes without the need to connect any external components. It even has its own tiny resistor, soldered directly to the surface of the board

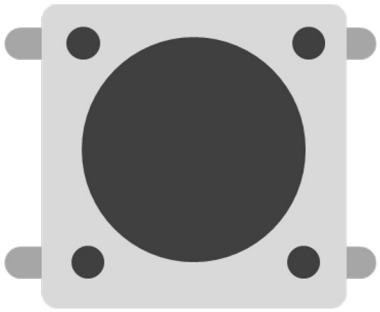
Virtual Activity – 2
Arduino Switch

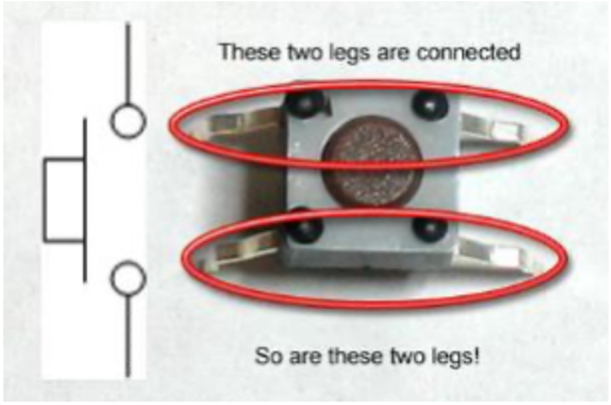
The switch

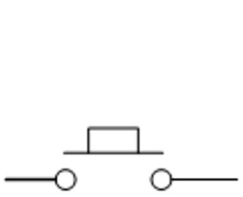
A switch is a simple, digital **sensor**

Switches come in different forms, but all of them in some way **open** or **close** a gap in a wire

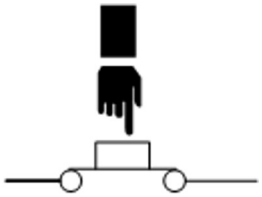
The **pushbutton** switch has four legs for easier mounting, but only two of them are needed





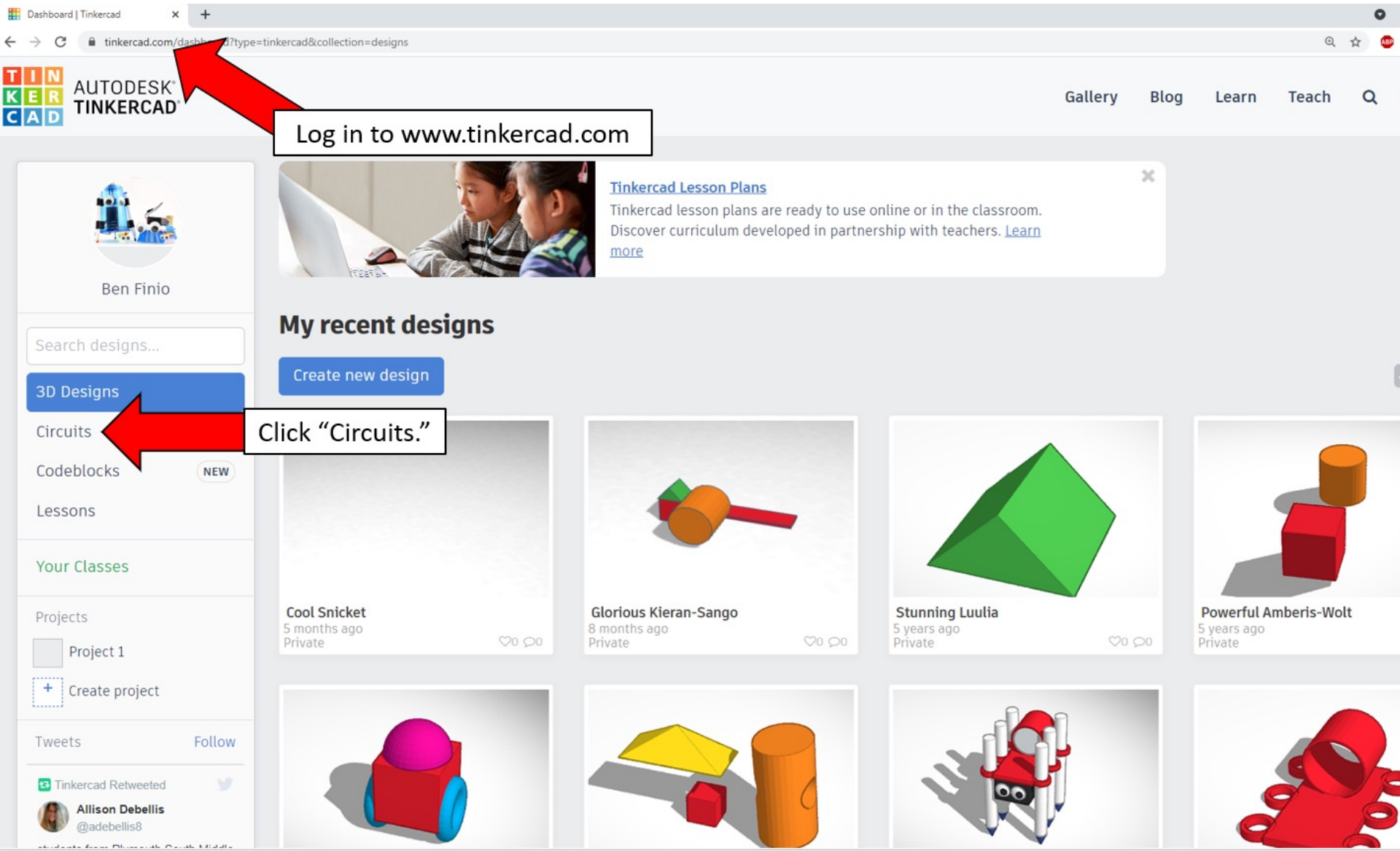


Normally Open
When not pressed

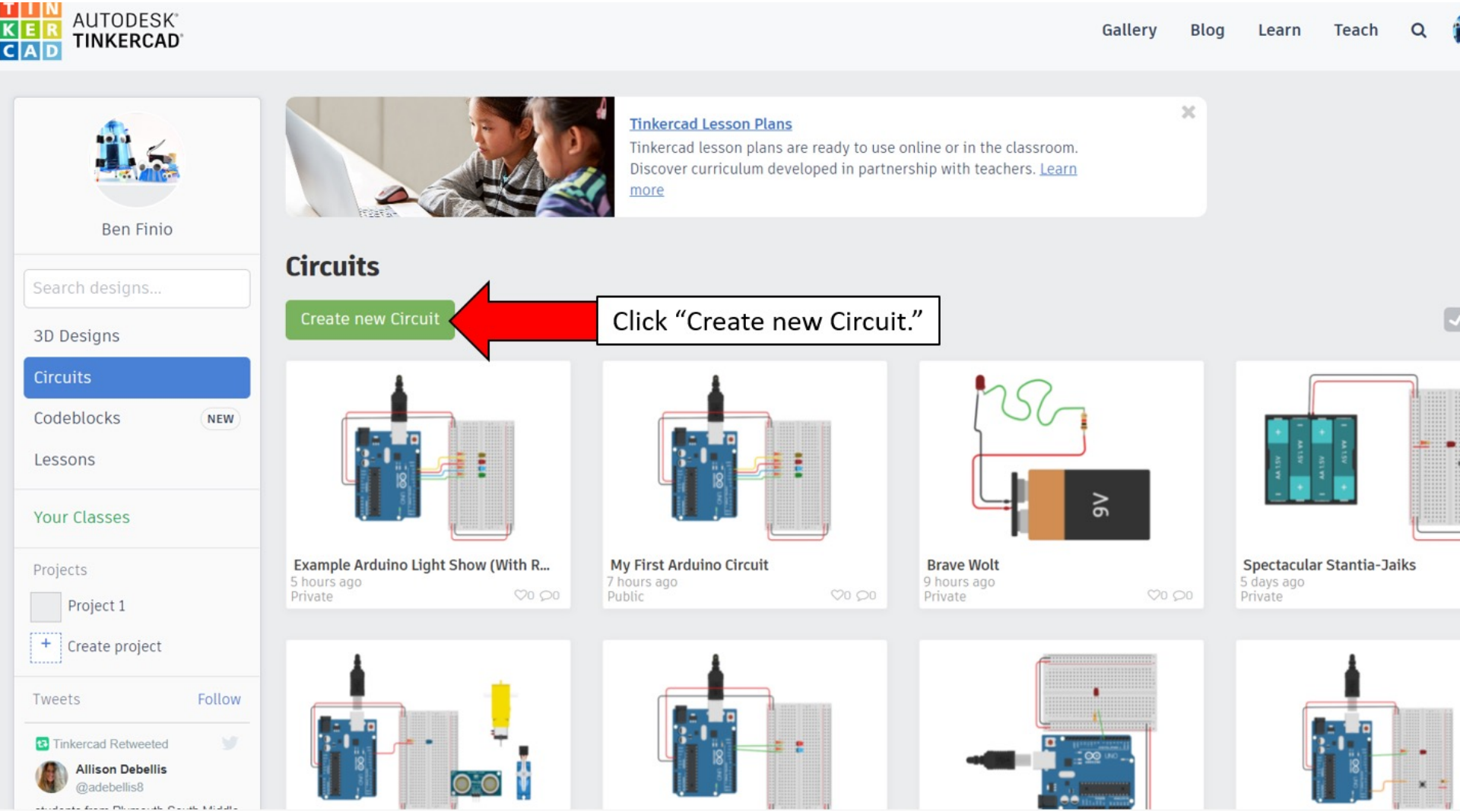


Closed
When pressed

Starting a new circuit



Starting a new circuit

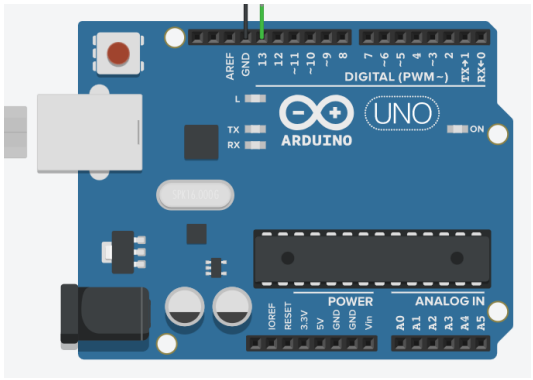


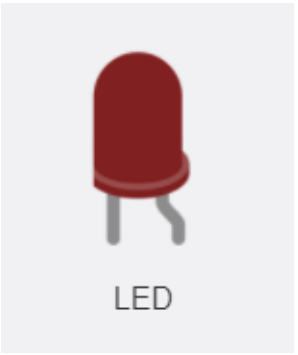
Components List

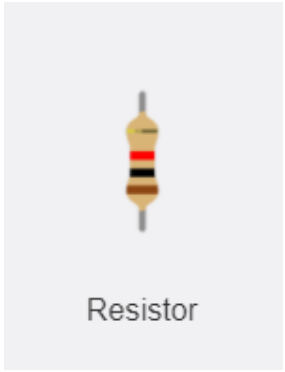
We will introduce how to blink the on-board LED and how to blink an external LED with a switch.

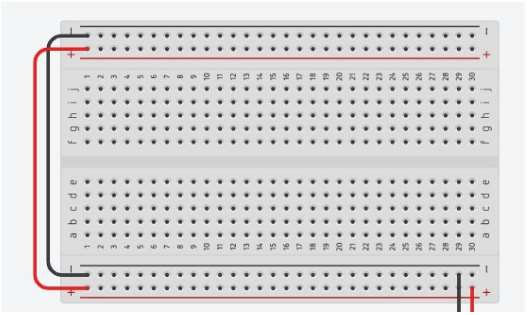
Components

- Arduino x1
- Push Button x1
- LED x1
- 10 KΩ Resistor x1
- 200 Ω Resistor x1
- Bread Board x1



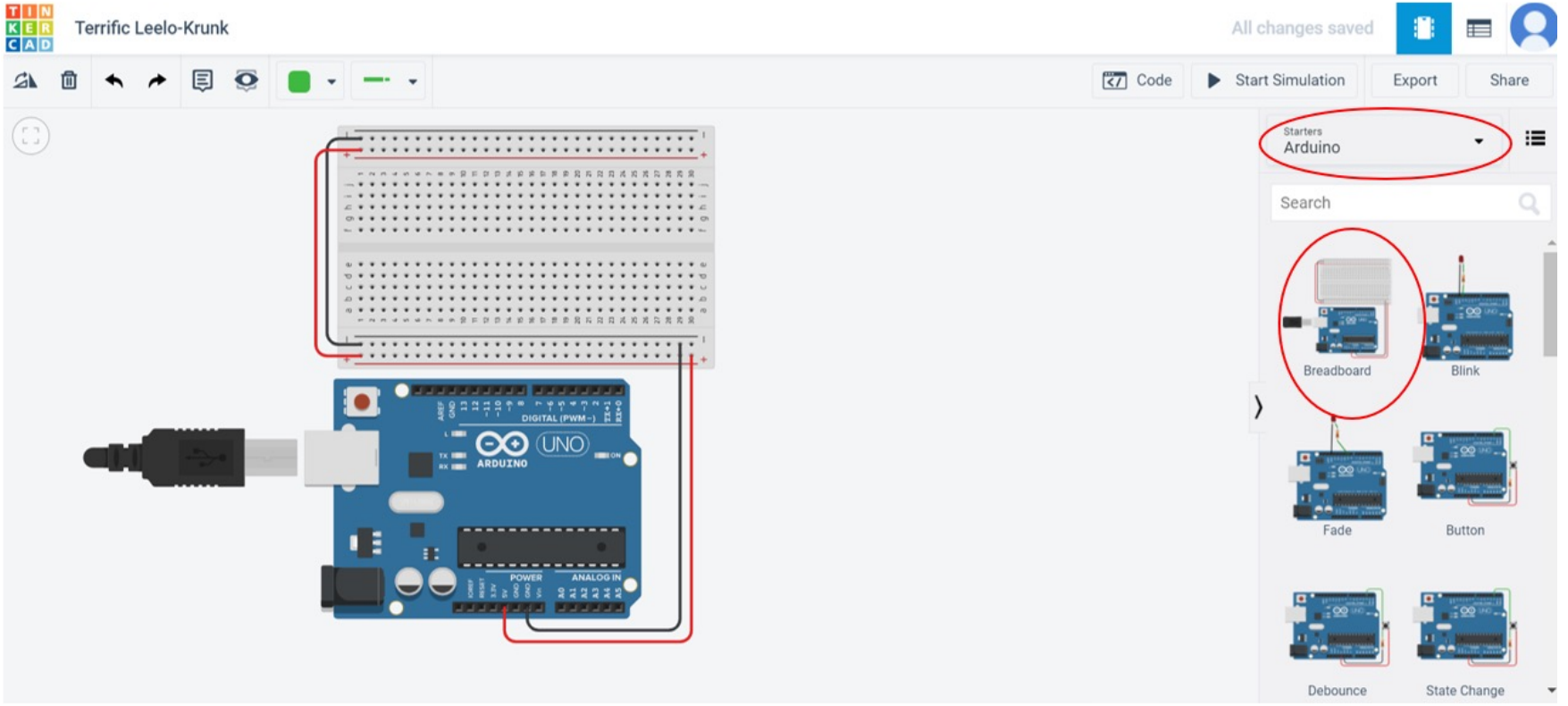






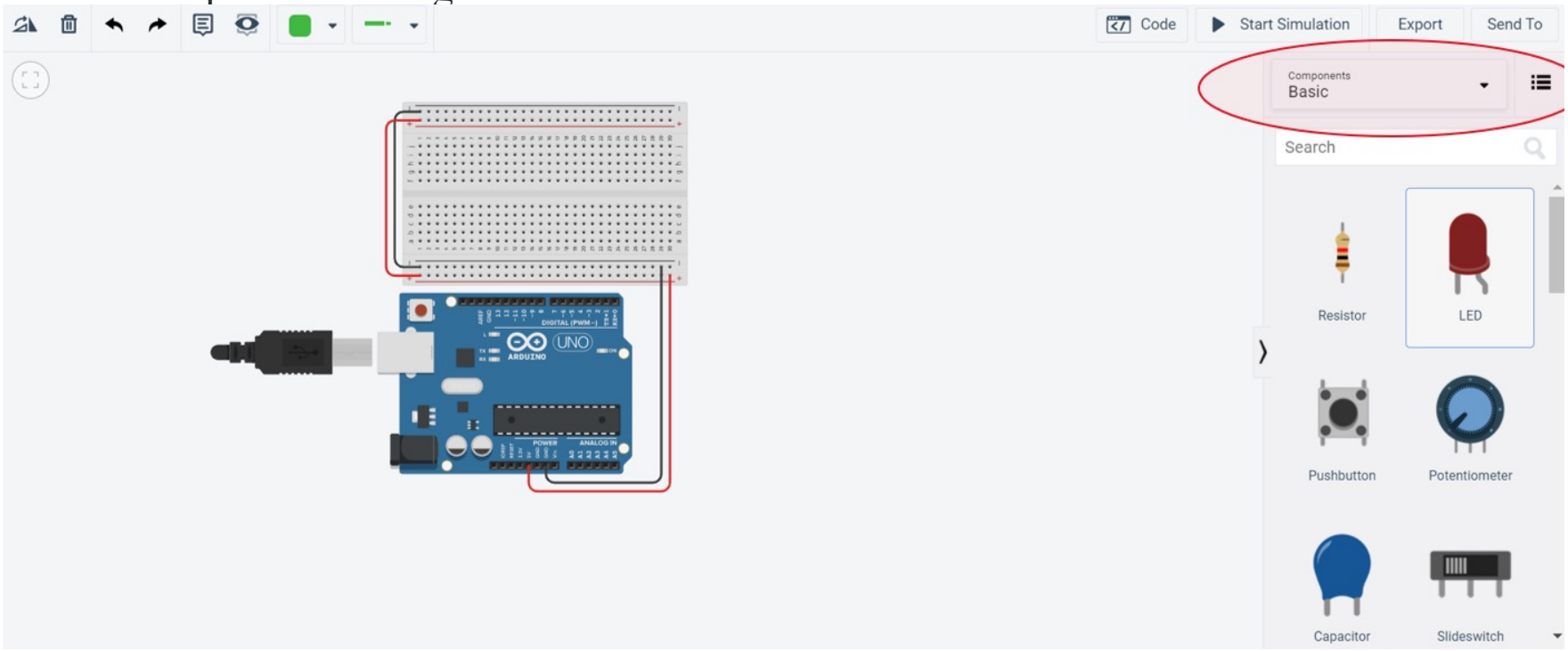
Setting up our Workplace

1. Under Starters/Components: click Arduino.
2. Click and drag on the first component, place it on the workplace and let go.



Setting up our Workplace

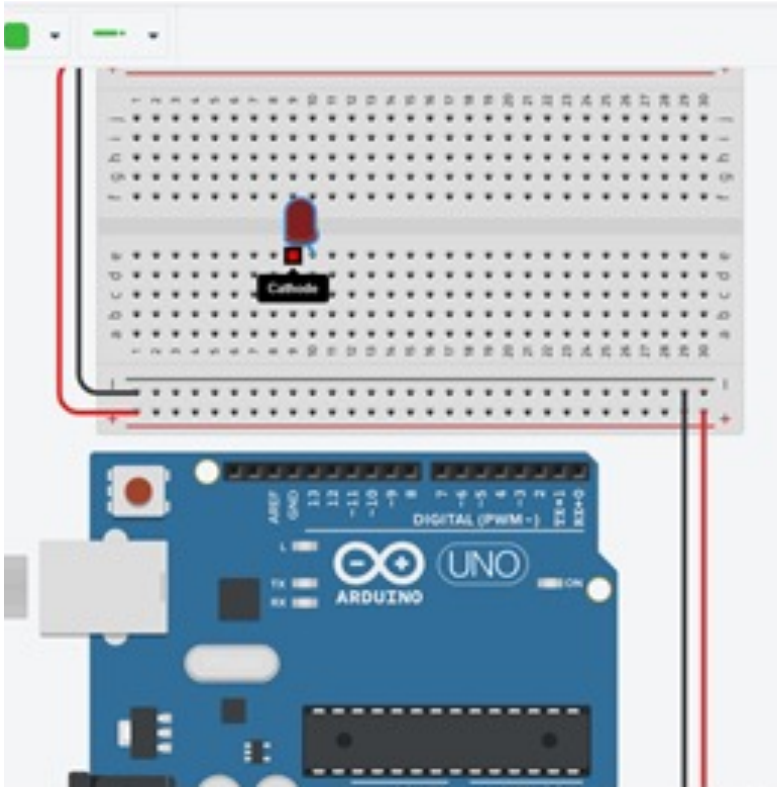
1. Under Basic/Components: click on the LED.
2. Click and drag on the first component, place it on the workplace and let go.

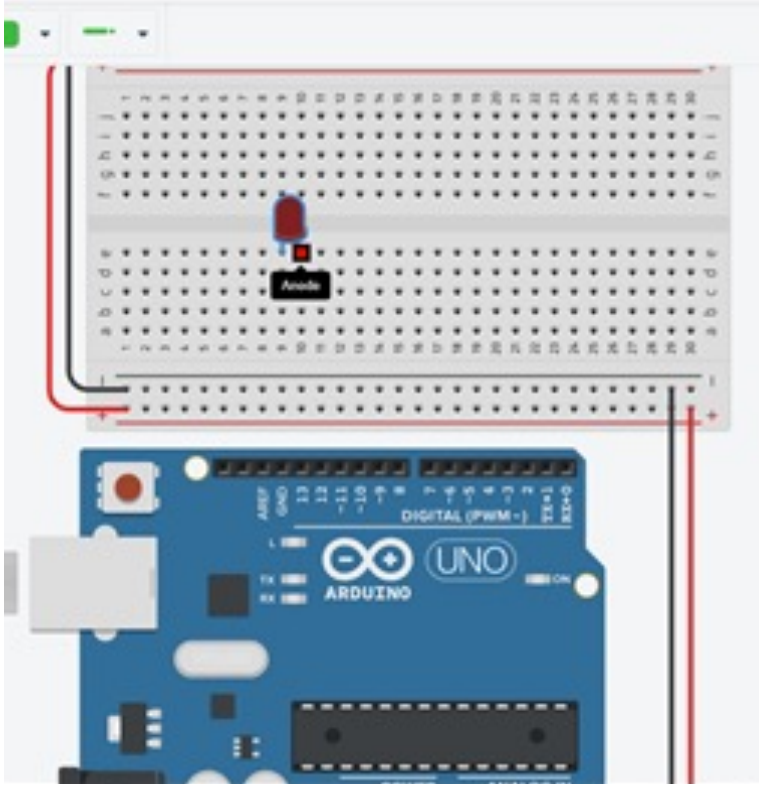


Setting up our Workplace

1. Under Basic/Components: click on the LED.
2. Click and drag on the first component, place it on the workplace and let go.

*Notice the Cathode is on Row 9, Column e. And the Anode is on Row 10, Column e.

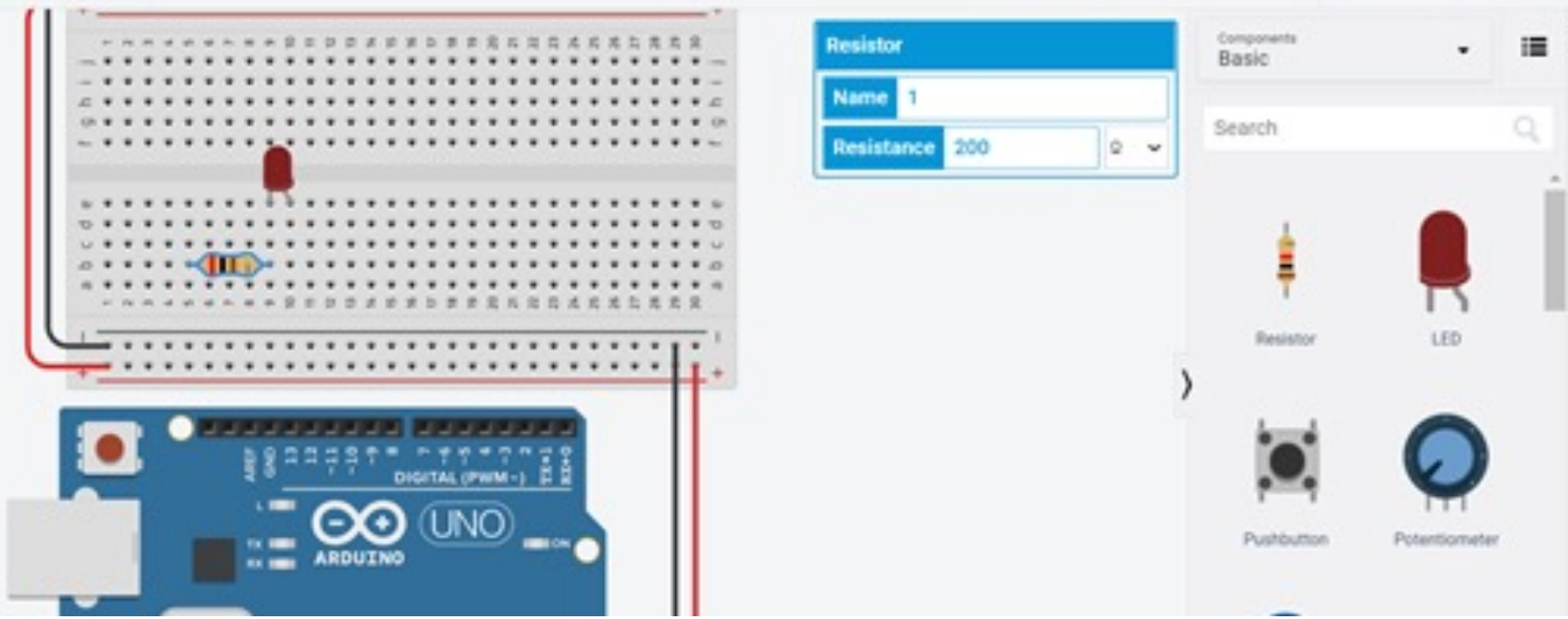




Setting up our Workplace

1. Under Basic/Components: click on the Resistor.
2. Click and drag on the component, place it on the workplace and let go.

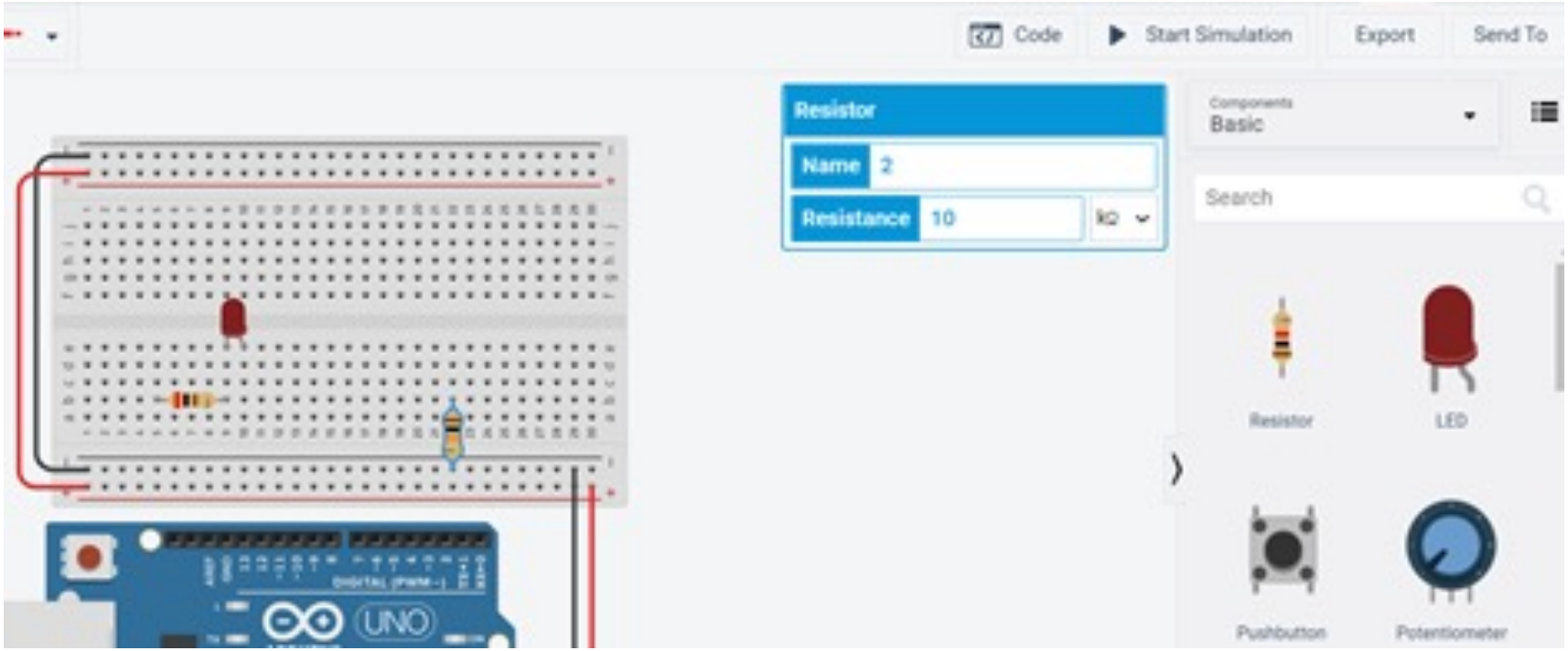
-Notice one end is on Row 5, Column b. And the other end is on Row 9, Column b.
-You can rotate the resistor by press "R" on your keyboard.
-Set your resistor to 200 Ohms



Setting up our Workplace

1. Under Basic/Components: click on the Resistor.
2. Click and drag on the component, place it on the workplace and let go.

