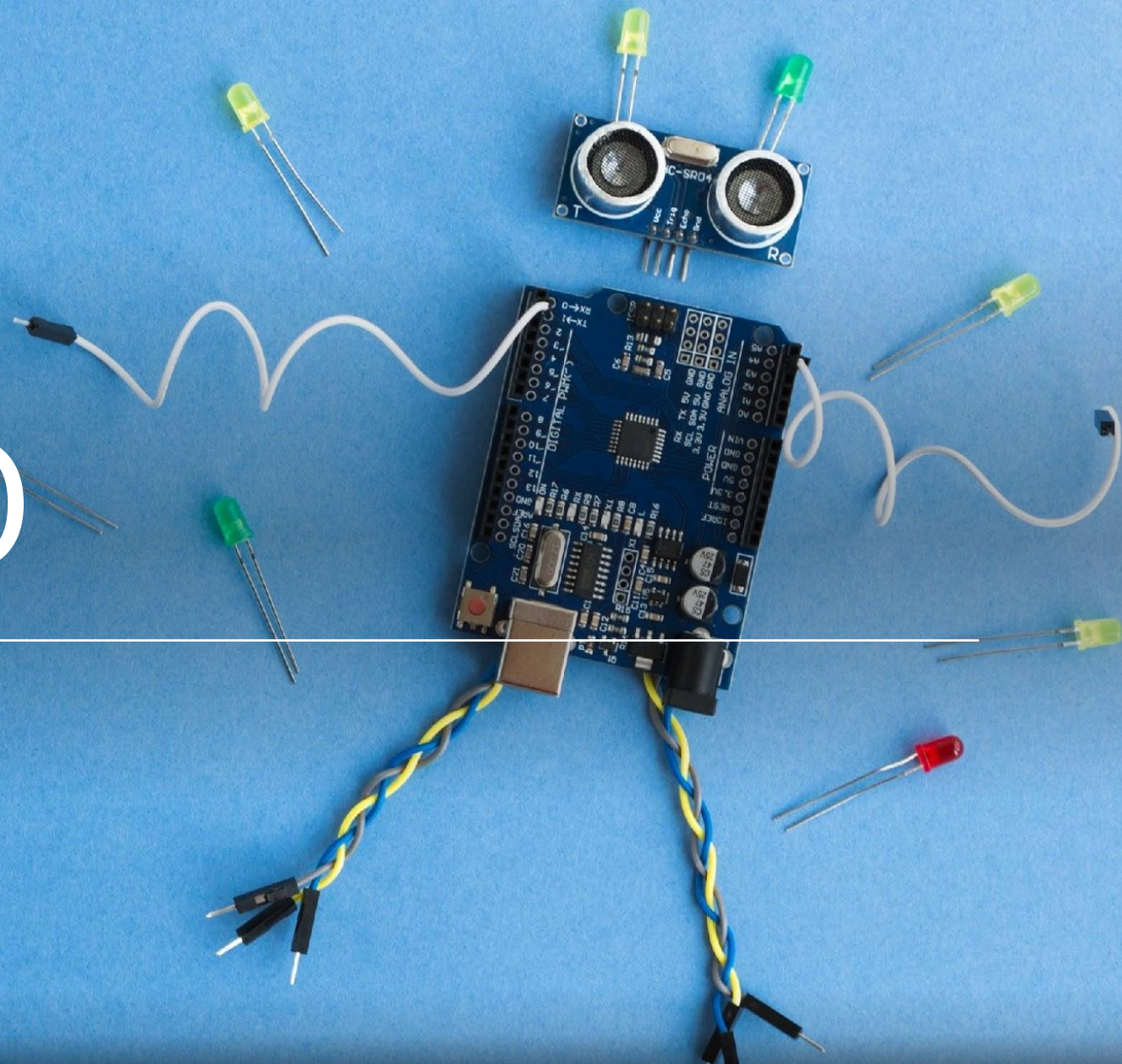


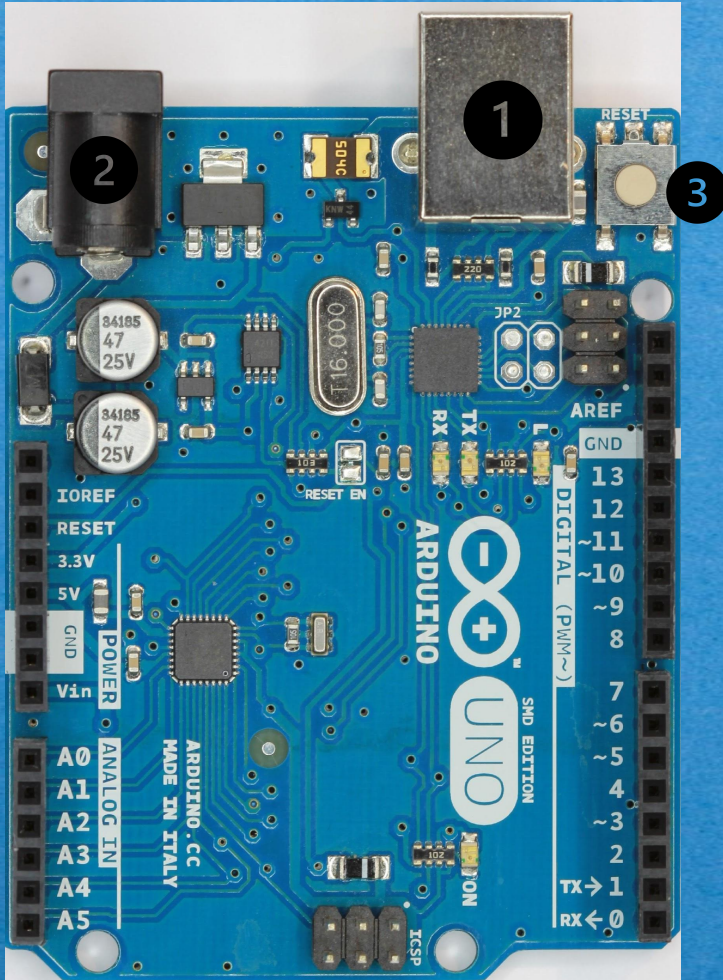
ARDUINO



What is Arduino?

- an open-source platform which is used for building electronics projects.
- comprises of a programmable circuit board (often called a microcontroller) and a piece of software (Integrated Development Environment).
- really common among hobbyists and professionals mainly because of its versatility and its simplicity...so *relax and you will most definitely get the hang of it.*
- Most of the time, working with Arduino is basically connecting components together and uploading your code to the Arduino board, using the IDE (integrated development environment) to instruct the microcontroller on how to manage the components and what tasks to do.

