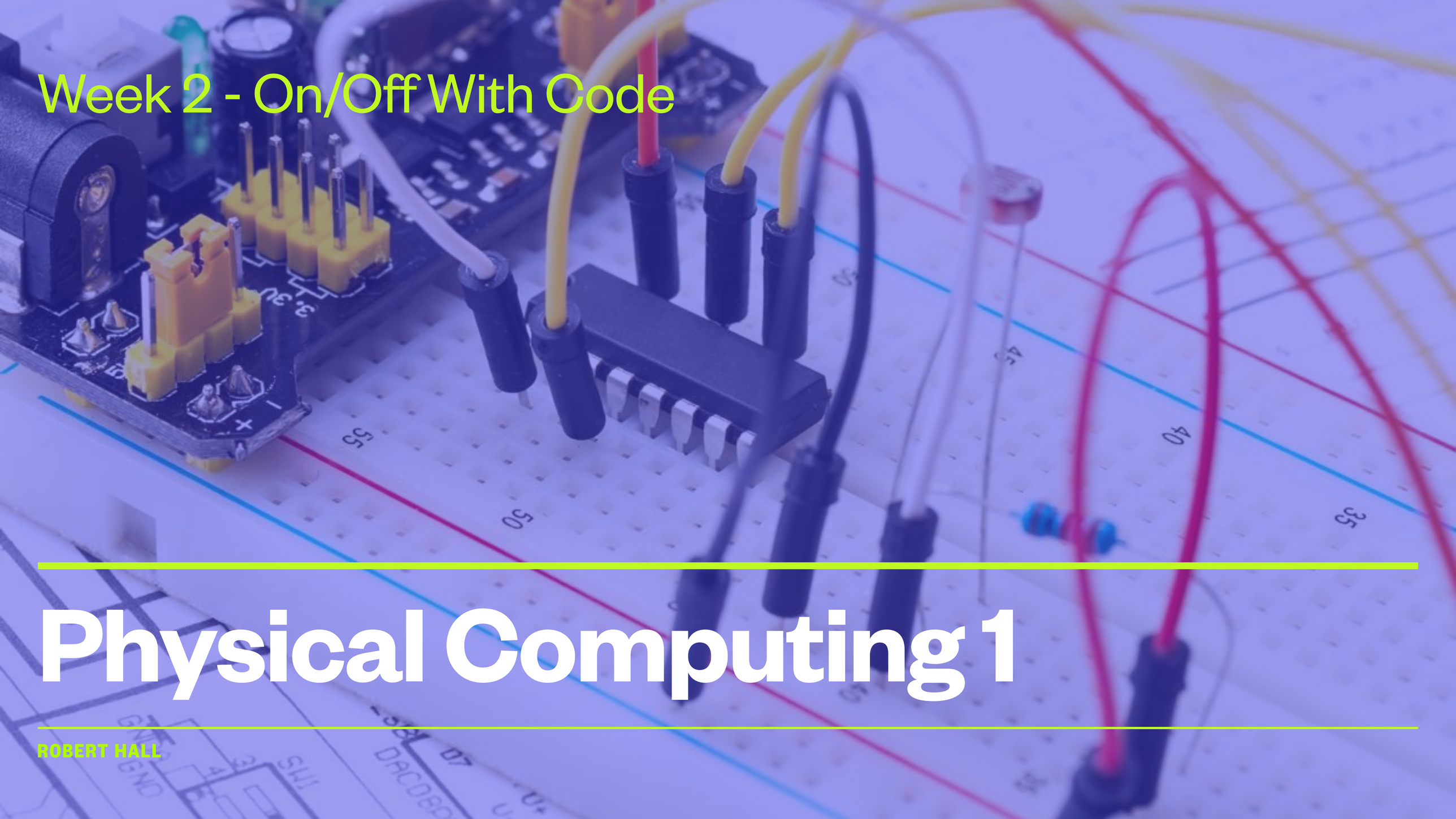


Week 2 - On/Off With Code

Physical Computing 1

ROBERT HALL

Sign The Register!

- Scan the QR Code

Sign Up
HatchLab Inductions

- https://learn.gold.ac.uk/course/view.php?id=6947
- You MUST Sign Up for Inductions to be ready when we use the machines in class.
- You don’t need to sign up for SOLDERING as you have done that in this class.

Legacy IDE (1.8.X)

Download the IDE
You'll need it later!

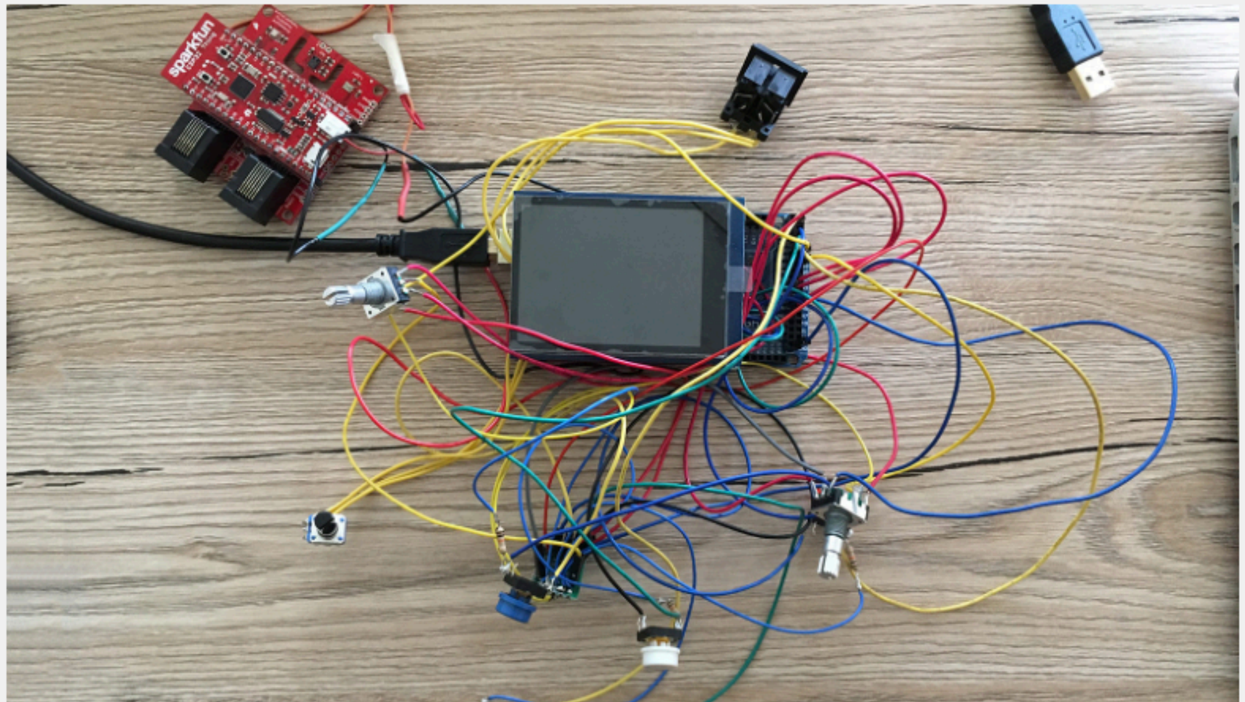
- Download Arduino IDE
- https://www.arduino.cc/en/software
- You can install either version but the old version will match the slides!

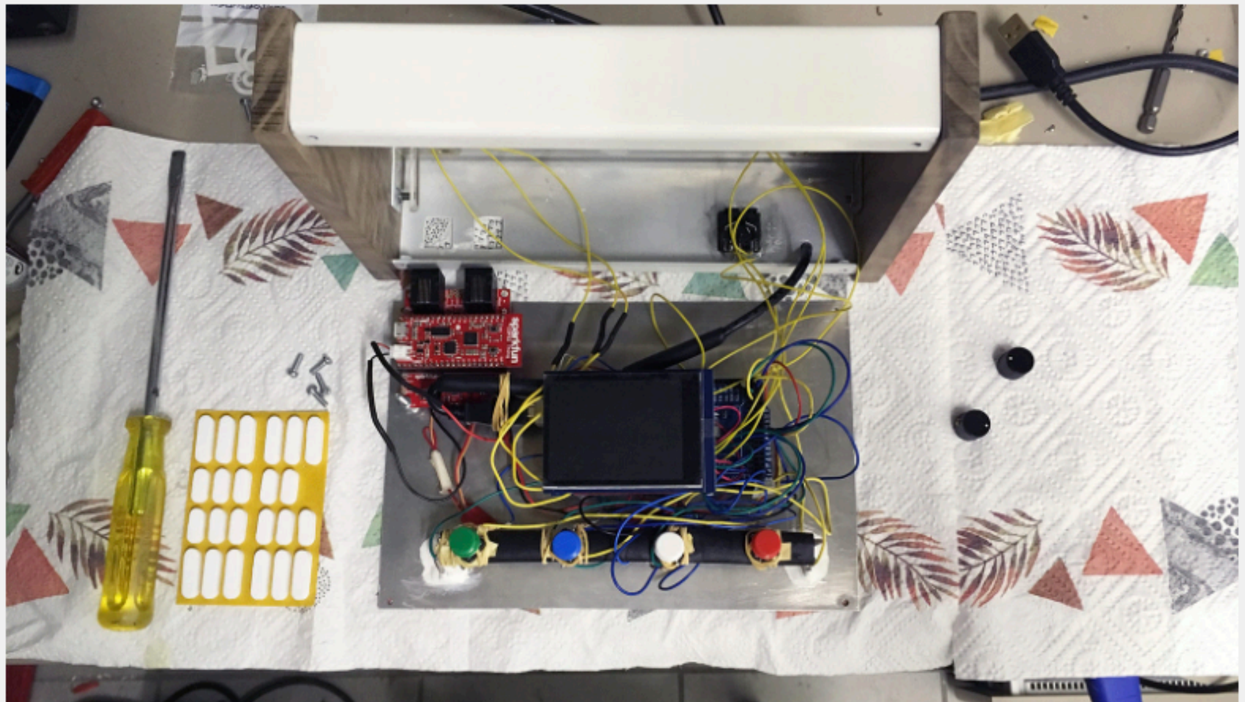
Any Questions so far?

Inspiration Time!

How was this done?

4


sparkfun



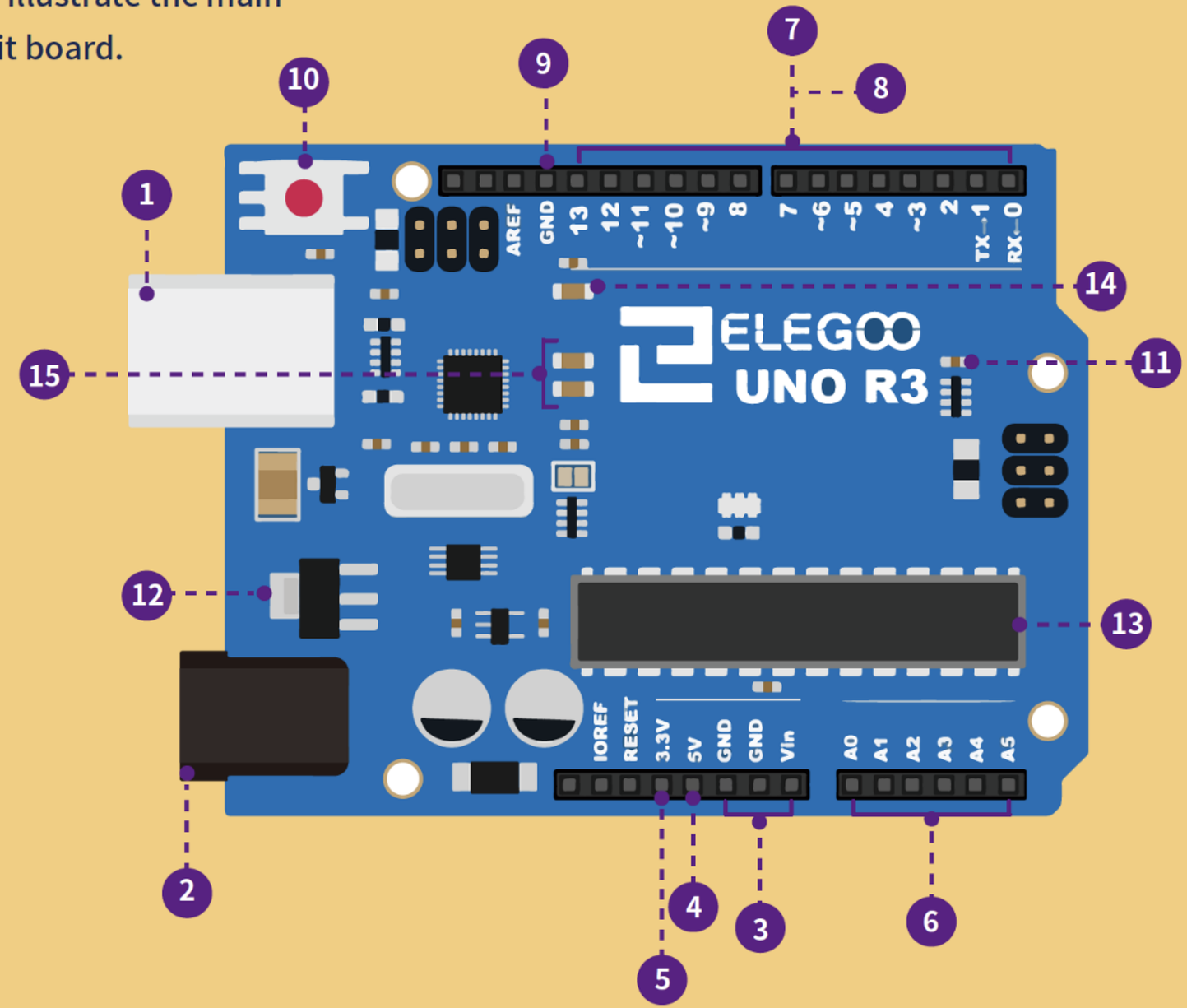
Plan for Today

- Recap of Last Week
- Multimeters
- Code + Electronics
- Practical Exercises
- Schematics
- Idea Generation
- 1st Assignment

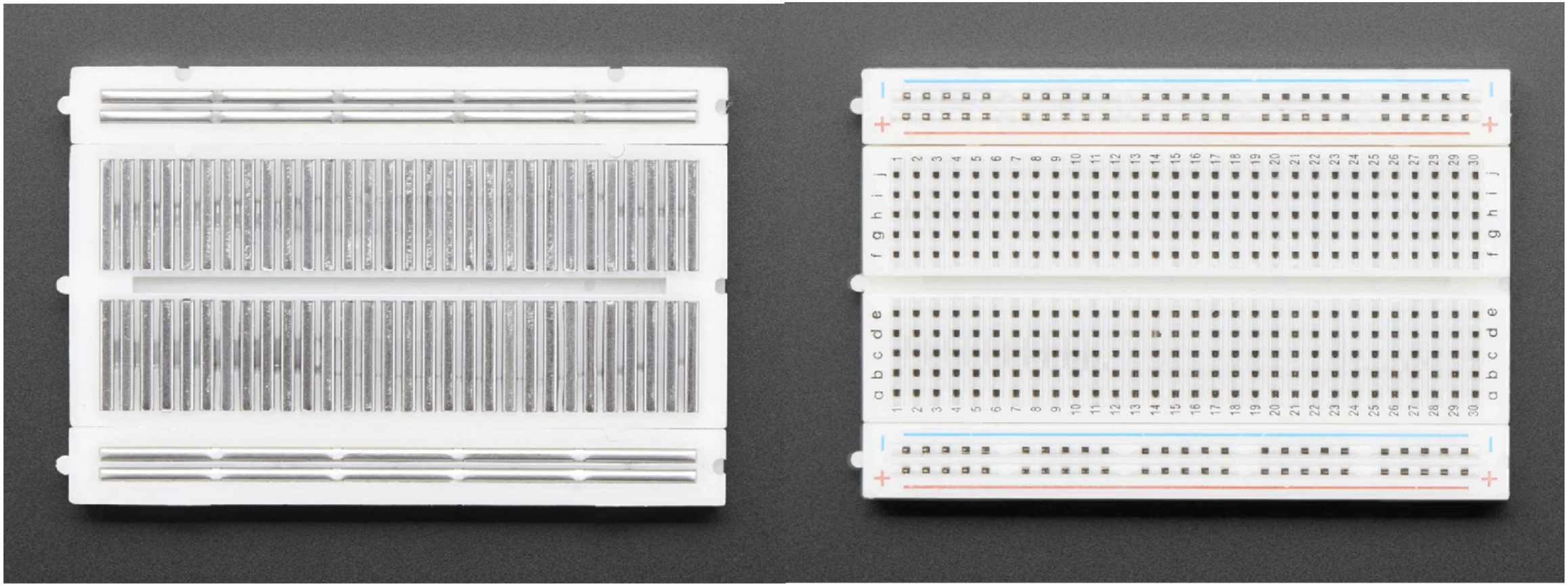
3.Introduction of board

We use elegoo uno as an example to illustrate the main circuit components of Arduino circuit board.