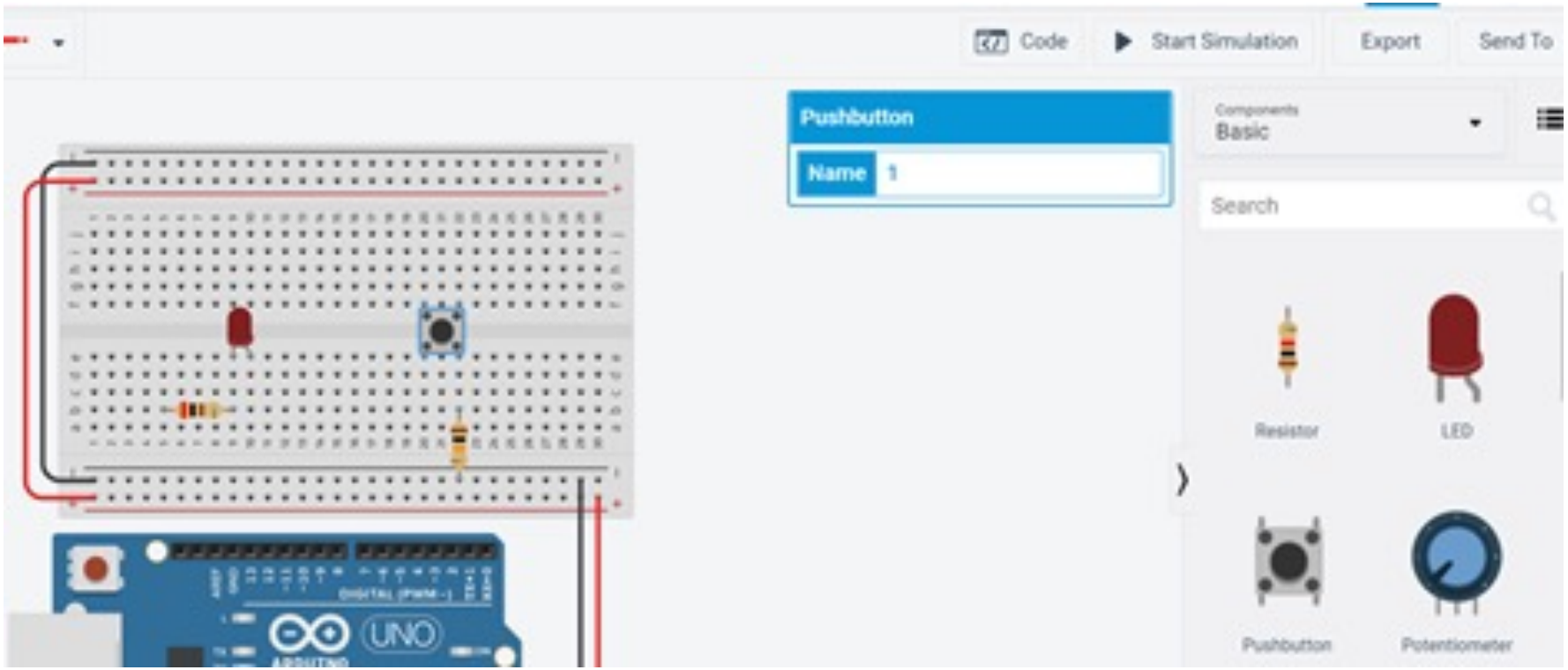
-Notice one end is on Row 22, Column b. And the other end is aligned to Row 22 on the Ground Bus.
-You can rotate the resistor by press "R" on your keyboard.
-Set your resistor to 10 kOhms



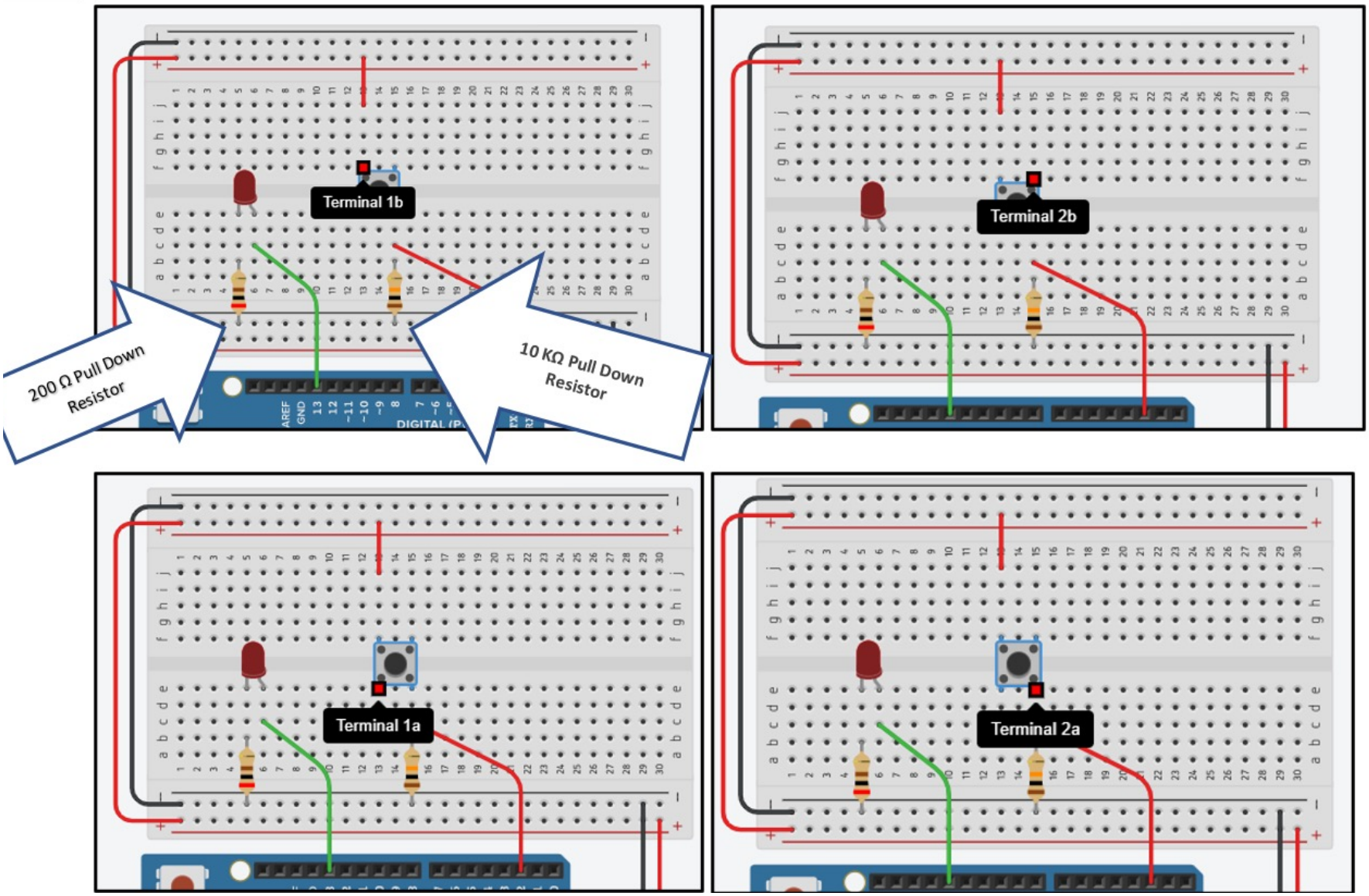
Setting up our Workplace

1. Under Basic/Components: click on the Pushbutton.
2. Click and drag on the component, place it on the workplace and let go.

-Terminals for the Pushbutton: Row 20, Column f (Terminal 1b) . Row 22, Column f (Terminal 2b). Row 20, Column e (Terminal 1a). Row 22, Column e (Terminal 2a)).
-You can rotate the resistor by press "R" on your keyboard.

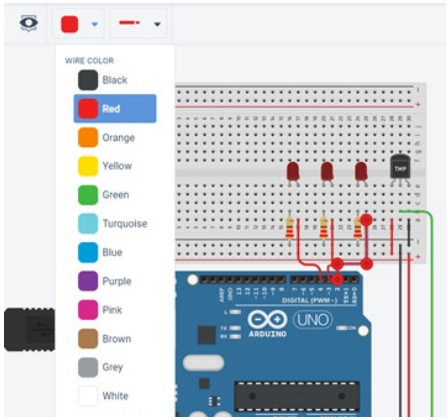


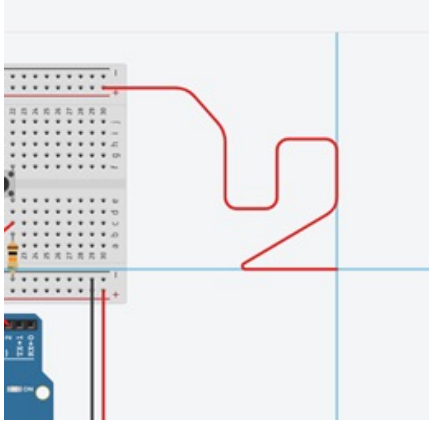
Switch Terminals

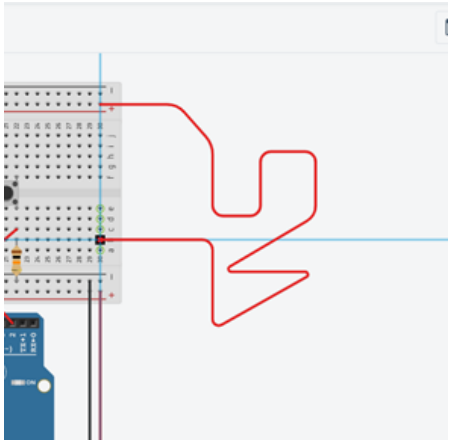


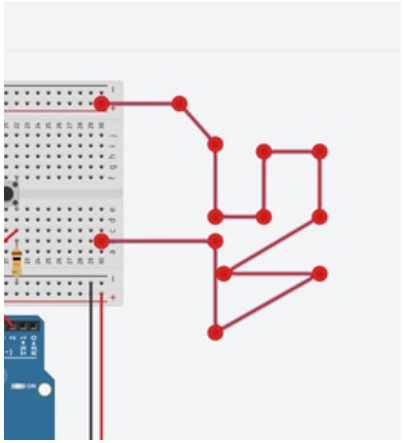
Wiring Orientation

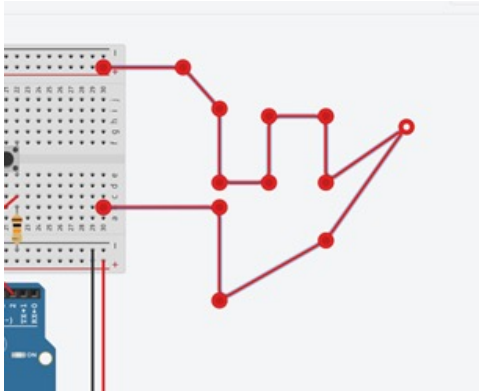
1. Choose a color for the wire. Commonly, your ground will be black, and power will be red.
2. Next choose a start point and click once, you must choose a port or a location on the bread board. You will now have a wire that can be stretched almost at any angle.
3. In order to make a your wire "curve" at different points, you can right click, as long as you do not click when hoovering over a port or else it'll assumingly connect to that port.
4. You add as many turning points as you want, and they will appear after you connect to a port.
5. After which you can stretch the wire through its' turning points.











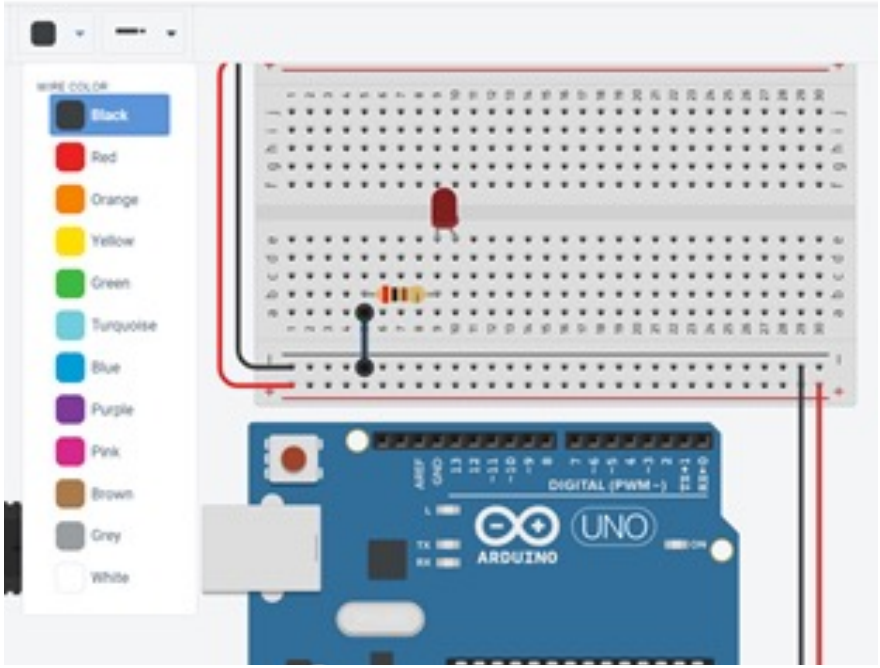
Wiring a LED with Arduino

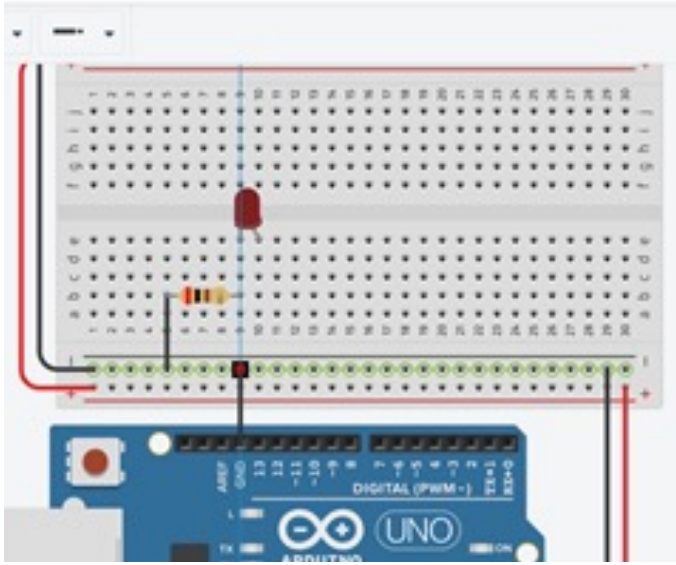
Set up

1. Select the color black for the following wires as seen in the image.

2. One end starts at the Ground bus (row 5). The other end connects to Row 5, Column a.

3. The second wire connects from the GND port on the Arduino to the Ground bus, aligned with row 9. This provide a return path to ground while using a pull down resistor on the Cathode side of the LED.





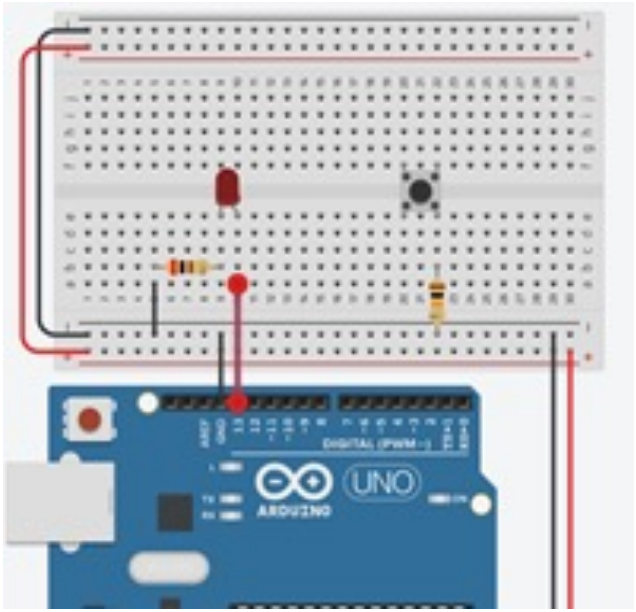
Wiring a LED with Arduino

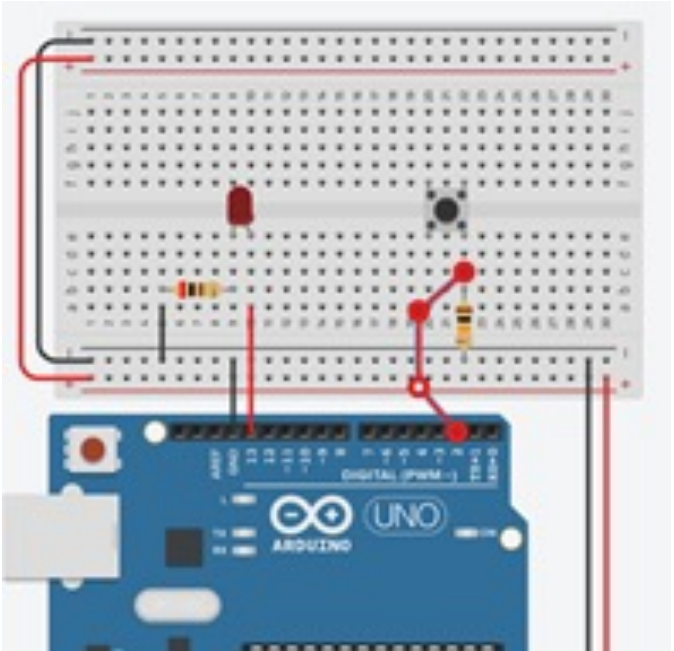
Set up

1. Select the color red for the following wires as seen in the image.

2. One end starts at Port 13 on the Arduino. The other end connects to Row 10, Column a.

3. The second wire connects from Port 2 on the Arduino to Row 22, Column c.

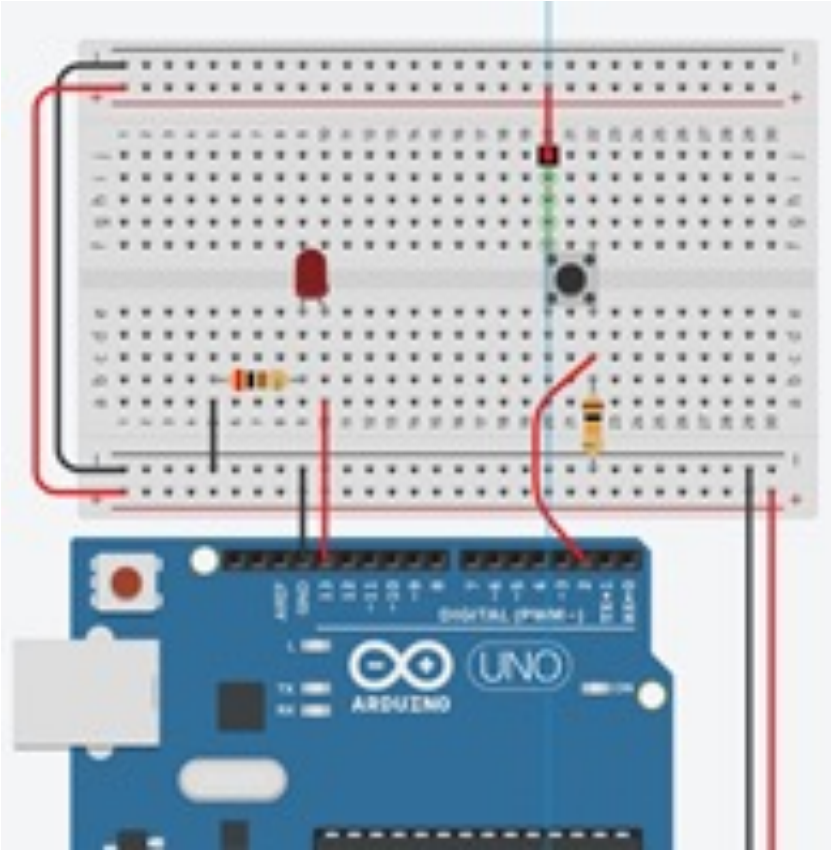




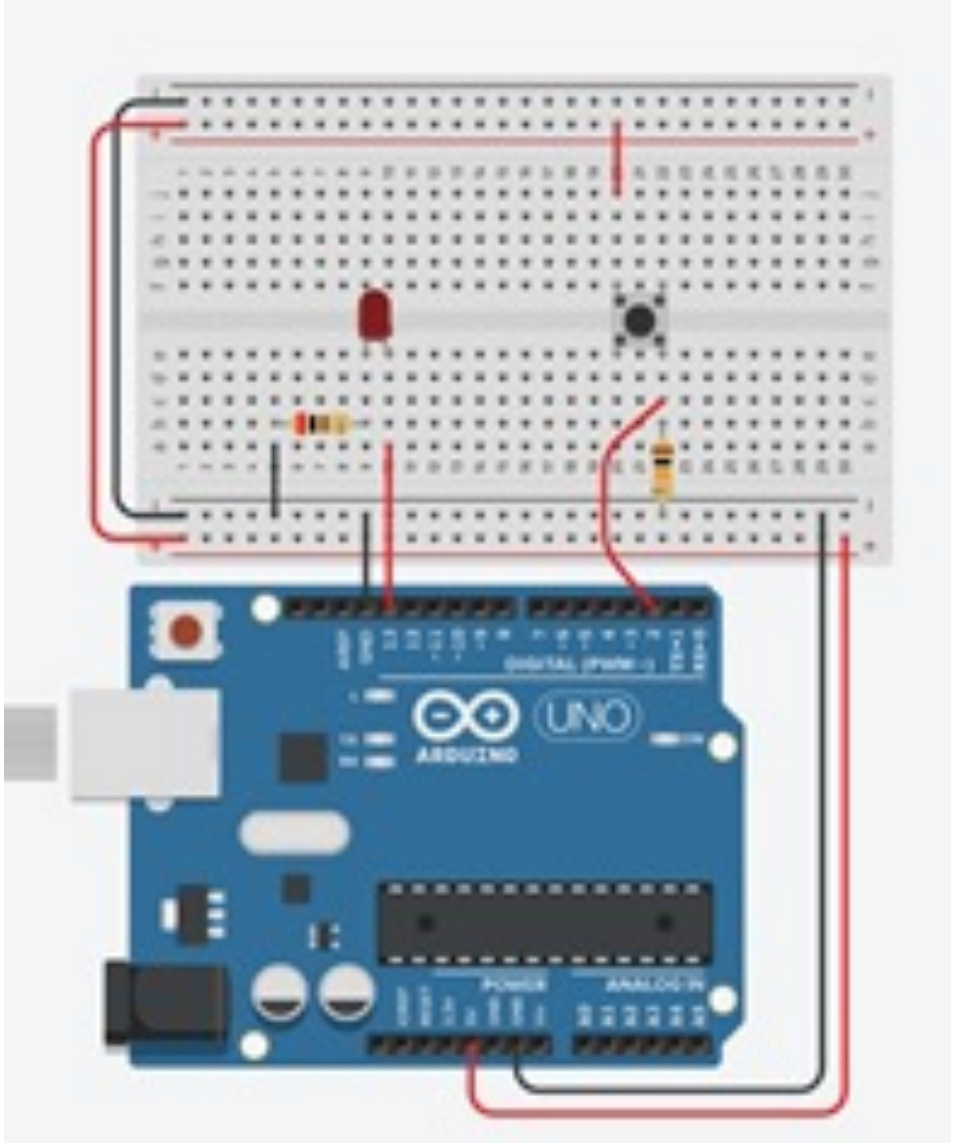
Wiring a LED with Arduino

Set up

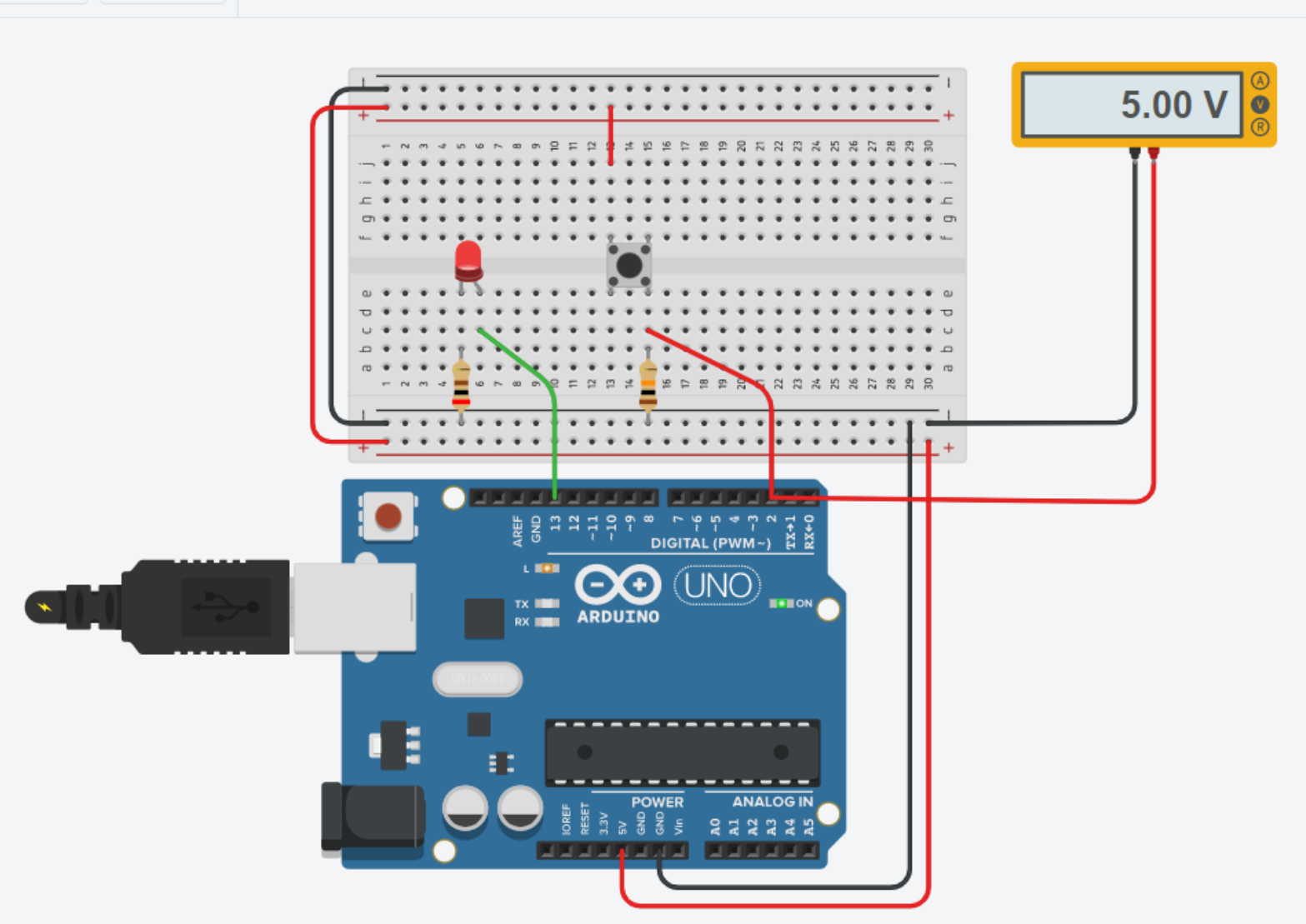
1. One end starts at Positive Bus aligned with Row 20. The other end connects to Row 20, Column j.