Board layout (Uno)

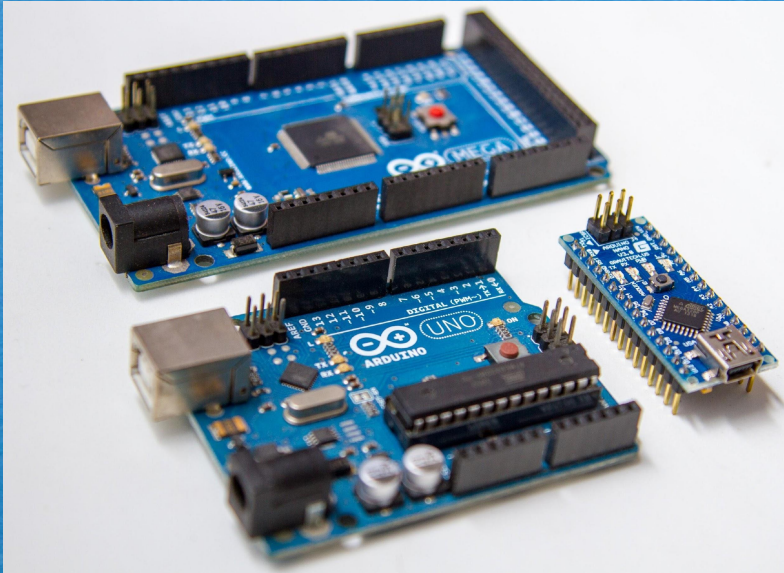


Pin / part	Function
3.3V	3.3V output
5V	5V output
GND	Ground pins
Vin	Input voltage (7 – 12V)
A0, A1, A2, A3, A4, A5	Analog I/O pins
2, 4, 7, 8, 12, 13	Digital I/O pins
3, 5, 6, 9, 10, 11	Digital I/O pins (PWM)
Label 1	USB port
Label 2	7 – 12V DC input
Label 3	Reset button

- PWM stands for “Pulse Width Modulation”
- Regular digital I/O pins work in a binary way, PWM pins can work with varying values.

Types of boards / Setup

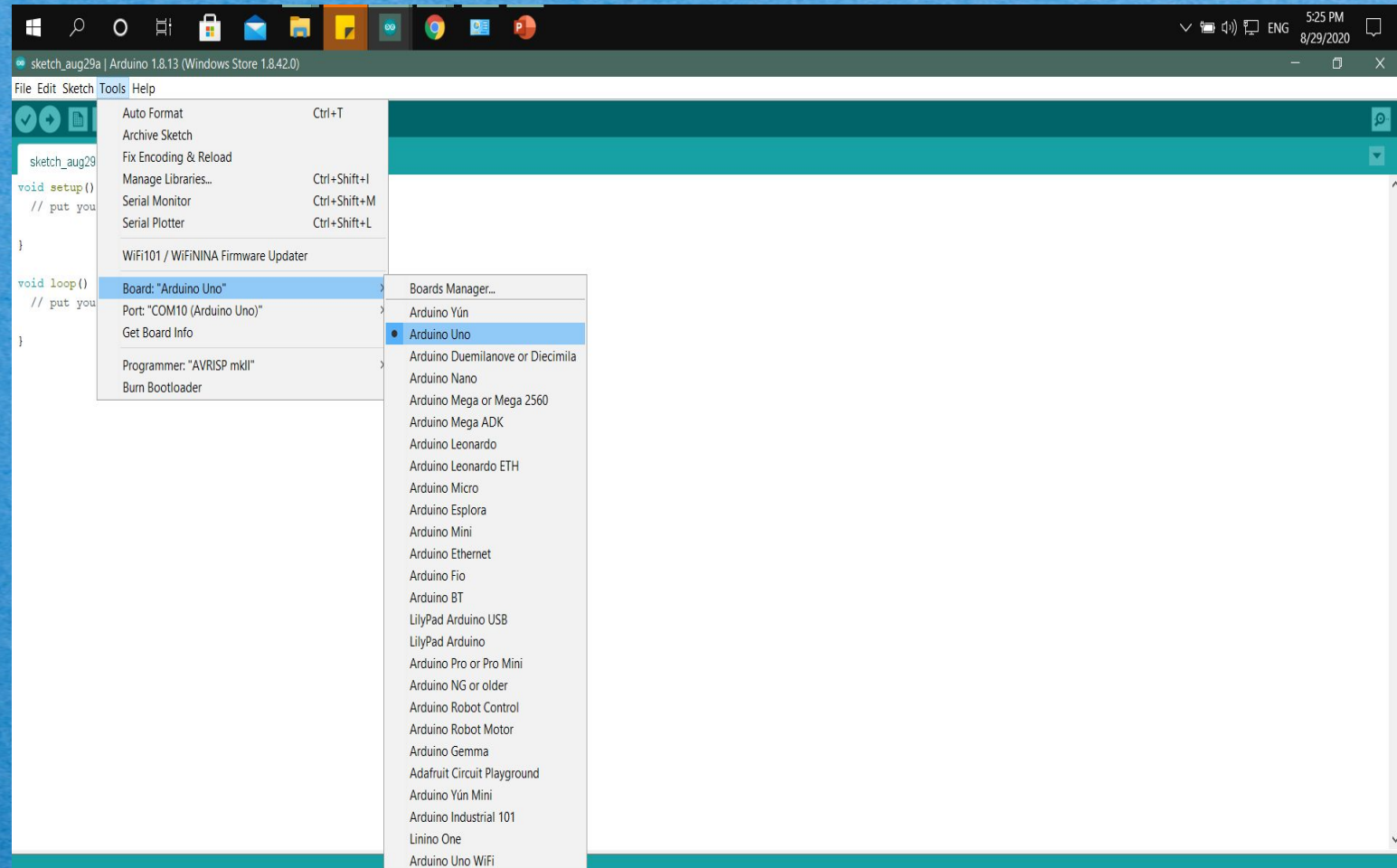
- There are tons of Arduino boards out there. They work similarly.



- To communicate with the Arduino, the Arduino IDE (integrated development environment) software needs to be installed on a laptop to write code and have it uploaded to the Arduino board.
- It can be gotten from the Microsoft store (PC users) or from the Arduino website (<https://www.arduino.cc/en/software>) for Linux, PC and Mac users.