① the USB connection
② the 6V~12V barrel jack
③ GND
④ 5V pin supplies
⑤ 3.3V pin supplies
⑥ analog pin
⑦ digital pin
⑧ (pwm~)
⑨ GND
⑩ Reset Button
⑪ Power LED Indicator
⑫ Voltage Regulator
⑬ Main IC
⑭ Pin D13 indicator LED
⑮ Serial communication indicator LEDs

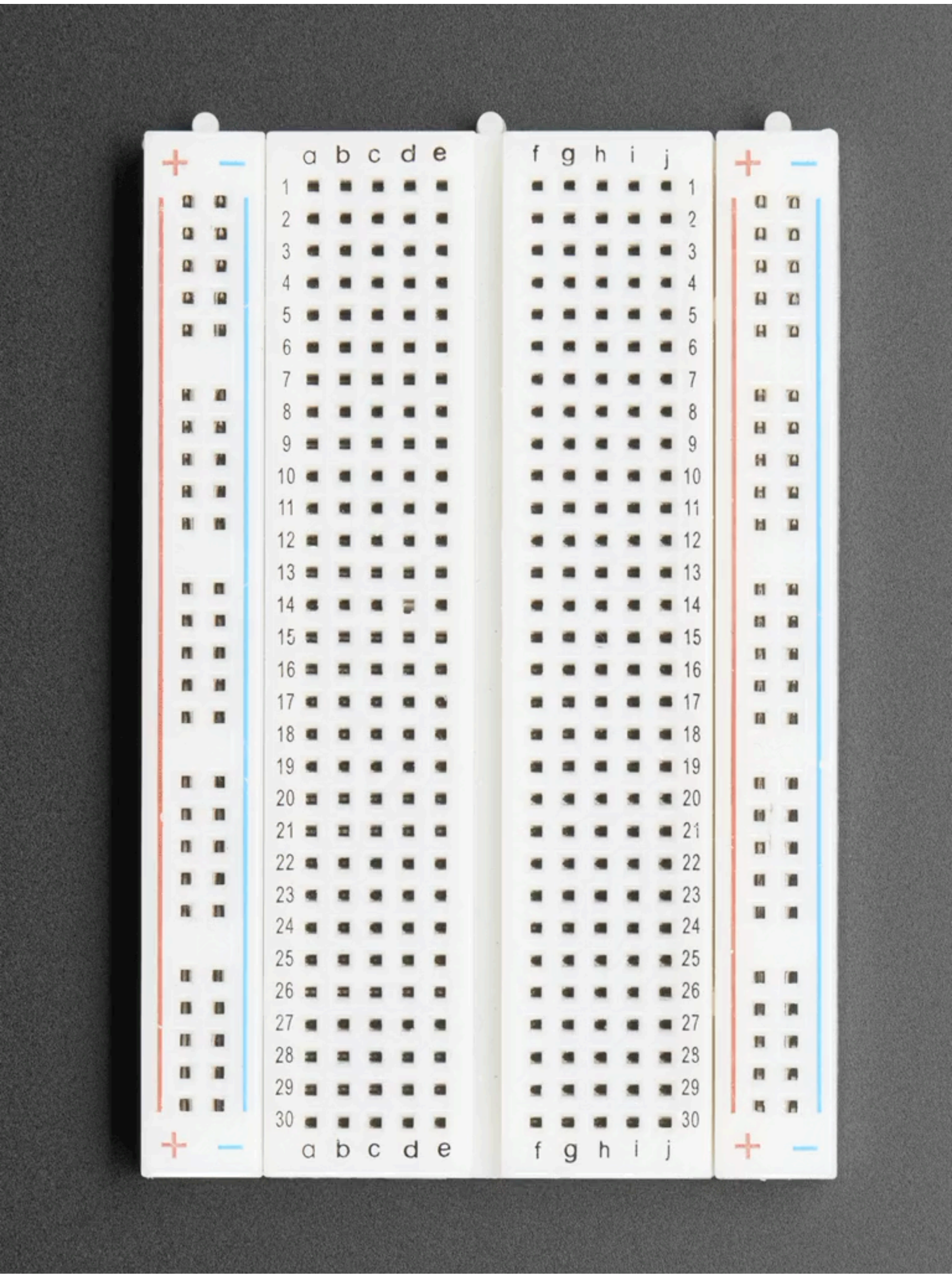


Breadboard

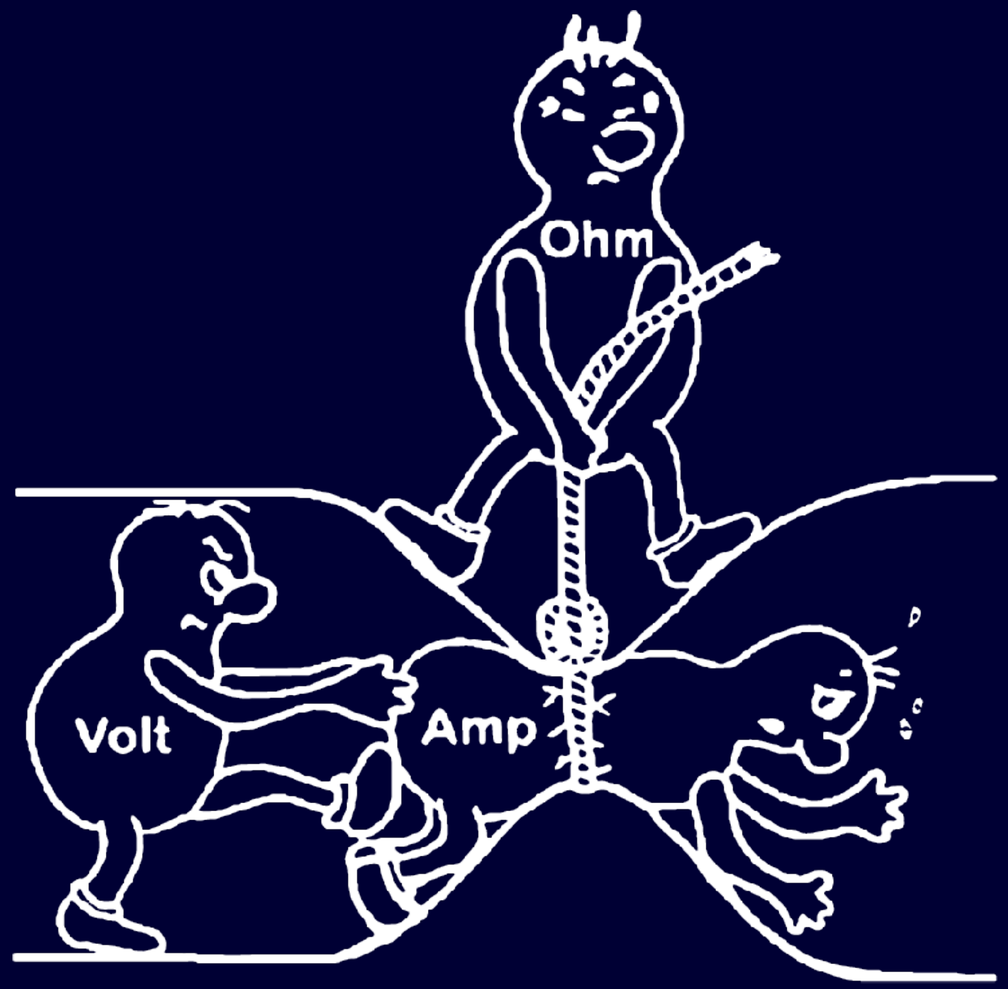


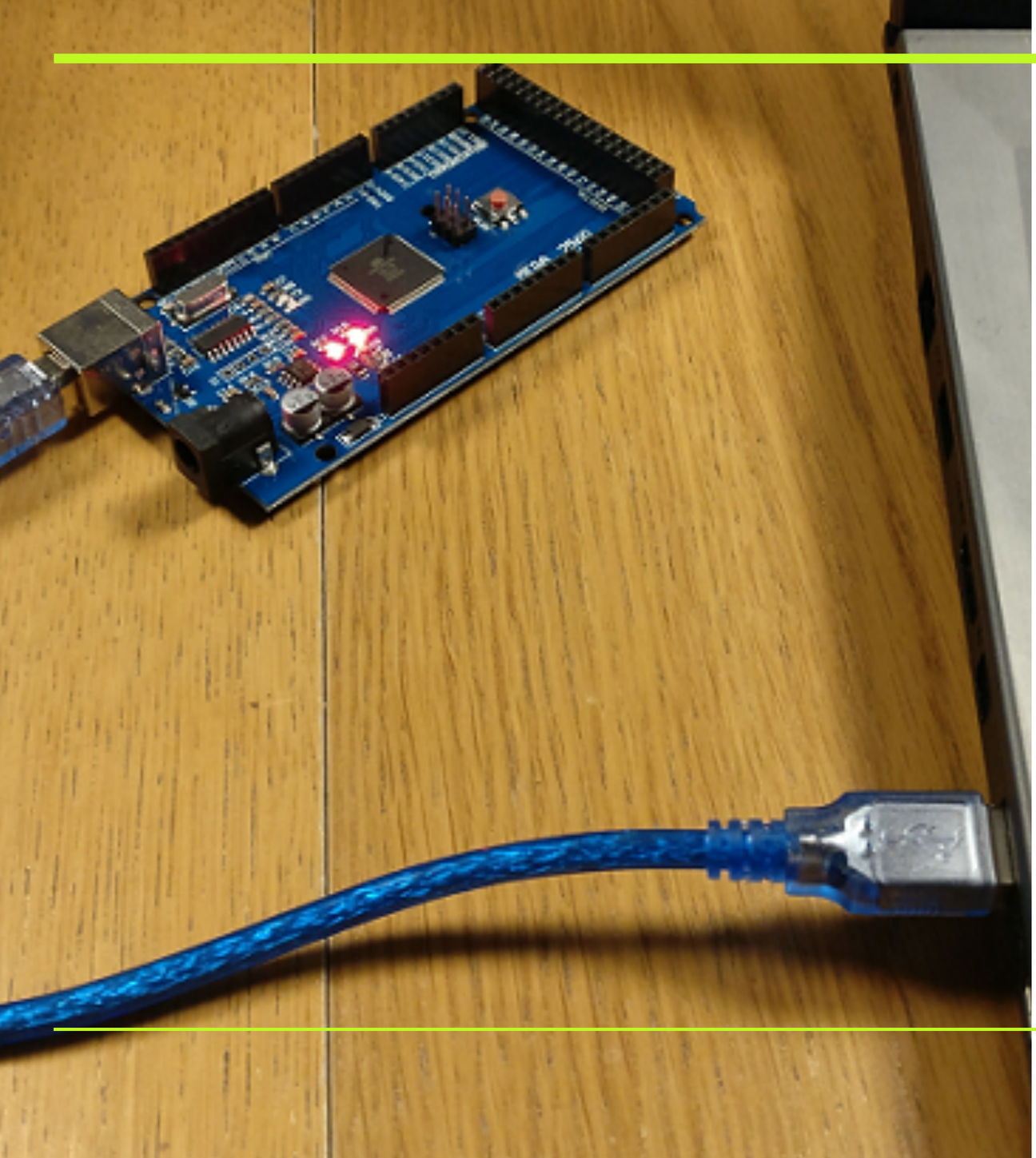
Inside

Outside


a b c d e
f g h i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Electricity!



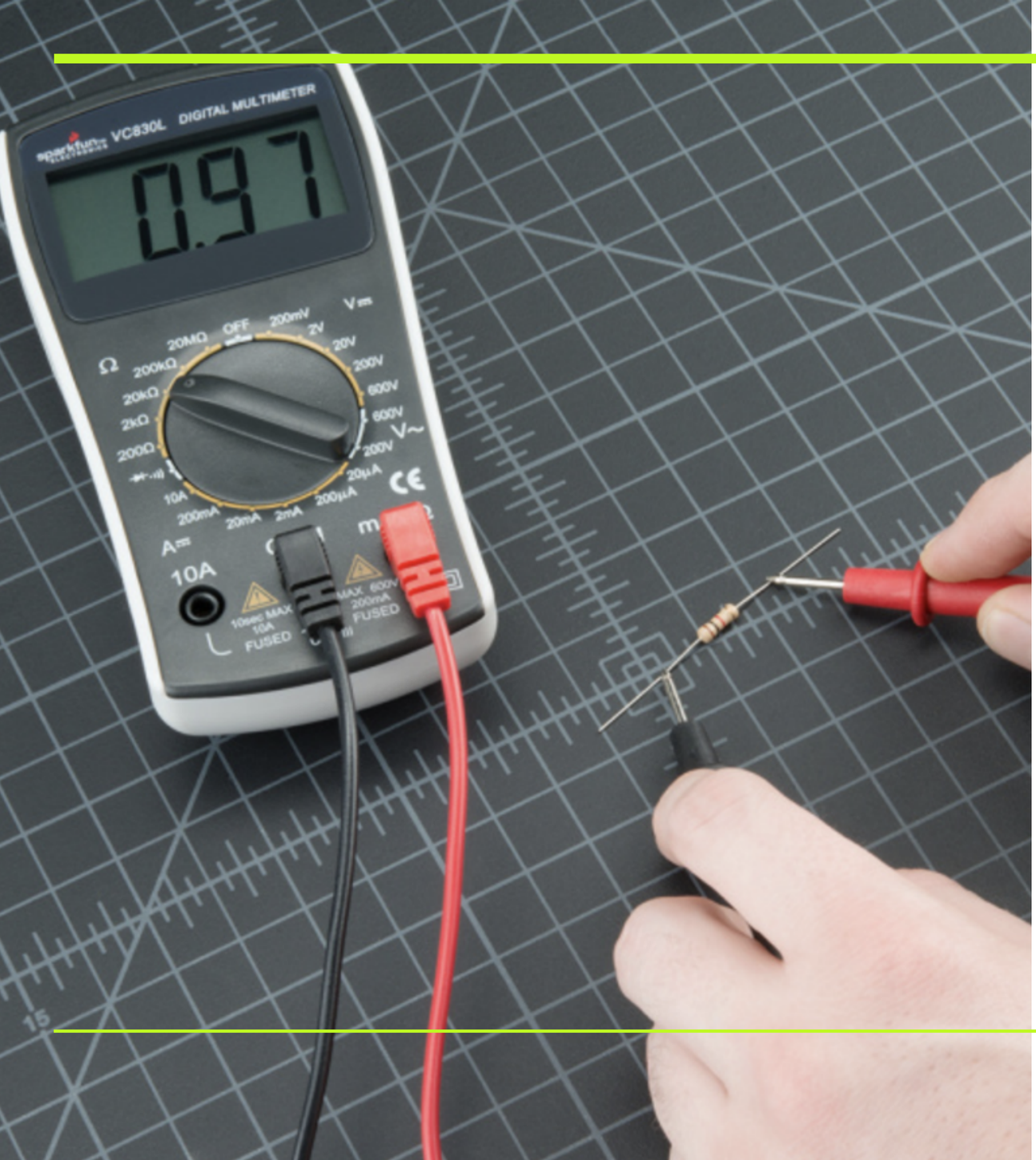


Arduino Remembers!

- Arduino can work on it’s own, once you have uploaded your code you can unplug it from your computer.
- Once it is connected to power it will run the code whether it is connected to the computer or not.

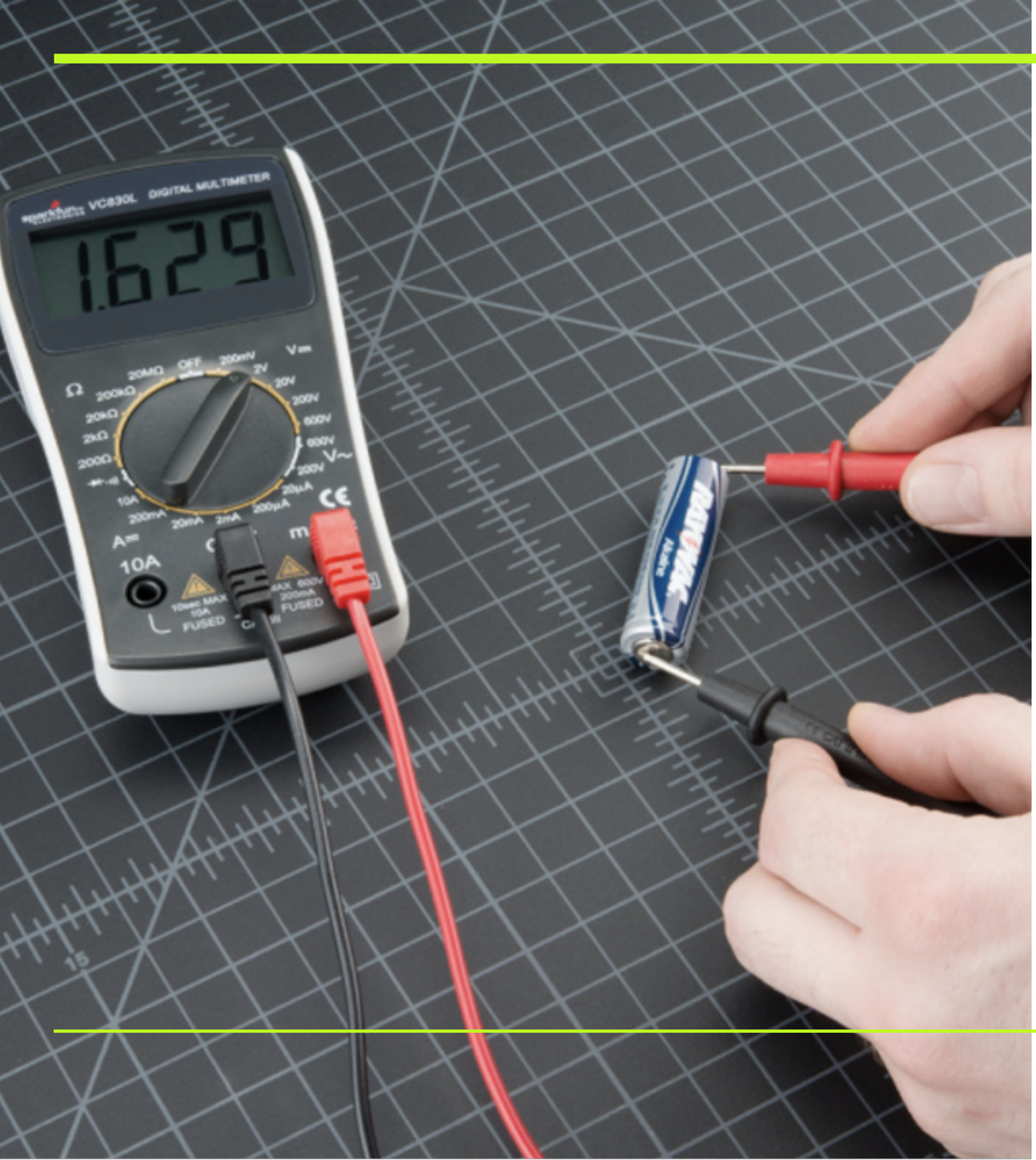
Multimeter

- This is a really common tool in electronics.
- You will use it to :
- Test Continuity
- Find value of Resistors
- Find voltage of Power Supplies
- Find voltage drop across a component
- Find current used by component