Final Product

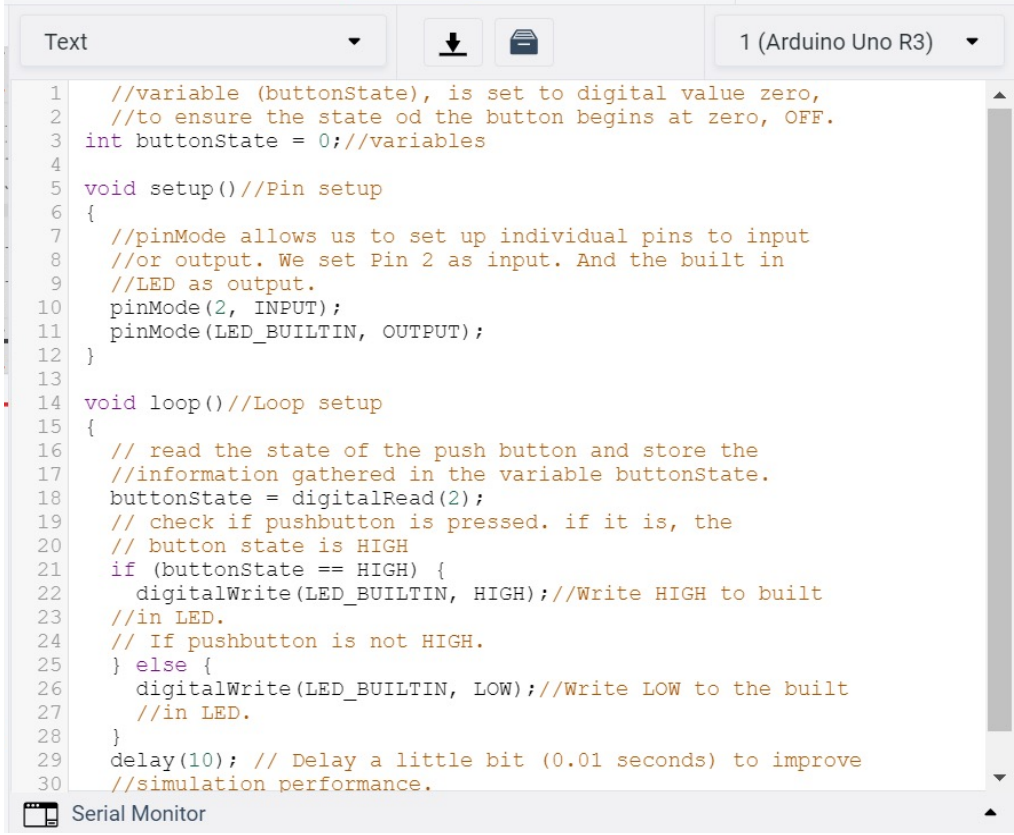


Analyzing your Switch



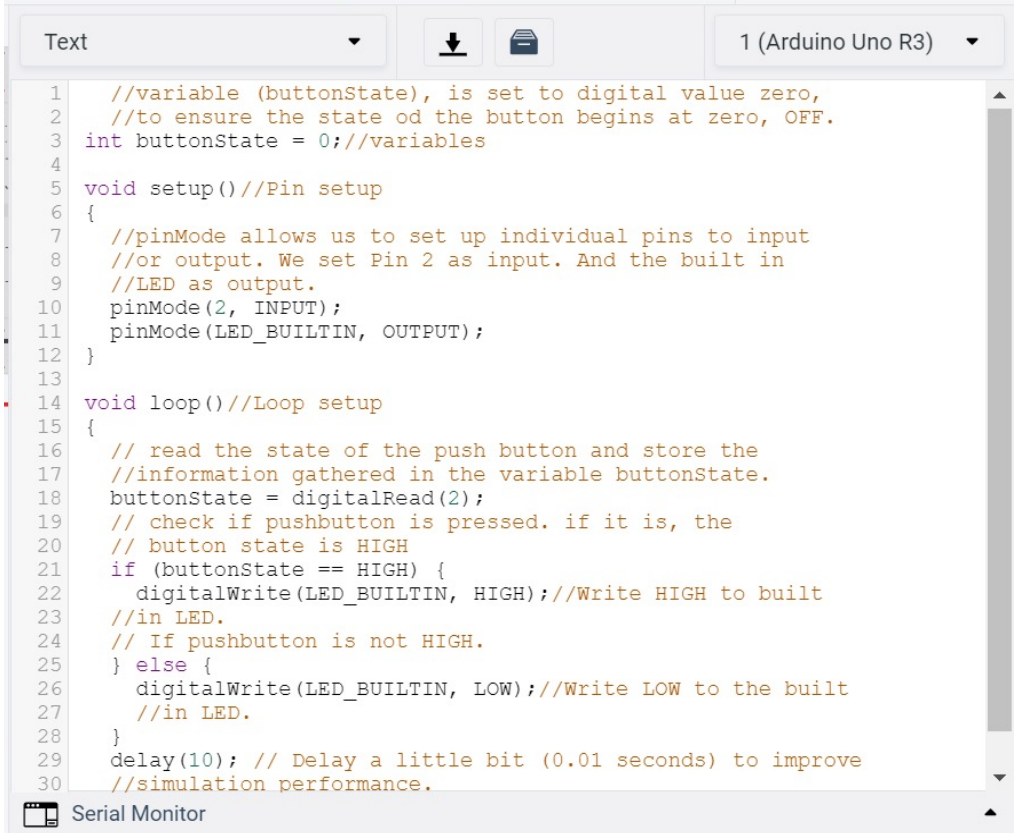
C++ First Half of Code

```
//C++
  //variable (buttonState), is set to digital value
zero,
  //to ensure the state od the button begins at
zero, OFF.
int buttonState = 0;//variables

void setup()//Pin setup
{
  //pinMode allows us to set up individual pins to
input
  //or output. We set Pin 2 as input. And the built
in
  //LED as output.
  pinMode(2, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}
```

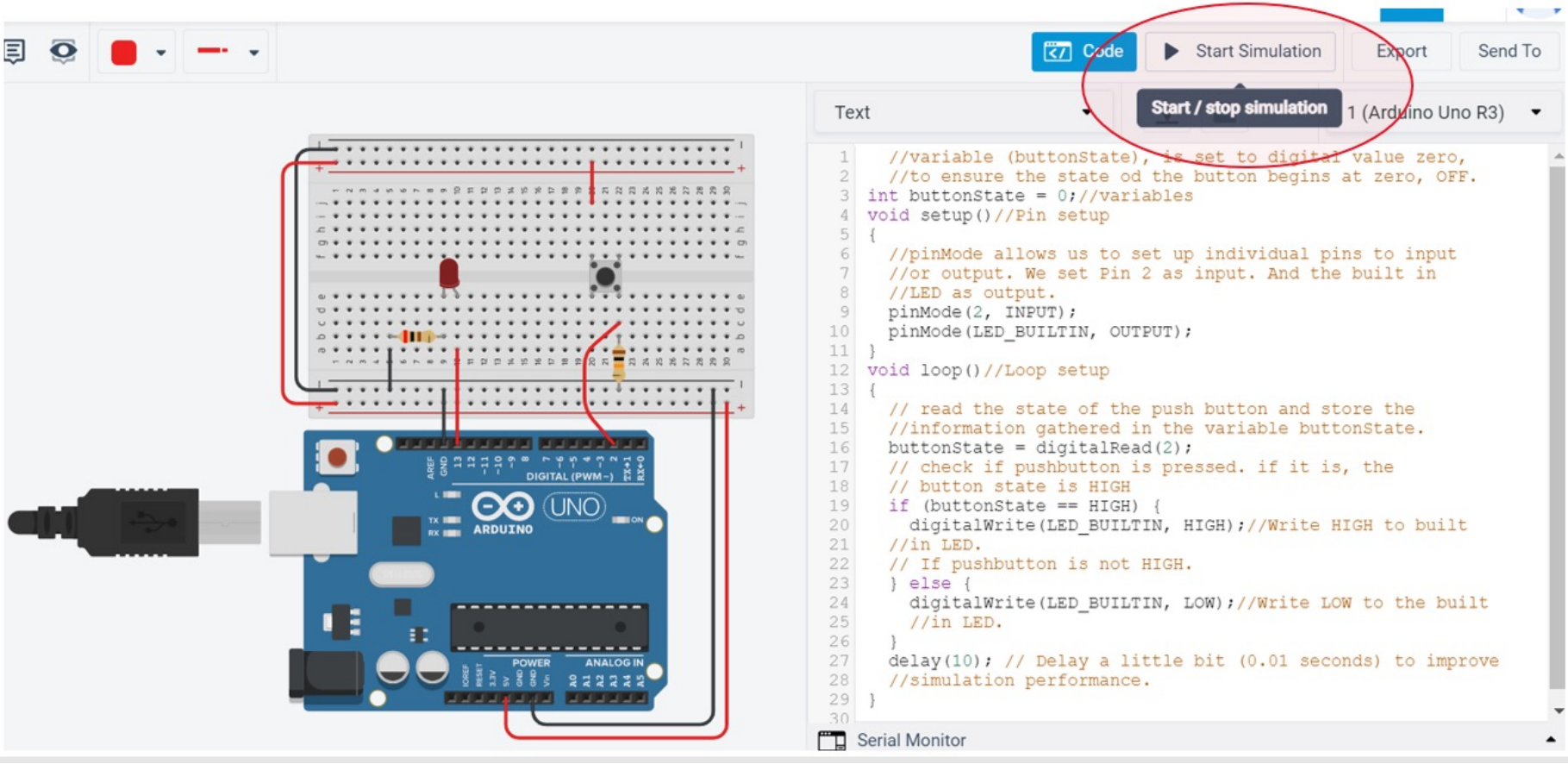


C++ Second Half of Code

```
void loop()//Loop setup
{
  // read the state of the push button and
store the
  //information gathered in the
variable buttonState.
  buttonState = digitalRead(2);
  // check if pushbutton is pressed. if it is, the
  // button state is HIGH
  if (buttonState == HIGH) {
    digitalWrite(LED_BUILTIN,
HIGH);//Write HIGH to built
  //in LED.
  // If pushbutton is not HIGH.
  } else {
    digitalWrite(LED_BUILTIN, LOW);//Write
LOW to the built
    //in LED.
  }
  delay(10); // Delay a little bit (0.01 seconds)
to improve
  //simulation performance.
}
```



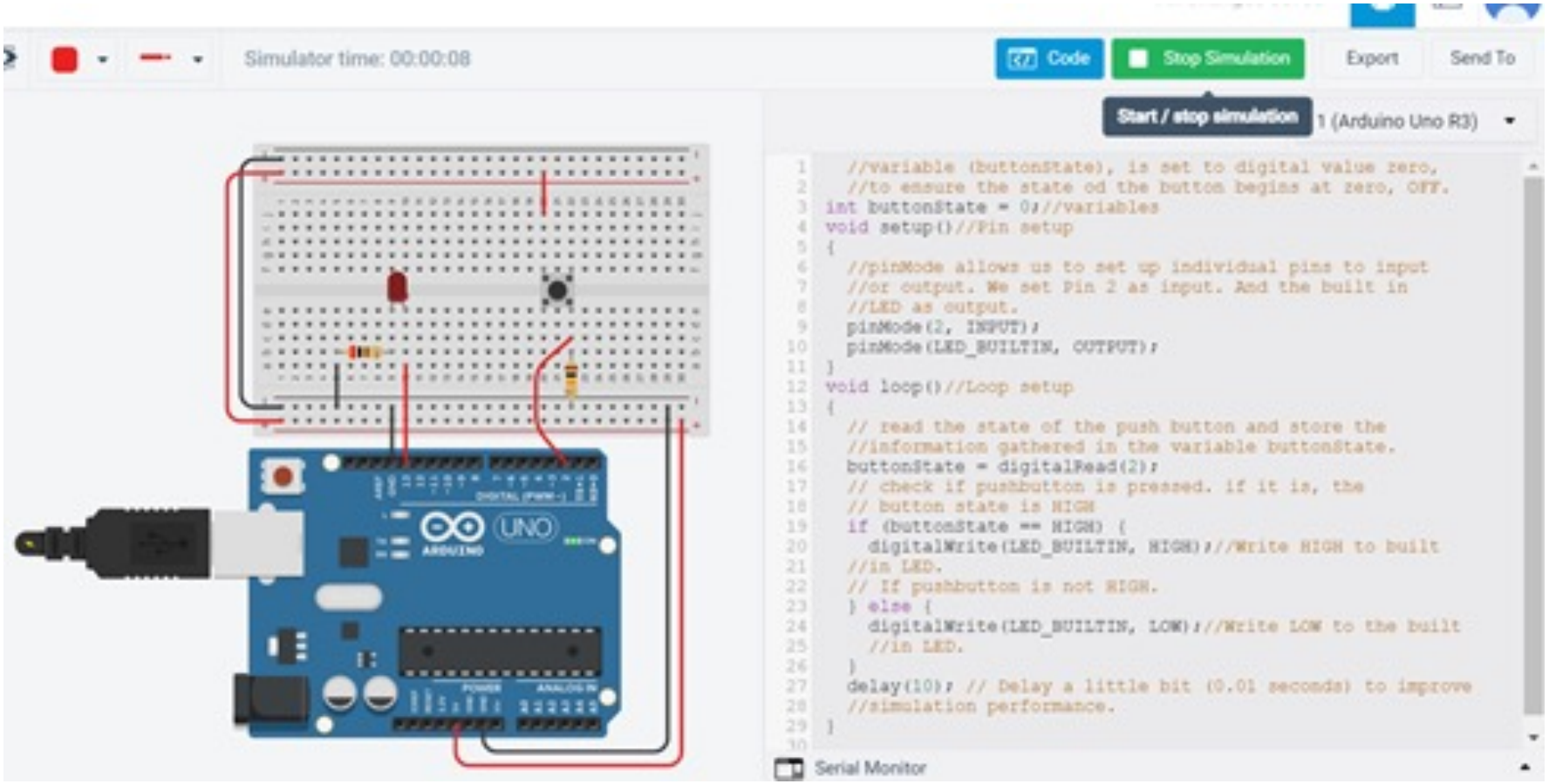
Begin Simulation



Simulation and Review

- Click on the Start Simulation to review your results. Your push button should turn on the LED, and turn off when you let go.
- Can you figure out a way to swap the logic in your circuit, so that the LED is lit by default and goes dark when the button is pressed?

Stop Simulation

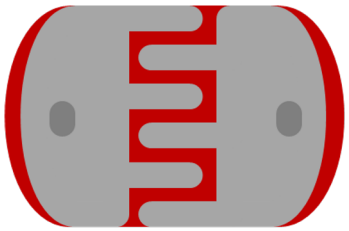


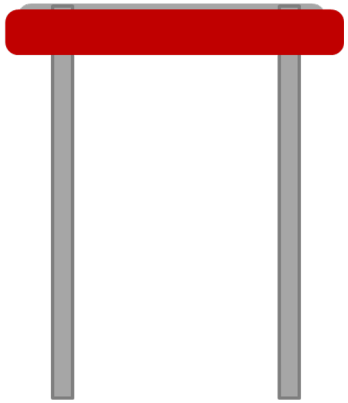
Virtual Activity - 3

Photoresistor

Photoresistor (LDR)

A photoresistor or **LDR** (**l**ight **d**ependent **r**esistor) is a resistor whose resistance depends on light intensity

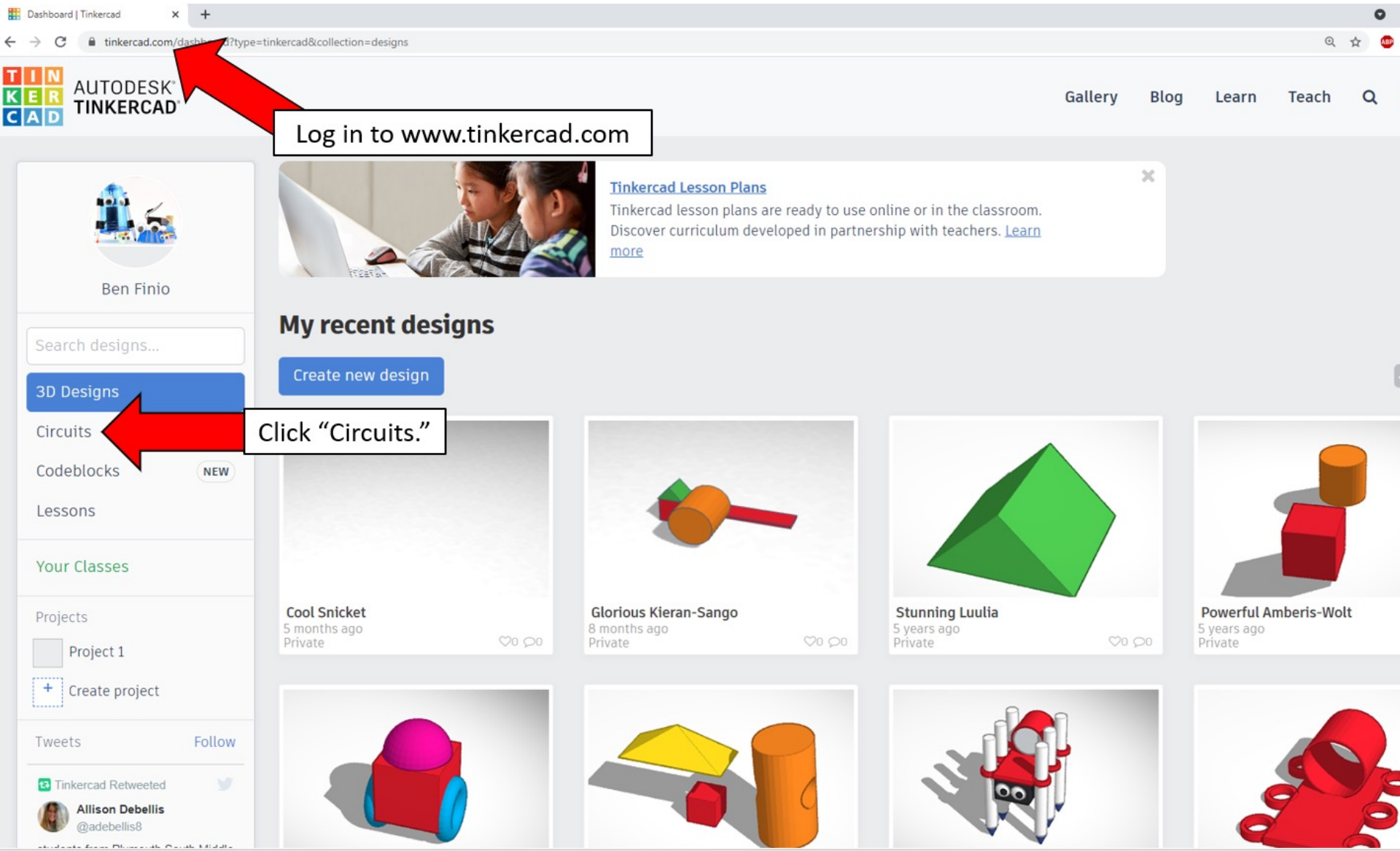




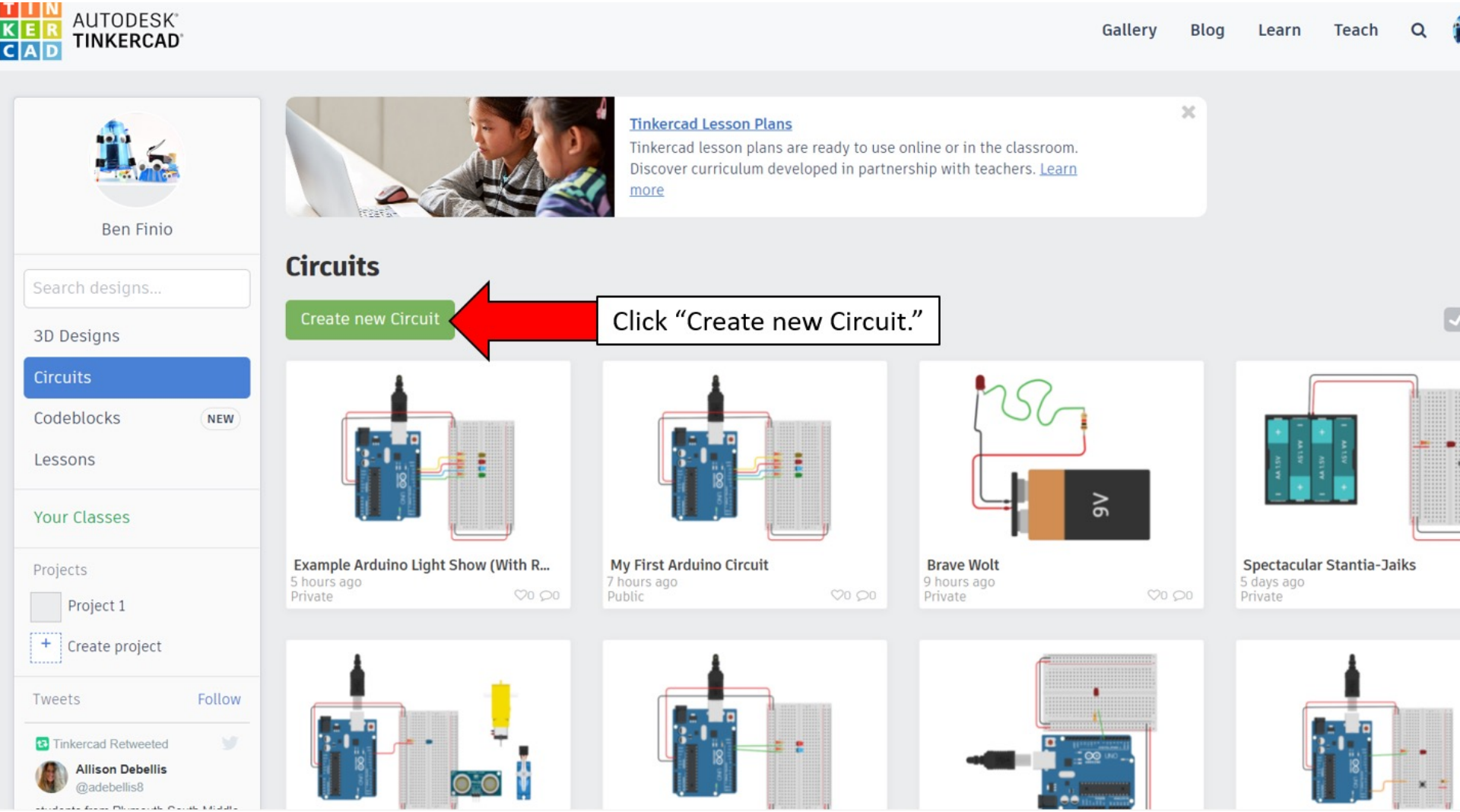
An LDR can be used as a simple, **analog sensor**

The orientation of an LDR does not matter

Starting a new circuit



Starting a new circuit

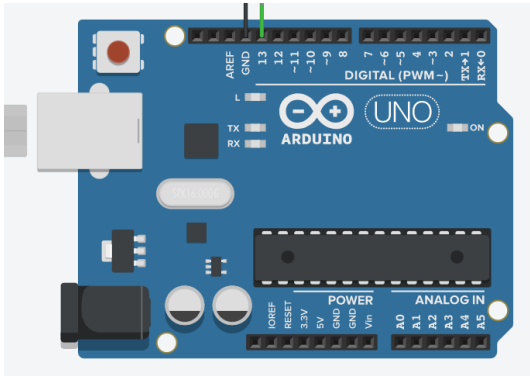


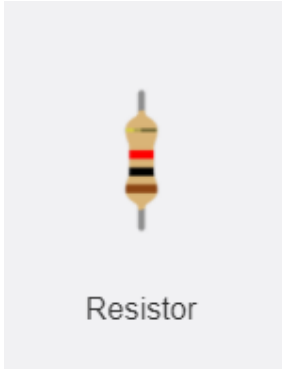
Components

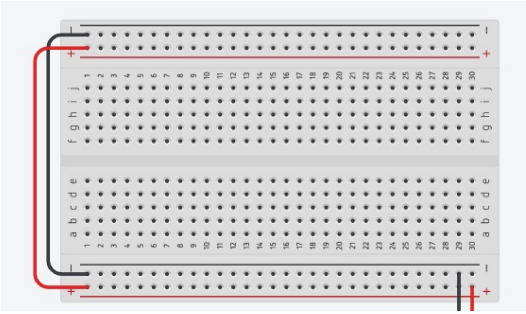
We will introduce how to blink the on-board LED and how to blink an external LED using an LDR.

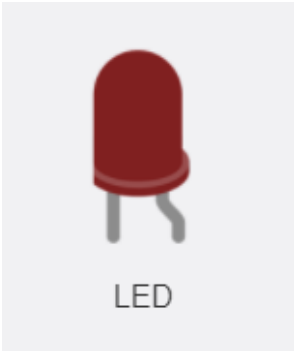
Components

- Arduino
- LDR / photoresistor
- Two resistors
- Breadboard
- LED



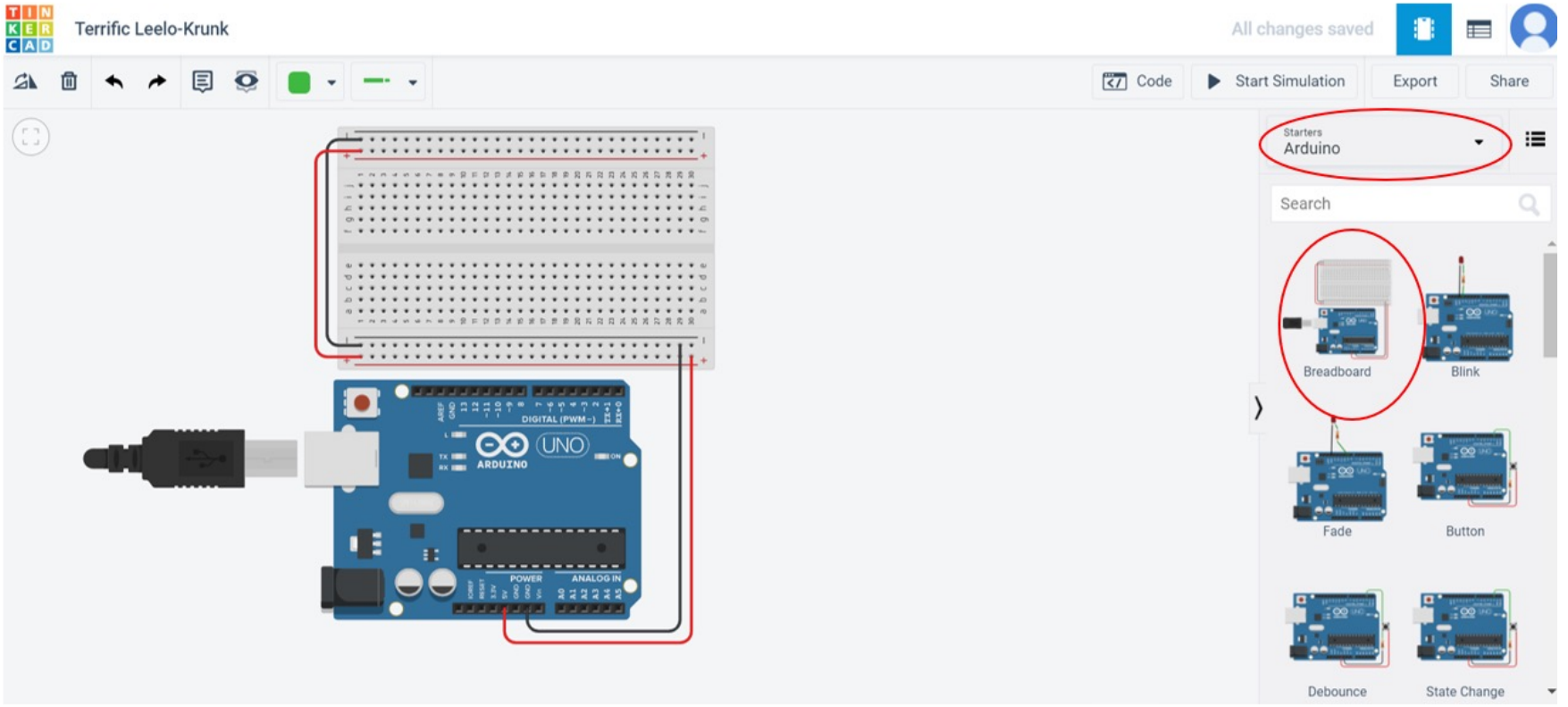






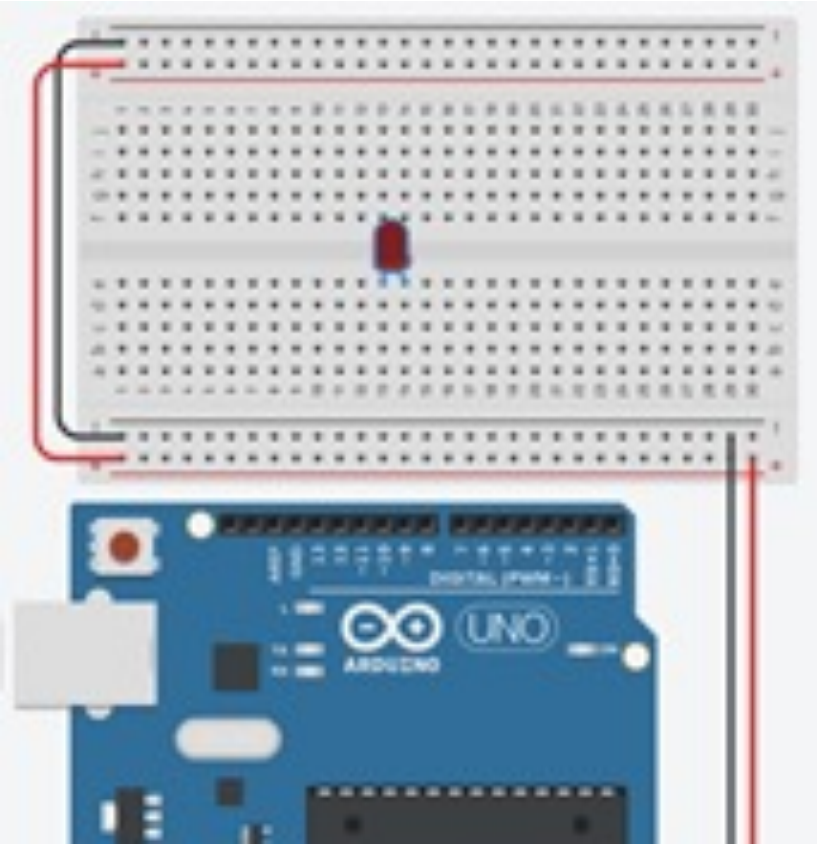
Setting up our Workplace

1. Under Starters/Components: click Arduino.
2. Click and drag on the first component, place it on the workplace and let go.
3. Repeat step two for the following components: two resistors, one LDR, and one LED.