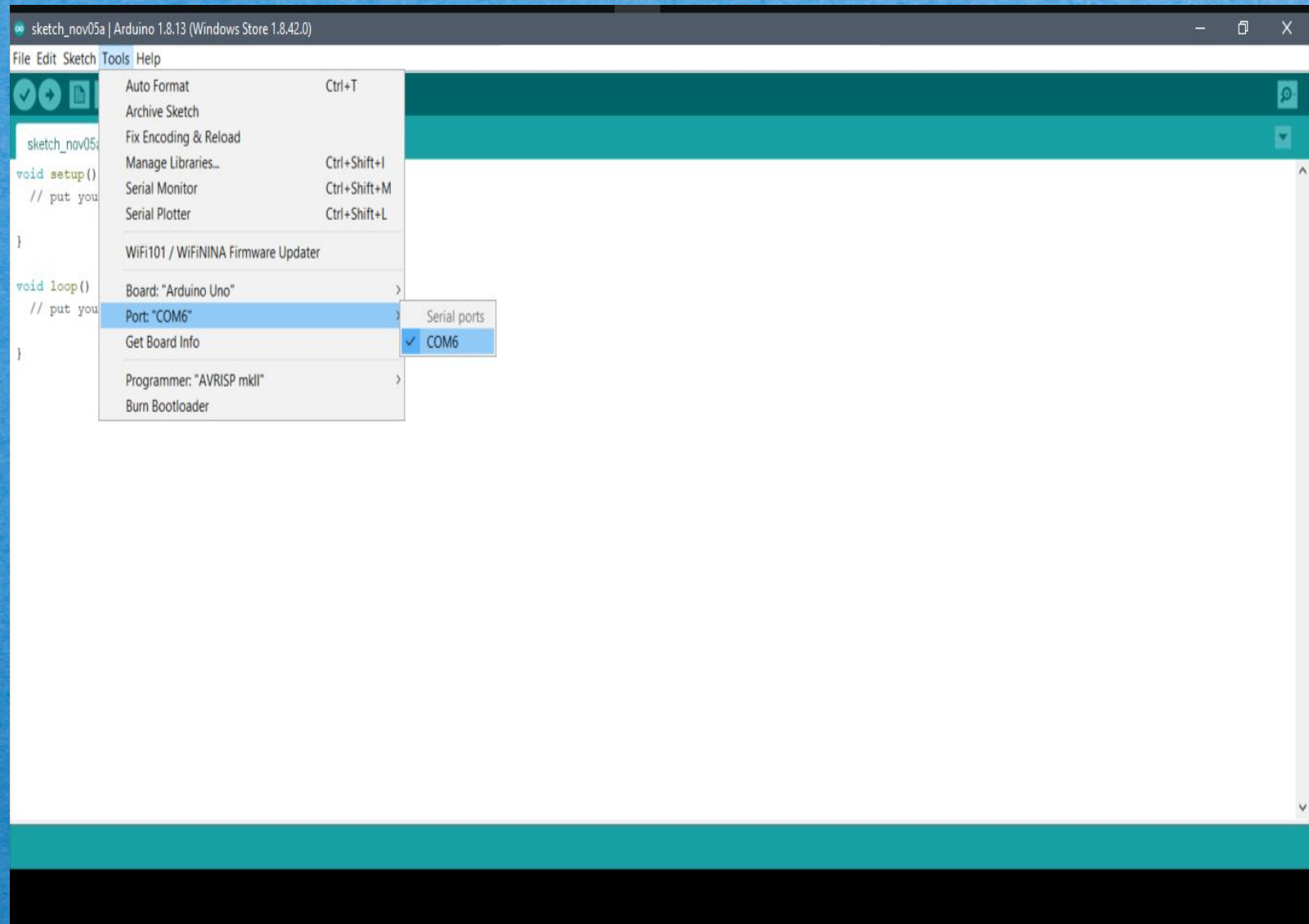
Working with the Arduino #1

- After having the software installed, connect your Arduino board and choose the board type by going to...
tools >> board and then choosing what board you are using.



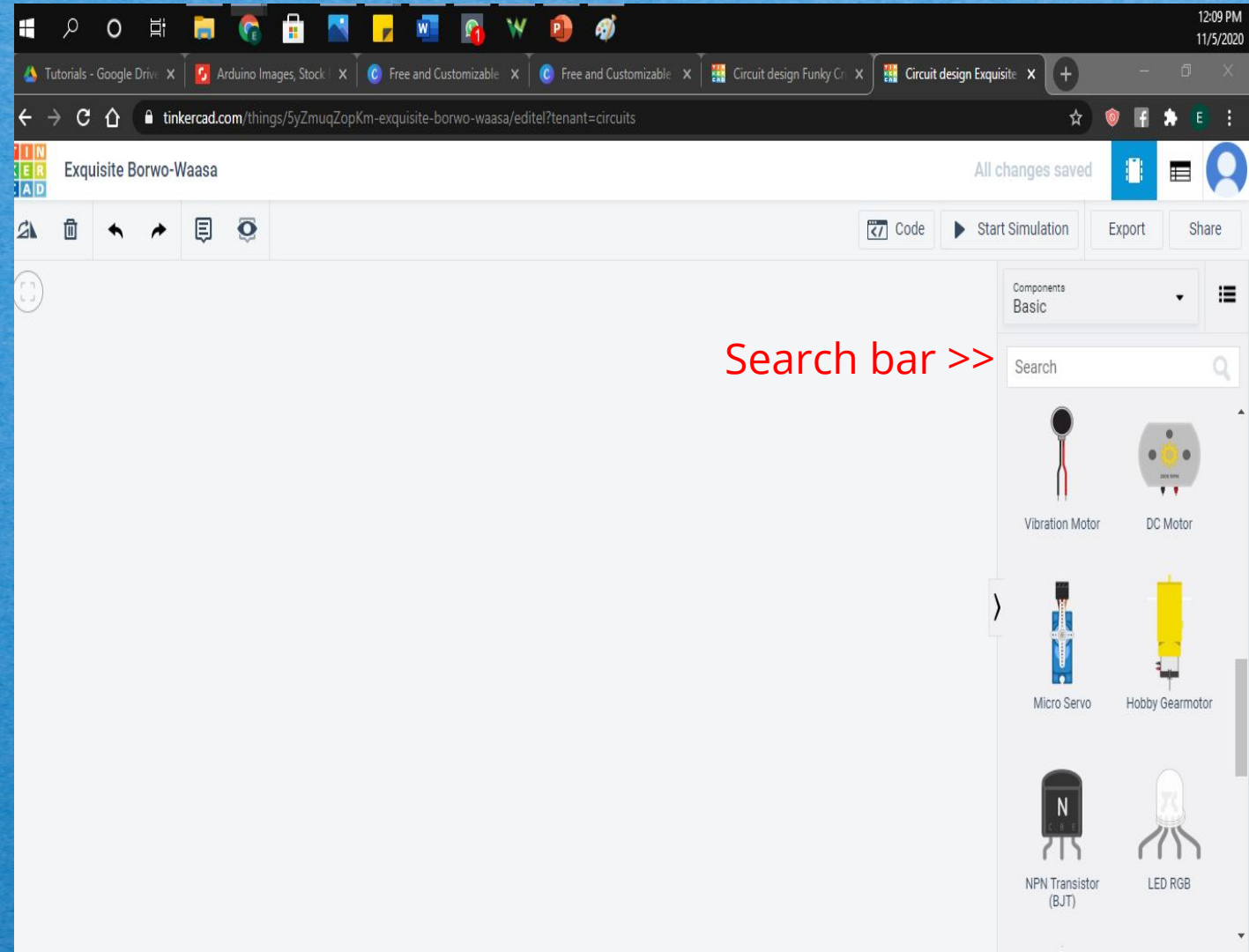
#2

- Next, we choose what USB port our Arduino is connected to by going to... **tools** >> **port** and clicking the port where our Arduino is connected to.
- Now we are ready to start working with the Arduino.



#3

- TinkerCAD is a simulation software which we are going to use to emulate the working of an Arduino.
- We can create circuits by going to the TinkerCAD website (<https://www.tinkercad.com/dashboard>), creating an account, and going to **circuits** >> **create new circuit**
- To create a circuit we can get the components or parts required, from the component window on the right. We can scroll down to see more components or by clicking on the search bar and typing in the name of the component we need.