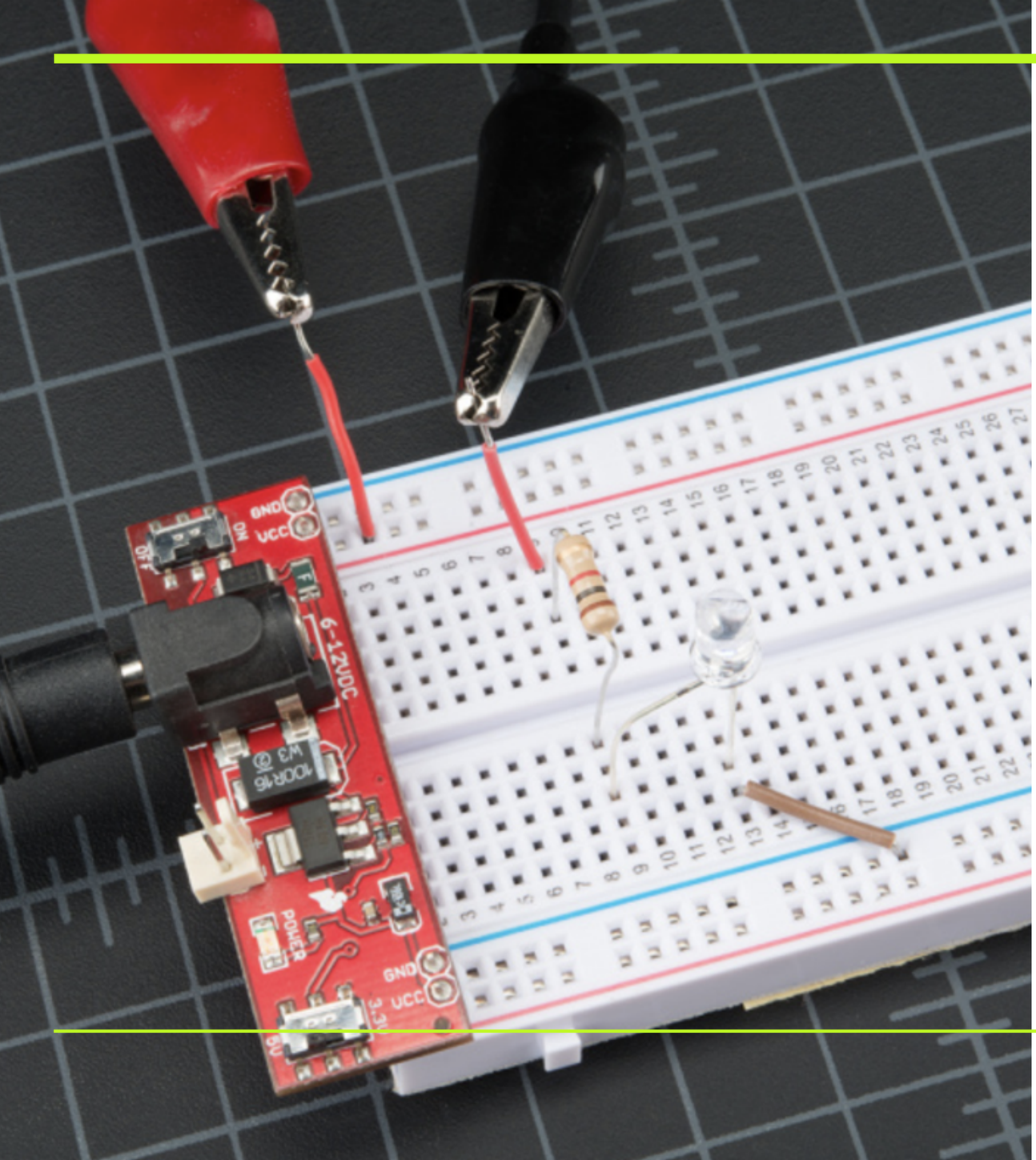
Measuring Resistance

- Select the Ohm (Ω) setting on the multimeter and place the probes on either side of the resistor.
- You might need to change the sensitivity by selecting different settings under the Ohm (Ω) bracket.
- Try measuring some different resistors!



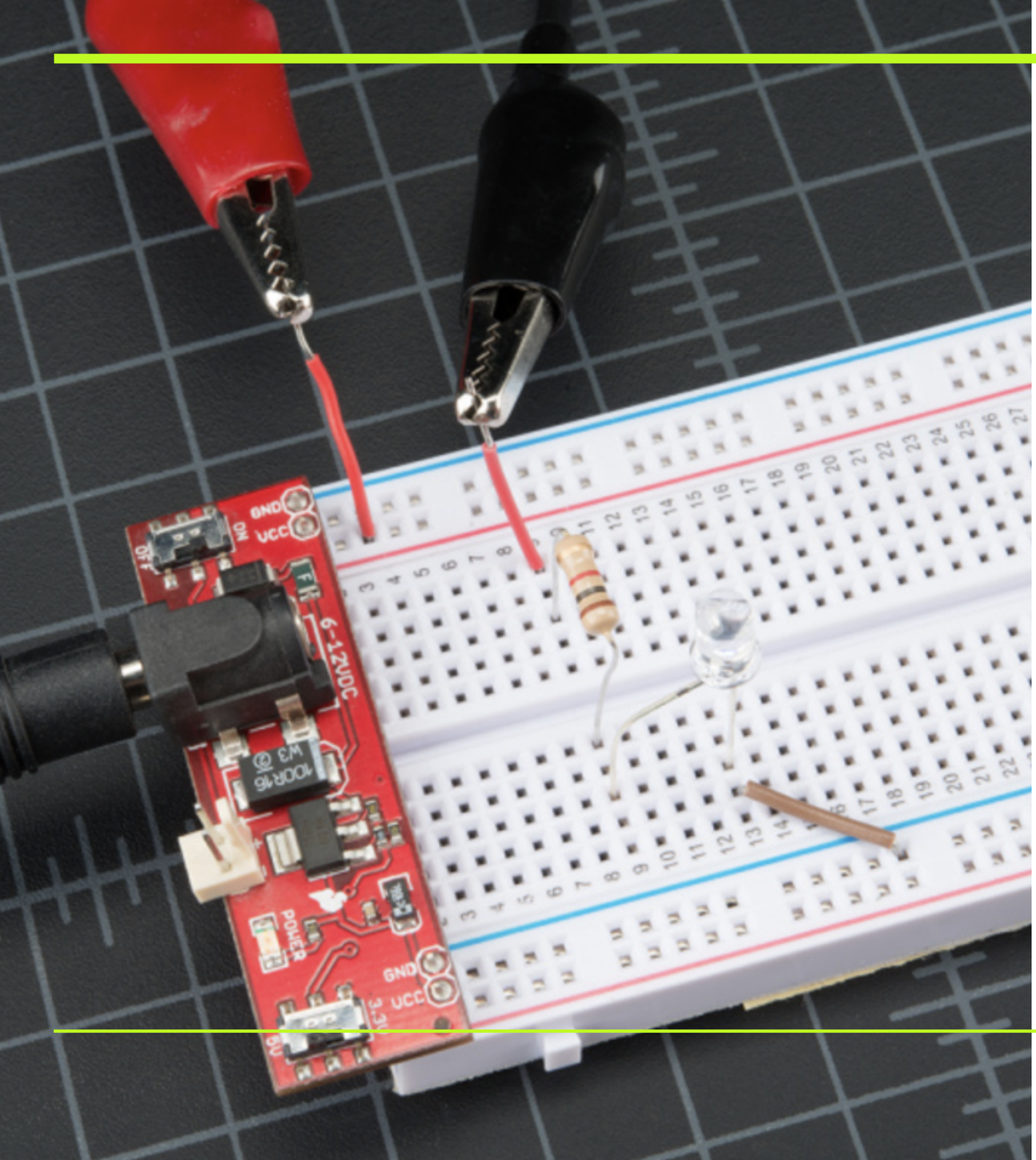
Measuring Voltage

- Select the Volts (V) mode on the multimeter and place the probes on either contact of the battery in your kit.
- Now try using wires to connect the probes to 5V and GND on your Arduino!



Measuring Current

- Measuring Current works slightly differently. We measure current in SERIES rather than PARALLEL.
- The easiest way to measure current is to use one of the Bench Power Supplies in the lab.
- Just power your project from that power supply and the display will show you how many amps are being used.

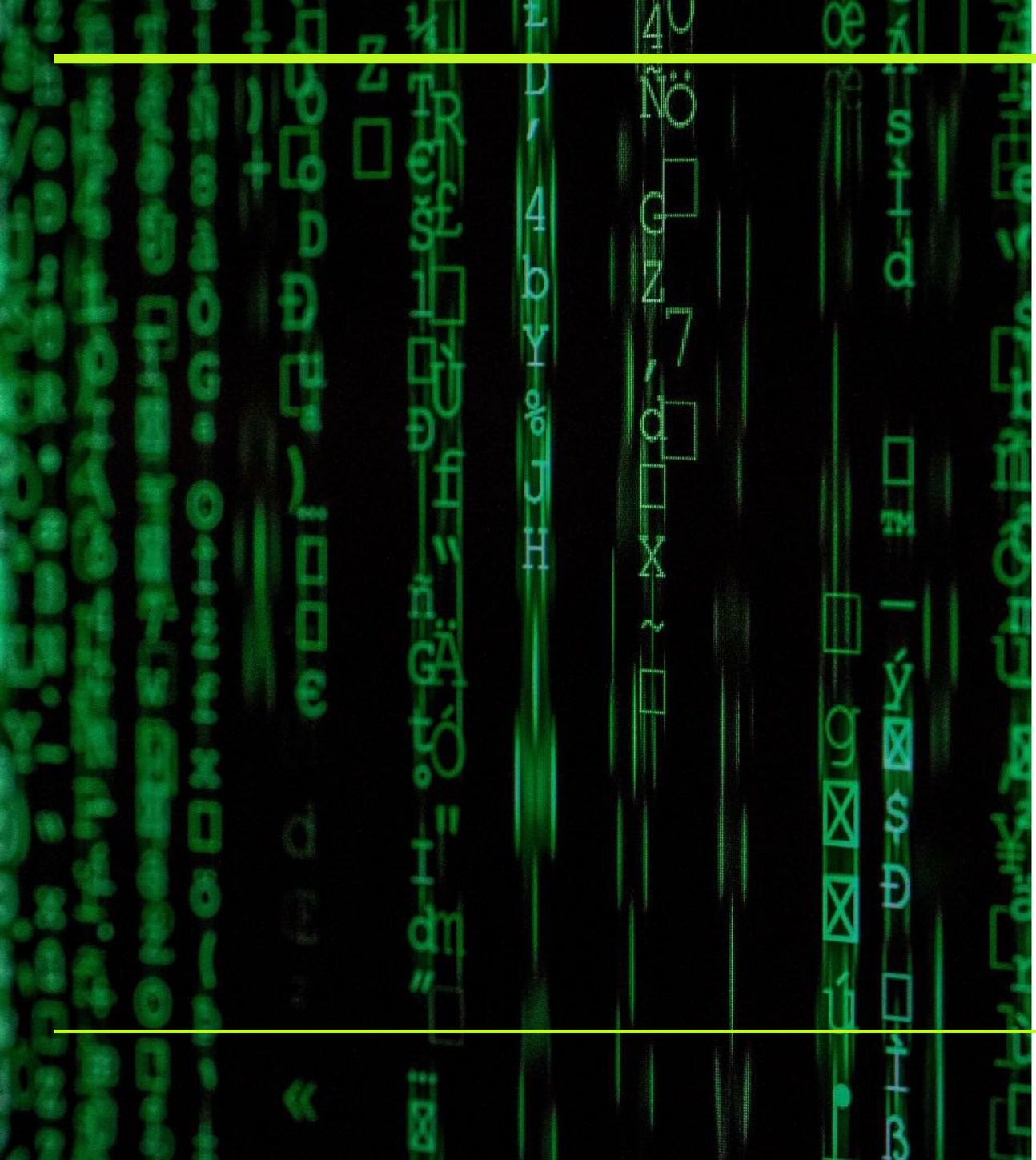


Measuring Current

- This isn’t something you really need to worry about at this point, nothing in your kit uses more current than your Arduino can supply on it’s own, it’s only when you want to use several of something that you need to start measuring current.

Demo

Now For Code!



The Structure of Code

- Actions
- Memory
- Decisions
- Loops (Next Week)

Action

Statement :
An individual instruction, usually a function call or setting a variable to a number.

```
int ledLevel = 8;

analogWrite(ledLevel);
```

Action

Functions :
Re-usable code blocks. Take arguments as inputs that help them know what to do.

```
void setup() {
 pinMode( 1 , OUTPUT ) ;
 delay(200);
}
```

Memory

Variables :
Ways of storing and keeping track of values (numbers, text, anything else!)

You can think of them as folders that you can put information in.

Have a specific TYPE unlike in P5.js

Memory

Variables in Arduino look different from what you might be used to from P5.js

In P5.js you might say "`var number = 5;`"

In Arduino you have to specify what TYPE of variable you have.
"`int number = 5;`"

Memory

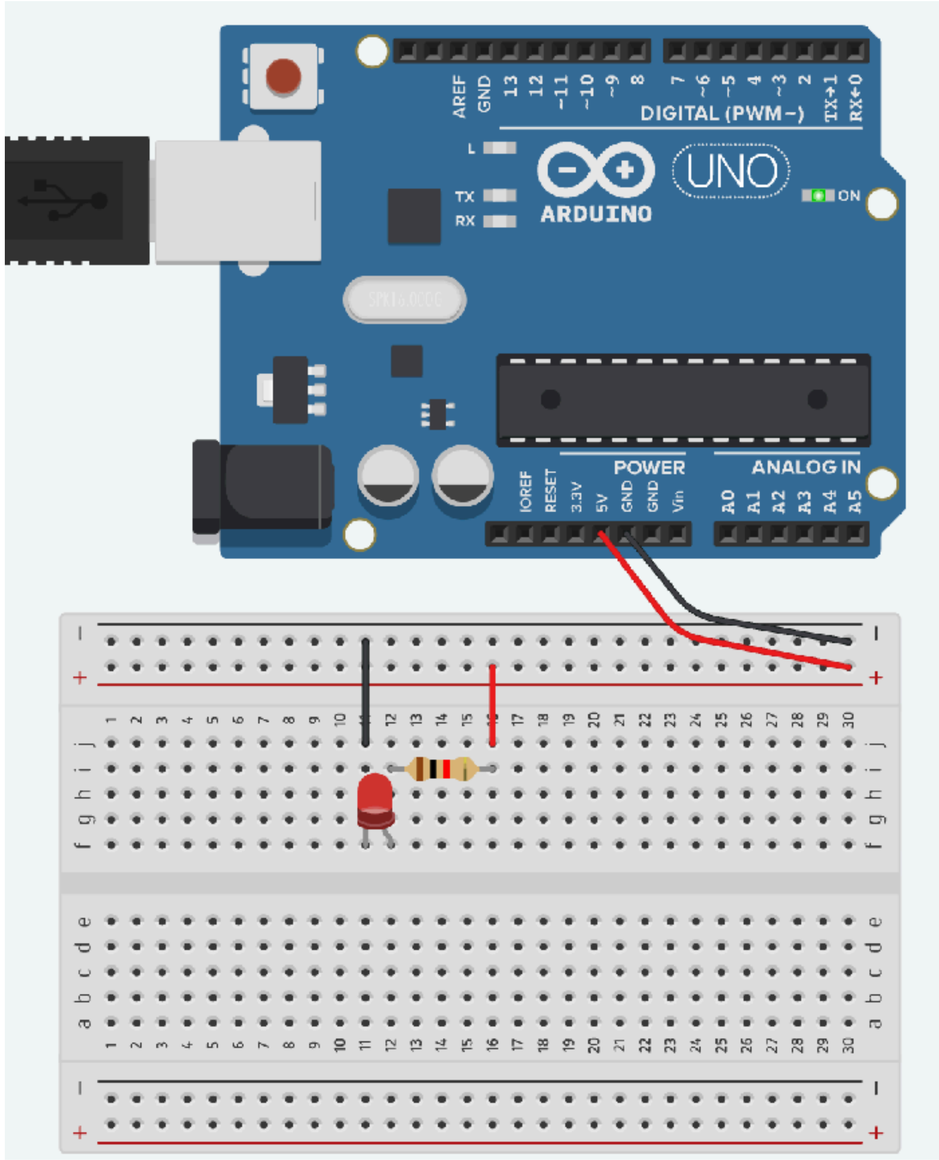
The most common TYPES are:

`int` - numbers with no decimal point
`float` - numbers with a decimal point
`string` - words and sentences made of letters

Decisions:
We make decisions primarily with ‘if’ statements.

```
if(resistorLevel > 500){
 digitalWrite(LED_BUILTIN, HIGH);
}else{
 digitalWrite(LED_BUILTIN, LOW);
}
```

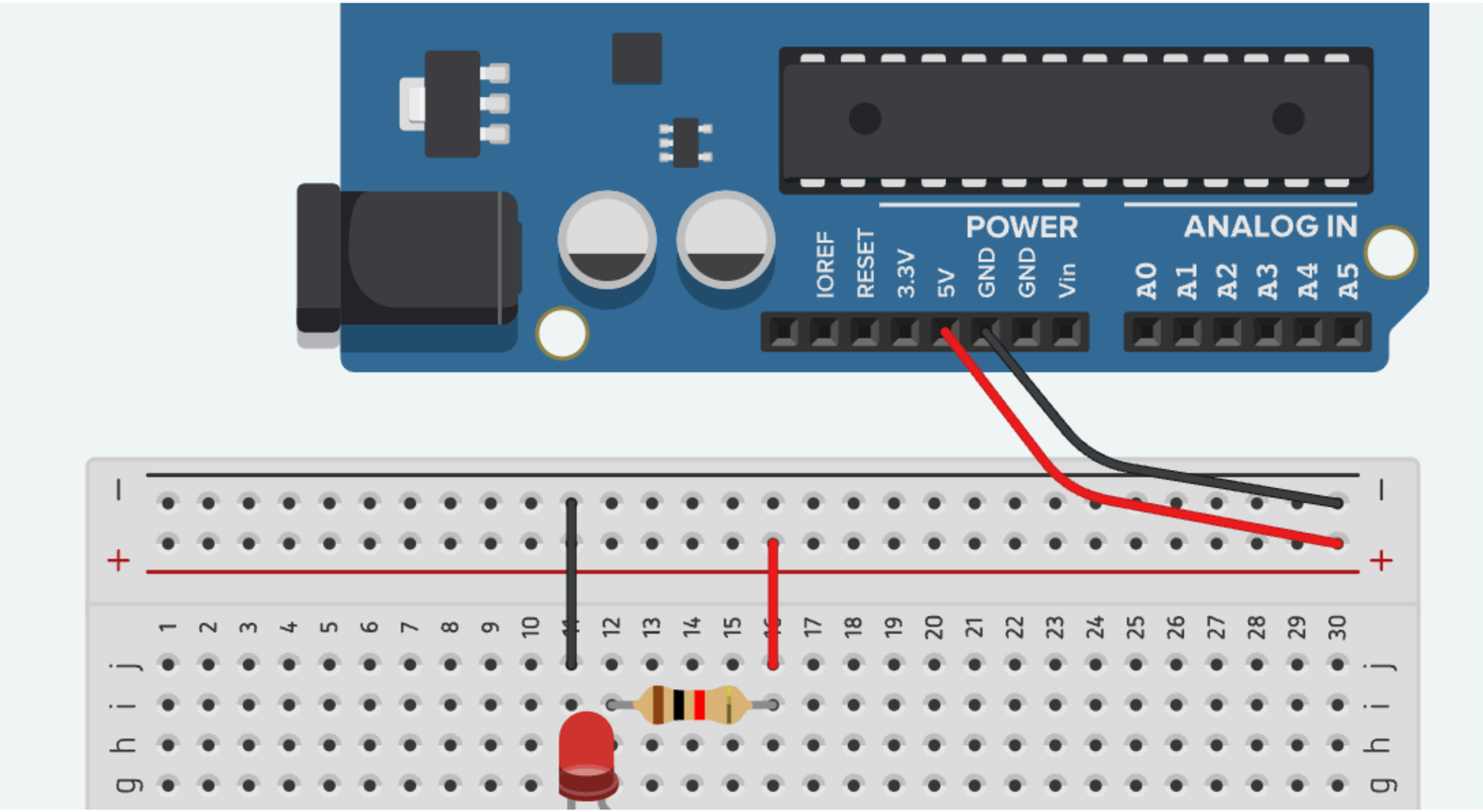
Any Questions so far?

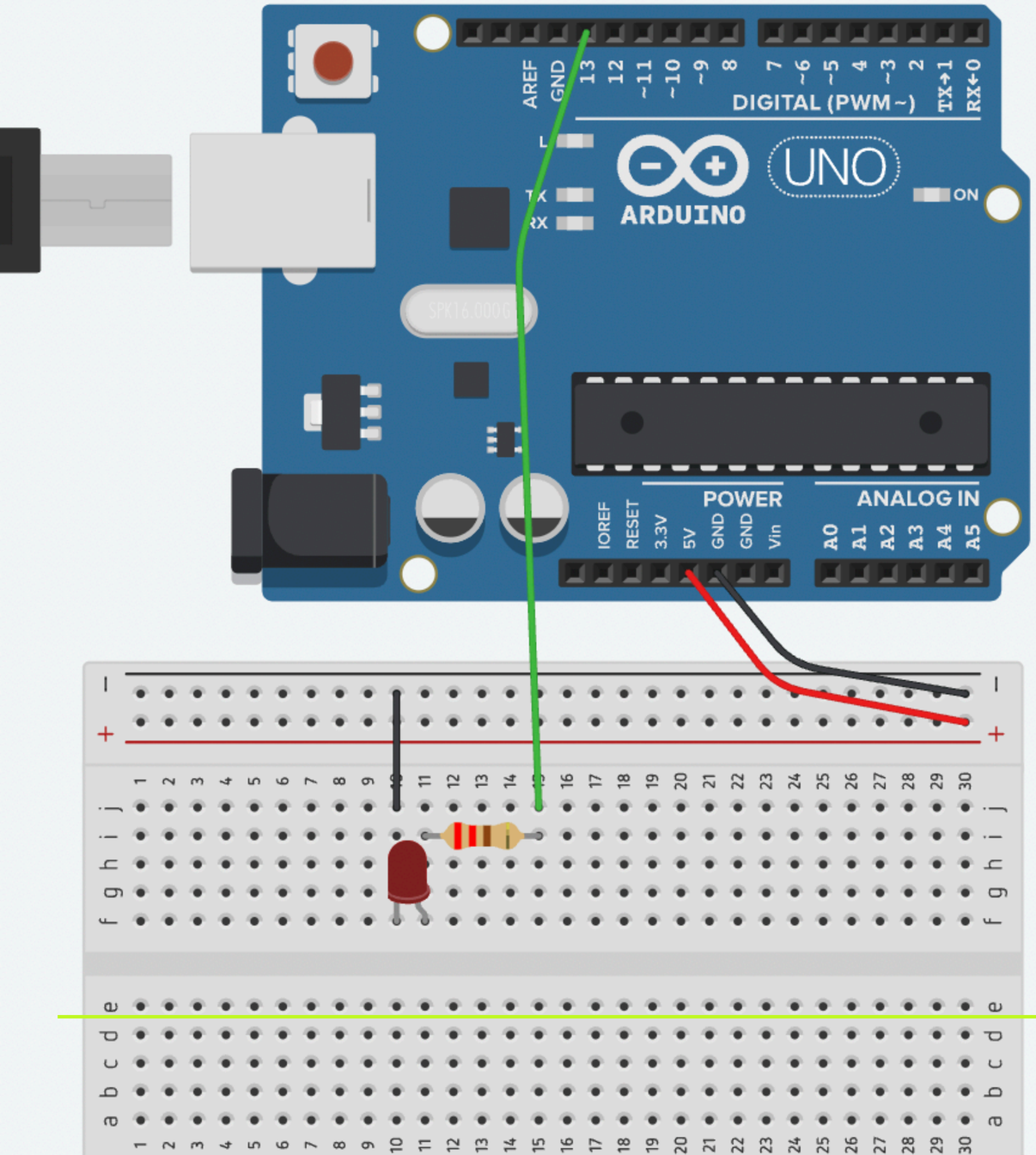


Starting with last week's circuit

- Let's start by building the circuit from last week.
- While you're building it get the Arduino software up and running on your computer.

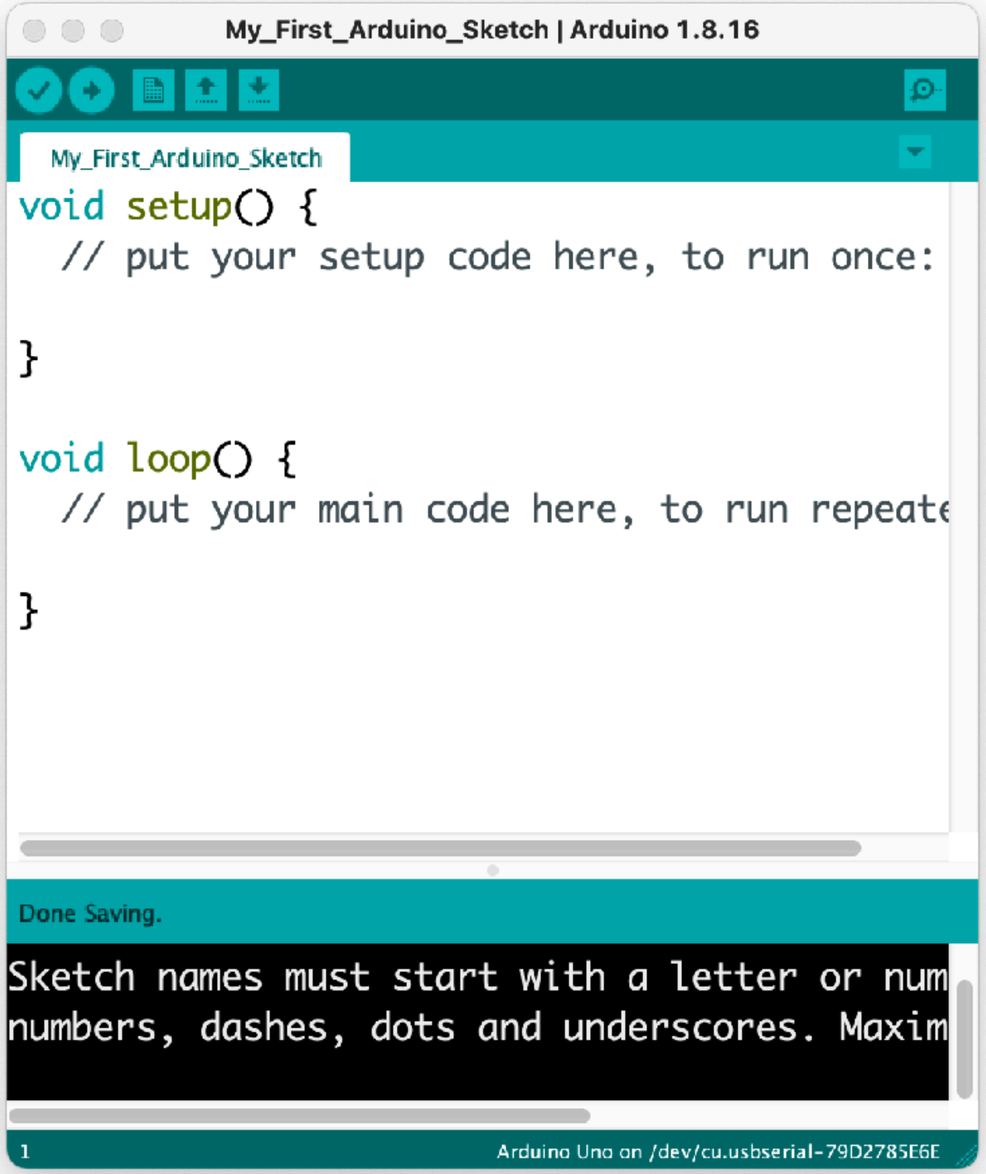
Closeup





Making one small change

- Instead of connecting the LED to the 5V line, we instead connect it to pin 13 on the Arduino.
- This pin is one of the Digital Pins that we can control with code.
- When you plug in the Arduino, the LED might begin to flash, this is because your Arduino comes pre-loaded with some code.

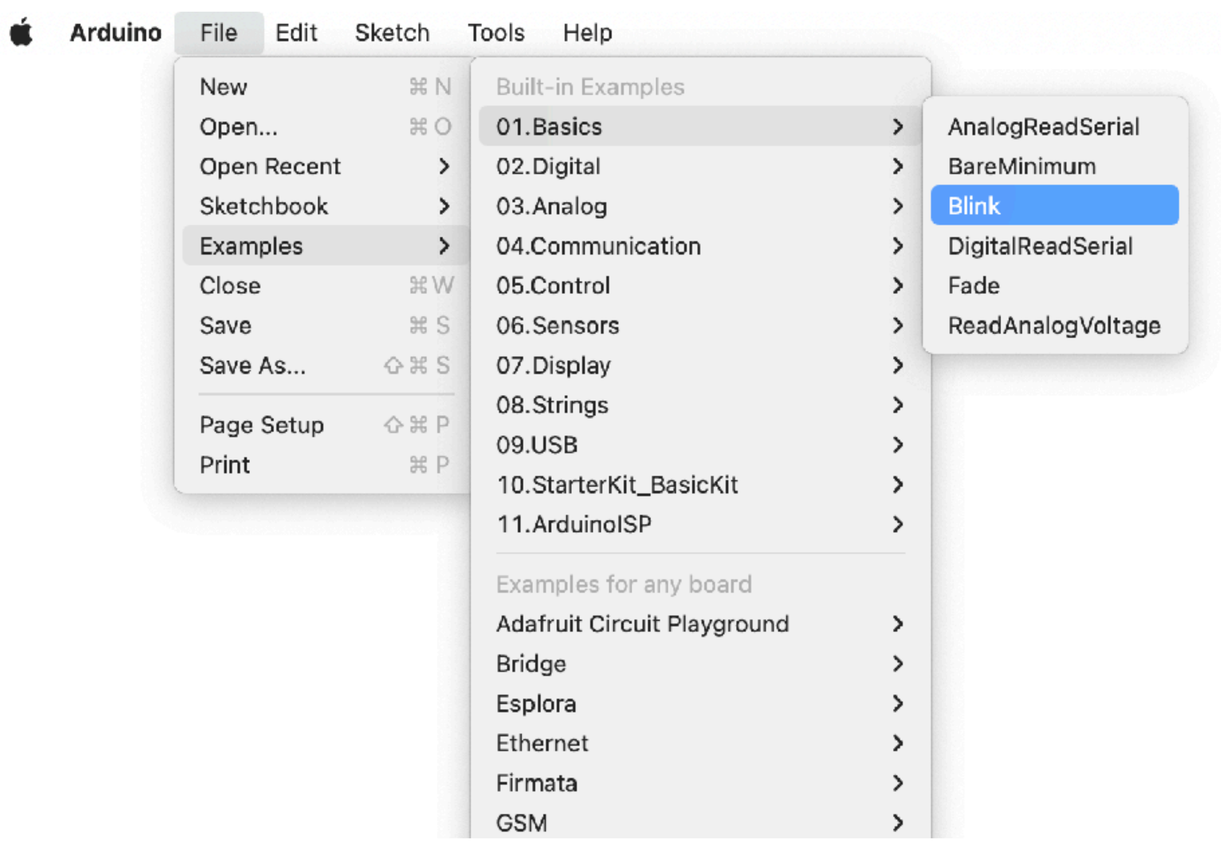


The Arduino IDE

- This is the program we will use to write code for the Arduino and upload it to the board.
- When it opens we see the basic sketch with setup() and loop() but no code inside.
- This might be familiar from P5.js, we use the same structure, only in P5.js we call the functions setup() and draw() instead.

Compile and Upload

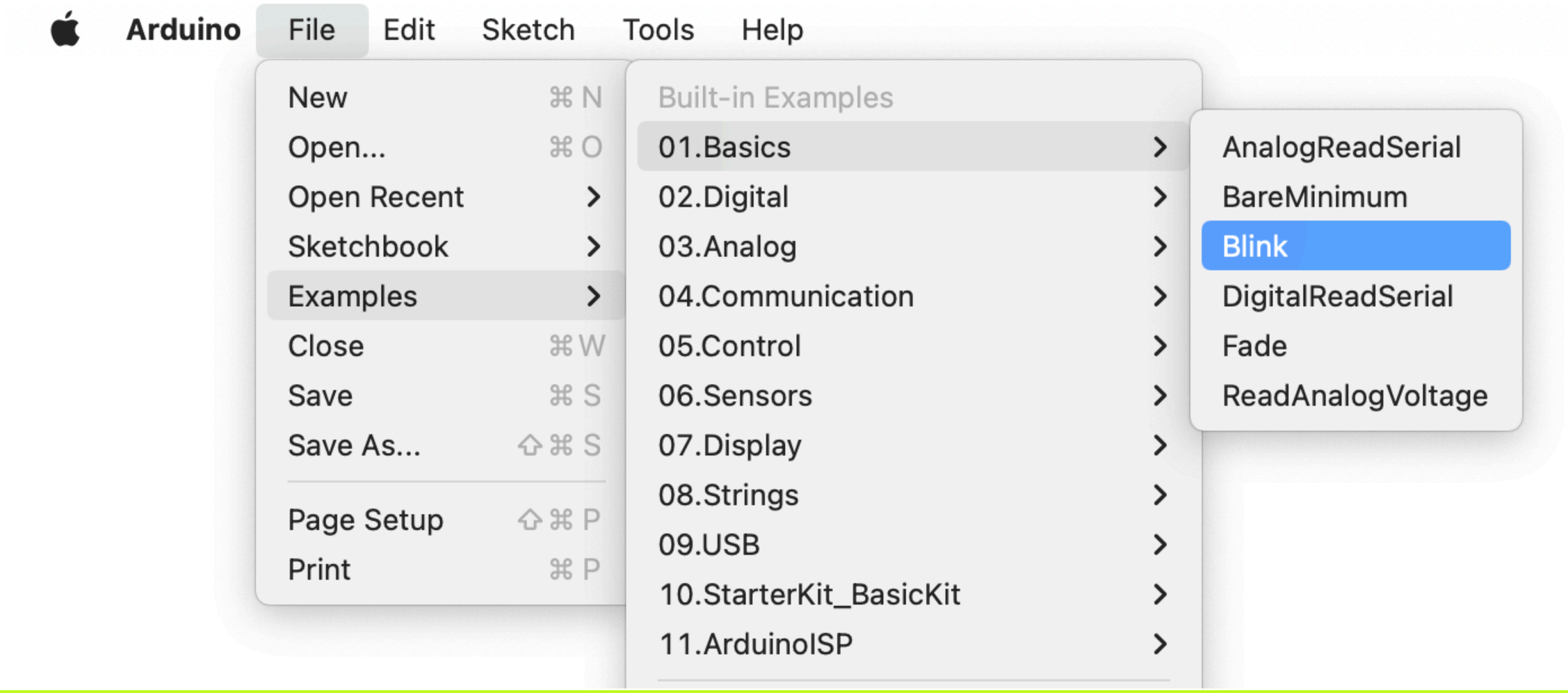
- In the top left, we have buttons for compiling and uploading the code.
- Compiling is when the code is converted into Ones and Zeros that the Arduino can understand.
- Uploading is transferring the code to the Arduino, like copying files to a USB stick.
- The Upload button will automatically compile it, so you don't need to press both buttons every time.