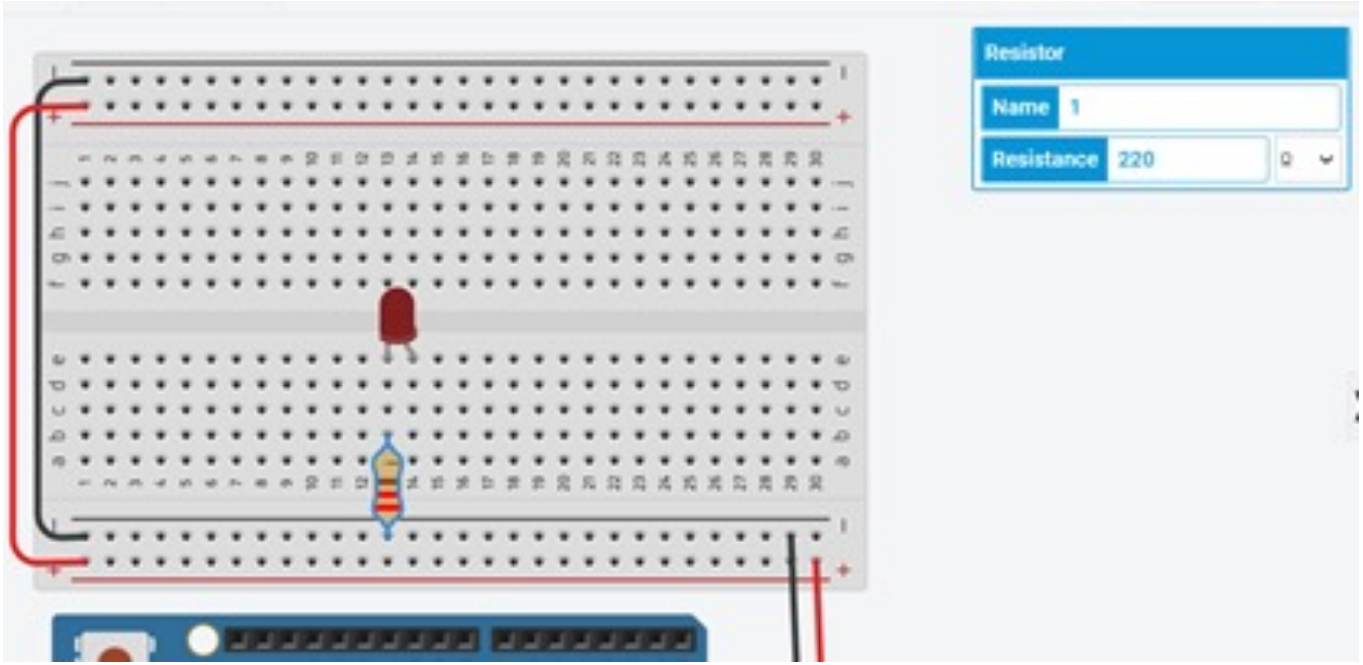
Setting up our Workplace

1. Under Starters/Components: click LED.
2. Click and drag on the first component, place it on the workplace and let go.
3. Cathode will be on Row 13, Column e. Anode will be on Row 14, Column e.



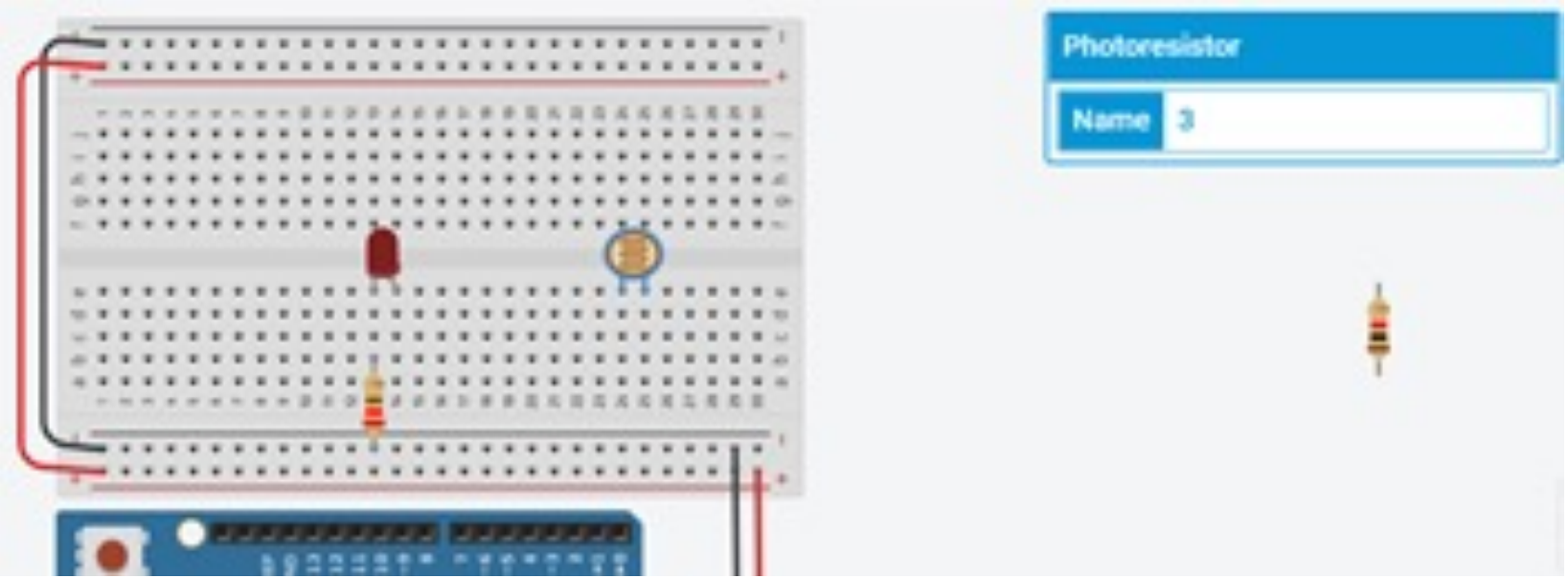
Setting up our Workplace

1. Under Starters/Components: click Resistor.
2. Click and drag on the first component, place it on the workplace and let go.
3. One end is on Row 13, Column b. The other end is aligned to Row 13 on the Ground Bus.
4. Set this Resistor to 220 Ohms



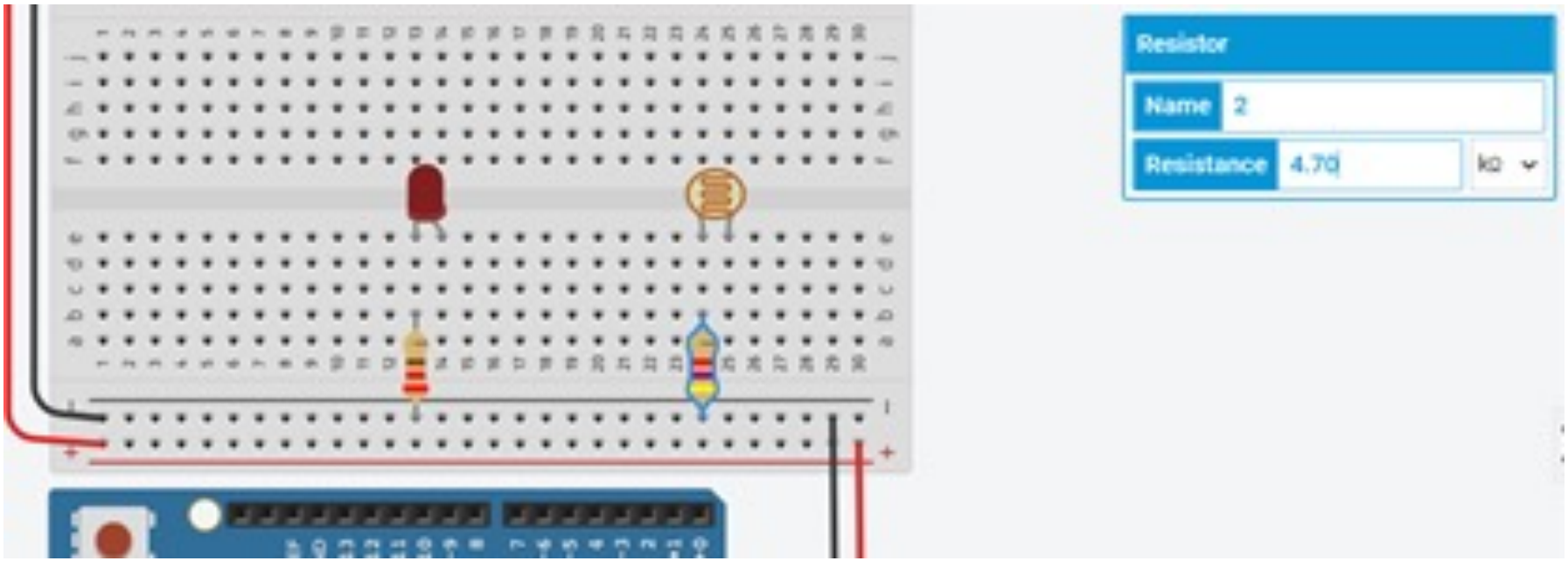
Setting up our Workplace

1. Under Starters/Components: click Photoresistor.
2. Click and drag on the component, place it on the workplace and let go.
3. One end is on Row 24, Column e. The other end is on Row 25, Column e.
4. You can rotate the resistor by press "R" on your keyboard.



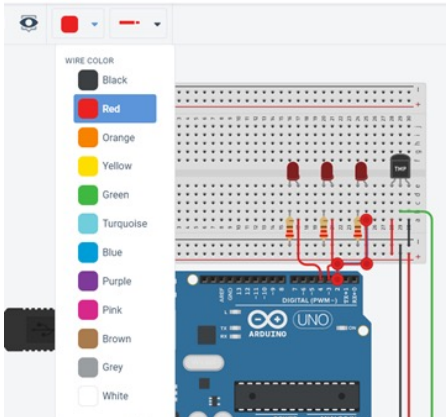
Setting up our Workplace

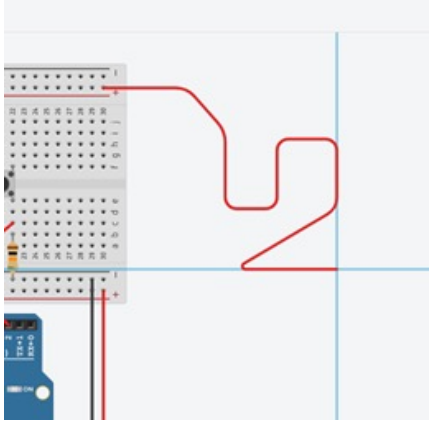
1. Under Starters/Components: click Resistor.
2. Click and drag on the first component, place it on the workplace and let go.
3. One end is on Row 24, Column b. The other end is aligned to Row 24 on the Ground Bus.
4. Set this Resistor to 4.70 kOhms

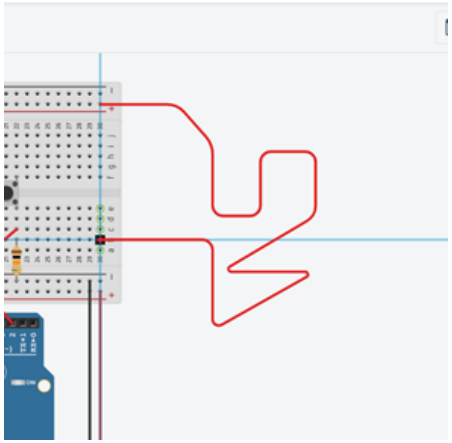


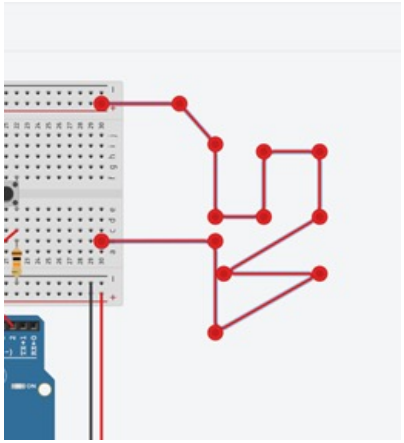
Wiring Orientation

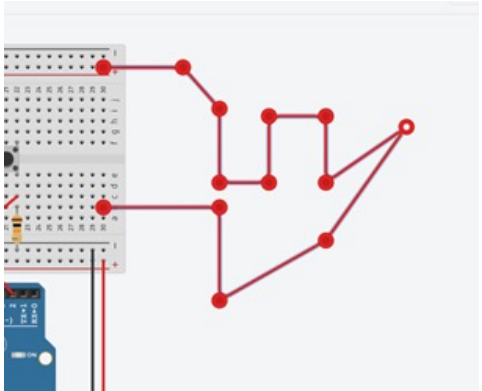
1. Choose a color for the wire. Commonly, your ground will be black, and power will be red.
2. Next choose a start point and click once, you must choose a port or a location on the bread board. You will now have a wire that can be stretched almost at any angle.
3. In order to make a your wire "curve" at different points, you can right click, as long as you do not click when hoovering over a port or else it'll assumingly connect to that port.
4. You add as many turning points as you want, and they will appear after you connect to a port.
5. After which you can stretch the wire through its' turning points.





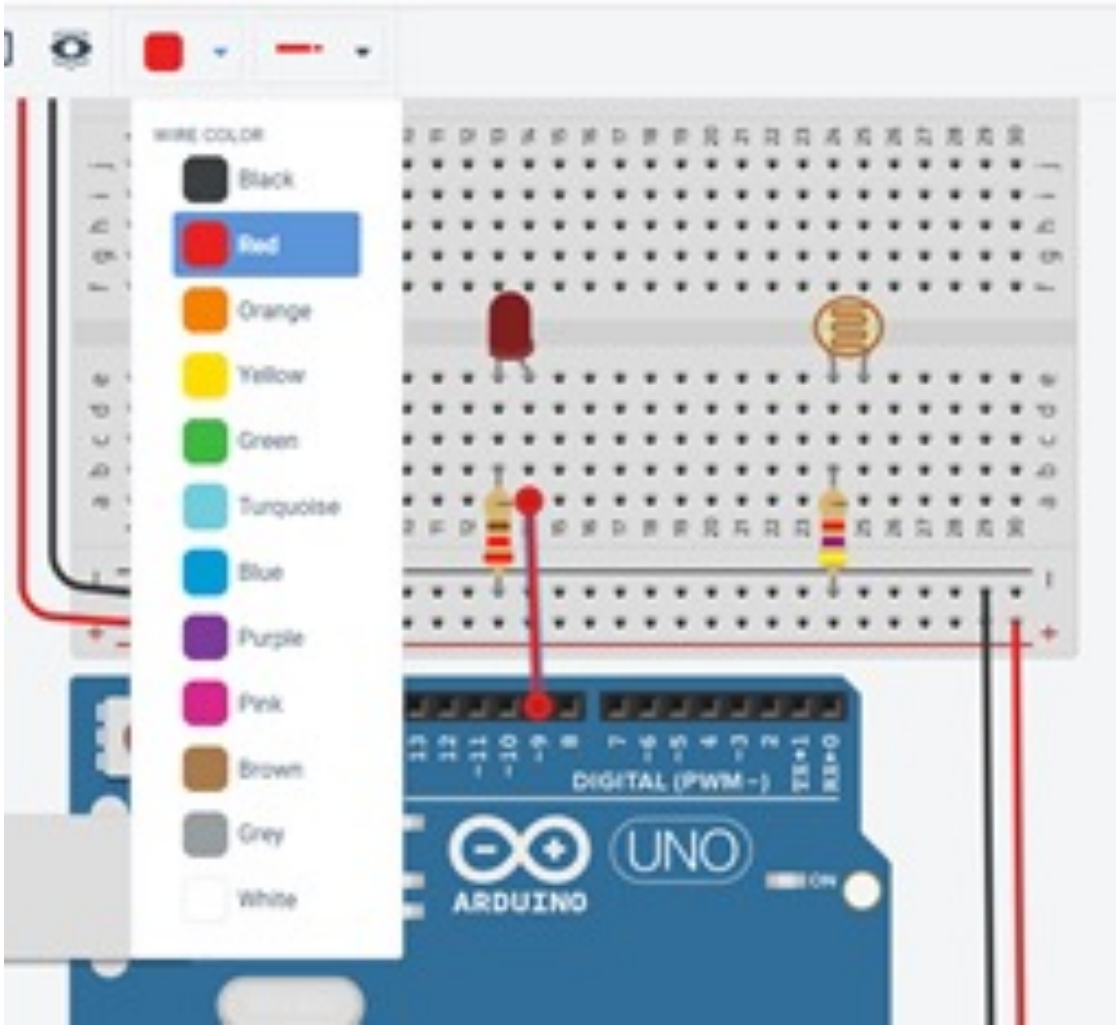






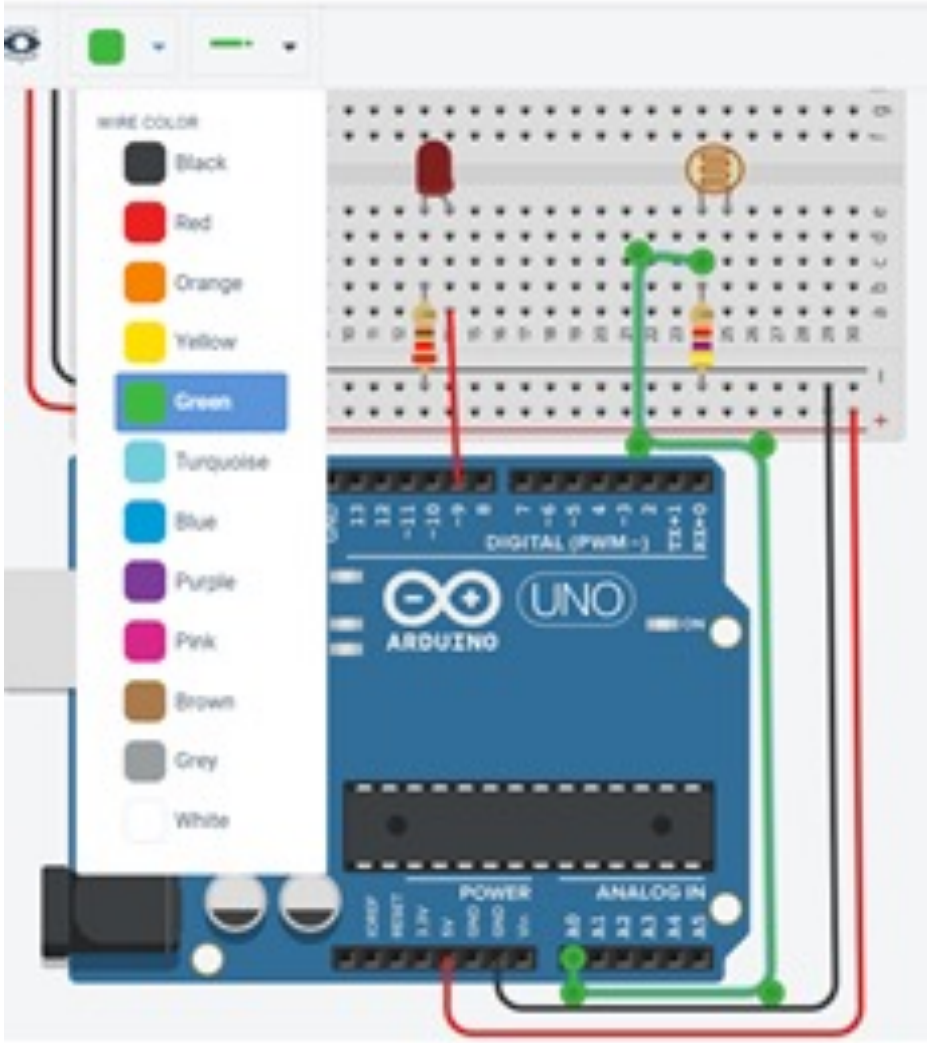
Wiring components

1. For this wire select the color red.
2. One end will begin at the Arduino Port 9. The other end will connect to Row 14, Column a.



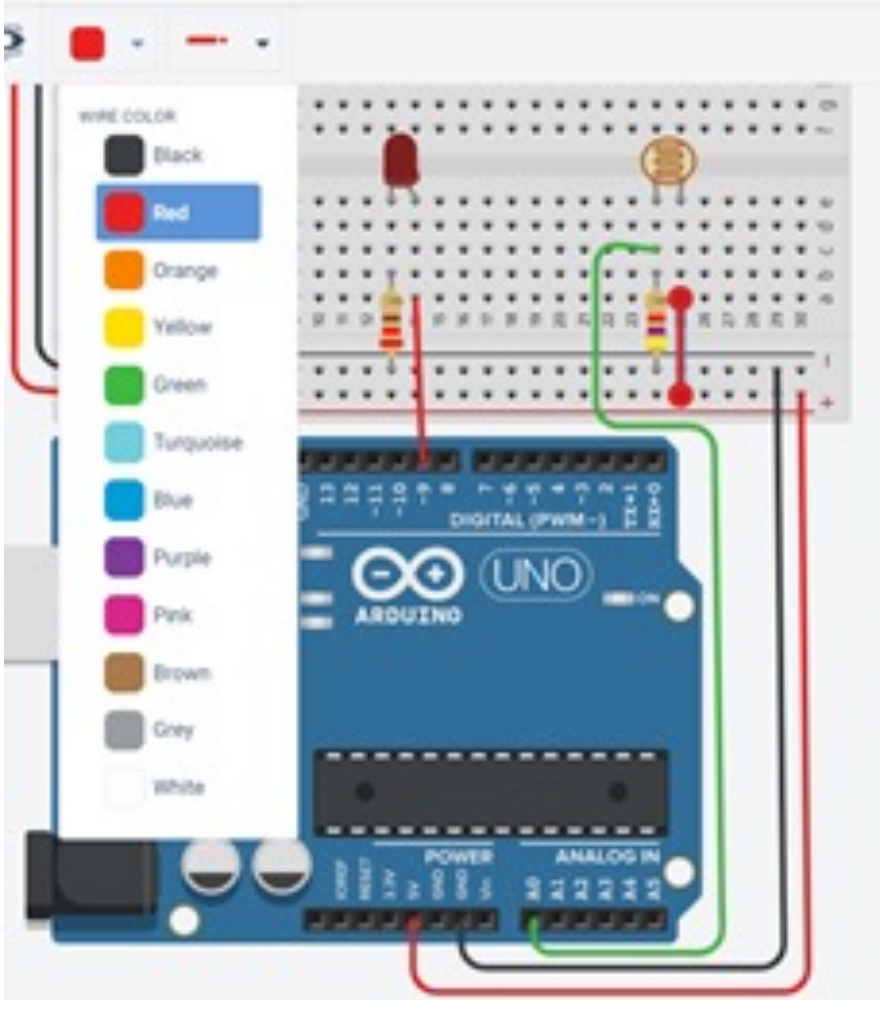
Wiring components

1. For this wire select the color green.
2. One end will begin at the Arduino Analog Port 0. The other end will connect to Row 24, Column c.

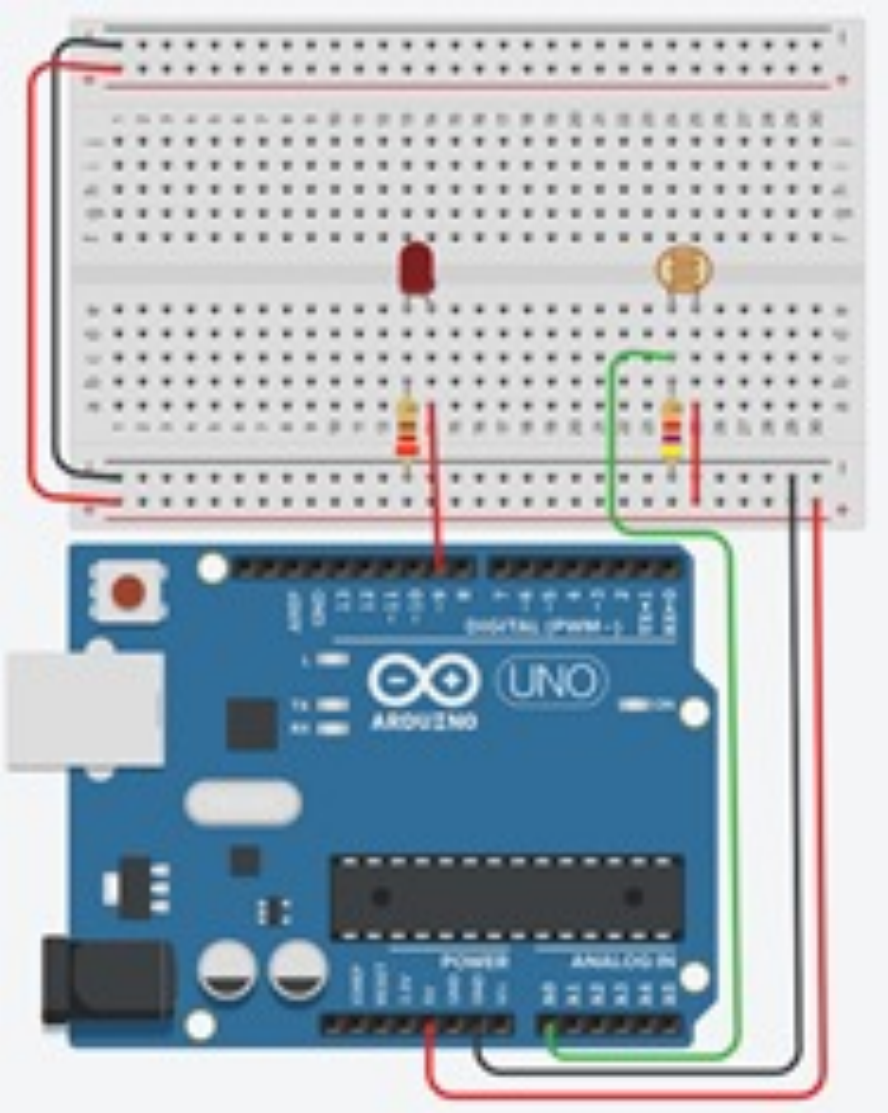


Wiring components

1. For this wire select the color red.
2. One end will begin at the positive bus. The other end will connect to Row 25, Column a.