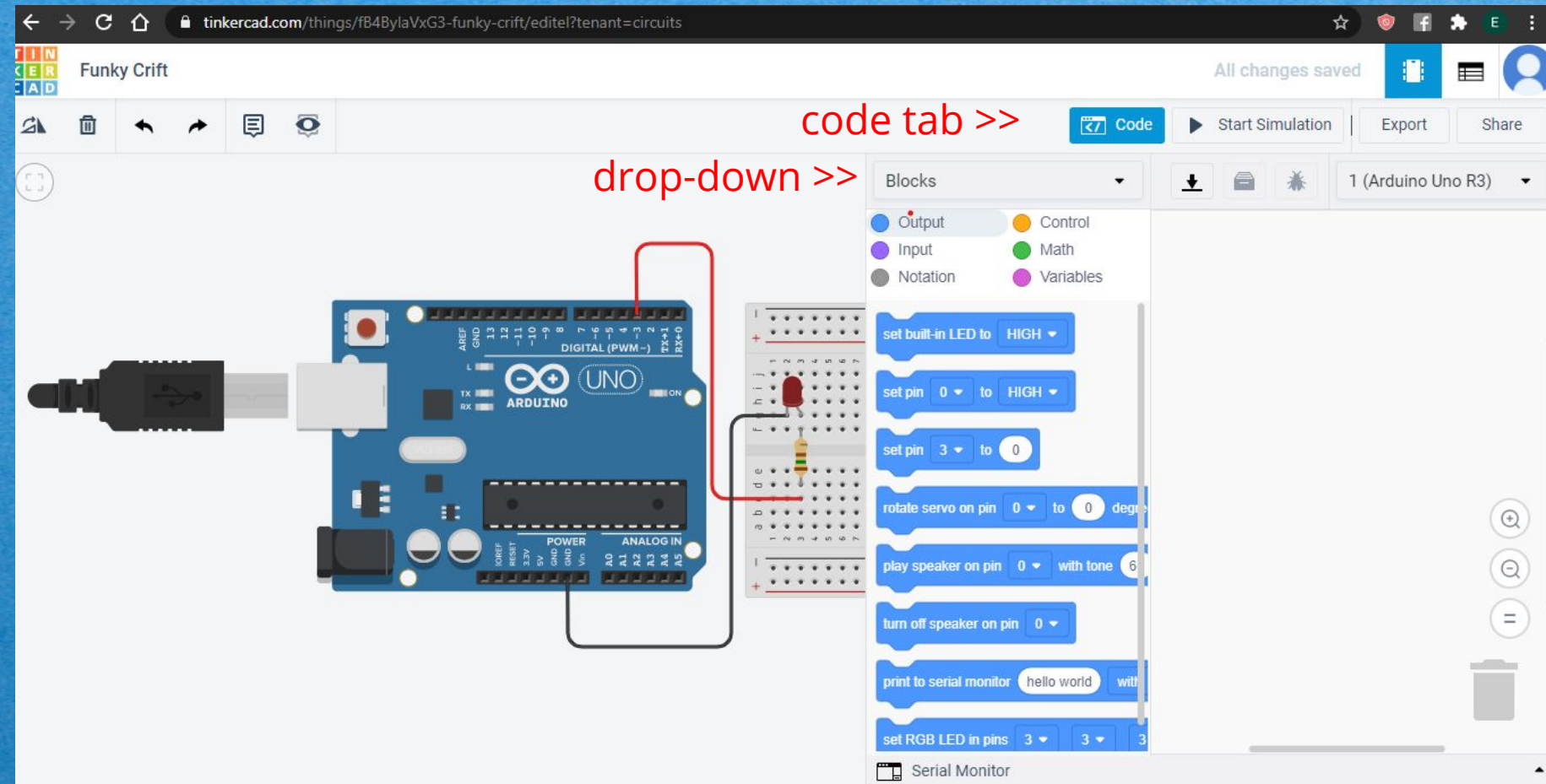


#4

- After selecting the components and connecting them, we are ready to program the Arduino.
 - TinkerCAD provides a simple way of coding Arduino by using “blocks” or “block method”.
 - Alternatively we could type in the code manually using the Arduino language. This is a “language of code” that the Arduino understands, and we use to tell it what to do.
- In the next slide, we will be seeing how to program the Arduino using the block method to create a simple L.E.D blink circuit.

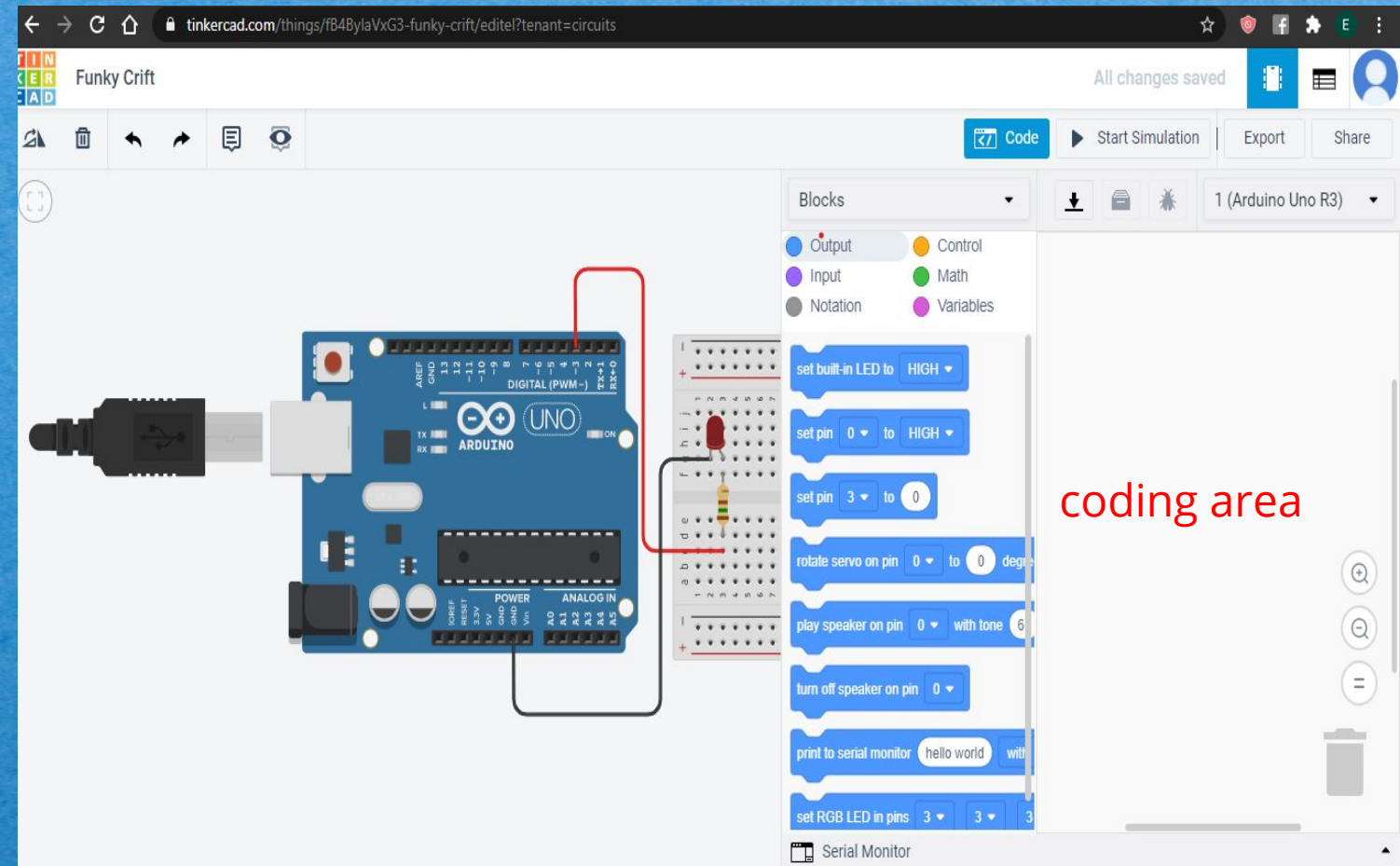
#5

- After placing the components on the workspace and connecting them, we click the “**code**” tab, and a window opens.
- There is a **drop-down** menu which we can use to choose how we want to program the Arduino, either with blocks, or text.