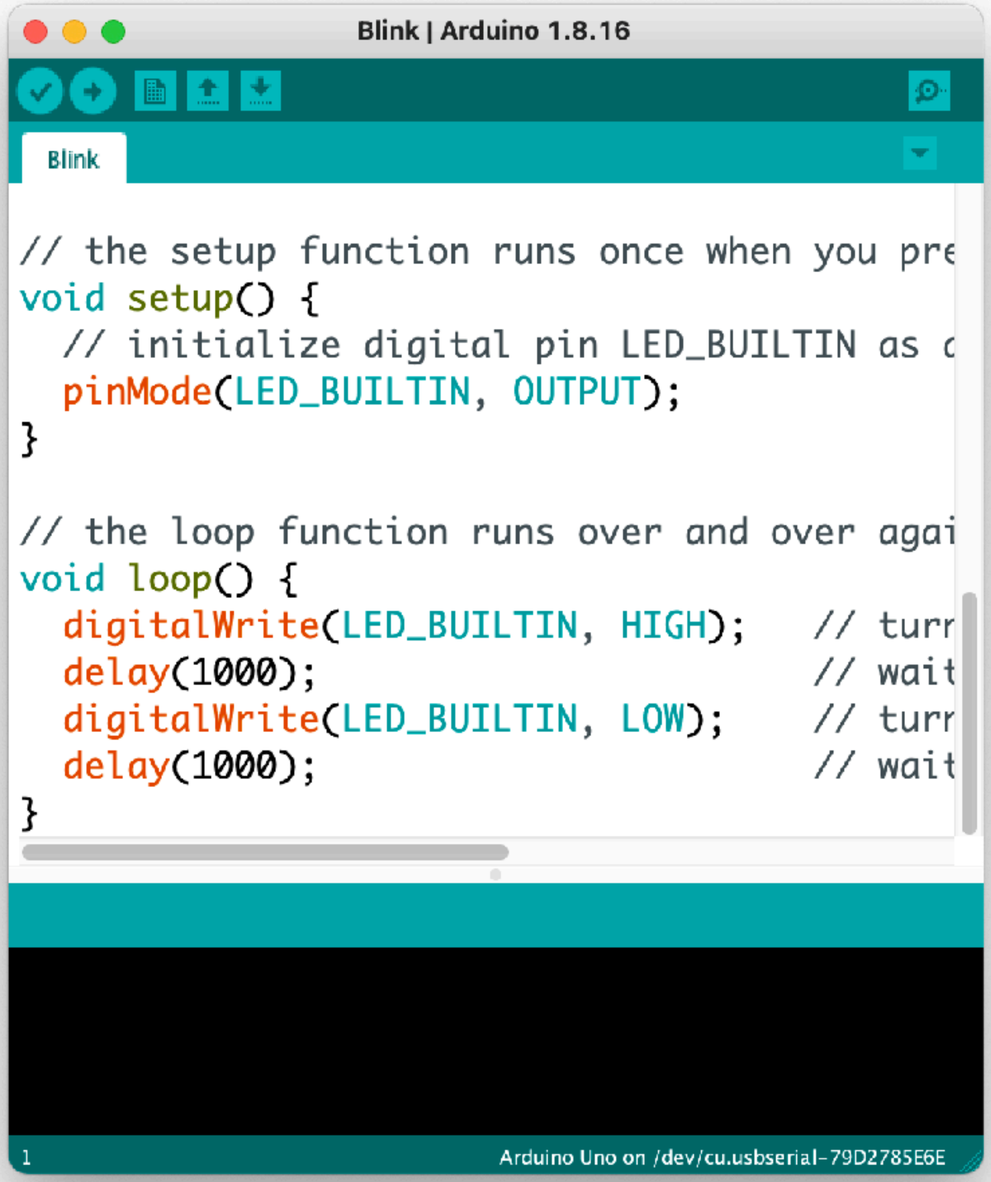


The Blink Sketch

- To test we can upload code to our board we use the Blink example sketch.
- This sketch turns the LED on Pin 13 on and off repeatedly.

Closeup


Arduino
File
Edit
Sketch
Tools
Help
New ⌘ N
Open... ⌘ O
Open Recent
Sketchbook
Examples
Close ⌘ W
Save ⌘ S
Save As... ⇧ ⌘ S
Page Setup ⇧ ⌘ P
Print ⌘ P
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
AnalogReadSerial
BareMinimum
Blink
DigitalReadSerial
Fade
ReadAnalogVoltage



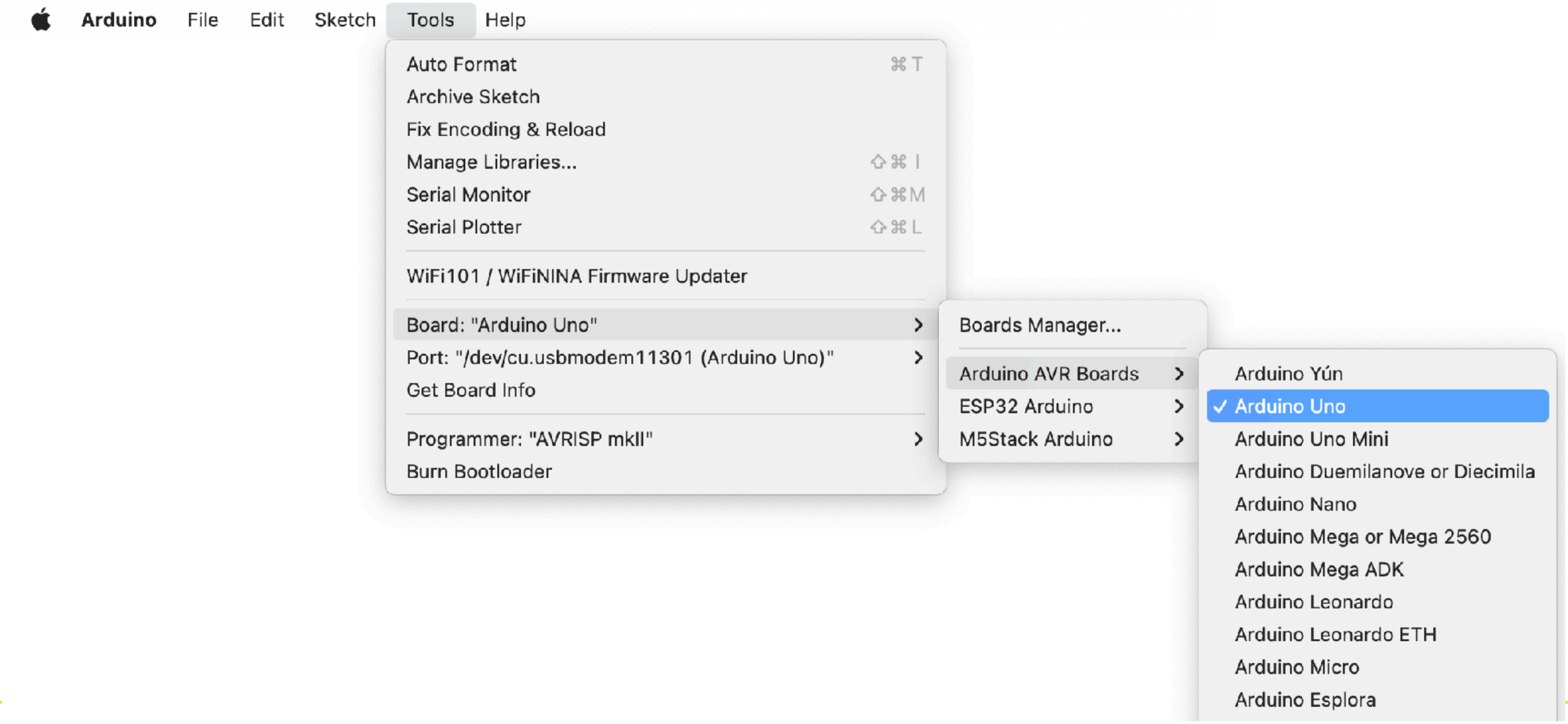
The Blink Sketch

- The setup() function runs only once when we first plug in our Arduino.
- The loop() function runs over and over forever as long as our Arduino has power.
- They work in a similar way to the setup() and draw() functions in p5.js

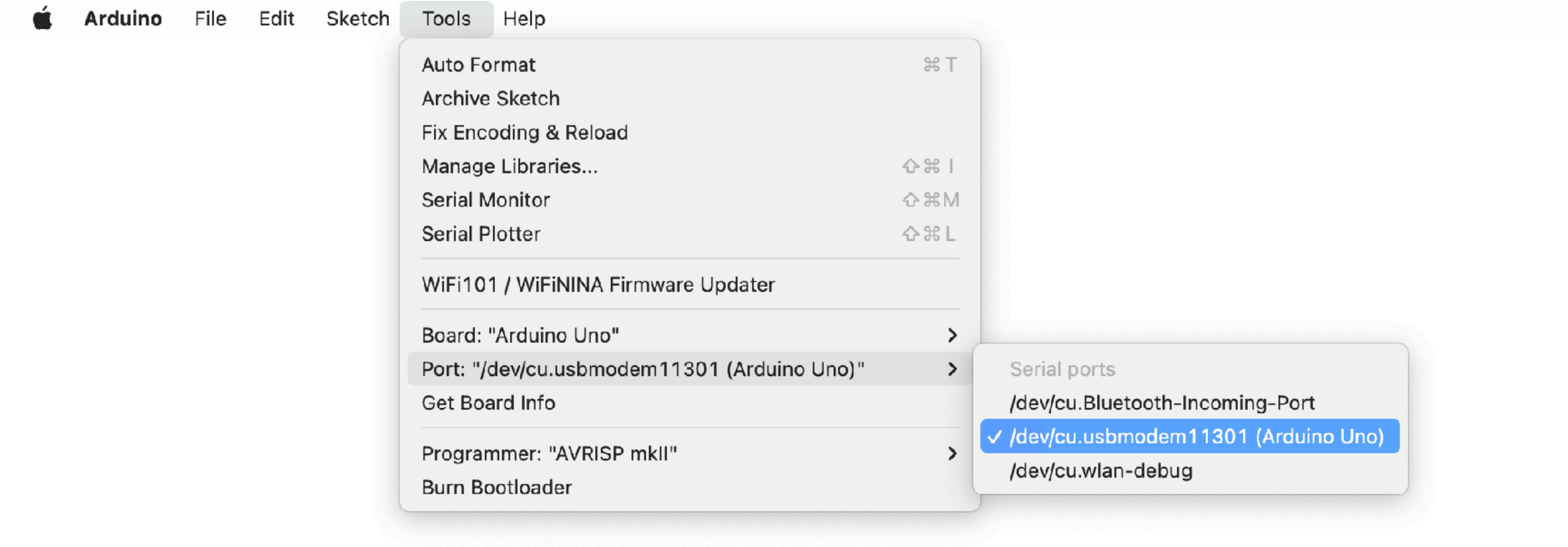
A Quick Glossary

- Sketch : The name given to programs we write for Arduino, similar to how p5.js programs are called Sketches. Arduino Sketches are file type .ino and will only open in the Arduino IDE.
- Library : A ‘Plug-In’ that gives extra functionality to our Arduino Sketch. For Example, the Servo Motor Library allows us to talk to Servos.
- Port : This is how our Arduino connects to our computer, through a Serial Port. We will talk more about that in a future class.
- Board : This is a more generic term for the Arduino hardware you are using. It is worth noting that all Arduinos are boards but not all boards are Arduinos!

Select Board



Select Port


Arduino
File
Edit
Sketch
Tools
Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... ⇧⌘I
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "/dev/cu.usbmodem11301 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
/dev/cu.Bluetooth-Incoming-Port
/dev/cu.usbmodem11301 (Arduino Uno)
/dev/cu.wlan-debug

Blink

```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Setup Function

- The setup() function runs once when your Arduino is first plugged in, or when you press the reset button.
- This works the same way as the setup() function of P5.js.
- Any statements you put inside the brackets { and } will happen once.
- This is most commonly pinMode() statements and Serial.begin() statements as they only need to happen once.

Blink

```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Loop Function

- The loop() function runs over and over forever while the Arduino is plugged in.
- The loop() function works like the draw() function in P5.js.
- delay() functions pause the loop so everything stops when they are used.

Blink

```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

pinMode()

- The pinMode() function is run in the setup as it only needs to happen once.
- It takes two arguments as input.
- The first argument is which pin to set the mode for. In this case LED_BUILTIN, which is a variable containing the number 13.
- The second argument is which mode to select on that pin. In this case we want the pin to be an OUTPUT so it can power an LED.

Blink

```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

digitalWrite()

- The digitalWrite() function allows us to set a pin to be HIGH (5V) or LOW (0V / GND).
- When a pin is HIGH the LED will be switched ON, when it is LOW the LED will be OFF.
- The function takes two arguments.
- The first argument is which pin to set the level for. In this case LED_BUILTIN, which is a variable containing the number 13.
- The second argument is which level we want to set the pin to, in this case HIGH.

Blink

```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

delay()

- The delay() function allows us to pause our program for a specific duration.
- It only takes one argument, which is the duration of the pause in milliseconds. In this case 1000 milliseconds or 1 second.

Blink

```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

delay()

- BUT BEWARE!
- The delay() function while very useful, pauses EVERYTHING that is happening on your Arduino.
- This can cause unexpected problems once your programs become more complicated.