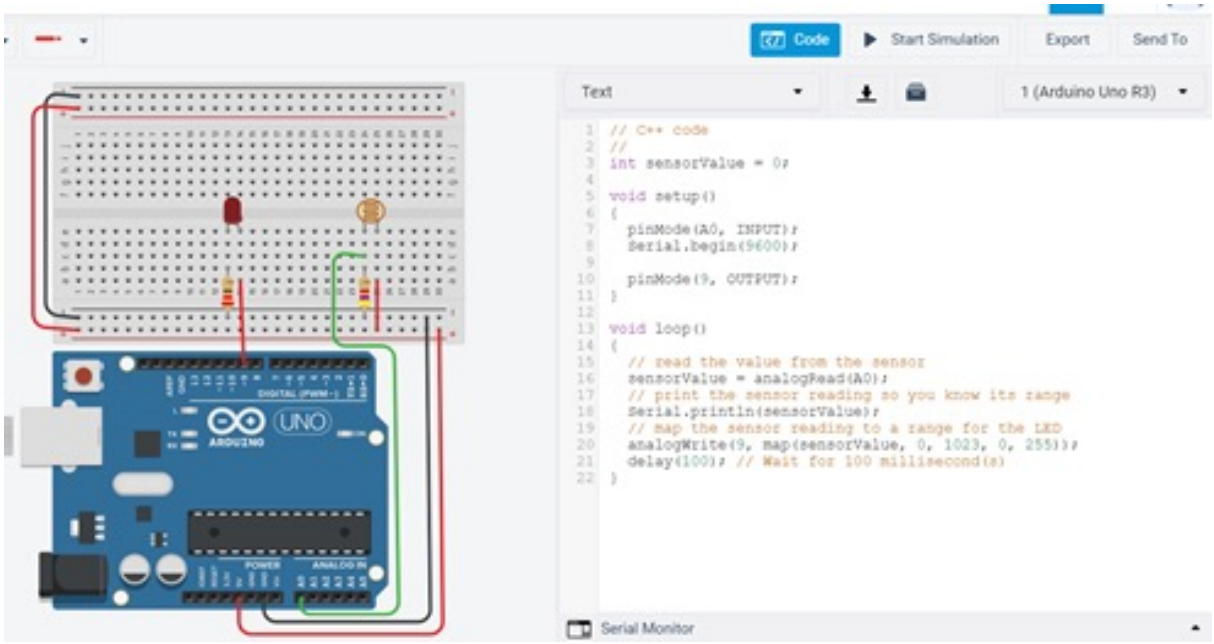
Final Product



Analog input with Arduino

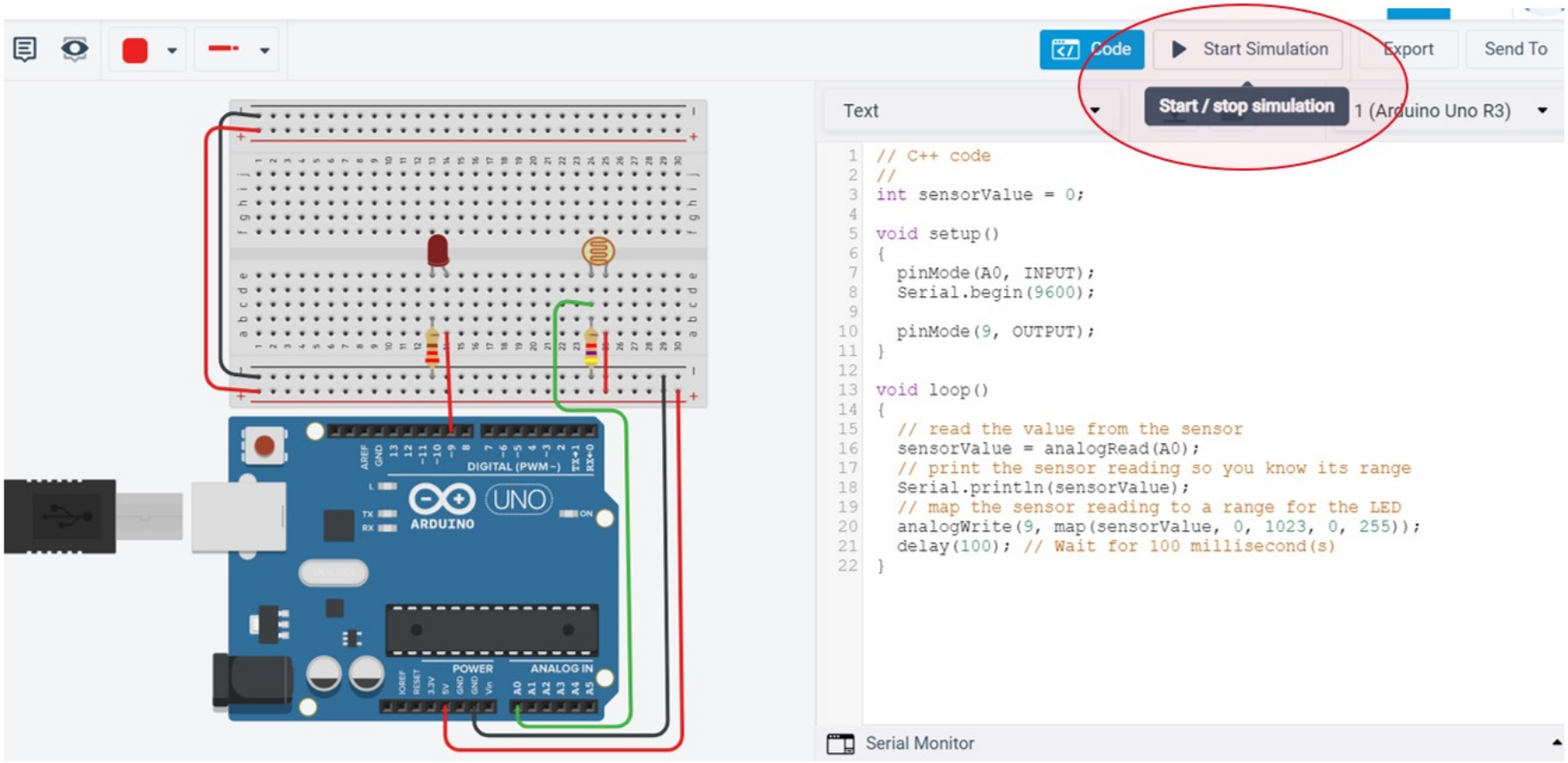
Code

➔ Copy the following code



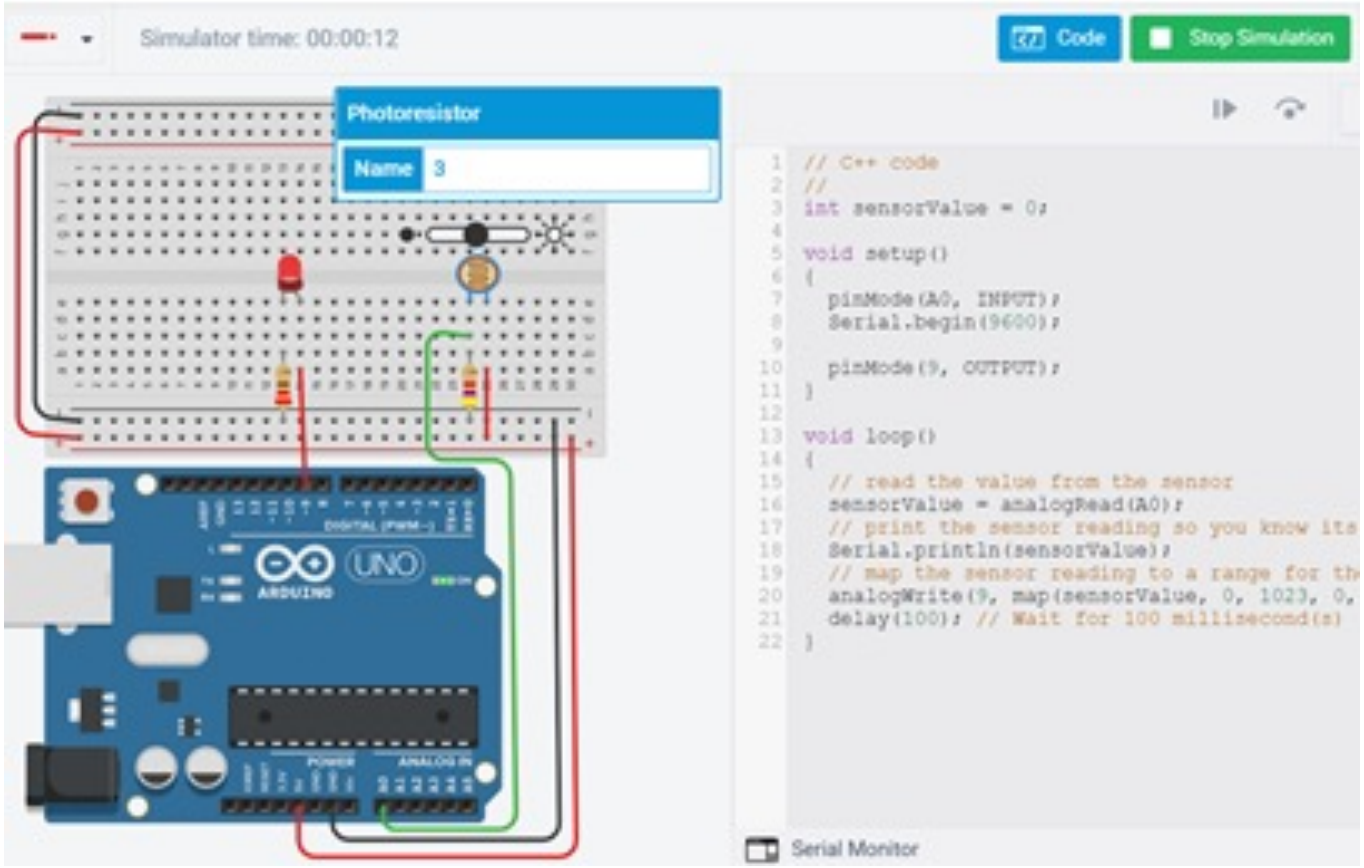

```
// C++ code
//
int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);

  pinMode(9, OUTPUT);
}

void loop()
{
  // read the value from the sensor
  sensorValue = analogRead(A0);
  // print the sensor reading so you know its range
  Serial.println(sensorValue);
  // map the sensor reading to a range for the LED
  analogWrite(9, map(sensorValue, 0, 1023, 0, 255));
  delay(100); // Wait for 100 millisecond(s)
}
```

Begin Simulation

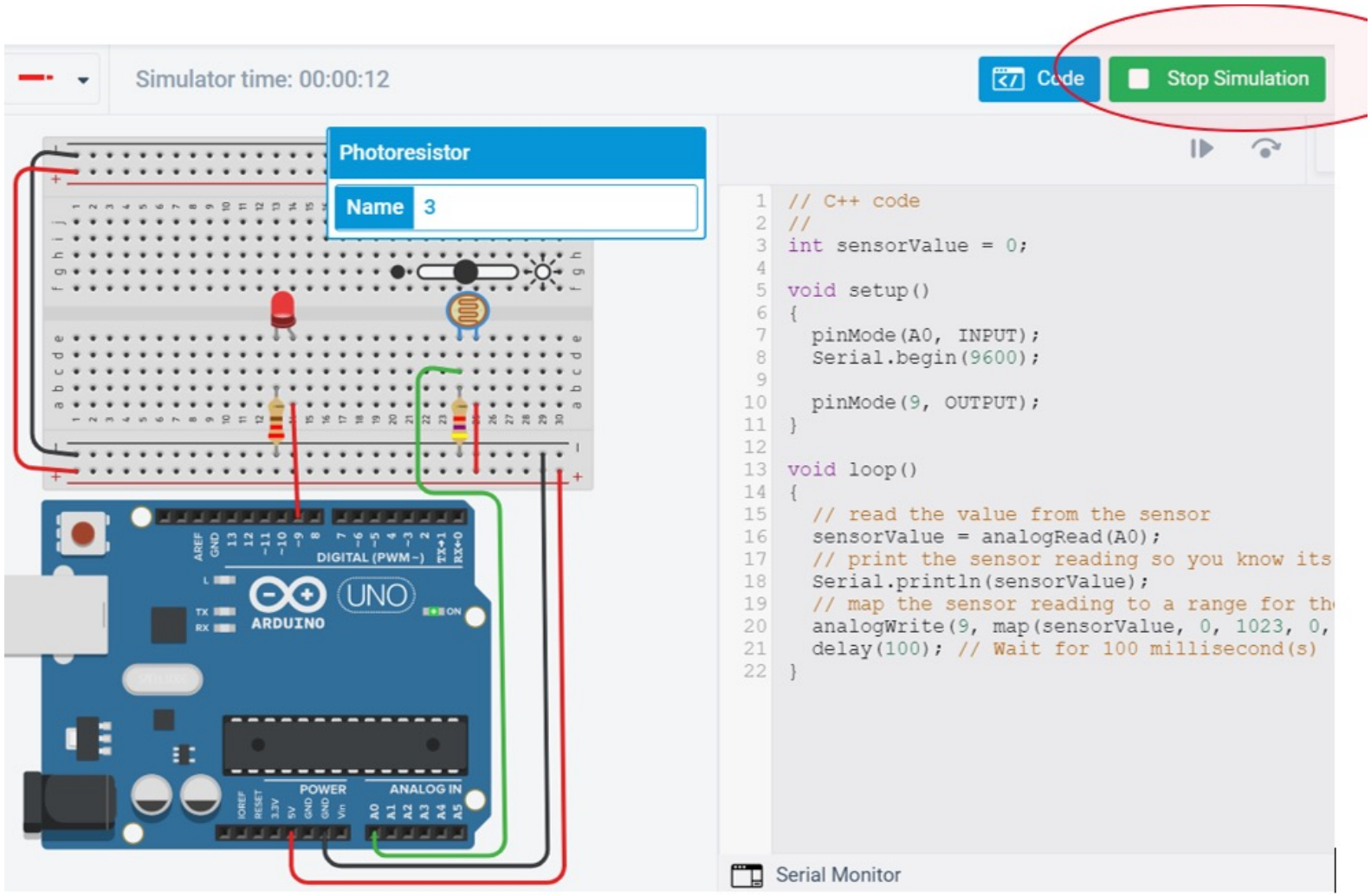


Simulation and Review

- Click on the Start Simulation to review your results. Your Photoresistor will display a dimming scroll for you to change from left to right. By scrolling from left to right you can see the LED turn on and off at your control.



Stop Simulation



Virtual Activity - 4

Temperature sensor

Temperature Sensor TMP36

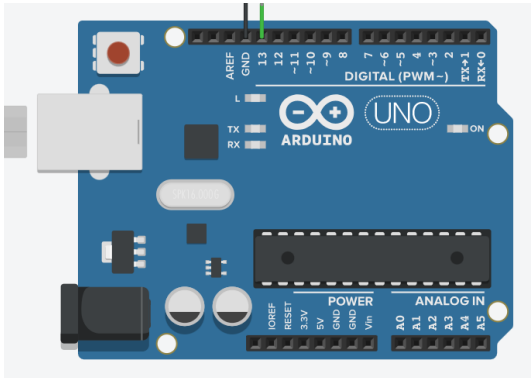
Temperature Sensor (Analog Input):

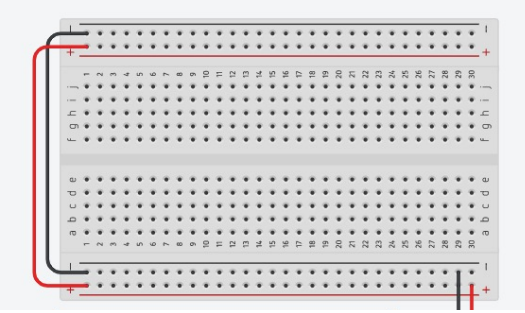
In this project, you will turn the Arduino into a thermometer. Use a temperature sensor to measure your skin temperature, and register the output with three LEDs. Even though the Arduino is a digital tool, it can interpret signals from an analog input, like the temperature sensor, using the built in Analog-to-Digital (ADC) converter, accessed through the analog pins.

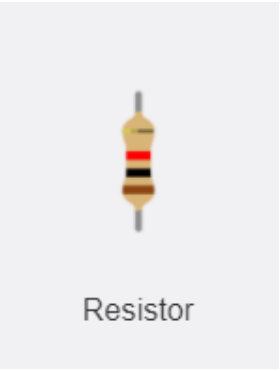
Components

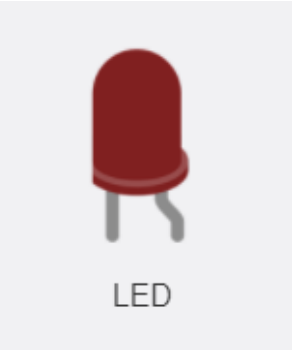
Components

- Arduino
- TMP36 Temperature sensor
- Breadboard
- Three Resistors
- Three LEDs

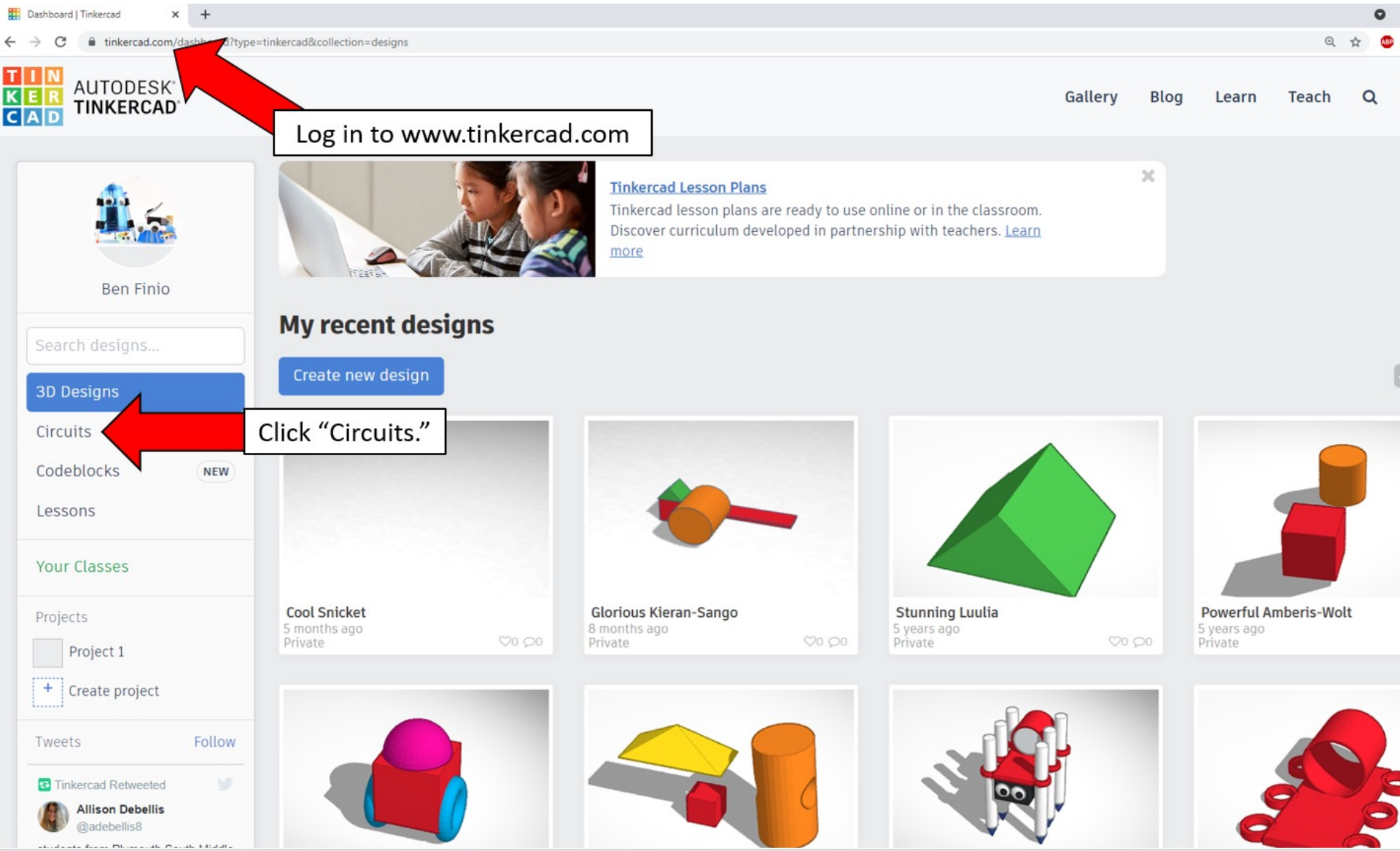




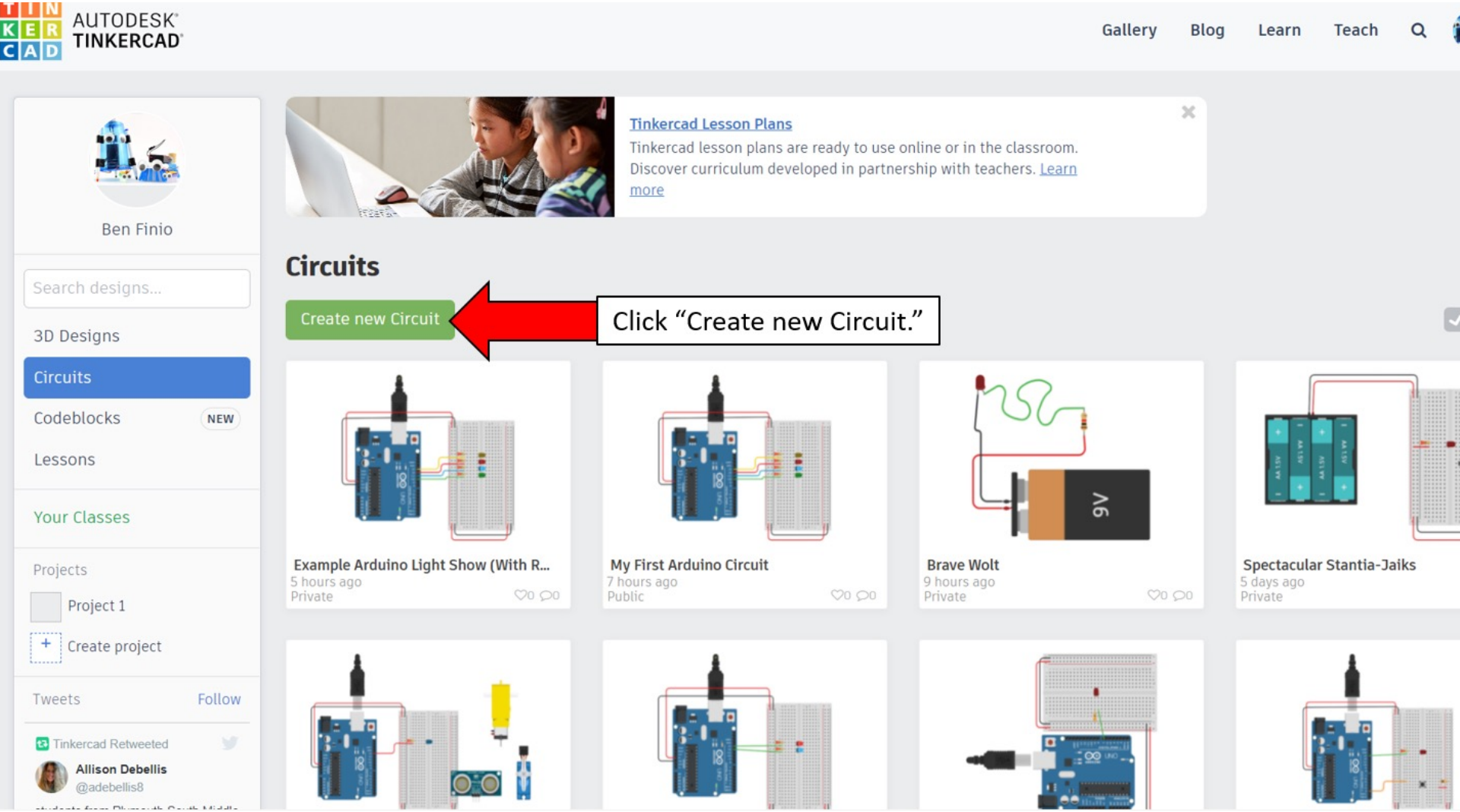




Starting a new circuit

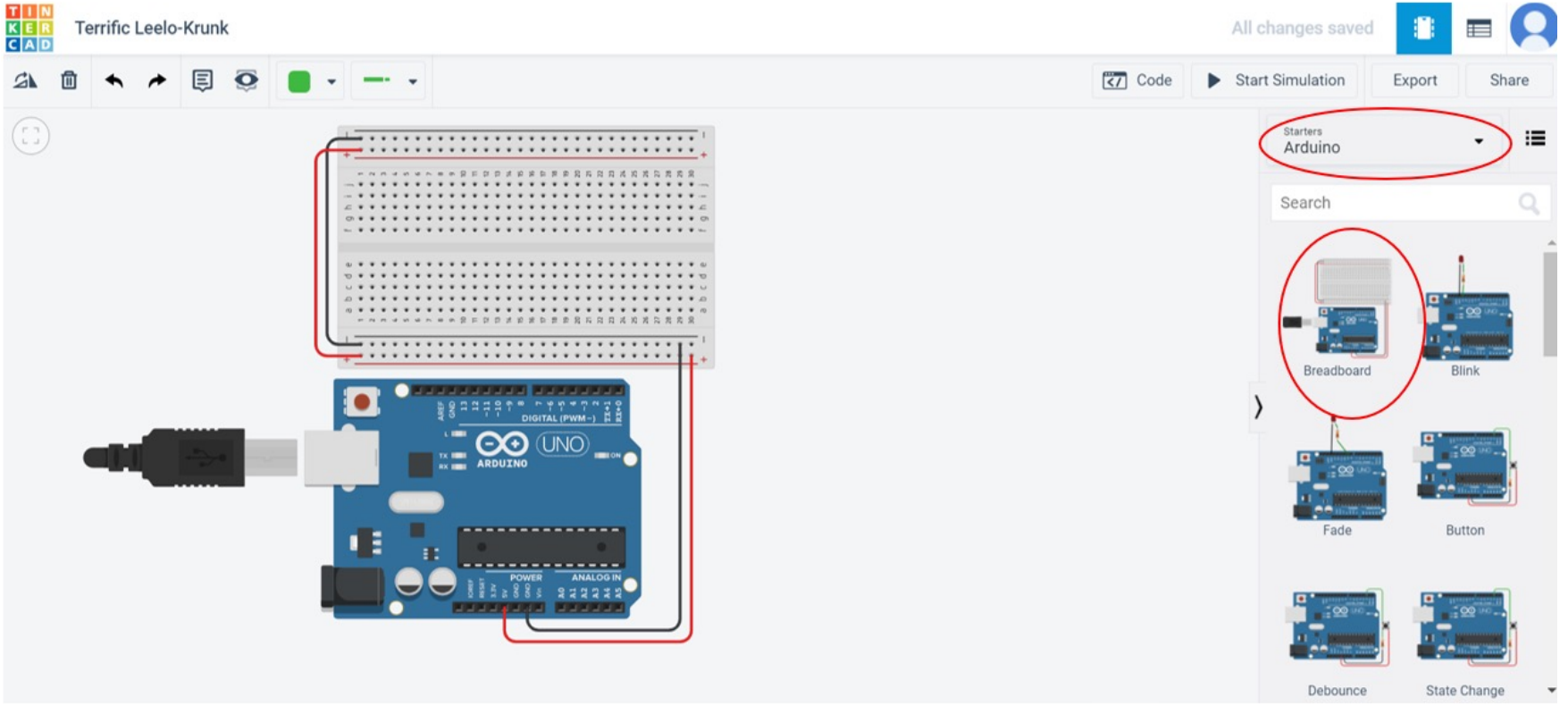


Starting a new circuit



Setting up our Workplace

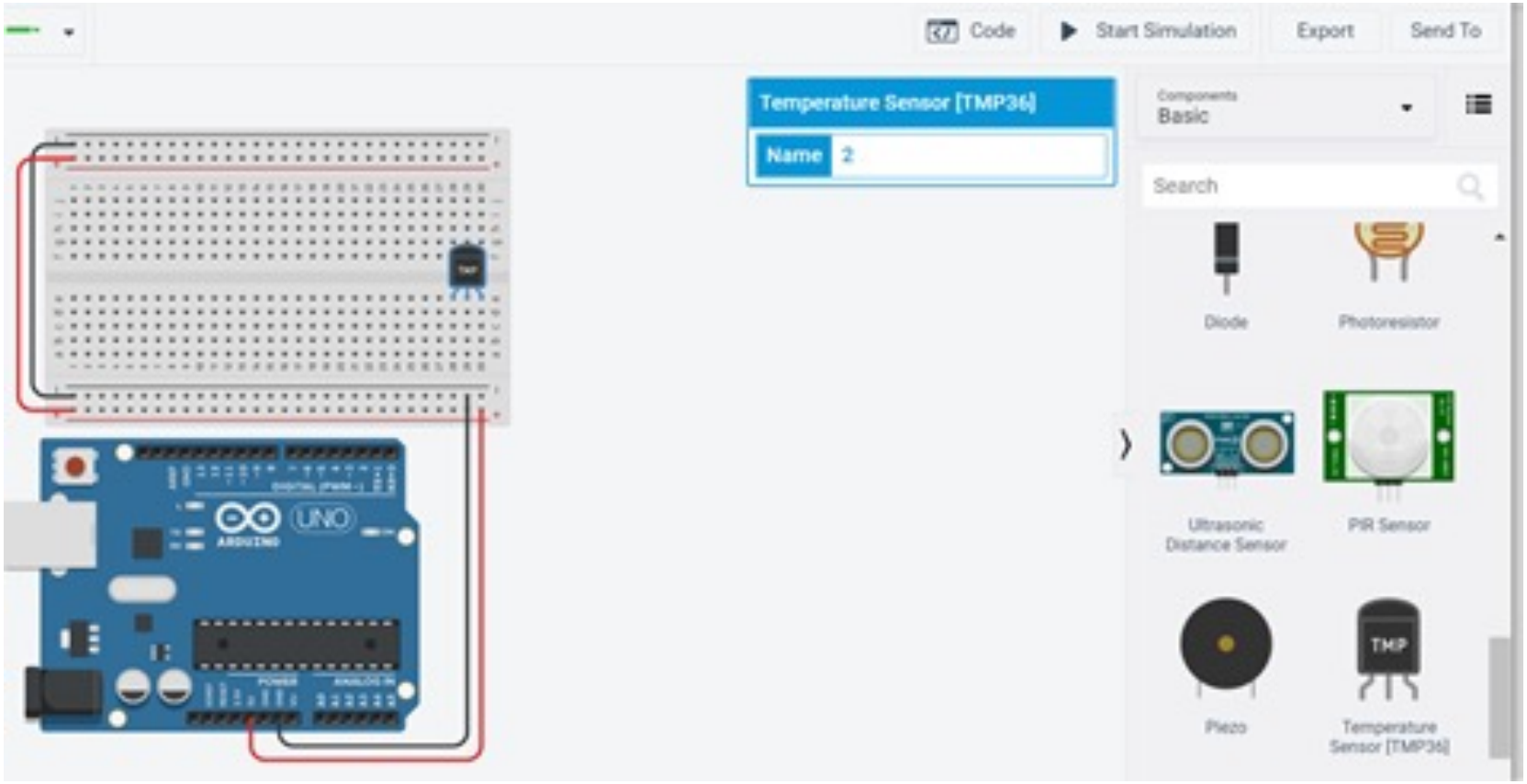
1. Under Starters/Components: click Arduino.
2. Click and drag on the first component, place it on the workplace and let go.
3. Repeat step two for the following components: two resistors, one LDR, and one LED.



Setting up our Workplace

1. Under Basic/Components: click Temperature Sensor.
2. Click and drag on the component, place it on the workplace and let go.