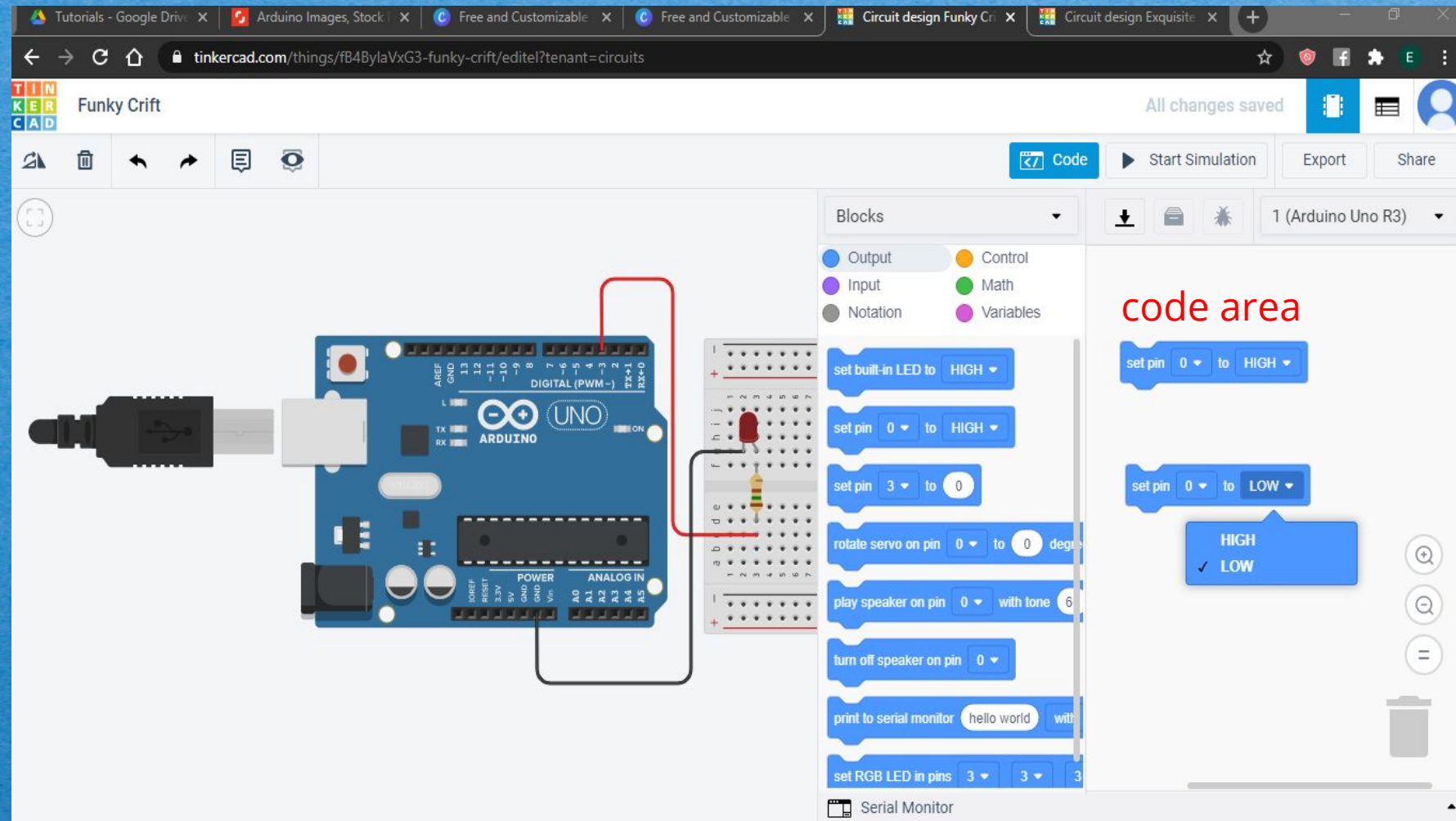
#6

- When programming the Arduino, it helps to visualize what we want to do.
- For this blink circuit, we want the pin 3, the pin where the L.E.D gets power from, to be in a “**HIGH**” state (or in an ON or 1 state) for 1 second and to be in a “**LOW**” state (or in an OFF or 0 state) for another second.
- Basically we want a **HIGH** output, a time **delay**, then a **LOW** output.
- Note that any code blocks we use, are going to run in a loop (or the process make is going to run continuously).