Blink

```
// the setup function runs once when
void setup() {
  // initialize digital pin LED_BUILT
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ov
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

How do they work together?

- The Blink sketch flashes the LED by turning it on :
  - digitalWrite(LED_BUILTIN, HIGH);
- Waiting 1 second :
  - delay(1000);
- Turning it off :
  - digitalWrite(LED_BUILTIN, LOW);
- Waiting 1 second :
  - delay(1000);

Arduino Language Reference

- The Arduino website has a great page that lists all of the possible functions you can use on an Arduino.

Language Reference

Arduino programming language can be divided in three main parts: functions, values (variables structure.

FUNCTIONS

For controlling the Arduino board and performing computations.

Digital I/O

digitalRead()
digitalWrite()
pinMode()

Analog I/O

analogRead()
analogReference()
analogWrite()

Zero, Due & MKR Family

analogReadResolution()
analogWriteResolution()

Advanced I/O

Math

abs()
constrain()
map()
max()
min()
pow()
sq()
sqrt()

Trigonometry

cos()
sin()
tan()

Characters

Random Numbe

random()
randomSeed()

Bits and Bytes

bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

External Interrup

attachInterrupt(

Blink

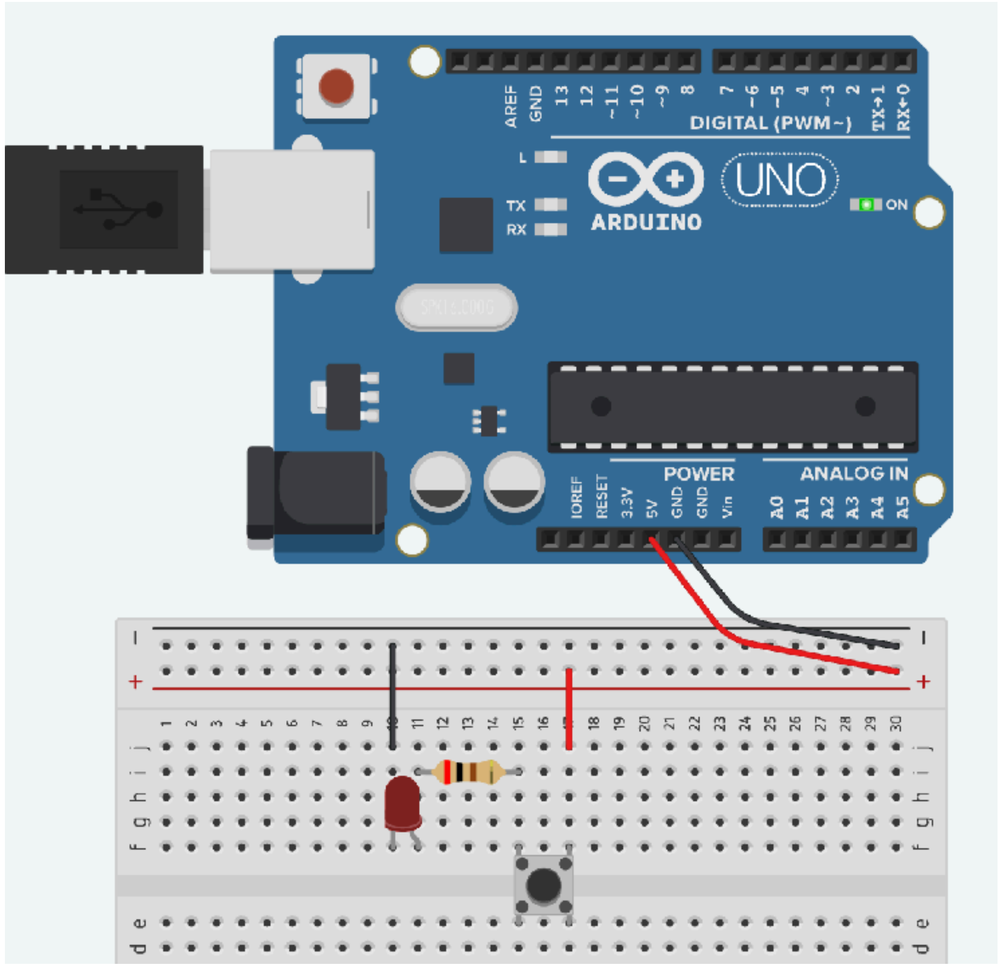
```
// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

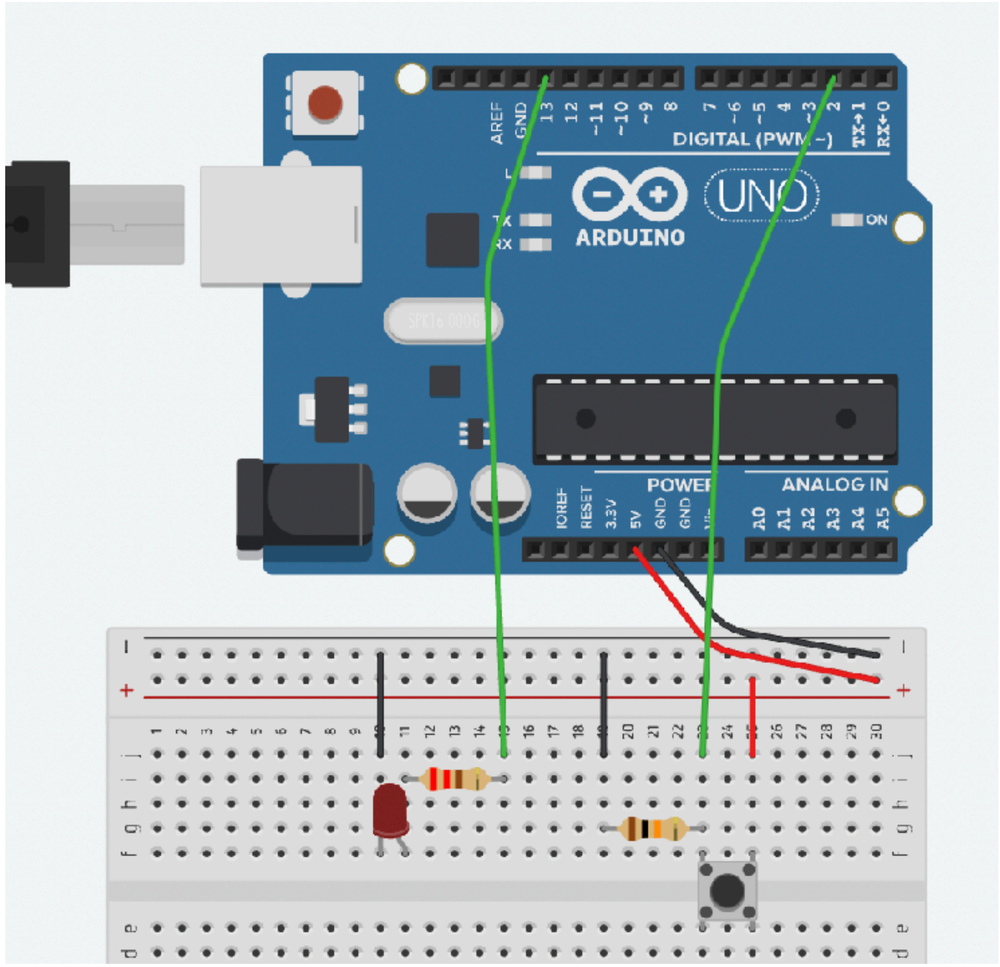
Your Turn

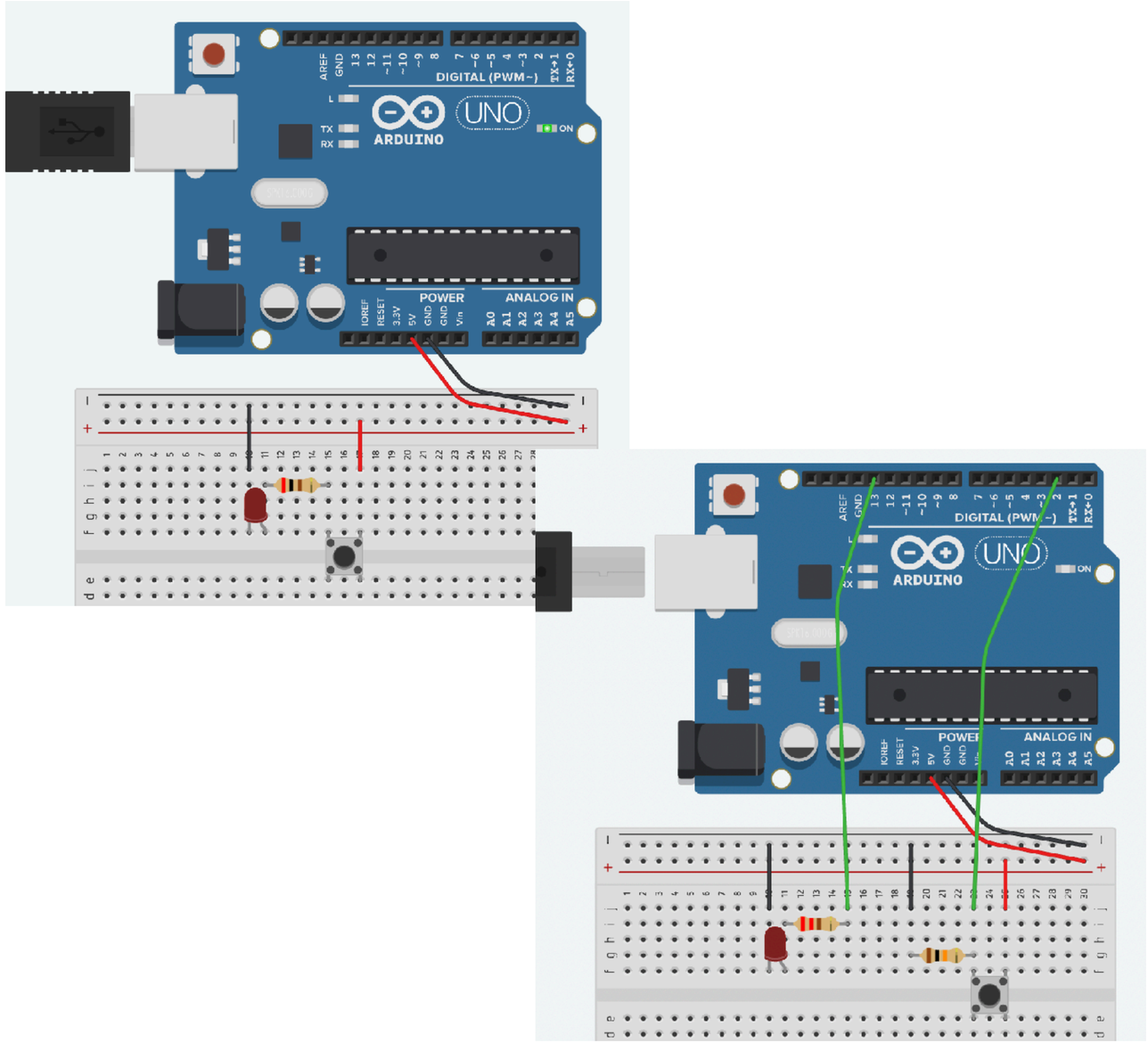
- Try changing the numbers in the delay functions to make your LED flash faster or slower.

Any Questions so far?

Spot the Difference!



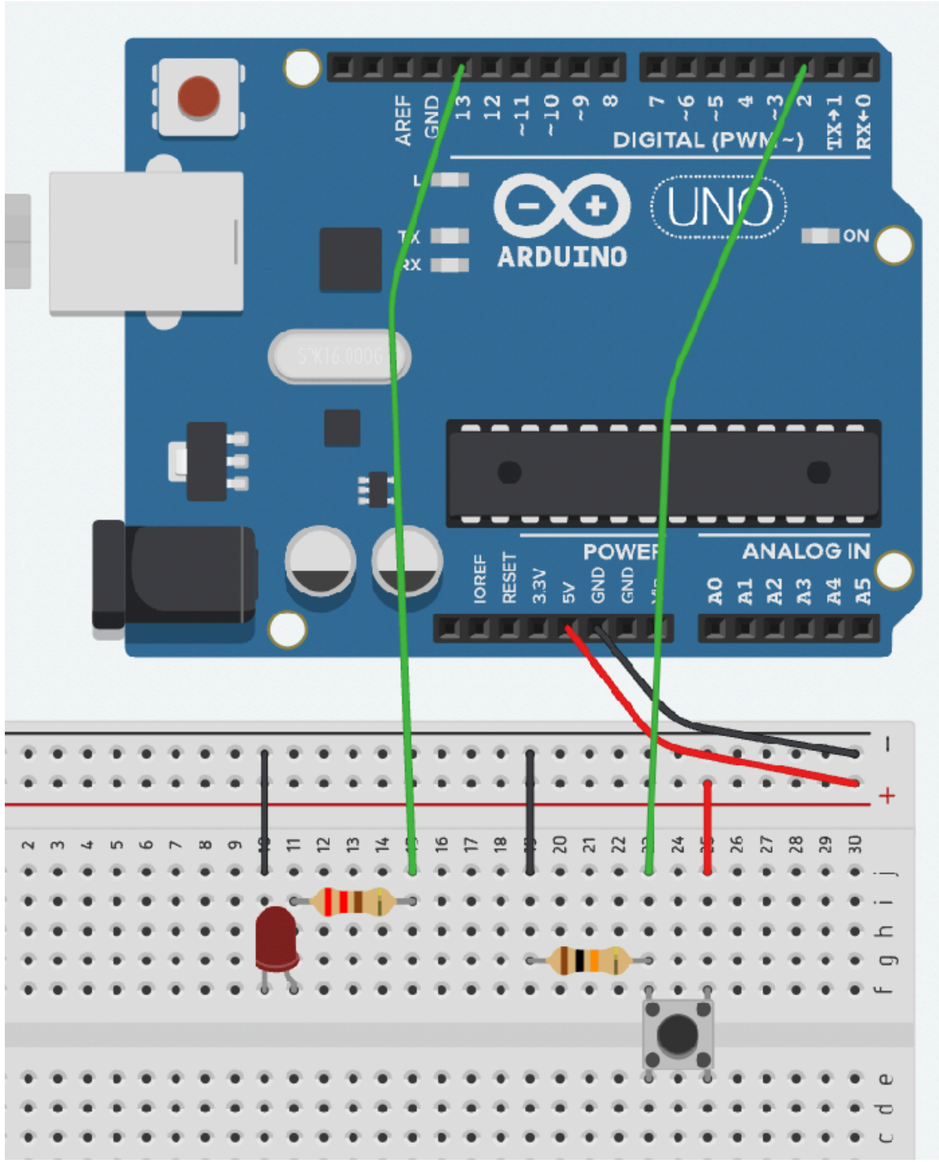




They can do the same thing

But the new one needs code

- We we use a resistor with the button to make sure the voltage is stable whether the button is pressed or not.
- This time the resistor is 10KΩ rather than 200Ω



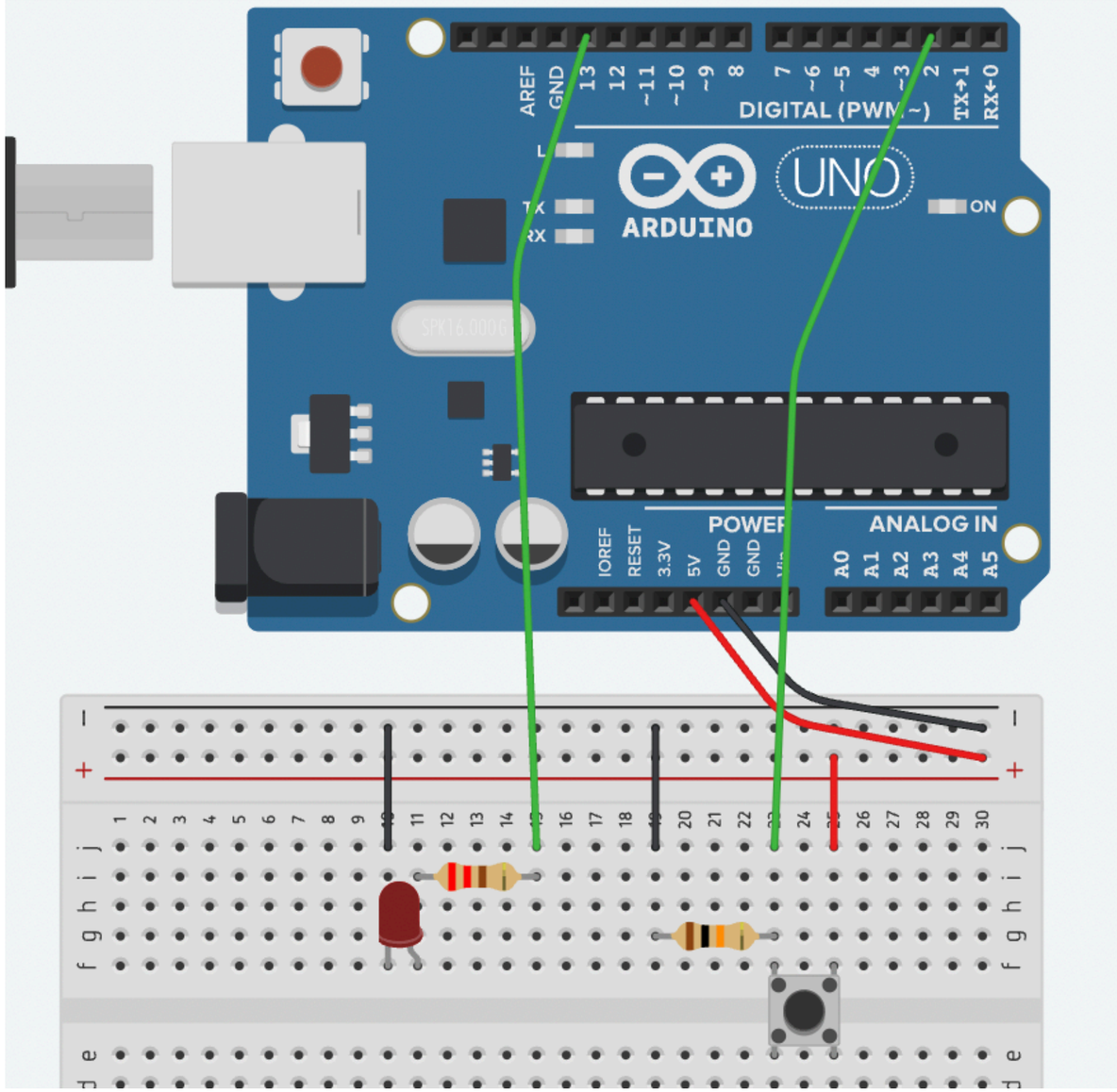
Let’s get building

- Set up the board as is shown.
- When you’re done you can upload the next code.
- This time it is Examples>Digital>Button