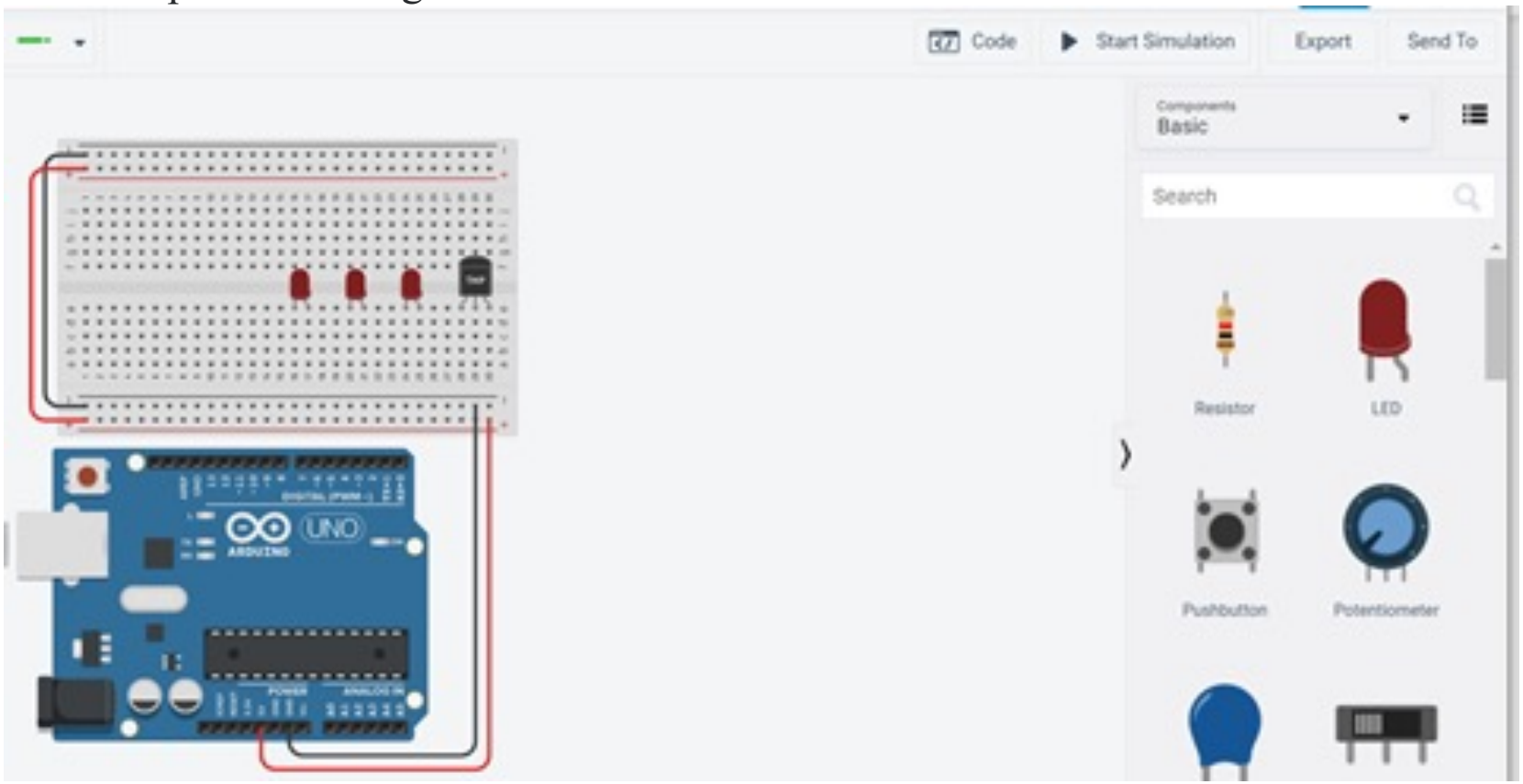
3. The following three terminal on the temperature sensor will be connected: Rows 28, 29, 30. Column e.



Setting up our Workplace

1. Under Basic/Components: click LED.
2. Click and drag on the component, place it on the workplace and let go.

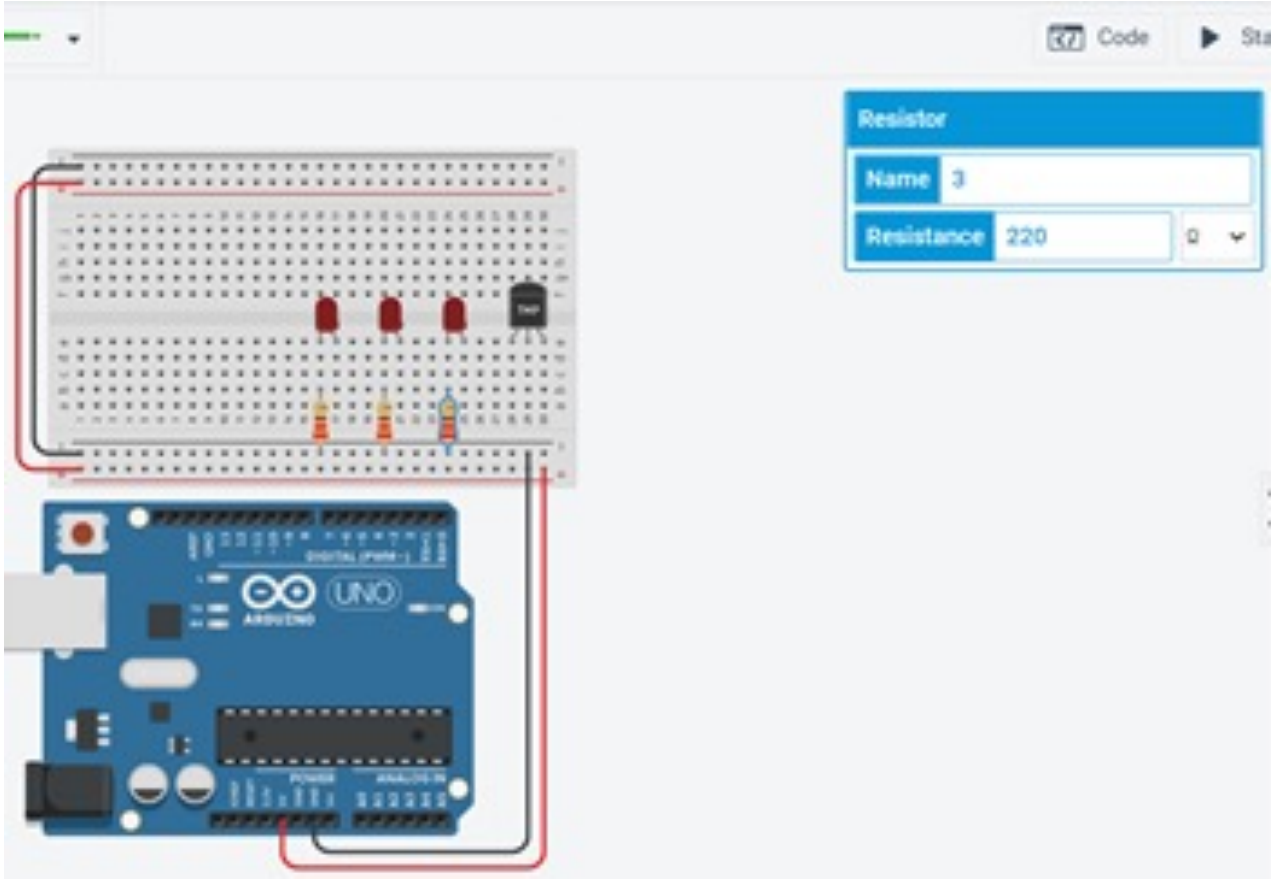
3. The following three LEDs will be connected to: Rows 16, 17, Column e. Rows 20, 21, Column e. Rows 24, 25, Column e.



Setting up our Workplace

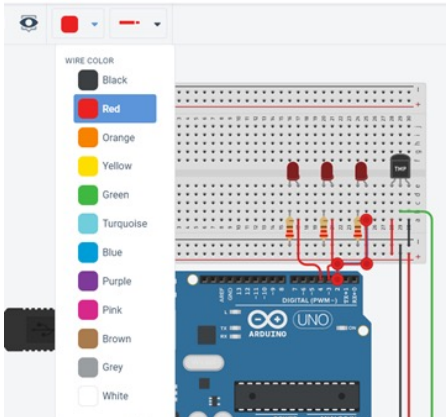
1. Under Basic/Components: click Resistor.
2. Click and drag on the component, place it on the workplace and let go.

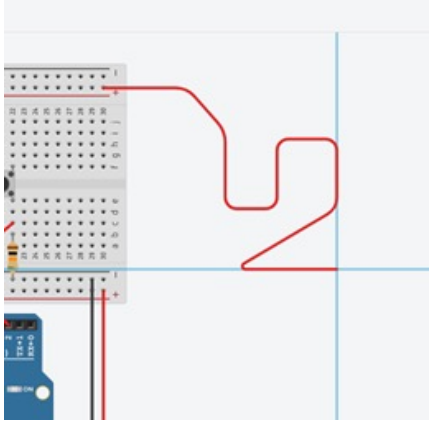
3. The following three Resistors will be connected to: (The top ends) Row 16, Column b. Row 20, Column b. Row 24, Column b. The bottom ends will connect to Ground bus, aligned with Rows 16, 20, 24. Set resistors to 220 Ohms.

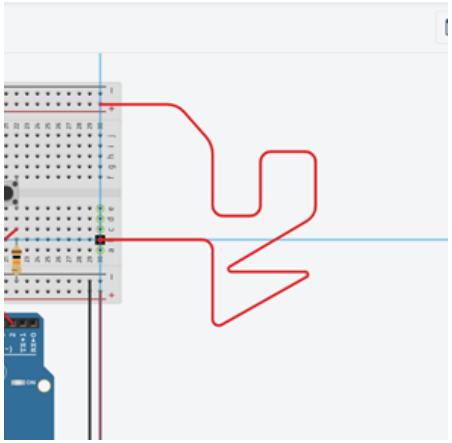


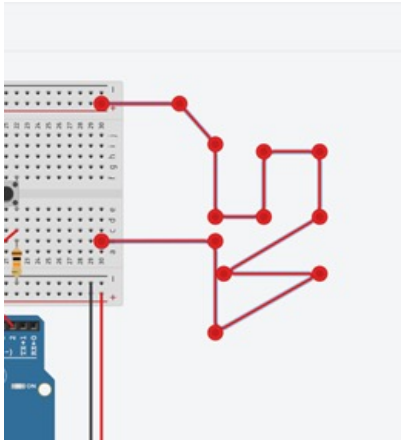
Wiring Orientation

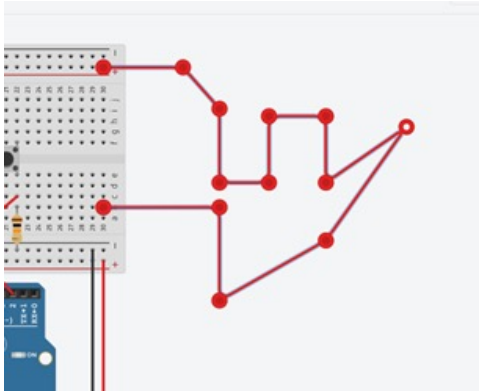
1. Choose a color for the wire. Commonly, your ground will be black, and power will be red.
2. Next choose a start point and click once, you must choose a port or a location on the bread board. You will now have a wire that can be stretched almost at any angle.
3. In order to make a your wire "curve" at different points, you can right click, as long as you do not click when hoovering over a port or else it'll assumingly connect to that port.
4. You add as many turning points as you want, and they will appear after you connect to a port.
5. After which you can stretch the wire through its' turning points.





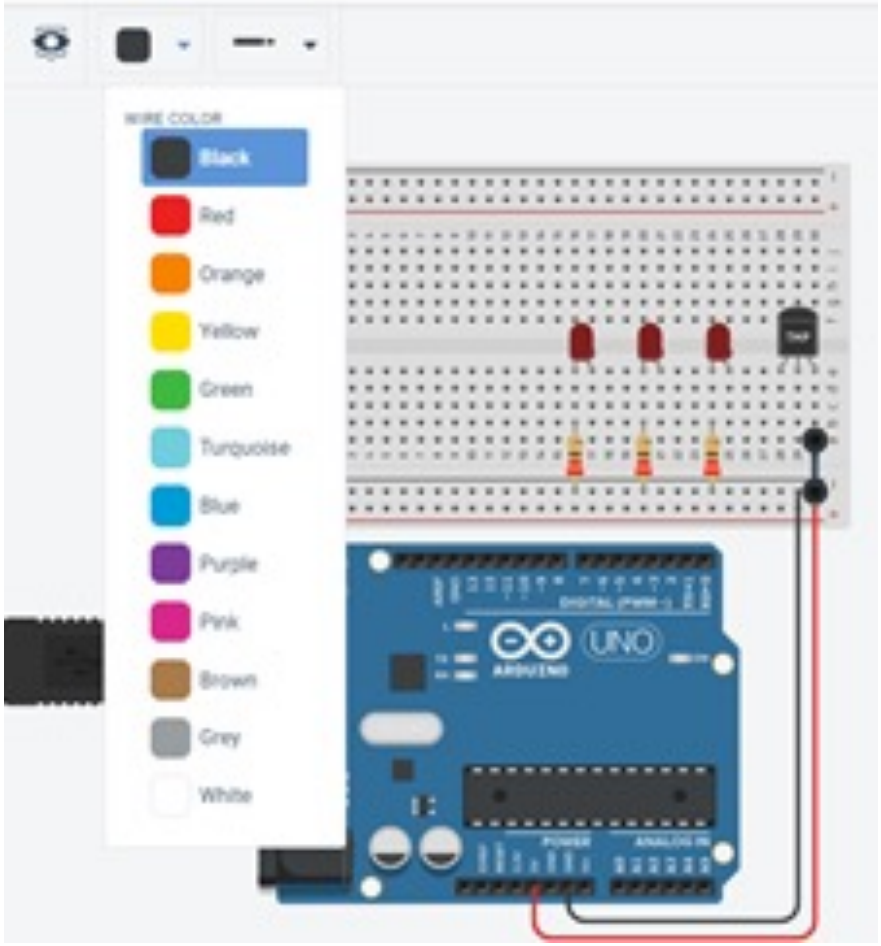






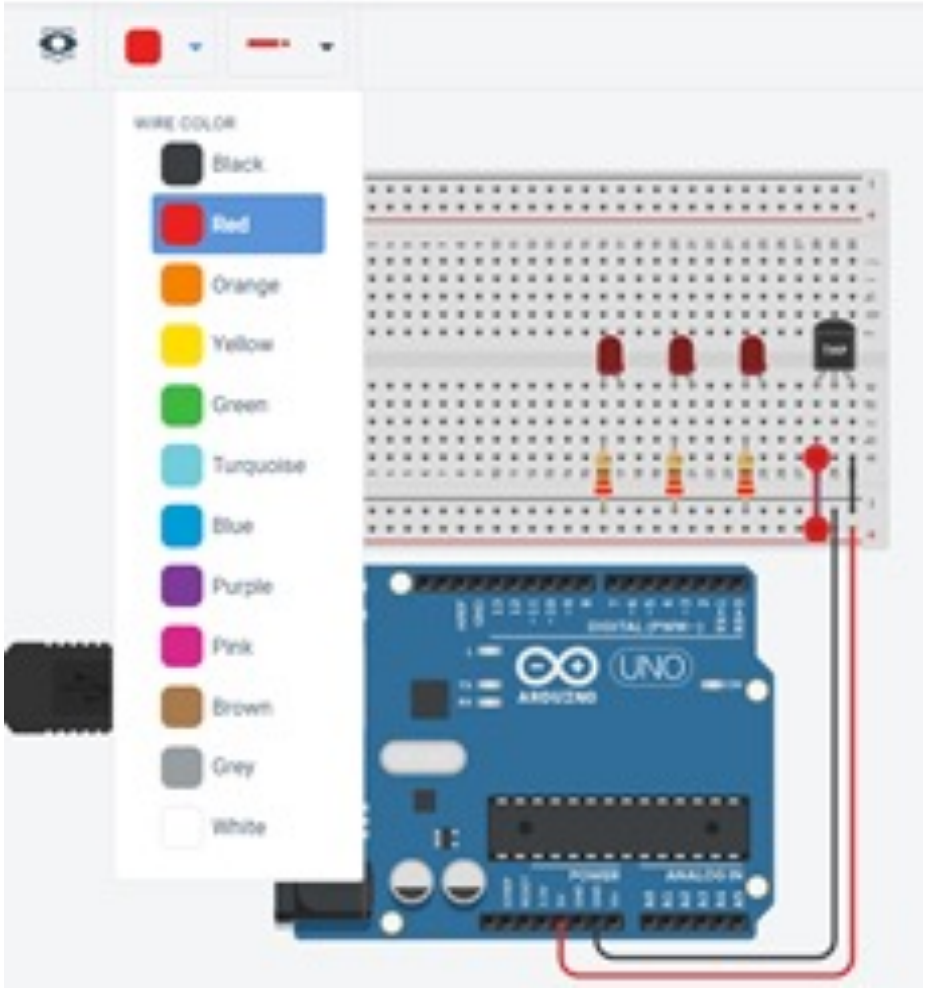
Wiring components

1. For this wire select the color black.
2. One end will begin on the Ground bus aligned to Row 30. The other end will connect to Row 30, Column a.



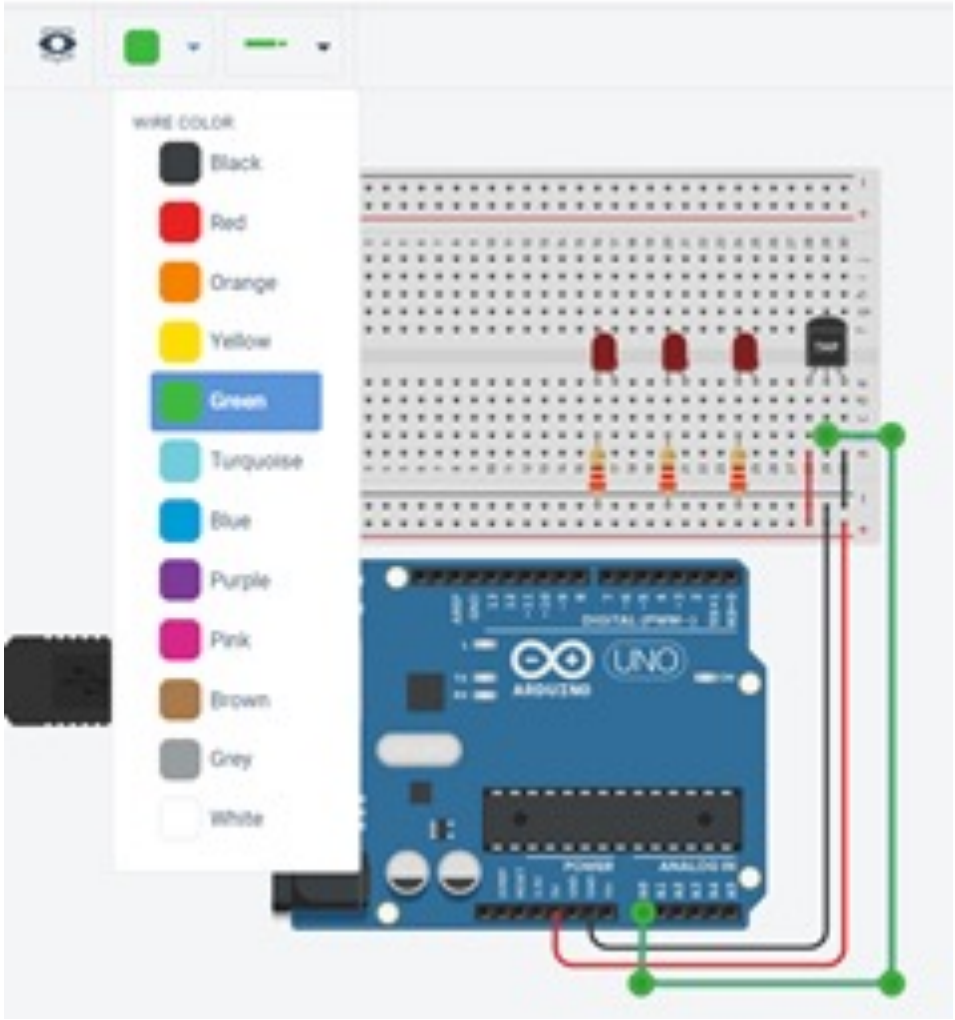
Wiring components

1. For this wire select the color red.
2. One end will begin on the Positive bus aligned to Row 28. The other end will connect to Row 28, Column a.



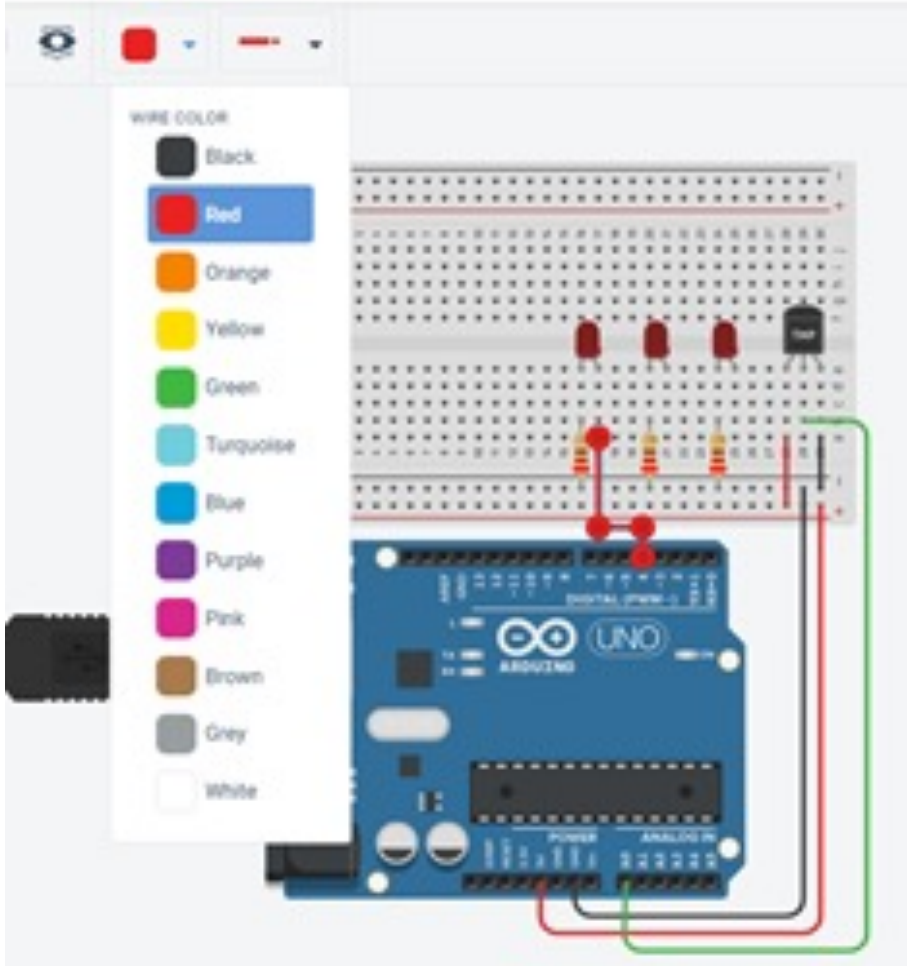
Wiring components

1. For this wire select the color green.
2. One end will begin on the Analog Port A0. The other end will connect to Row 29, Column b.