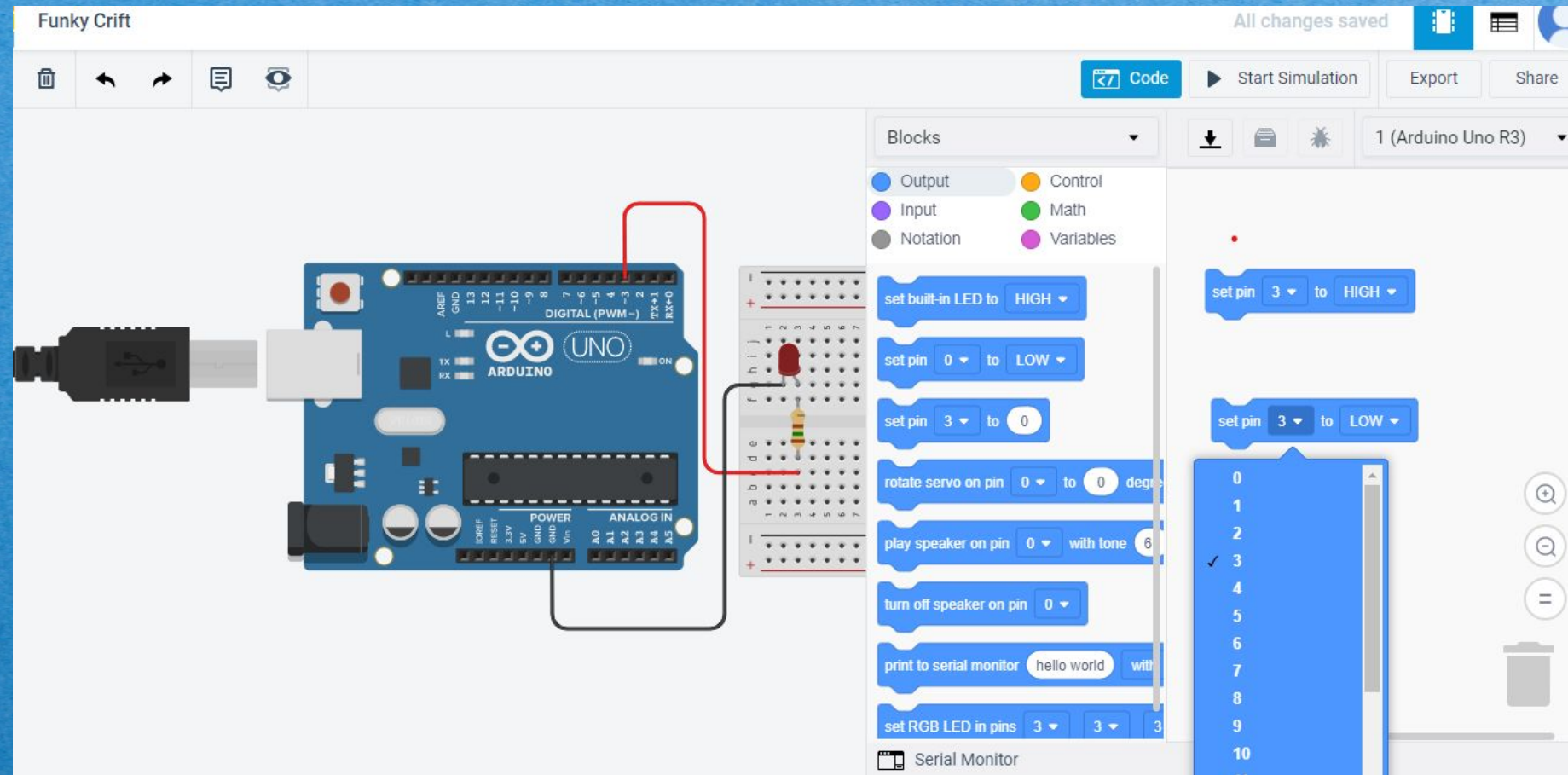
#7

- Now we understand what our process looks like. We need an output code block that turns the L.E.D on. So we drag the code block that says, "set pin 0 to HIGH" unto the code area
- We do the same thing again, but this time we click the area that says "HIGH", a menu pops up and we select "LOW" this is output that turns the L.E.D off



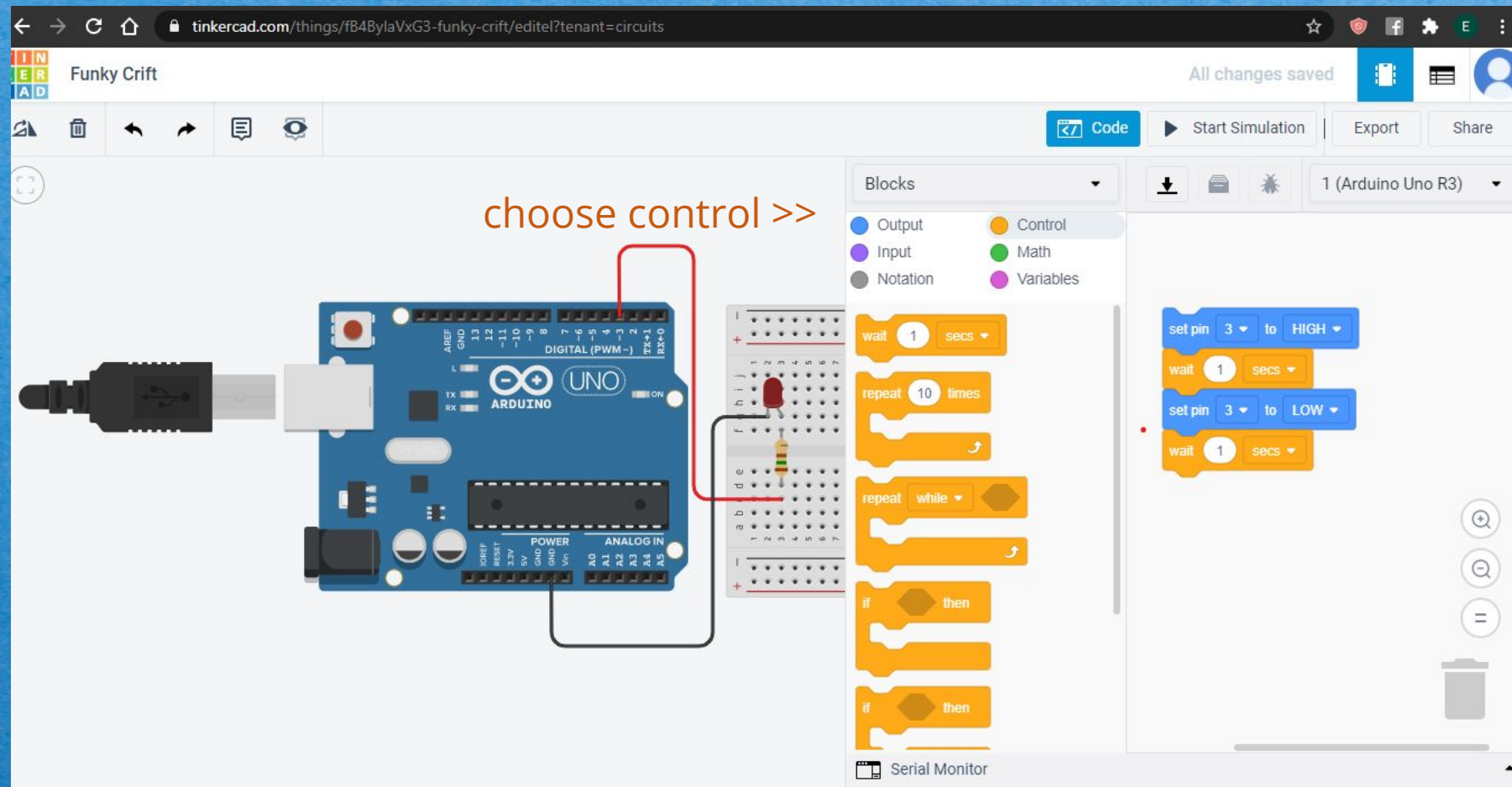
#8

- We should remember to choose the pin we want these signals to go to by clicking the drop-down menu on the code block and choosing the correct pin number, which is the one we have connected.