Closeup

220 Resistor

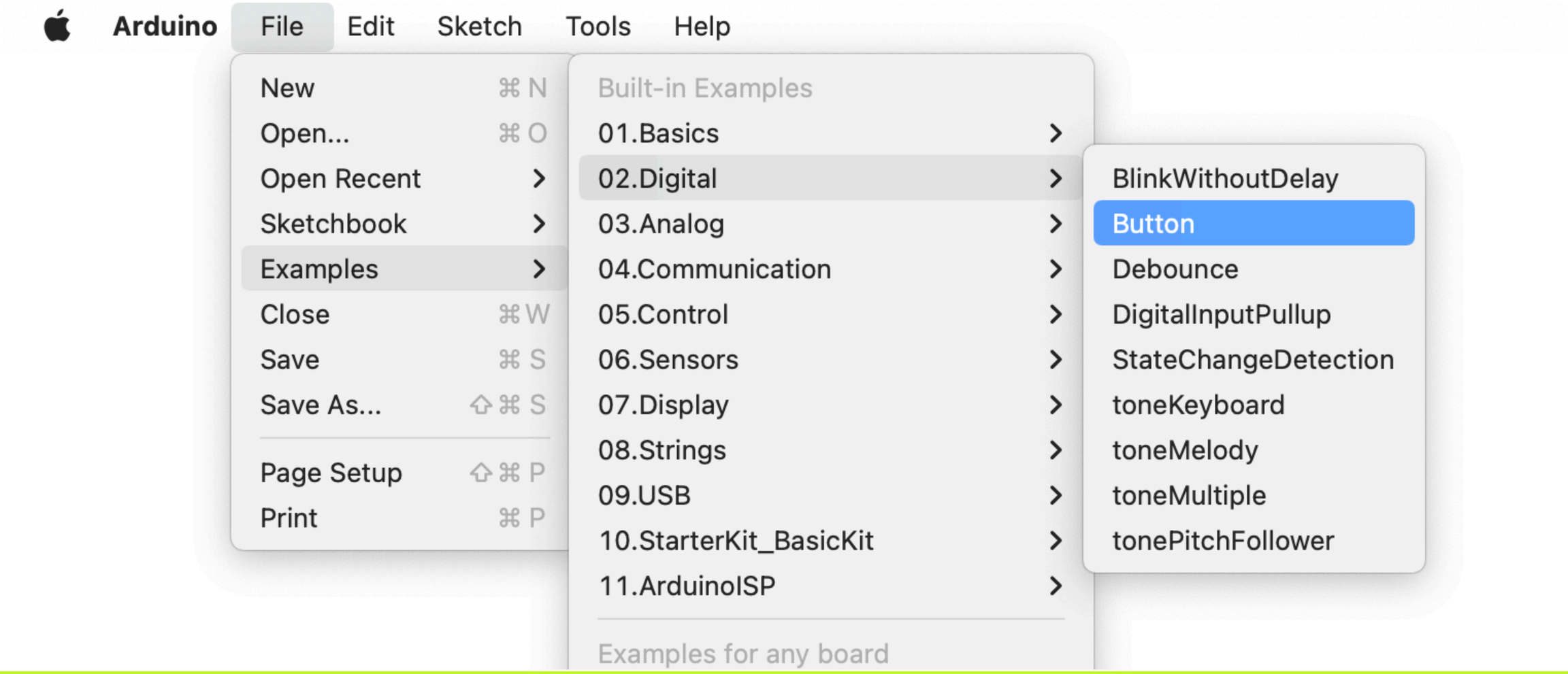
For LED



10K Resistor

For Button

Closeup


Arduino
File
Edit
Sketch
Tools
Help
New ⌘ N
Open... ⌘ O
Open Recent
Sketchbook
Examples
Close ⌘ W
Save ⌘ S
Save As... ⇧ ⌘ S
Page Setup ⇧ ⌘ P
Print ⌘ P
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
BlinkWithoutDelay
Button
Debounce
DigitalInputPullup
StateChangeDetection
toneKeyboard
toneMelody
toneMultiple
tonePitchFollower

Button | Arduino 1.8.16

Button

```
// constants won't change. They're used here to set pin numbe
const int buttonPin = 2;     // the number of the pushbutton
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the push

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttor
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

1 Arduino Uno on /dev/cu.usbserial-79D2785E6E

The Button Sketch

- This sketch is the first time we will encounter variables, and if statements.
- We'll go through how they work in this example.

The Button Sketch

Button | Arduino 1.8.16

Button

```
// constants won't change. They're used here
const int buttonPin = 2;     // the number o
const int ledPin =  13;      // the number o

// variables will change:
int buttonState = 0;         // variable for

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an inp
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
```

- First we set some variables
- We do that by stating the type of variable and the name we want it to have.

Button | Arduino 1.8.16

Button

```
// constants won't change. They're used here
const int buttonPin = 2;     // the number o
const int ledPin =  13;      // the number o

// variables will change:
int buttonState = 0;         // variable for

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an inp
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
```

The Button Sketch

- These variables just assign names to the pins we are using for various inputs and outputs.
- They are const int variables meaning they are integers and are constant, they cannot change.

Button | Arduino 1.8.16

Button

```
// constants won't change. They're used here
const int buttonPin = 2;     // the number o
const int ledPin =  13;      // the number o

// variables will change:
int buttonState = 0;         // variable for

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an inp
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
```

The Button Sketch

- This variable is just an int
- That means it is an integer that can change

The Button Sketch

Button

```
// constants won't change. They're used here
const int buttonPin = 2;     // the number o
const int ledPin =  13;      // the number o

// variables will change:
int buttonState = 0;         // variable for

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an inp
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If
```

- The setup function is the same as before, except we have a new call of pinMode()
- Rather than OUTPUT for the LED pin, we use INPUT, when we will using digitalRead()

```
// variables will change:
int buttonState = 0;         // variable for

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an inp
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
```

The Button Sketch

- Here in the loop we find our new function digitalRead()
- Rather than setting a pin to HIGH or LOW, digitalRead() returns a value of 1 (HIGH) or 0 (LOW).
- In this case the value is saved to the variable buttonState so we can use it later in the code, using the = sign.

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an inp
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

The Button Sketch

- Now we have an if statement
- An if statement takes a condition and decides if it is true or false.
- In this case the condition (buttonState == HIGH) is asking if the buttonState is equal to HIGH.
- If it is true it runs the code in the firs set of {} brackets, if it is false it runs the code in the {} brackets after else.

Button | Arduino 1.8.16

Button

```
// constants won't change. They're used here to set pin numbe
const int buttonPin = 2;     // the number of the pushbutton
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the push

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttor
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

1 Arduino Uno on /dev/cu.usbserial-79D2785E6E

The Button Sketch

- We use a digitalRead() statement to get the status of the button every loop.
- We then use an ‘if’ statement to check if the reading is HIGH or LOW and set the LED to be HIGH or LOW accordingly.
- Try adding a delay(1000); after digitalWrite(ledPin, HIGH);
- You will see that the LED always stays on for a second after you release the button.

Button | Arduino 1.8.16

Button

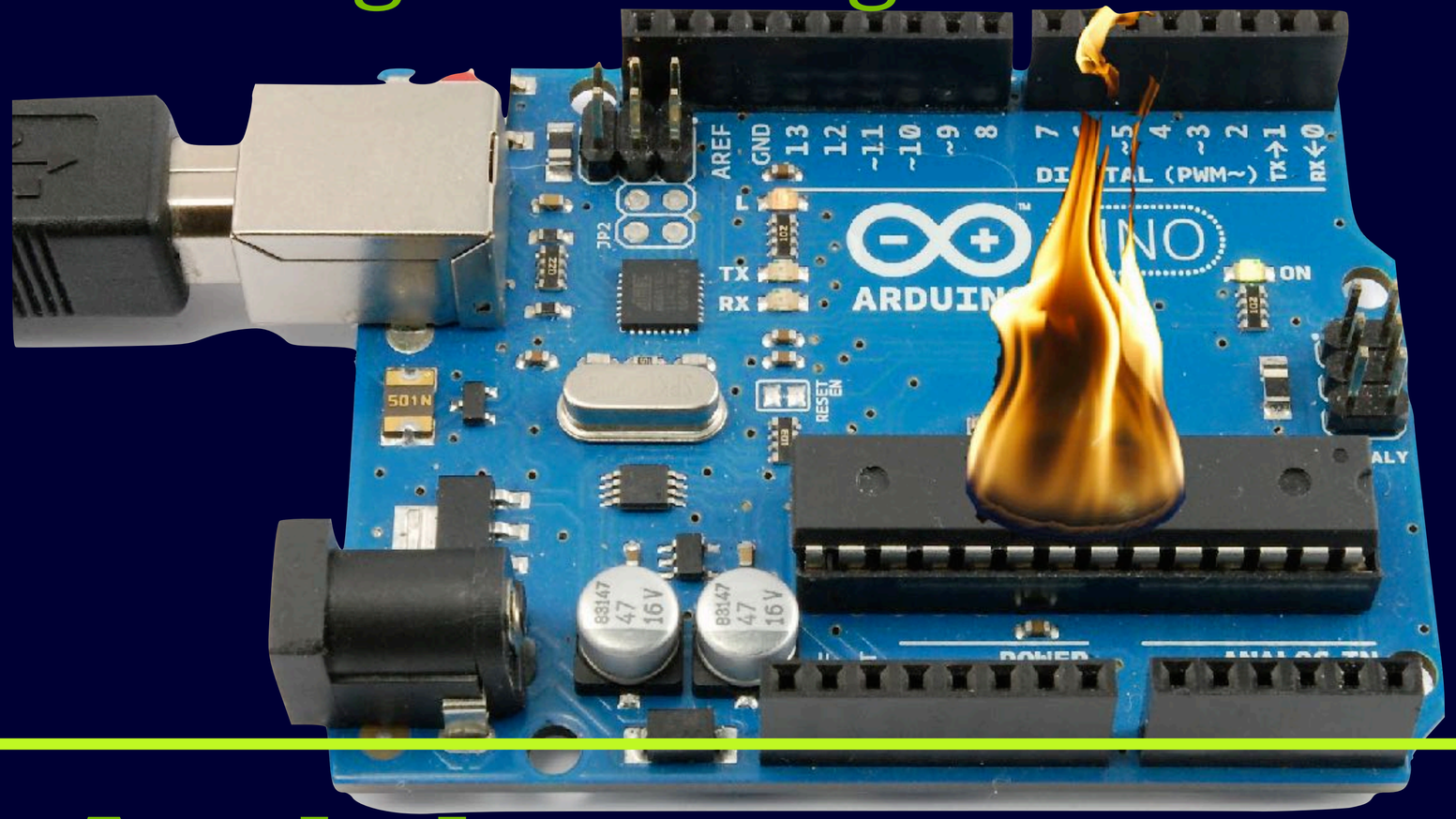
```
// constants won't change. They're used here to set pin numbe
const int buttonPin = 2;     // the number of the pushbutton
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the push

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttor
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

1 Arduino Uno on /dev/cu.usbserial-79D2785E6E

Extension

- Can you add a second button and/or a second LED to come up with some more complex systems?

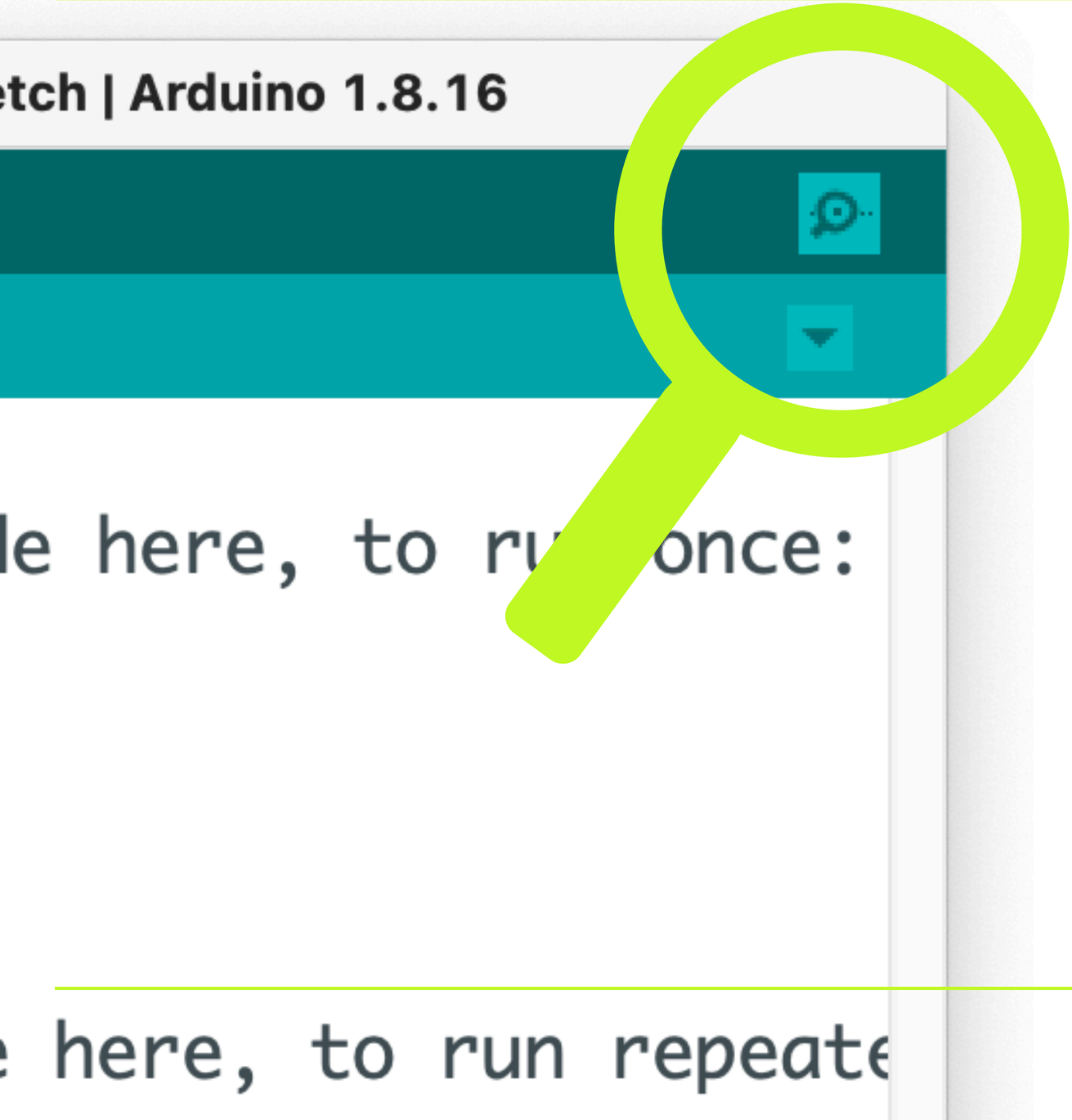
What happens when something isn’t working?



Debugging Arduino

Serial Debugging

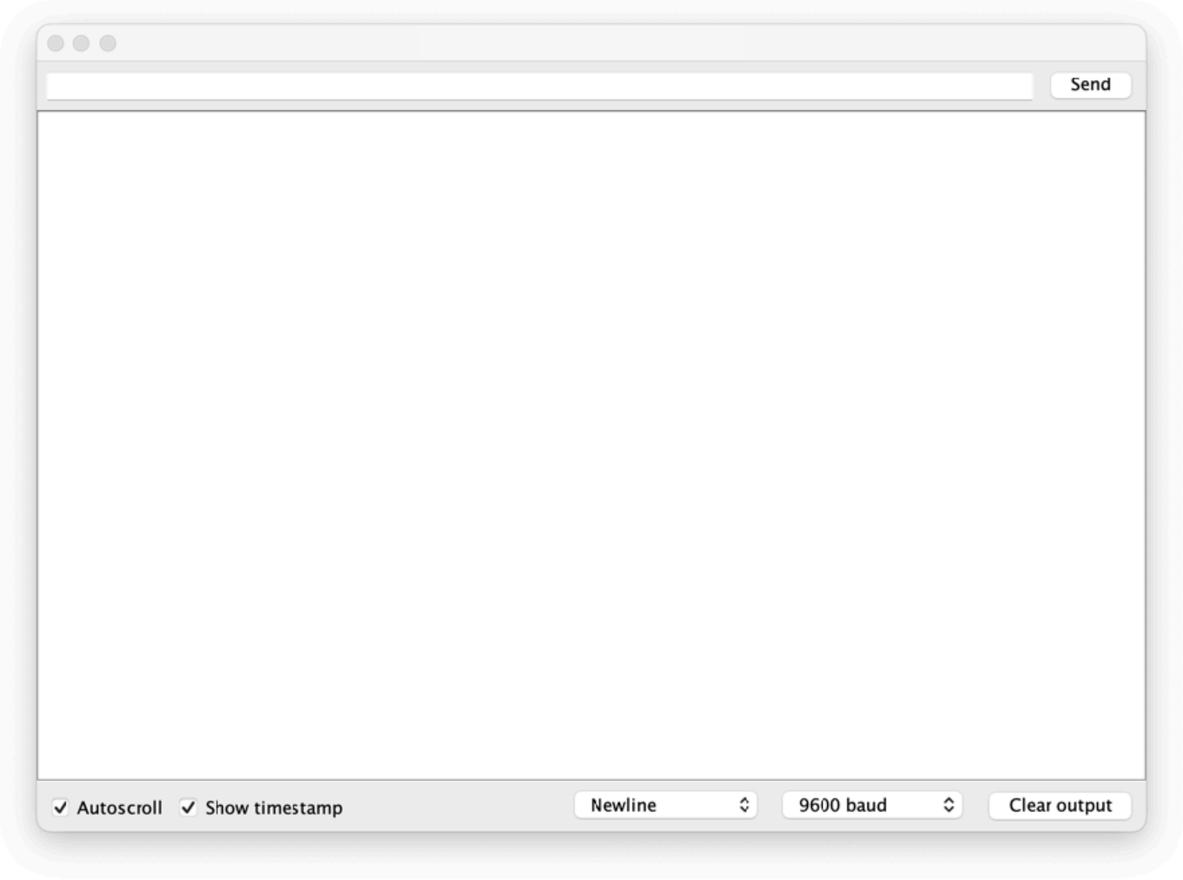
- Unlike p5.js the code you write is not running on your computer, it is running inside the Arduino.
- This means to get something like a console.log() we need to send messages from the Arduino to our computer.
- We do this using a protocol called Serial.
- This is the same way that sketches are transferred to the Arduino.
- Messages and sketches are sent down the USB cable as Ones and Zeros.



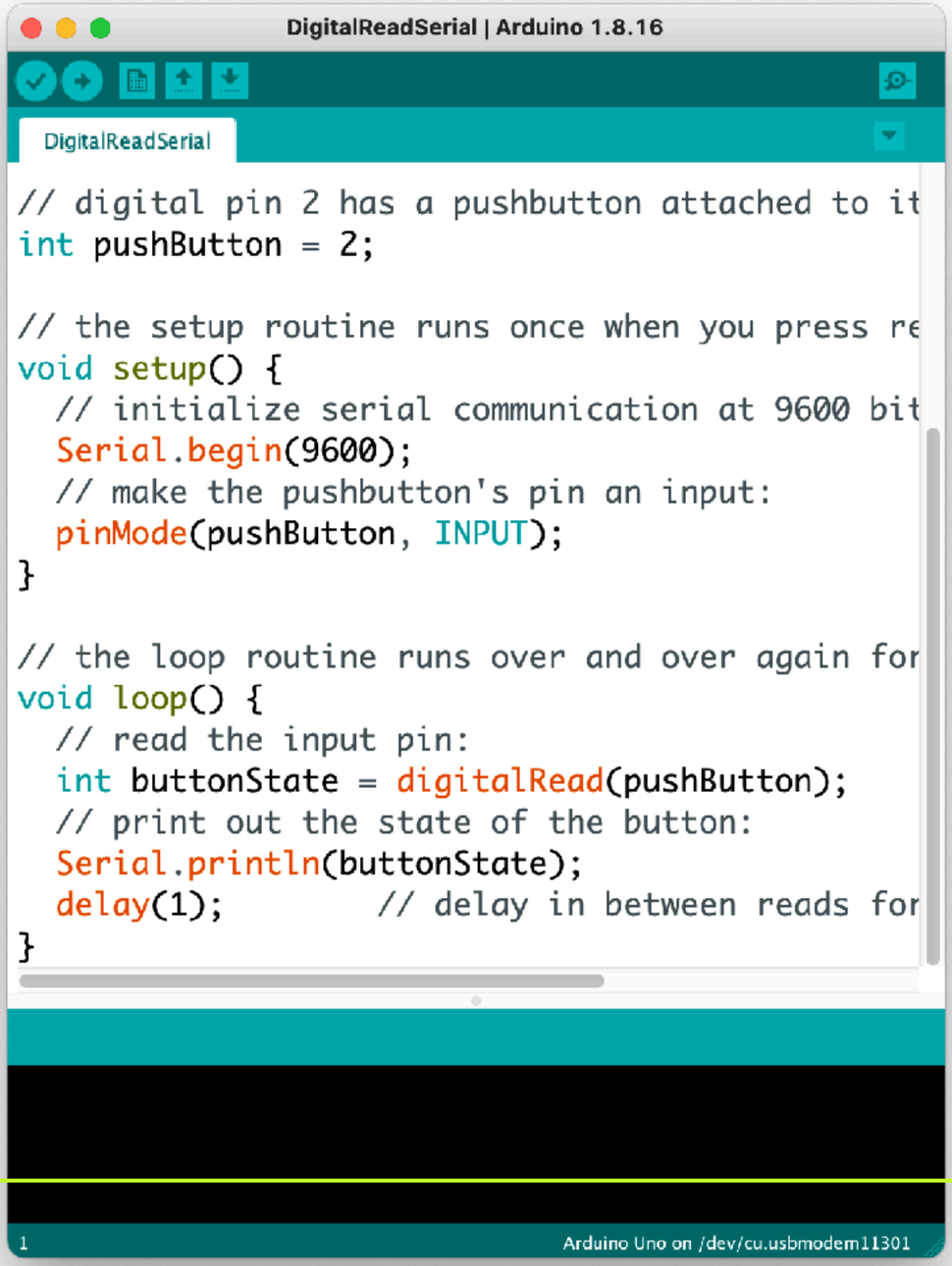
Serial Monitor

- The Arduino IDE has a tool for viewing messages from your Arduino.
- It is called the Serial Monitor and it can be accessed by pressing the button in the top right corner of the IDE.

The Serial Monitor



- When we use Serial.println("Message") in our code, the messages will pop up in this window.



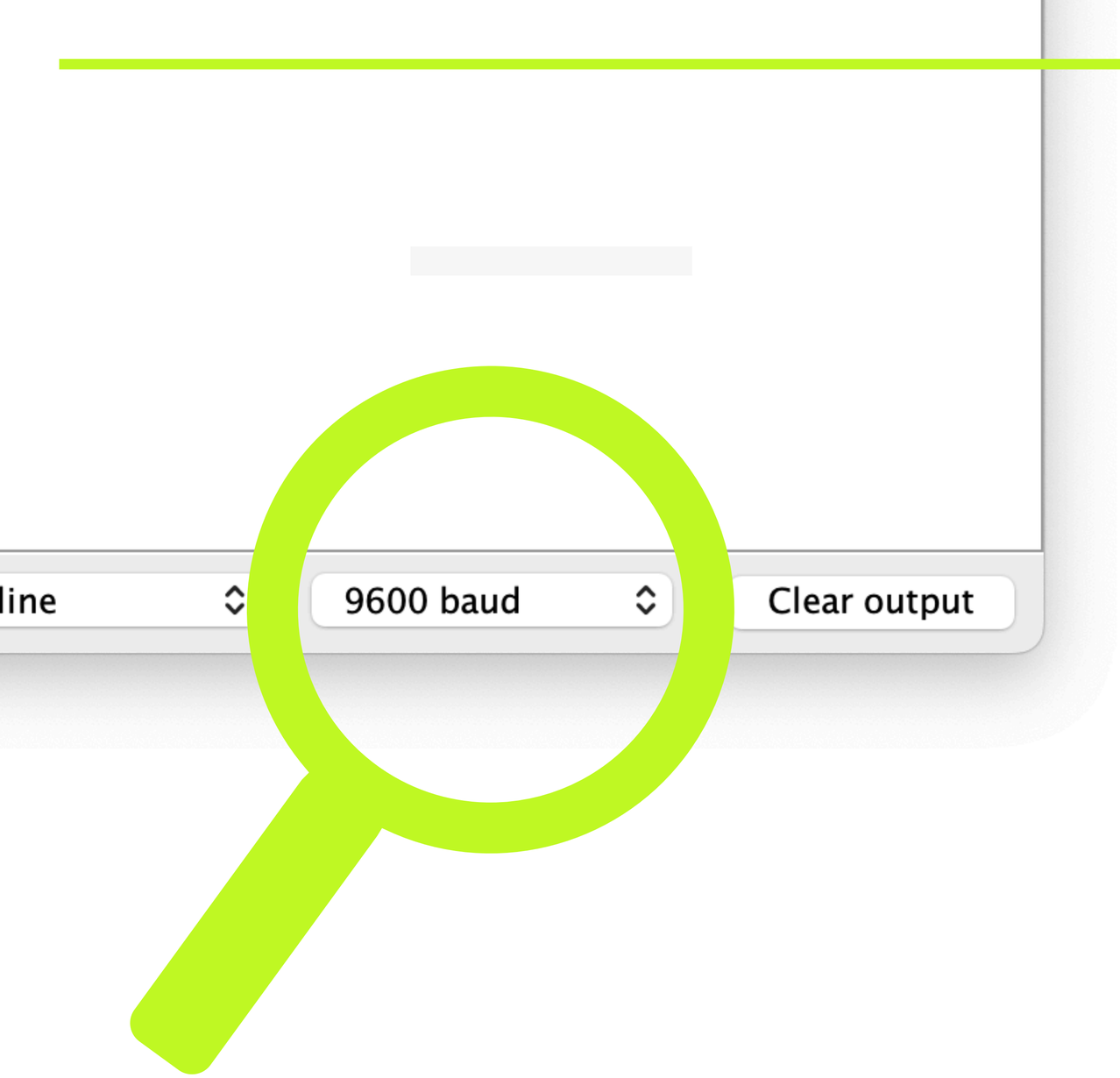
DigitalReadSerial

- This sketch uses Serial.begin() and Serial.println() to send messages to the serial monitor.