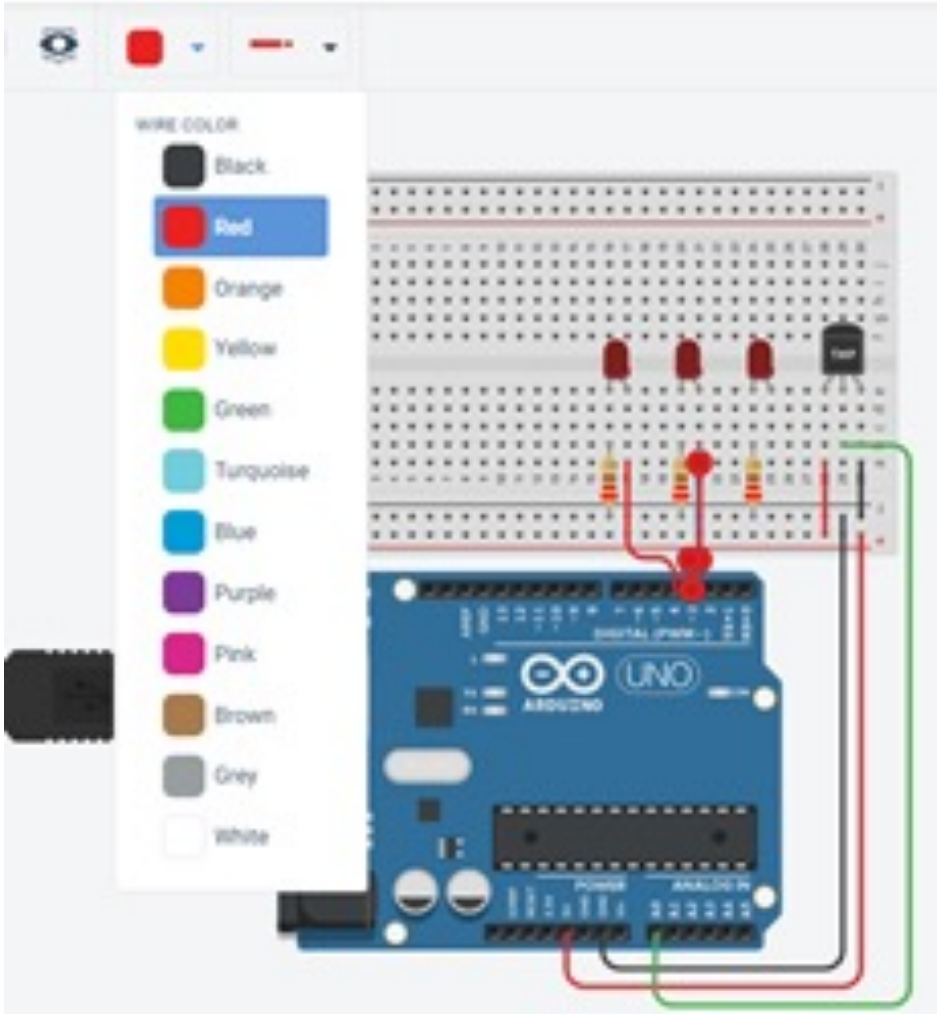
Wiring components

1. For this wire select the color red.
2. One end will begin on Port 4. The other end will connect to Row 17, Column a.



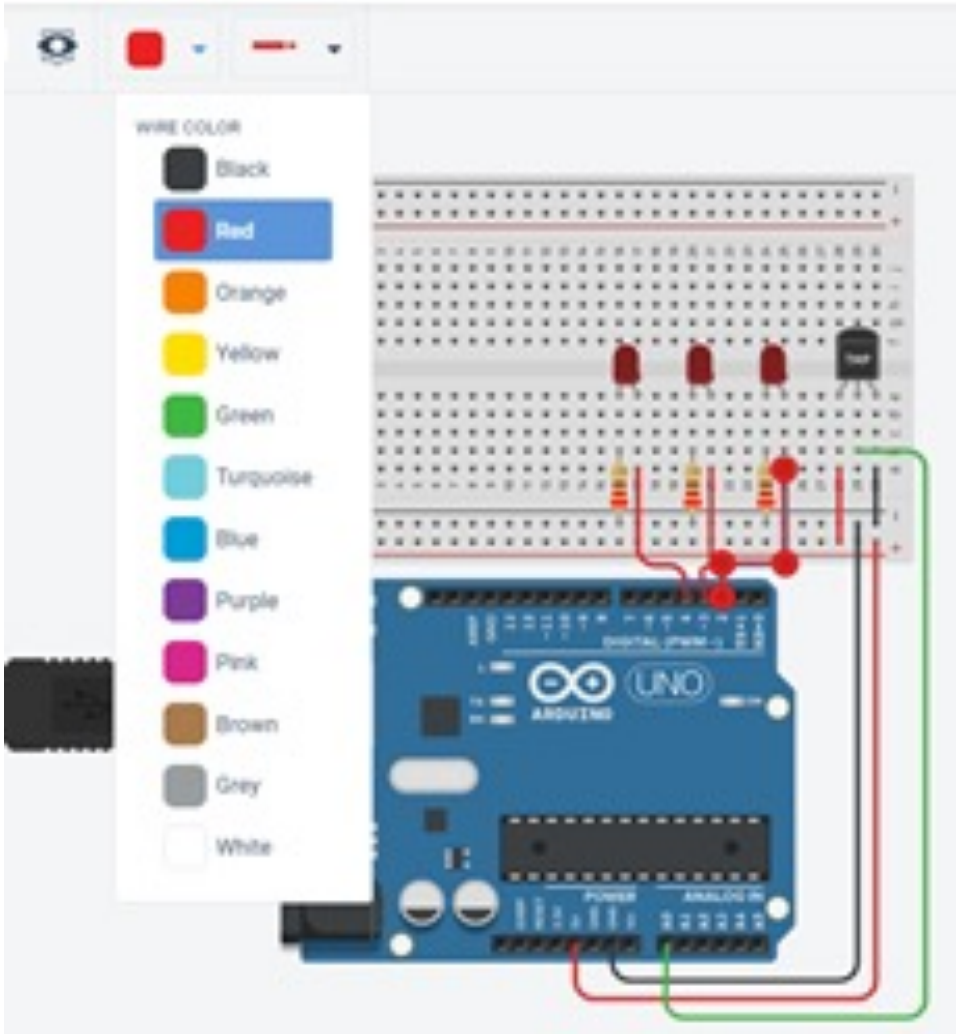
Wiring components

1. For this wire select the color red.
2. One end will begin on Port 3. The other end will connect to Row 21, Column a.

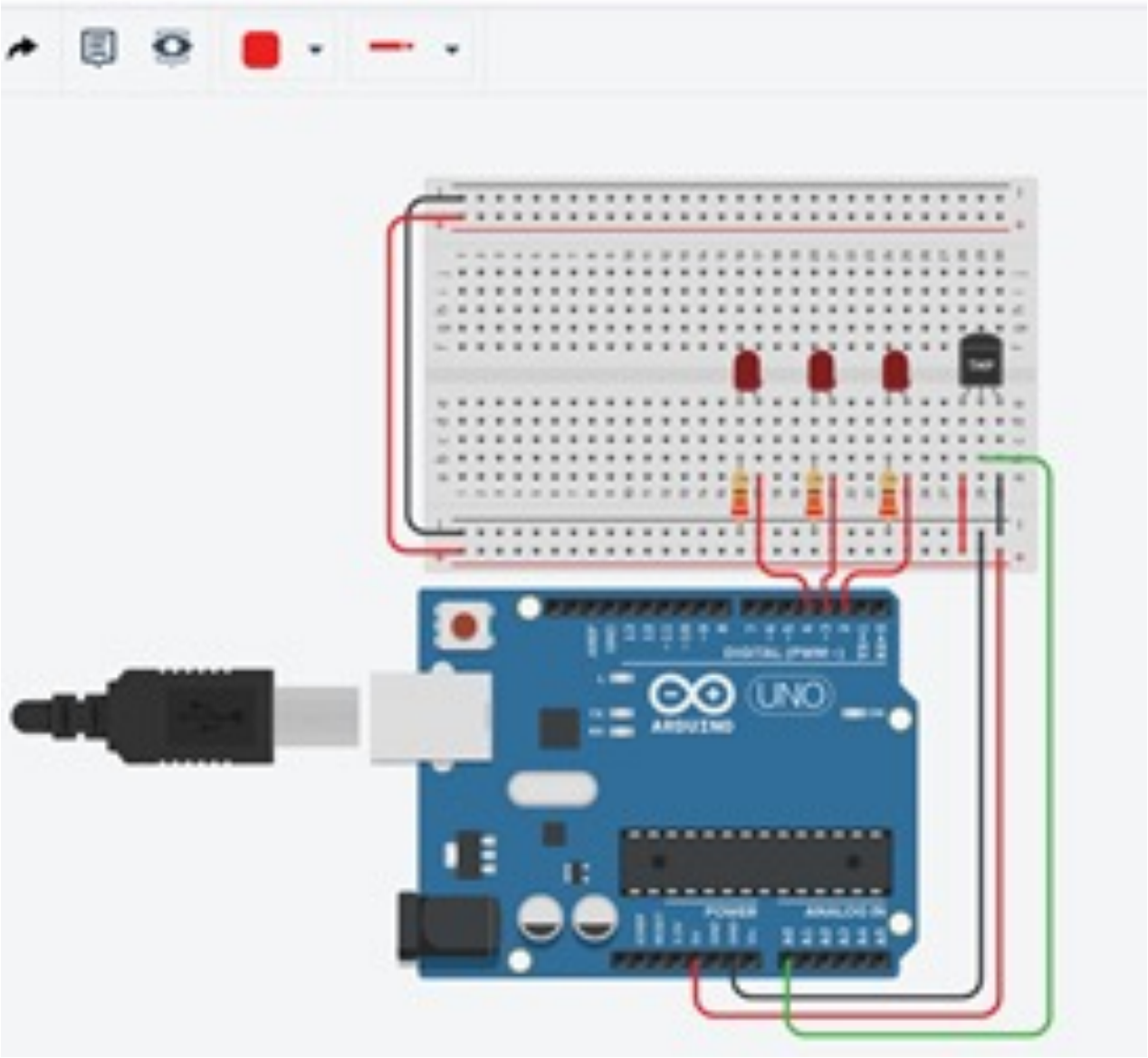


Wiring components

1. For this wire select the color red.
2. One end will begin on Port 2. The other end will connect to Row 25, Column a.



Final Product

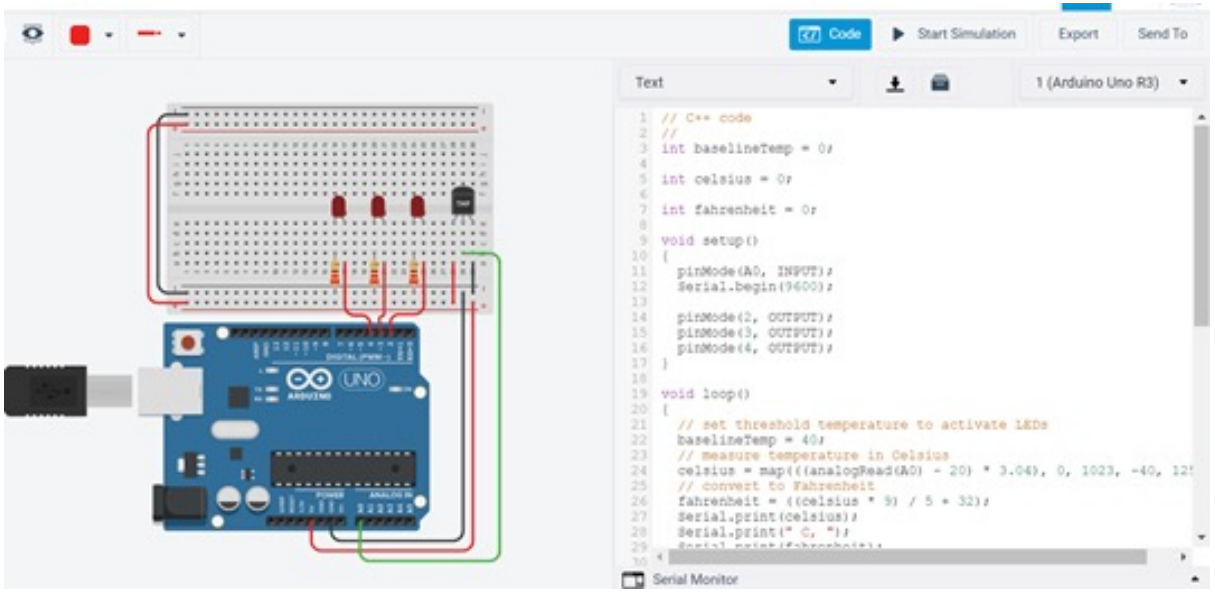


Analog input with Arduino

Code

➔ Copy the following code Part 1 of 3:

```
// C++ code
//
int baselineTemp = 0;

int celsius = 0;

int fahrenheit = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}
```



Analog input with Arduino

Code

➔ Copy the following code
Part 2 of 3:

```
void loop()
{
  // set threshold temperature to activate LEDs
  baselineTemp = 40;
  // measure temperature in Celsius
  celsius = map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40, 125);
  // convert to Fahrenheit
  fahrenheit = ((celsius * 9) / 5 + 32);
  Serial.print(celsius);
  Serial.print(" C, ");
  Serial.print(fahrenheit);
  Serial.println(" F");
  if (celsius < baselineTemp) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
  }
  if (celsius >= baselineTemp && celsius < baselineTemp + 10) {
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
```

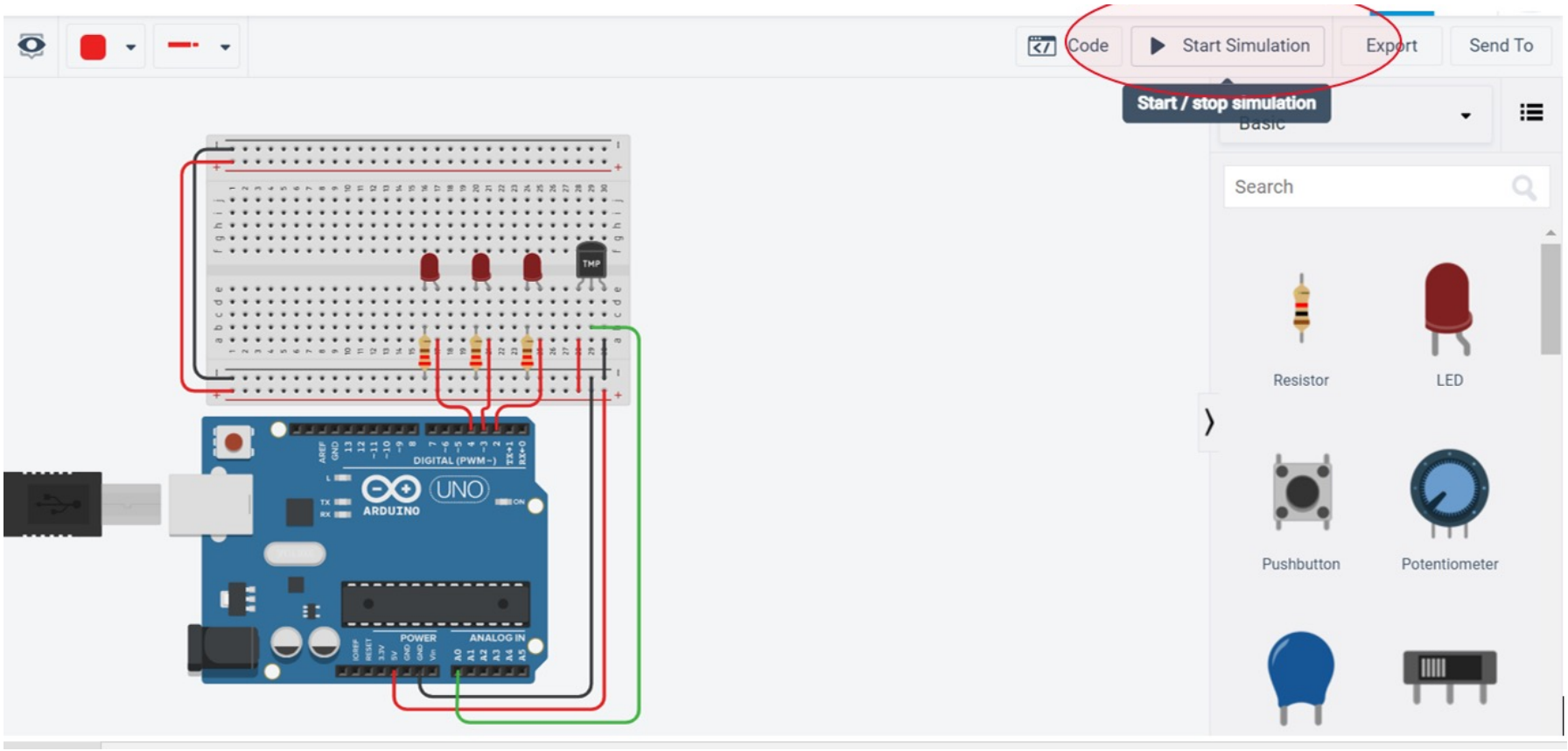
Analog input with Arduino

Code

- Copy the following code
  Part 3 of 3:

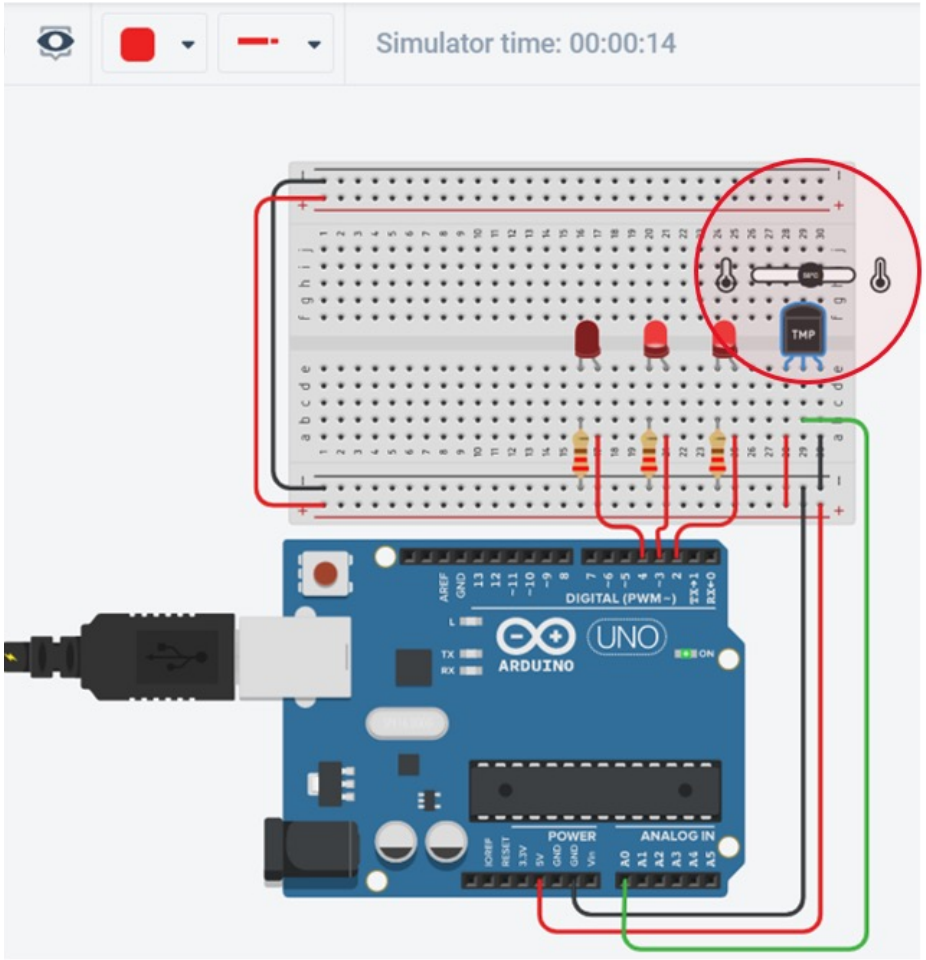
```
  }
  if (celsius >= baselineTemp + 10 && celsius < baselineTemp +
20) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
  }
  if (celsius >= baselineTemp + 20 && celsius < baselineTemp +
30) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
  }
  if (celsius >= baselineTemp + 30) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
  }
  delay(1000); // Wait for 1000 millisecond(s)
}
```

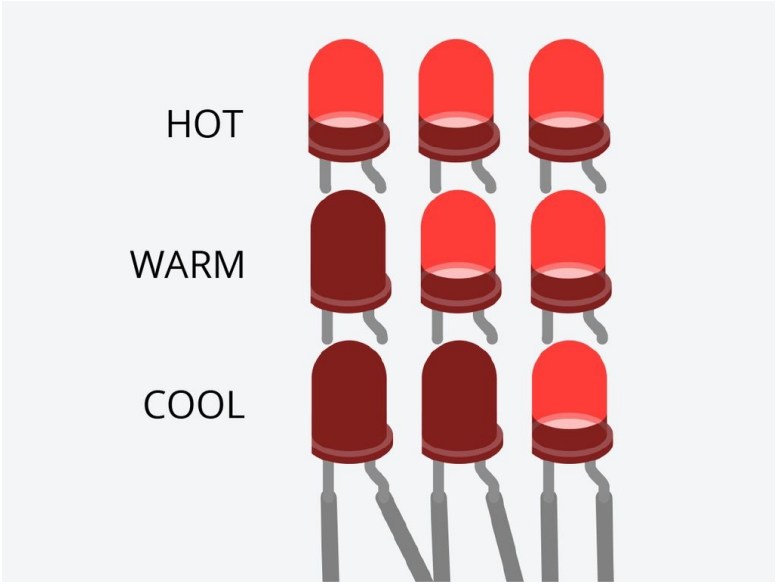
Begin Simulation



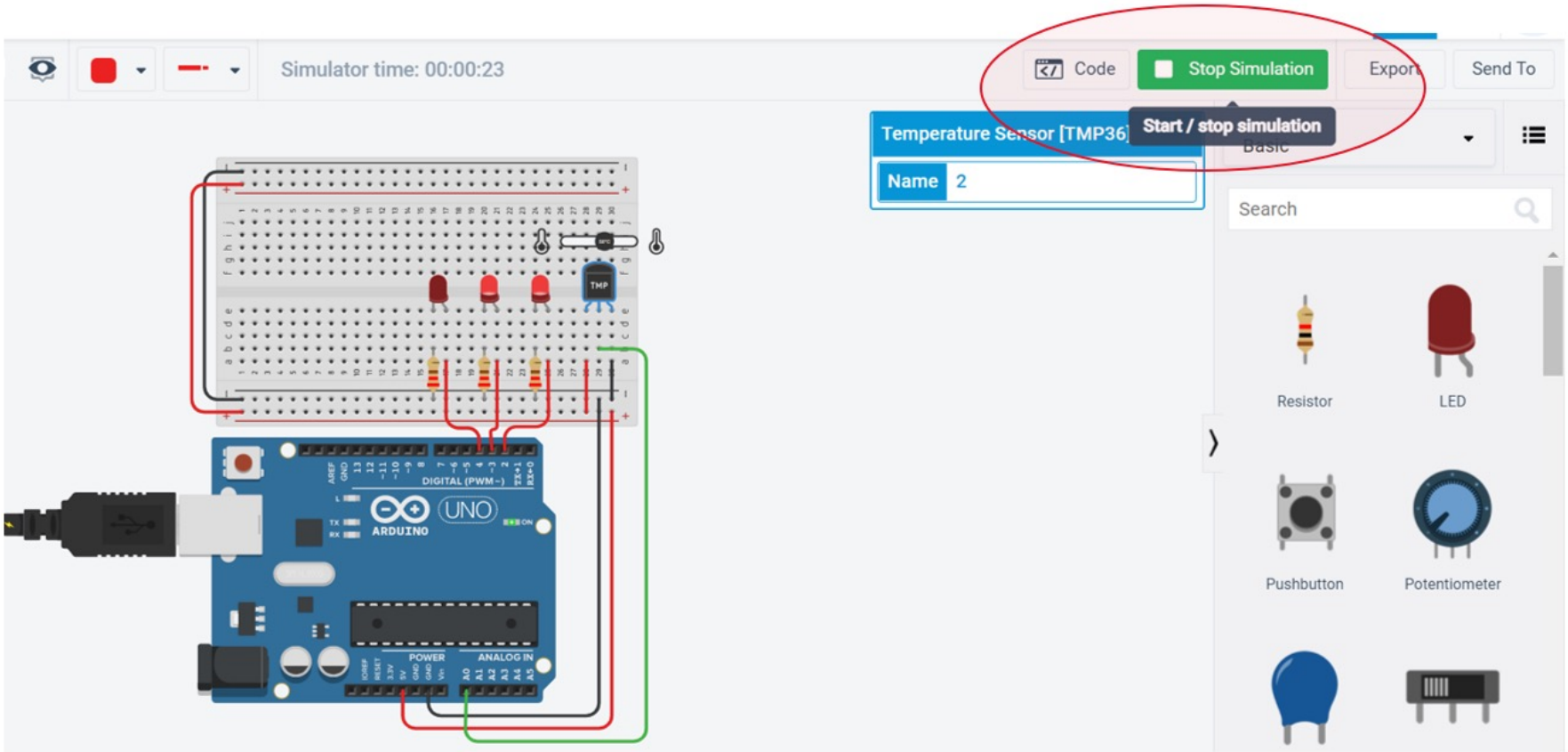
Review Simulation

Play with the temperature dial. This change in temperature represents the change in temperature in real life, like the temperature outside or your body temperature.





End Simulation



Virtual Activity - 5

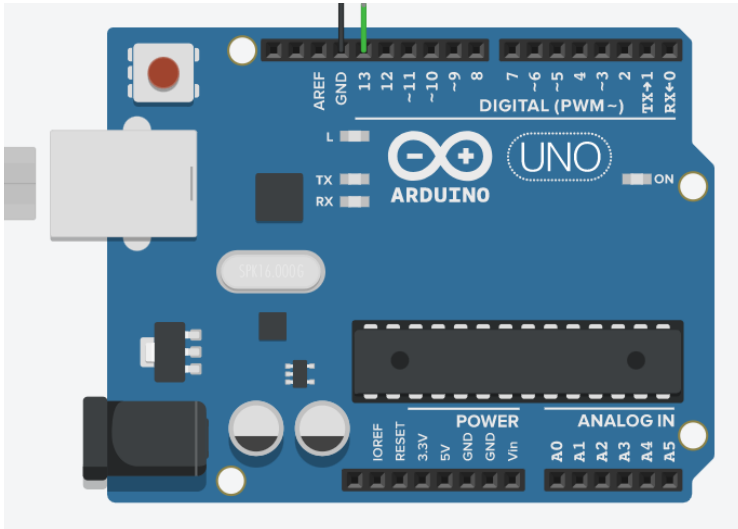
Keypad with Arduino

4x4 Keypad with Arduino

We will introduce how to use a 4x4 matrix keypad.

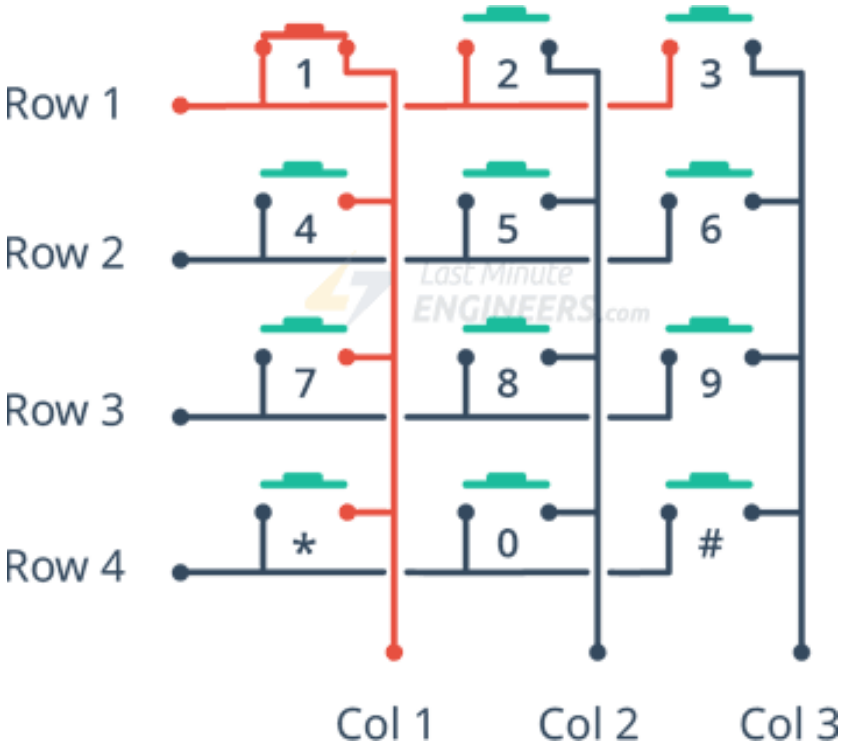
Components

- Arduino
- Keypad

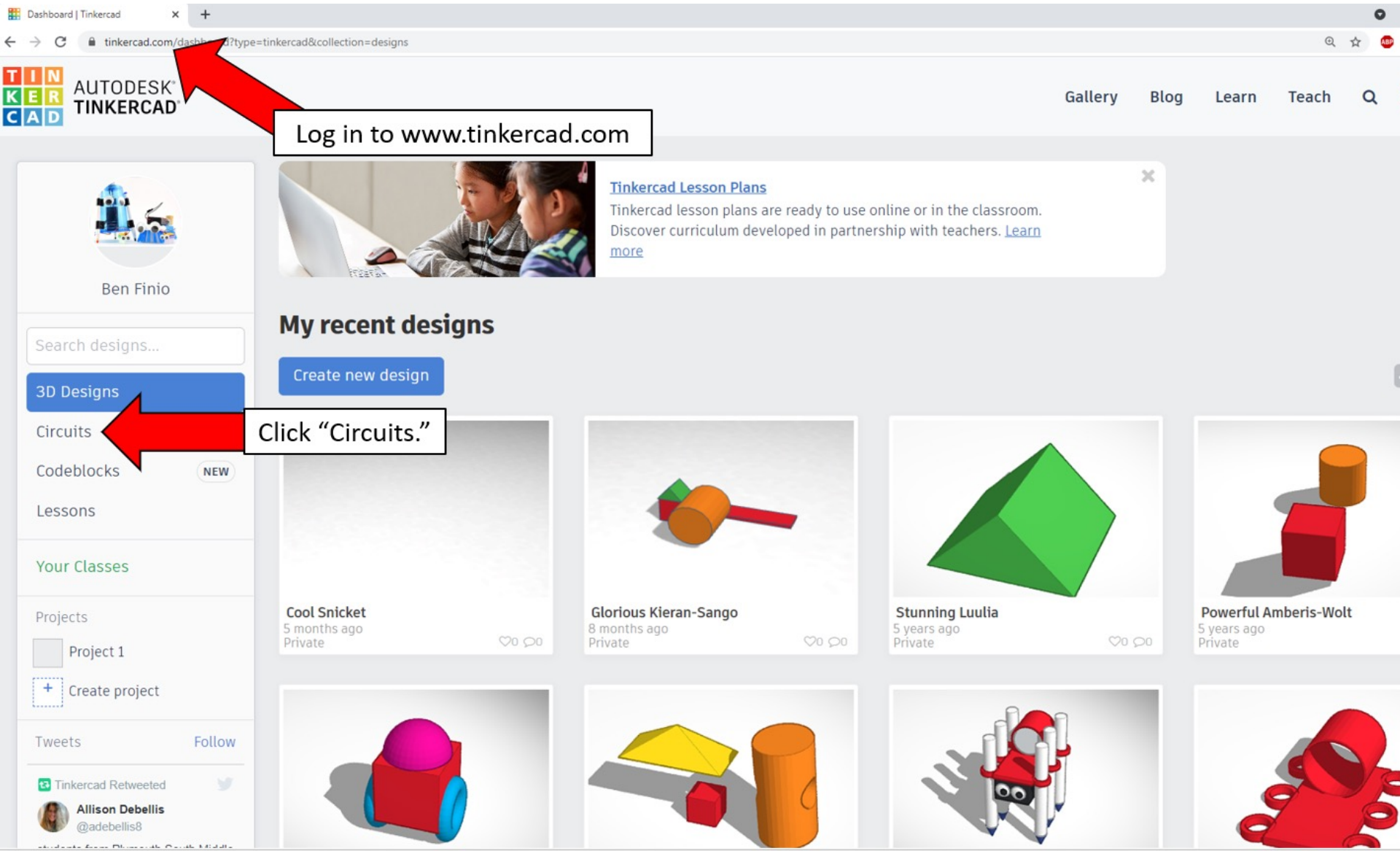


The Keypad

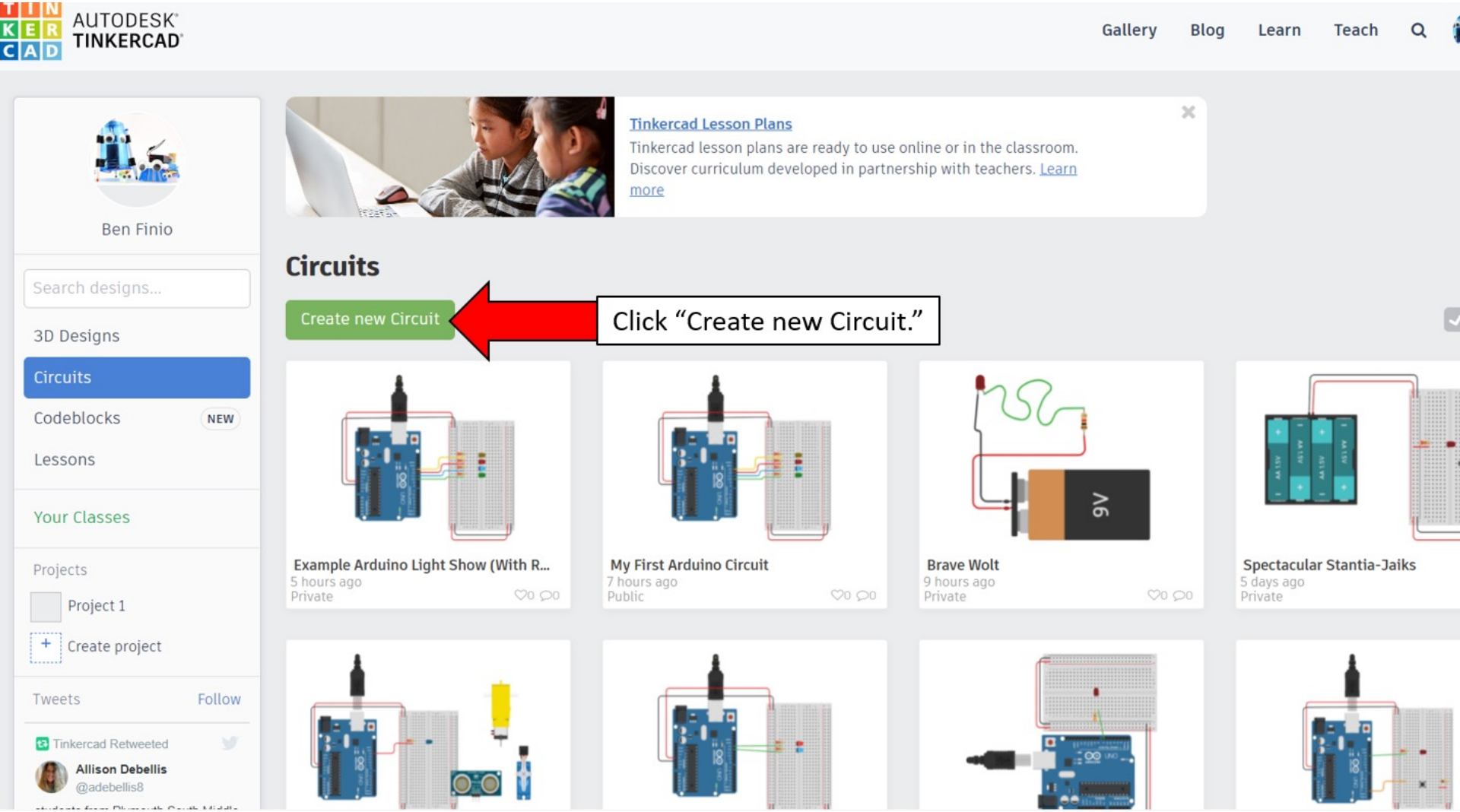
Shown below is a gif of a 4x3 keypad,
we will be using a 4x4 keypad.