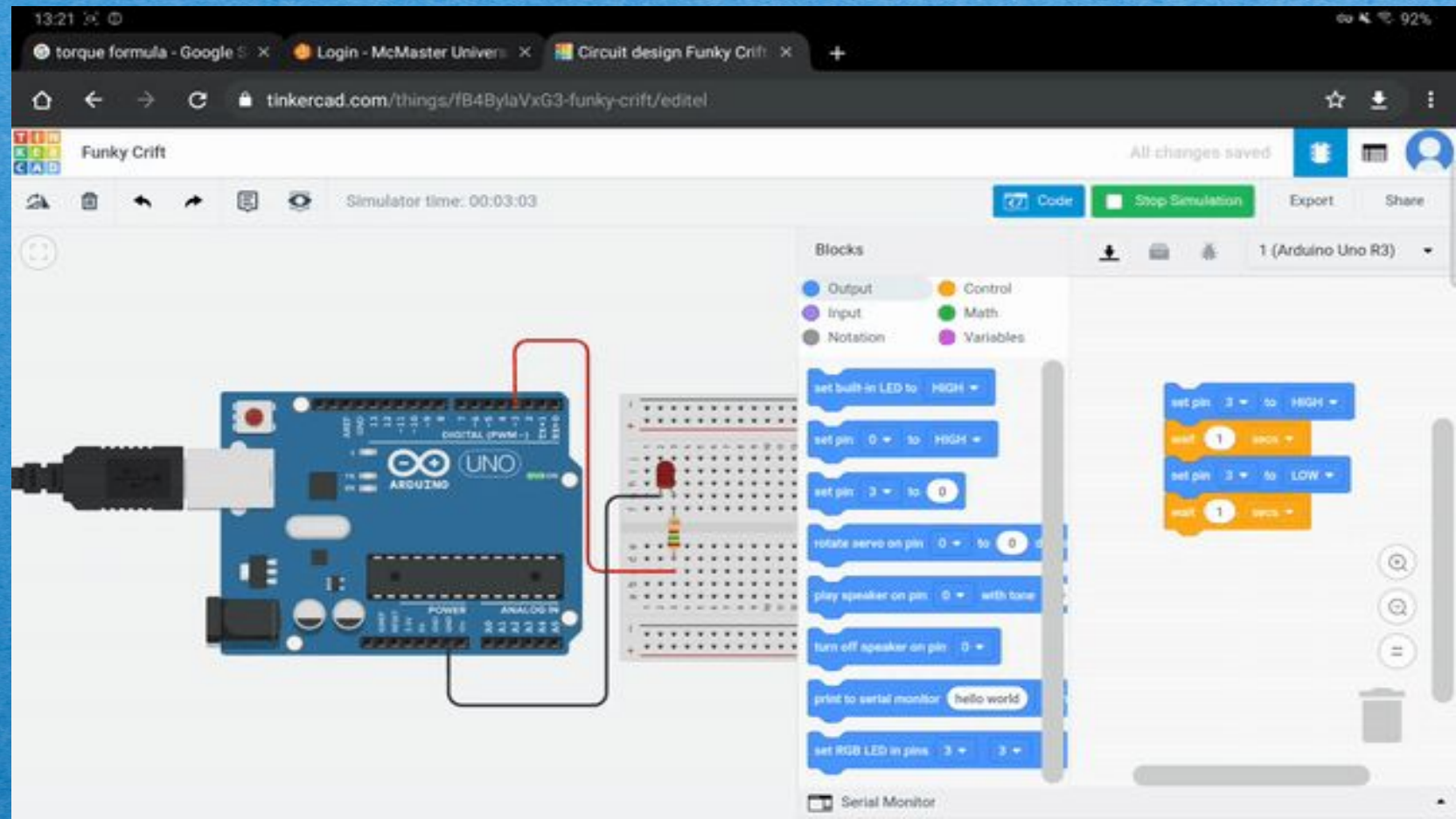
#9

- We are almost done, but we remember that we need a **HIGH** output, a **LOW** output and, two **time delays**, that control how long the L.E.D stays on and off for.
- Now we go to the “**control**” section under the blocks tab and drag the block that says “**wait 1 secs**” onto the code area.
- We do this twice to have two delays on the code area.



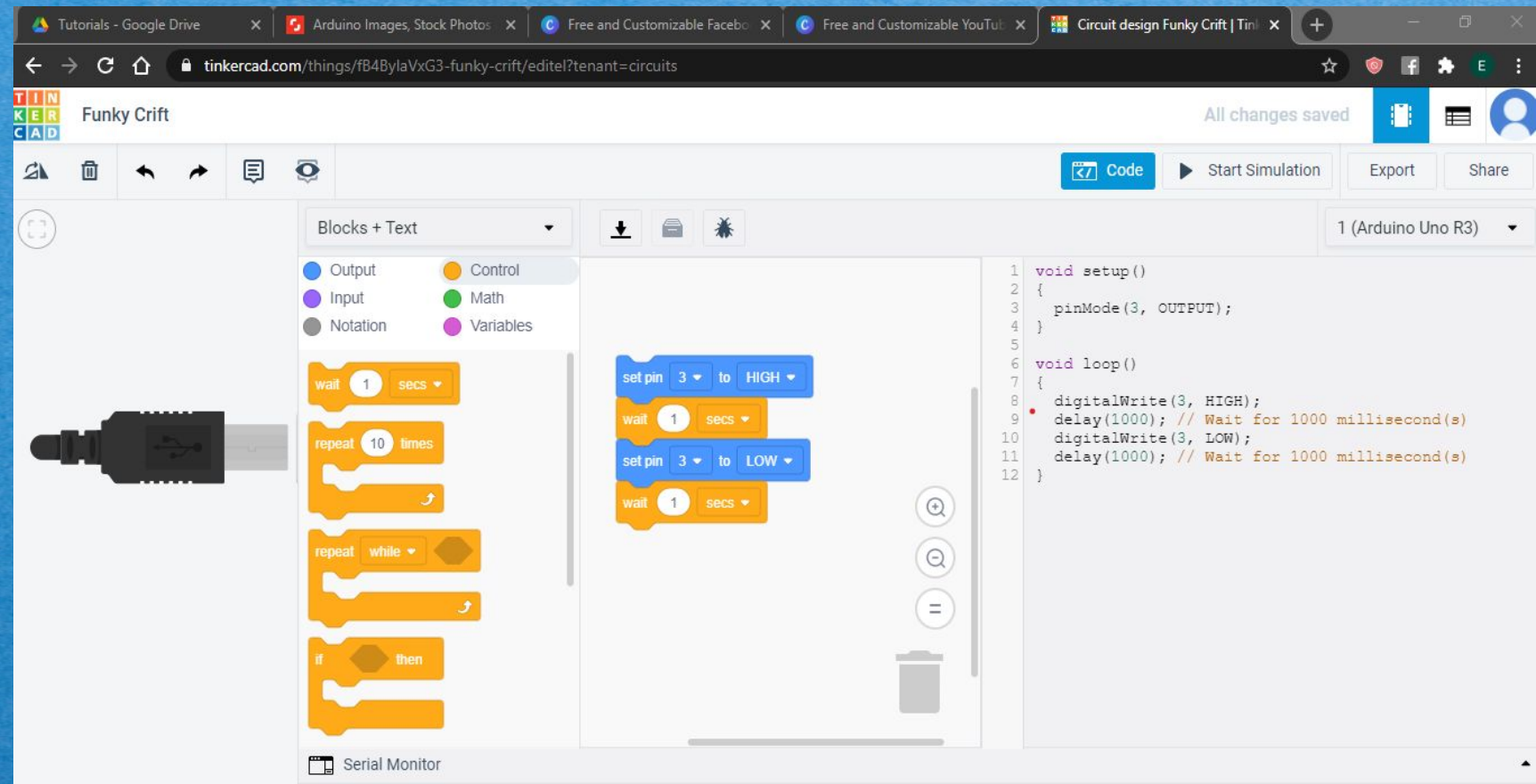
#10

- We are all done building the circuit! Click “so we click start simulation” to see the circuit in action.
- Remember that order matters, the blocks are placed in this order; to send a high signal to pin 3, wait for 1 second, send a low signal to pin 3, and wait another second.
- Play around with the timing, order of the blocks and even add more blocks to get a better understanding of how things work.