Starting a new circuit

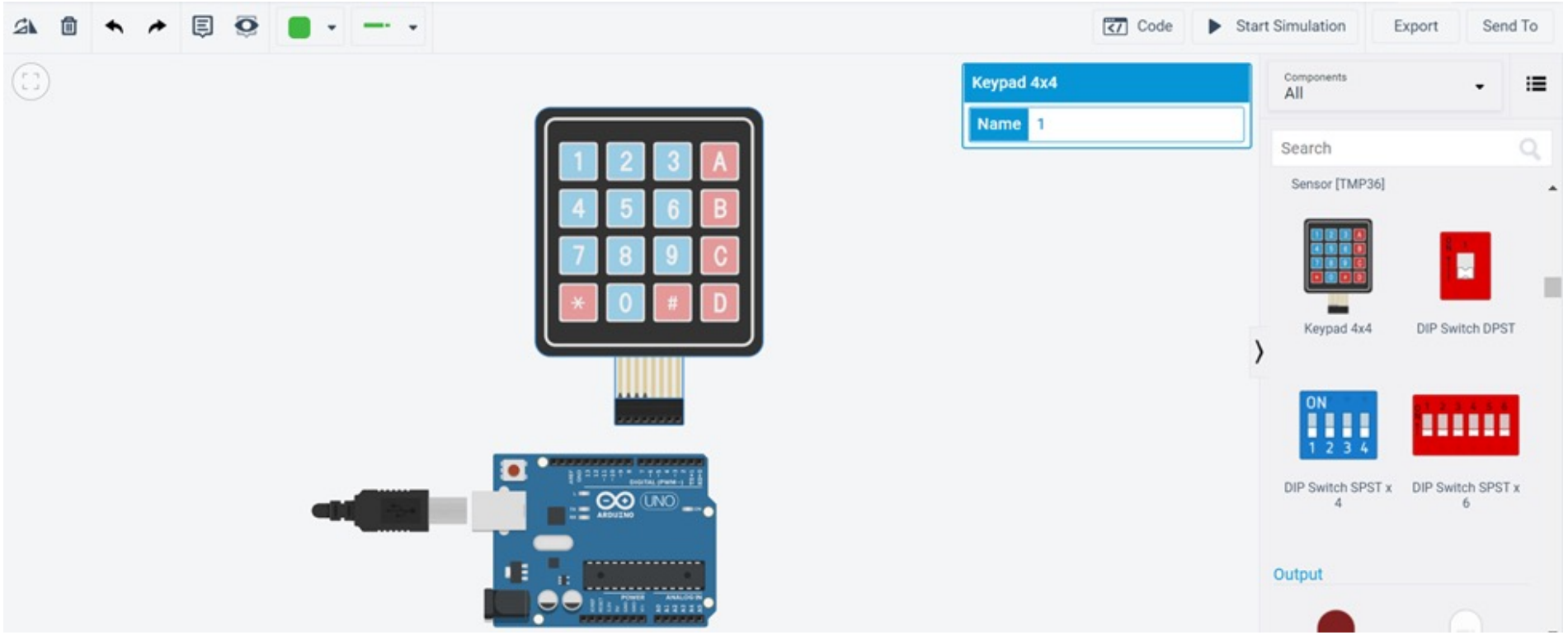


Starting a new circuit



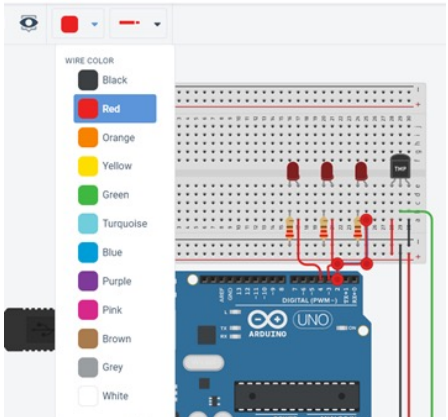
Setting up our Workplace

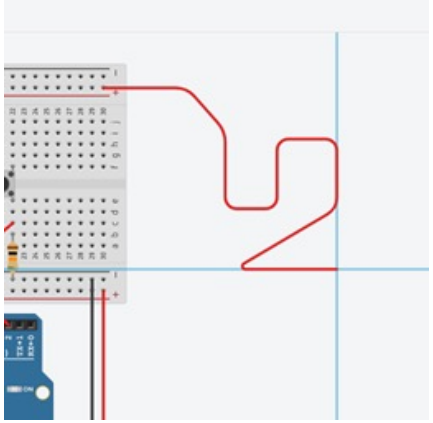
1. Under Starters/Components: click Arduino.
2. Click and drag on the first component, place it on the workplace and let go.
3. Repeat step two for the following components: 4x4 Keypad.

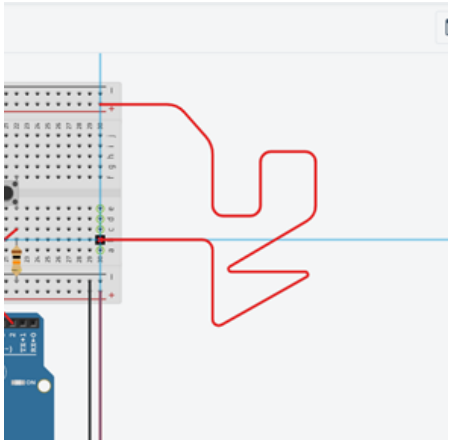


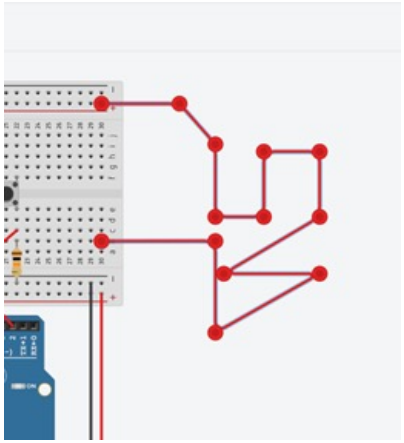
Wiring Orientation

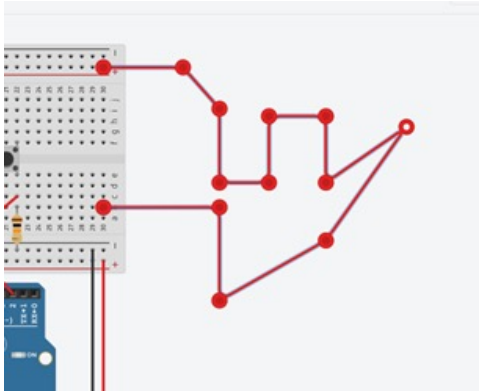
1. Choose a color for the wire. Commonly, your ground will be black, and power will be red.
2. Next choose a start point and click once, you must choose a port or a location on the bread board. You will now have a wire that can be stretched almost at any angle.
3. In order to make a your wire "curve" at different points, you can right click, as long as you do not click when hoovering over a port or else it'll assumingly connect to that port.
4. You add as many turning points as you want, and they will appear after you connect to a port.
5. After which you can stretch the wire through its' turning points.







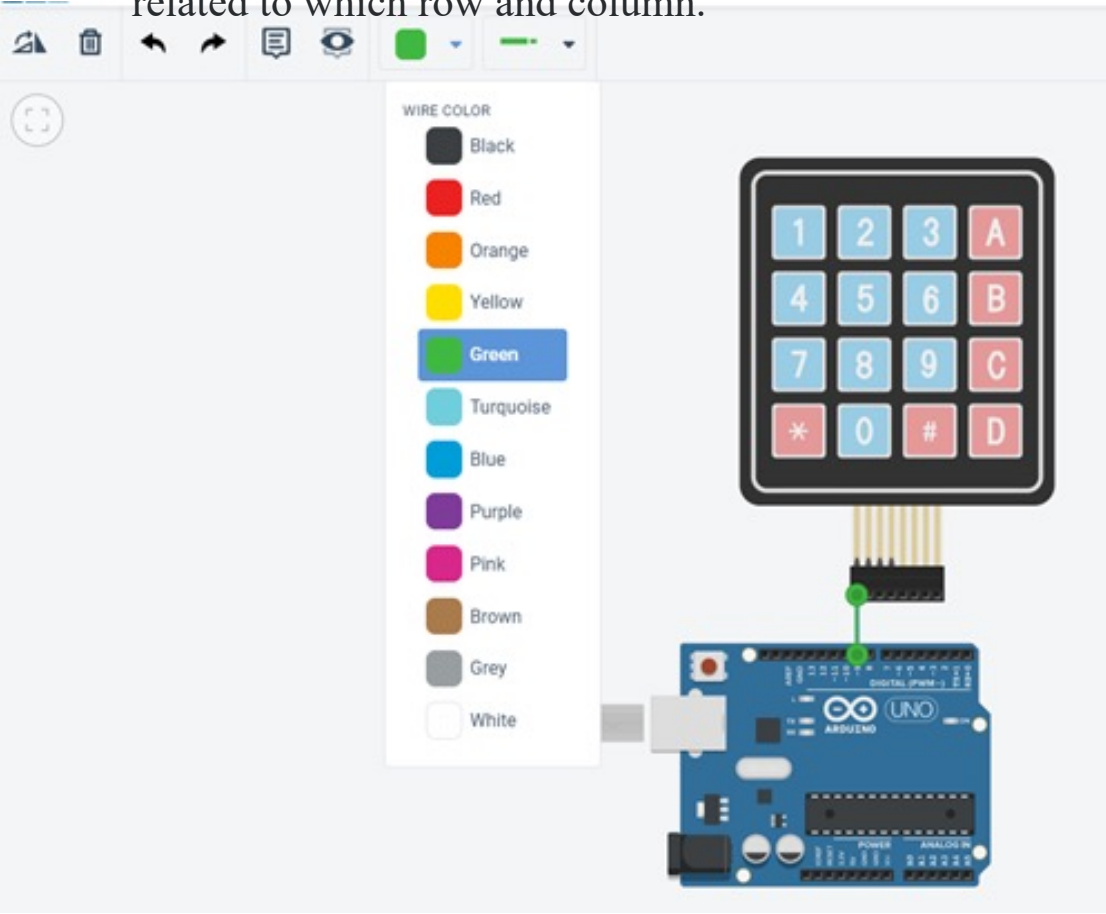


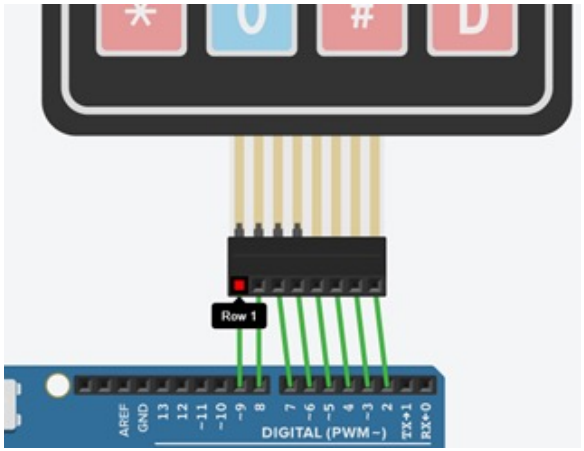


Wiring a 4x4 Keypad with Arduino

Set up

1. For this wire select the color green.
2. One end will begin on the Digital Port 9 and the other end will connect to "Row 1" port. If you hover your mouse over each port on the keypad you can see which port is related to which row and column.

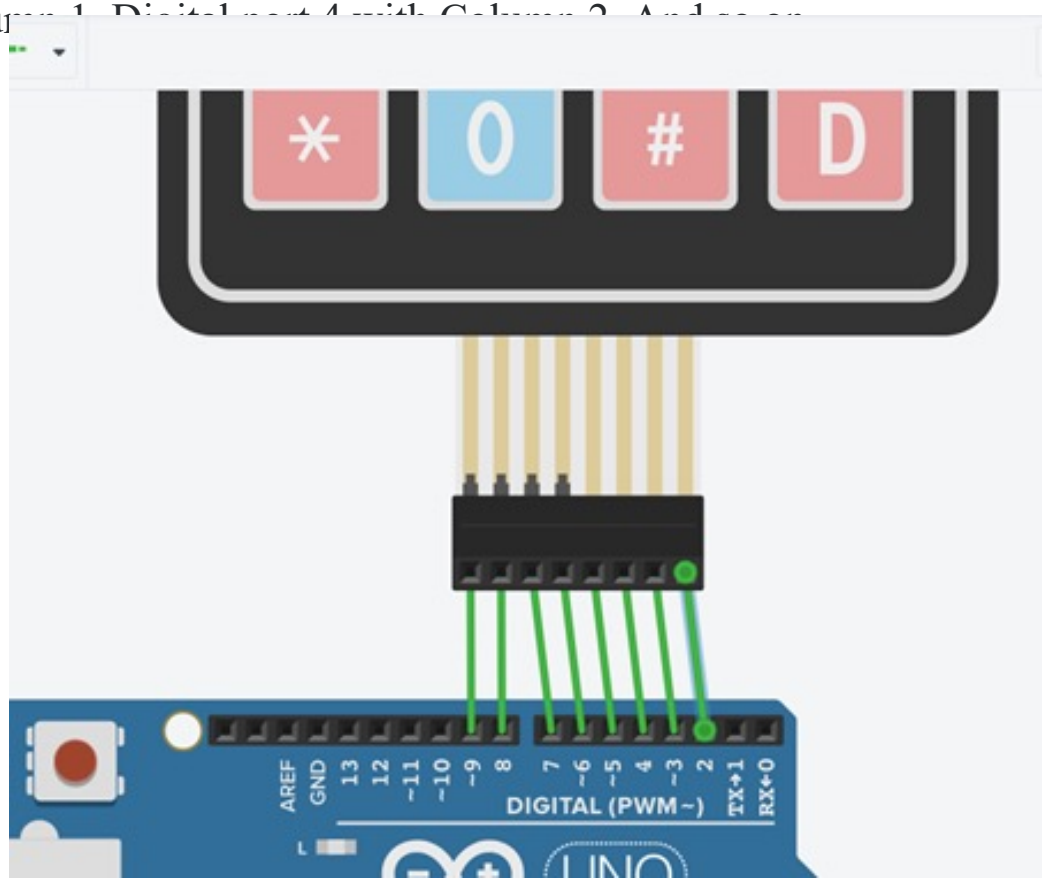




Wiring a 4x4 Keypad with Arduino

Set up

1. For this wire select the color green.
2. Continue the process from the last slide and wire from left to right. Notice how the digital ports on the Arduino align with keypad port. Digital port 9 with Row 1. Digital port 8 with Row 2. Digital port 7 with Row 3. Digital port 6 with Row 4. Digital port 5 with Colu[illegible]



Final Product

Keypad Code

Code

- Copy the following code
  Part 1 of 2:

```
#include <Keypad.h>

const byte ROWS = 4; //global variable that represents four rows
const byte COLS = 4; //global variable that represents four columns

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {5, 4, 3, 2}; //connect to the column pinouts of the keypad
```

Keypad Code

Code

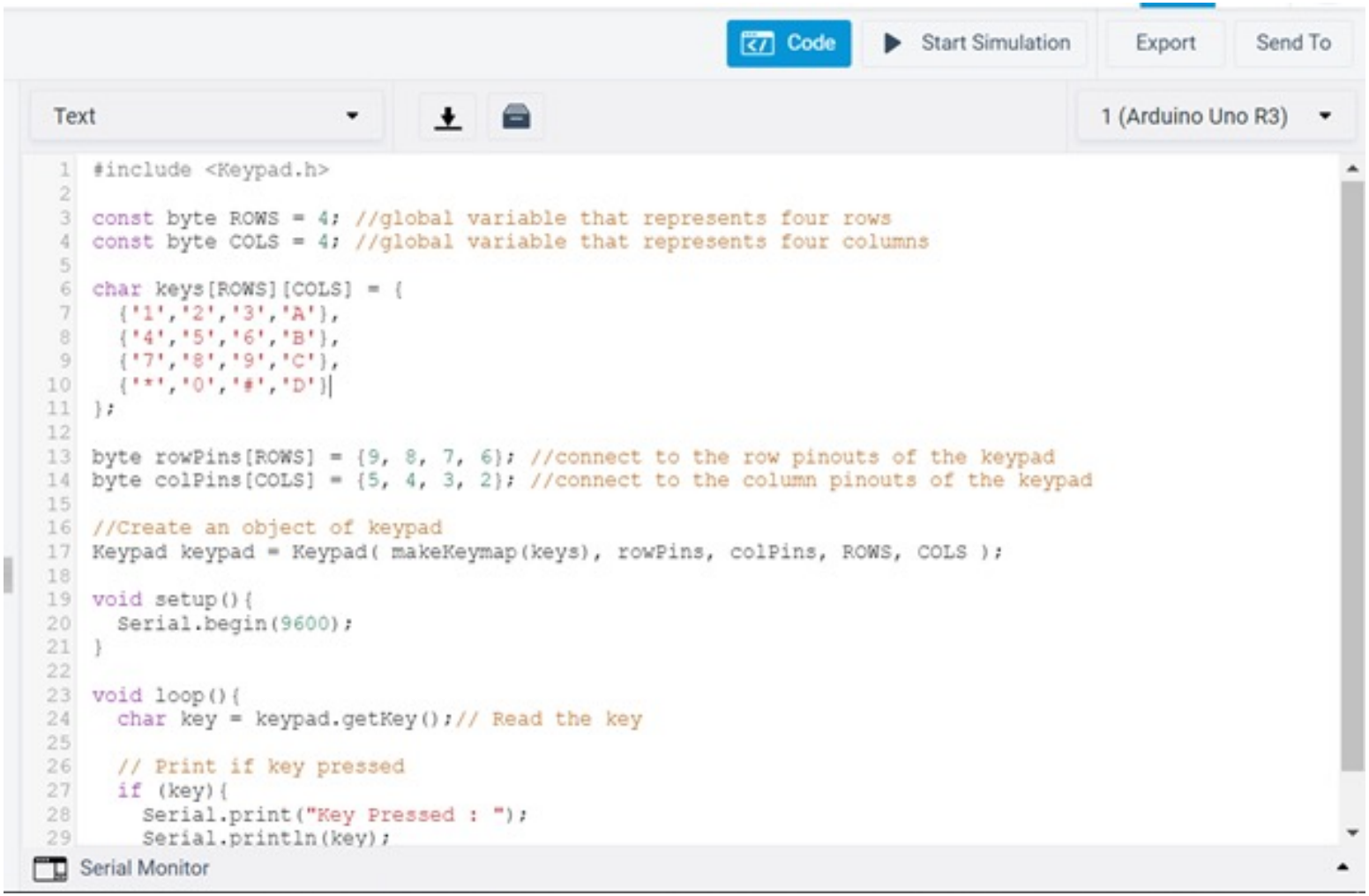
- Copy the following code
  Part 2 of 2:

```
//Create an object of keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600);
}

void loop(){
  char key = keypad.getKey();// Read the key

  // Print if key pressed
  if (key){
    Serial.print("Key Pressed : ");
    Serial.println(key);
  }
}
```

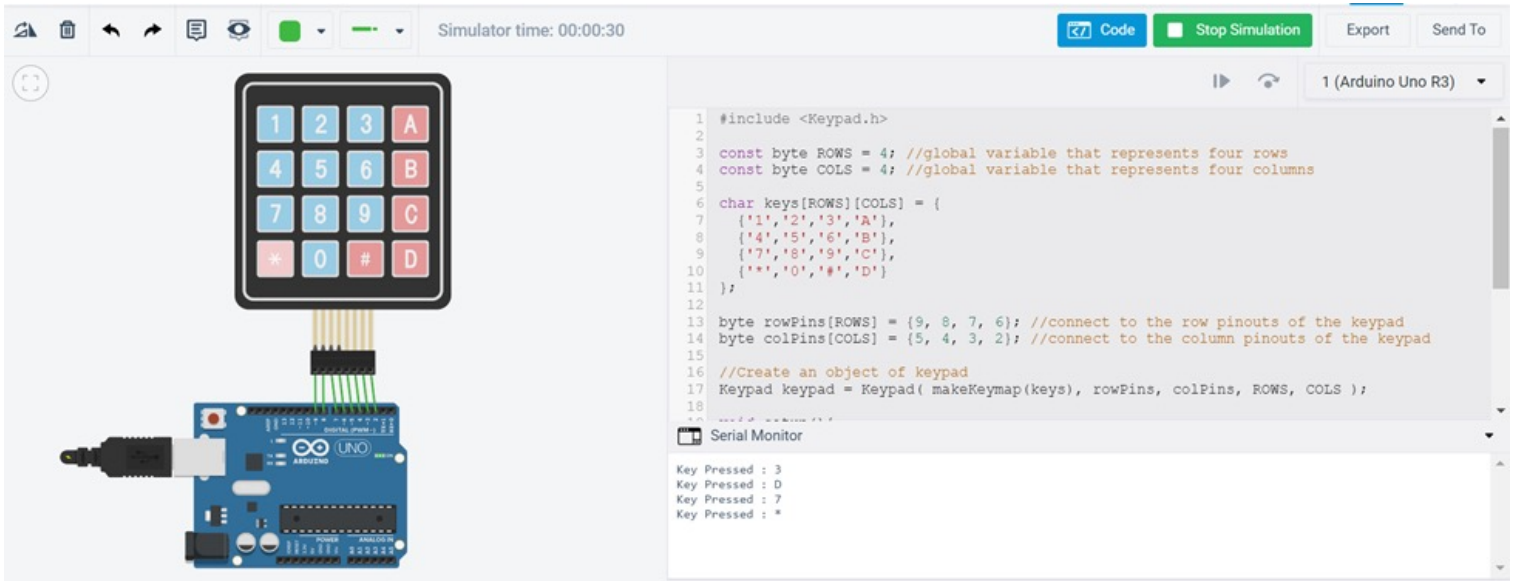
Keypad Code

Code

➔ Final Code Product:

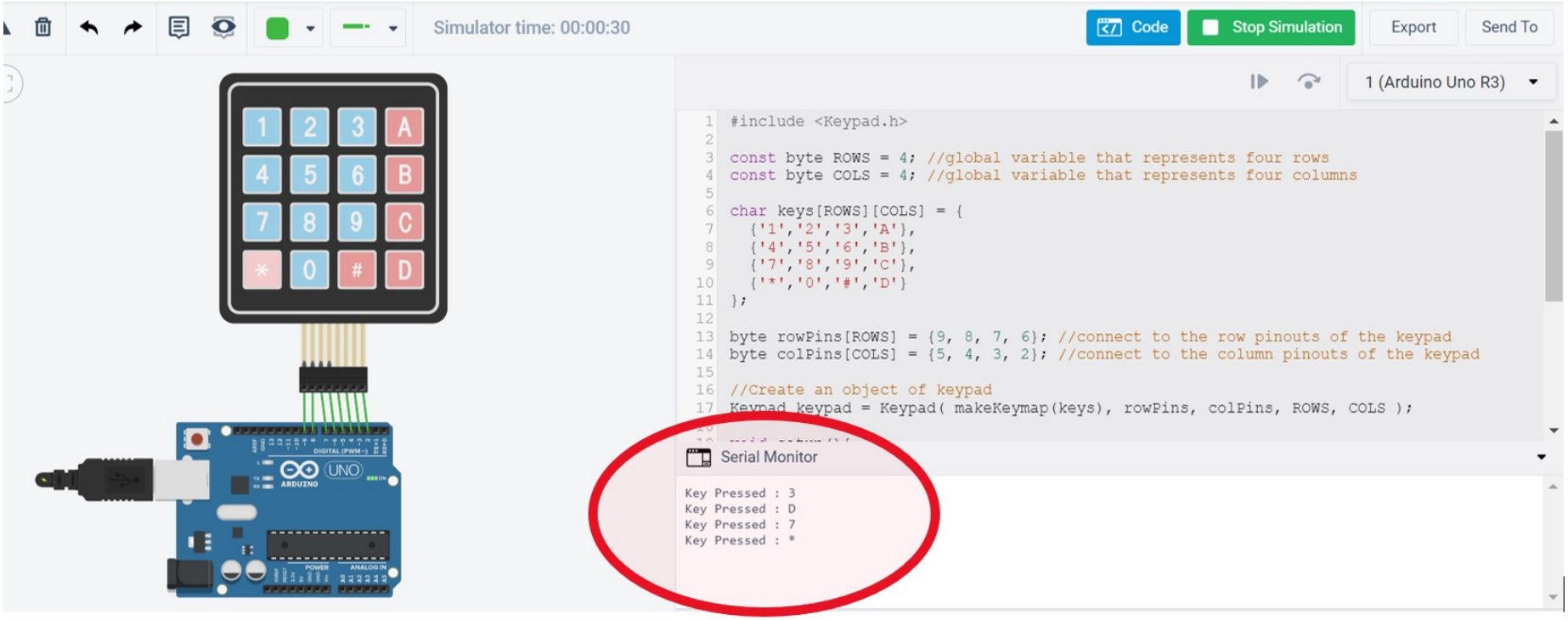


Begin Simulation

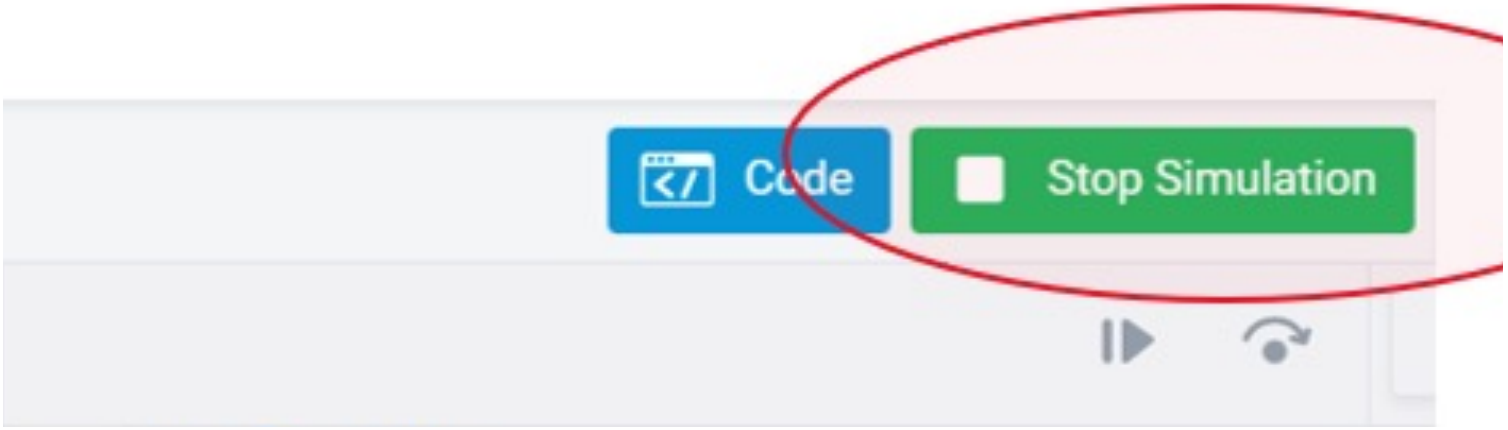


Review Simulation

Notice how the Serial Monitor displays and tracks what has been pressed on the keypad.



Stop Simulation



More to Explore!

There are a few useful functions you can use with Keypad object. Few of them are listed below:

- `char waitForKey()` Waits forever until someone presses a key. Warning – It blocks all other code until a key is pressed. That means no blinking LED's, no LCD screen updates, no nothing with the exception of interrupt routines.

- `KeyState getState()` Returns the current state of any of the keys. The four states are IDLE, PRESSED, RELEASED and HOLD.

- `boolean keyStateChanged()` Let's you know when the key has changed from one state to another. For example, instead of just testing for a valid key you can test for when a key was pressed.

- `setHoldTime(unsigned int time)` Sets the amount of milliseconds the user will have to hold a button until the HOLD state is triggered.

- `setDebounceTime(unsigned int time)` Sets the amount of milliseconds the keypad will wait until it accepts a new keypress/keyEvent.

- `addEventListener(keypadEvent)` Triggers an event if the keypad is used.

Dr. Murat Kuzlu
mkuzlu@odu.edu

Jorge Ramirez III
jrami013@odu.edu