#11

- No we understand how the Arduino works; we can see how to program it using the Arduino language. Click the “**Blocks**” drop-down and choose “**Blocks + Text**”
- The code on the right is what we would use to program the Arduino board if we were using the Arduino IDE and a physical Arduino to create this circuit.



The screenshot displays the Tinkercad web interface. On the left, a circuit diagram shows a USB-A to USB-B cable connected to a board. The central workspace contains a block-based program with the following sequence: a 'wait 1 secs' block, a 'repeat 10 times' loop containing a 'set pin 3 to HIGH' block and another 'wait 1 secs' block, followed by a 'repeat while' loop and an 'if then' block. A 'Blocks + Text' dropdown menu is open, showing categories: Output (blue), Input (purple), Notation (grey), Control (orange), Math (green), and Variables (pink). On the right, the code editor shows the following C++ code:

```
1 void setup()
2 {
3   pinMode(3, OUTPUT);
4 }
5
6 void loop()
7 {
8   digitalWrite(3, HIGH);
9   delay(1000); // Wait for 1000 millisecond(s)
10  digitalWrite(3, LOW);
11  delay(1000); // Wait for 1000 millisecond(s)
12 }
```

At the bottom, a 'Serial Monitor' tab is visible.

- After working with the blocks, we can hit the download button on the deck and have the already coded Arduino language version downloaded into our computer.

Take a break at this point to review what we've learned and re-energize!

fun_krift1 | Arduino 1.8.13 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help

fun_krift1

```
// this is how the arduino IDE looks.
// *note that anything written after "/" which is basically just a double forward slash will not affect the code. we call it commenting.
// *note that arduino language is case sensitive.

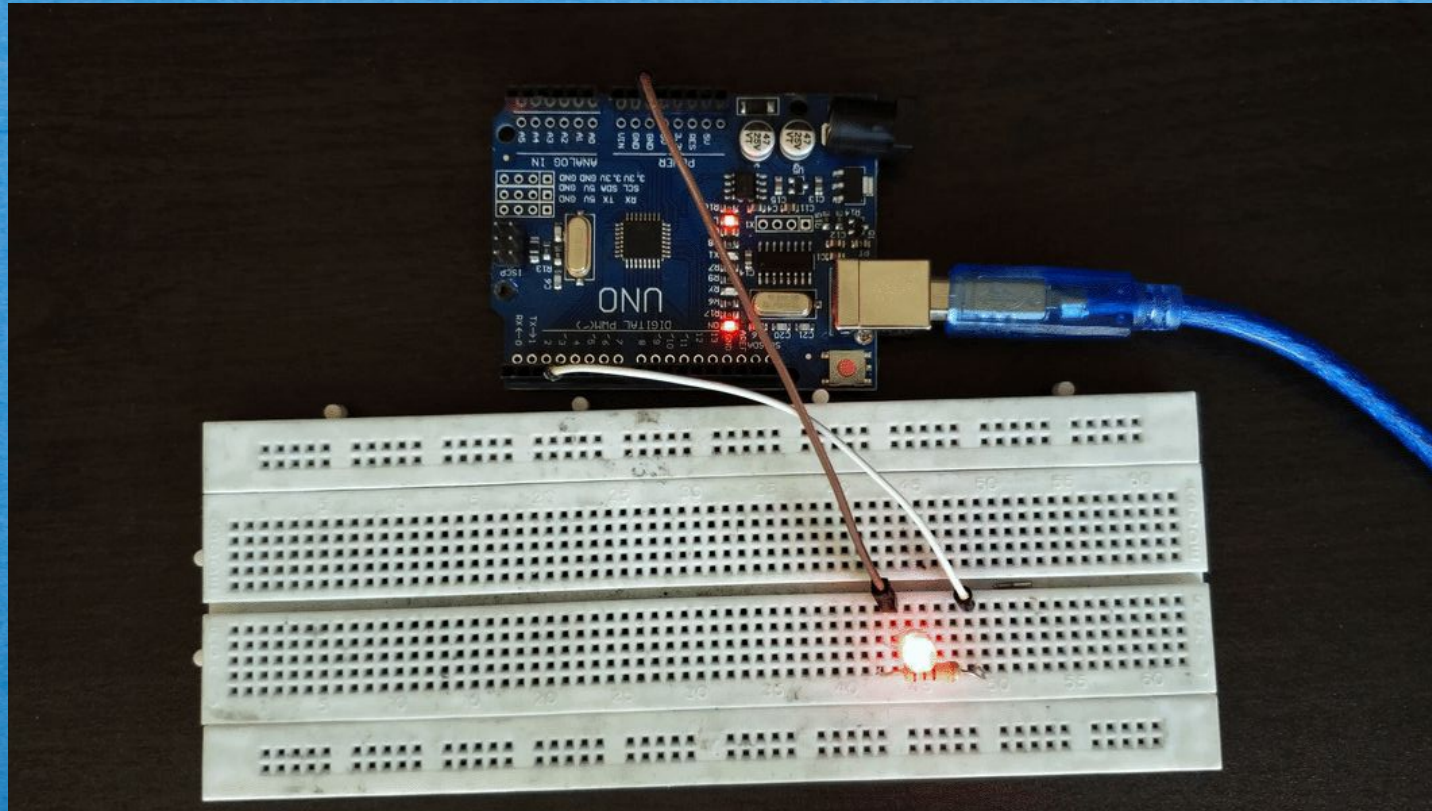
void setup() // code inside this function runs only once, and is used to setup the arduino and determine how the arduino behaves in this project. You want to always have this in your project.
{
  Serial.begin(9600); // This sets the Baud rate that the Arduino works with.
  pinMode(3, OUTPUT); // this piece of code configures the arduino pin 3 to be an output pin. we always have to set the behaviour of the pins we use.
} // a curly bracket at the end of a function closes the function. think of it like the back door of a house.

void loop() // code inside this function runs continuously, it does not stop. This is where our main code usually enters in projects.
{
  digitalWrite(3, HIGH); // digitalWrite is used to determine the state of a pin, by writing; digitalWrite(pin number, state). this sets pin 3 "HIGH" or "1" state or "ON"
  delay(1000); // Wait for 1000 millisecond(s). this is a delay
  digitalWrite(3, LOW); // this sets pin 3 "LOW" or "0" state or "OFF"
  delay(1000); // Wait for 1000 millisecond(s). this is a delay
}
```

With the Arduino connected to the computer, when we are done coding, we hit the “**verify**” button to make sure our work has no errors, then we hit the “**upload**” button to send the code to the Arduino.

Any error messages appear here!

- This is a physical example of the project.
It uses the same principles we learned!!



Important Information

[Parts list](#)

Team registrations are open!!