```
int pushButton = 2;

// the setup routine runs once when you pr
void setup() {
  // initialize serial communication at 96
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over aga
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);        // delay in between rea
}
```

DigitalReadSerial

- We start in setup() with Serial.begin(9600)
- The number in the brackets (in this case 9600) is the baud-rate.
- This is the speed that the Arduino communicates to the computer.
- If the Arduino and the Computer are talking and listening at different speeds, they won't be able to understand each other.



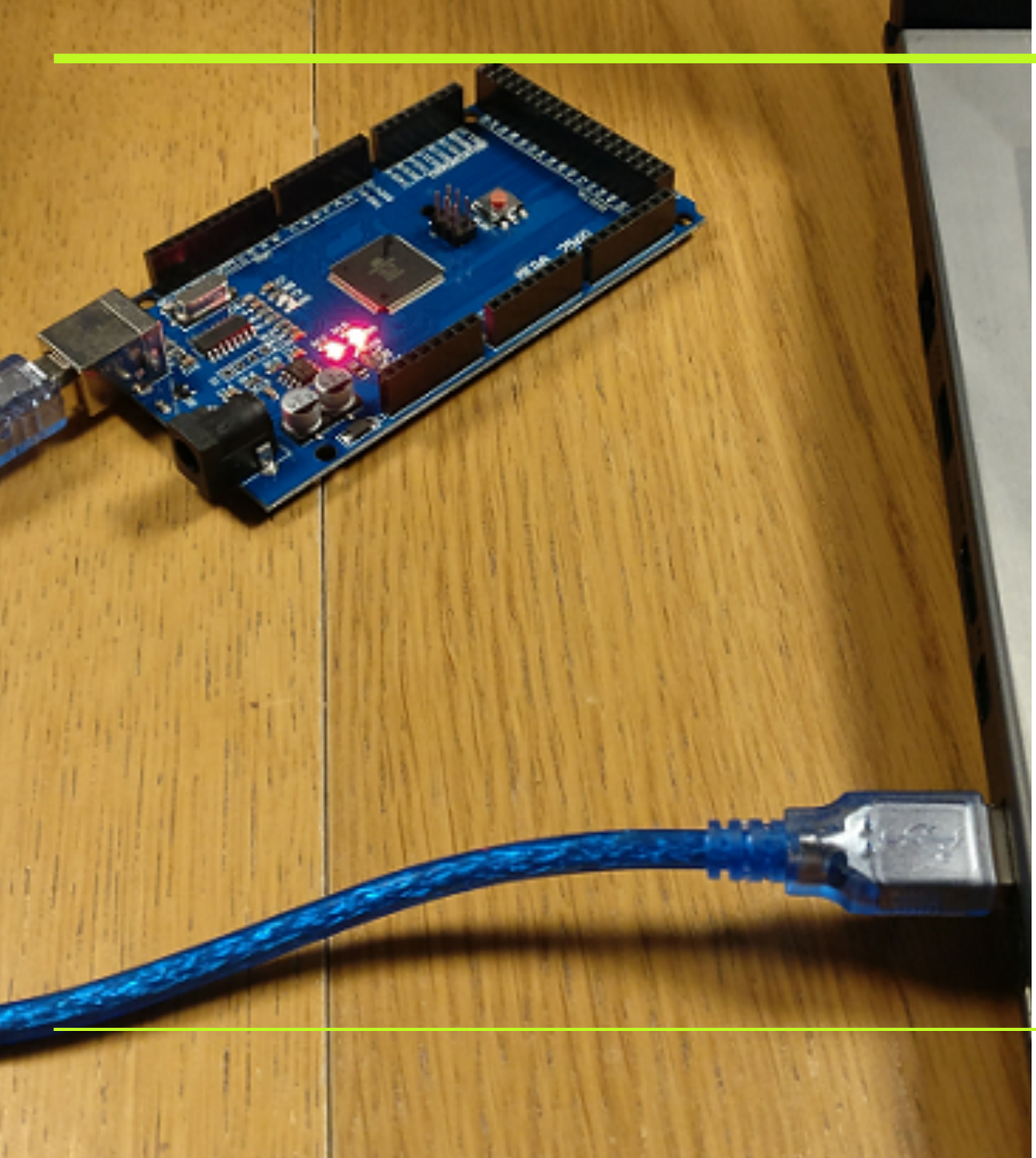
DigitalReadSerial

- We can set the baud-rate of the computer at the bottom right of the serial monitor.
- It doesn't matter too much what the number is, as long as they match.
- The higher the number, the faster the communication, but the more chance of mistakes.

```
nt pushButton = 2;

/ the setup routine runs once when you pres
oid setup() {
 // initialize serial communication at 9600
 Serial.begin(9600);
 // make the pushbutton's pin an input:
 pinMode(pushButton, INPUT);


/ the loop routine runs over and over again
oid loop() {
 // read the input pin:
 int buttonState = digitalRead(pushButton);
 // print out the state of the button:
 Serial.println(buttonState);
 delay(1);        // delay in between reads
```

DigitalReadSerial

- Now we read the state of the button, just like in the last sketch
- Instead of using it in an if statement, we are going to send it to the computer using Serial.println()
- Serial.print() also exists, but it does not make a new line after it finishes the message.



Serial only works with Computer

- But remember! Serial.println() only works when you are connected to a computer.
- When the Arduino is disconnected, your code will still run but any statements relating to serial will be ignored.

```
nt pushButton = 2;

/ the setup routine runs once when you pres
oid setup() {
 // initialize serial communication at 9600
 Serial.begin(9600);
 // make the pushbutton's pin an input:
 pinMode(pushButton, INPUT);


/ the loop routine runs over and over again
oid loop() {
 // read the input pin:
 int buttonState = digitalRead(pushButton);
 // print out the state of the button:
 Serial.println(buttonState);
 delay(1);        // delay in between reads
```

Your Turn

- Examples>Basics>DigitalReadSerial
- Upload the DigitalReadSerial code and see if you can read the messages in the Serial Monitor.
- Try changing the baud-rate and see what happens!

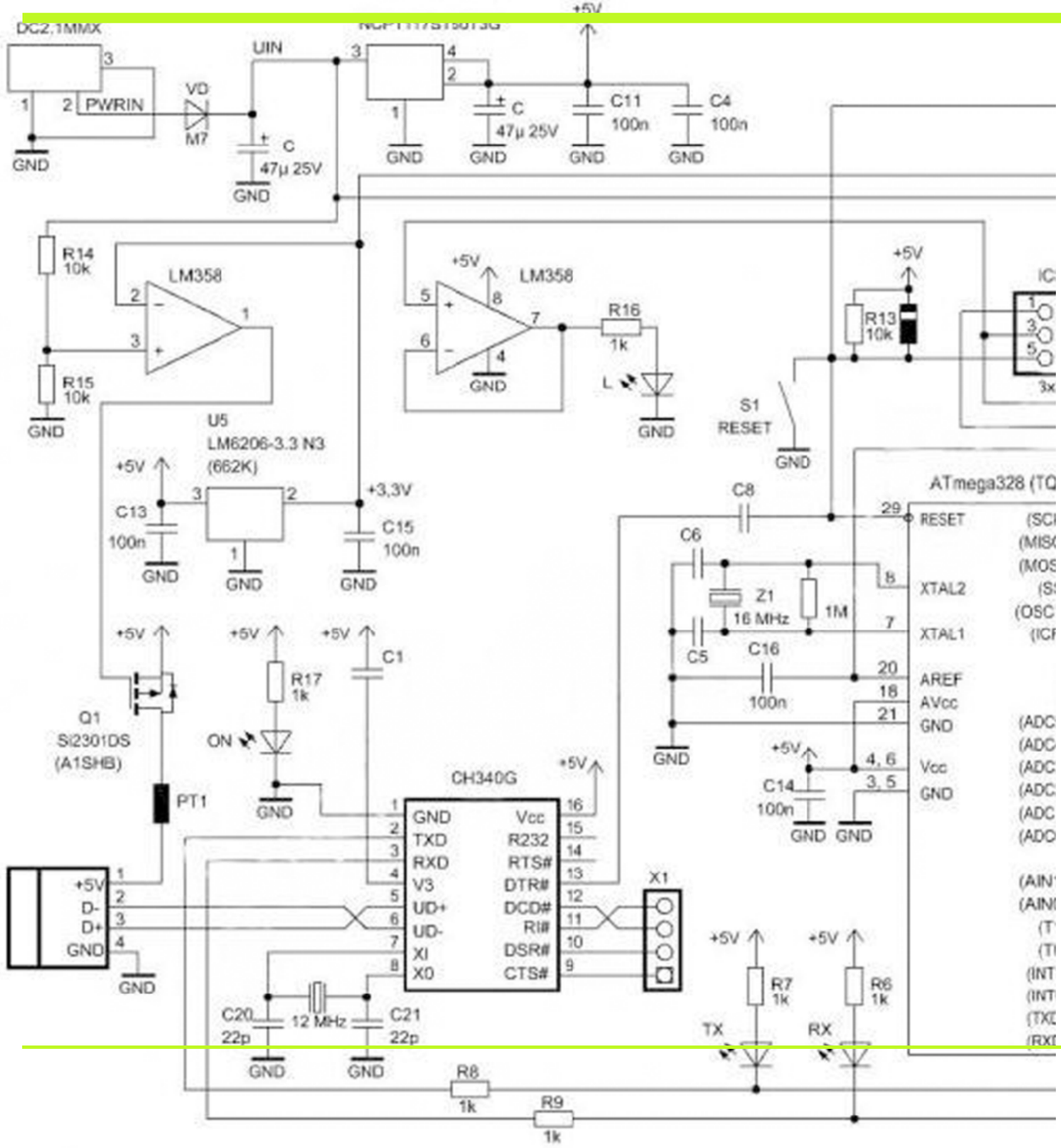
Can you use a variable to make the number on the serial monitor go up each time the button is pressed?

Now can you try and combine 2 sketches so you have one that sends a message to the console AND turns the LED on when you press the button?

Any Questions so far?

Sharing our Projects

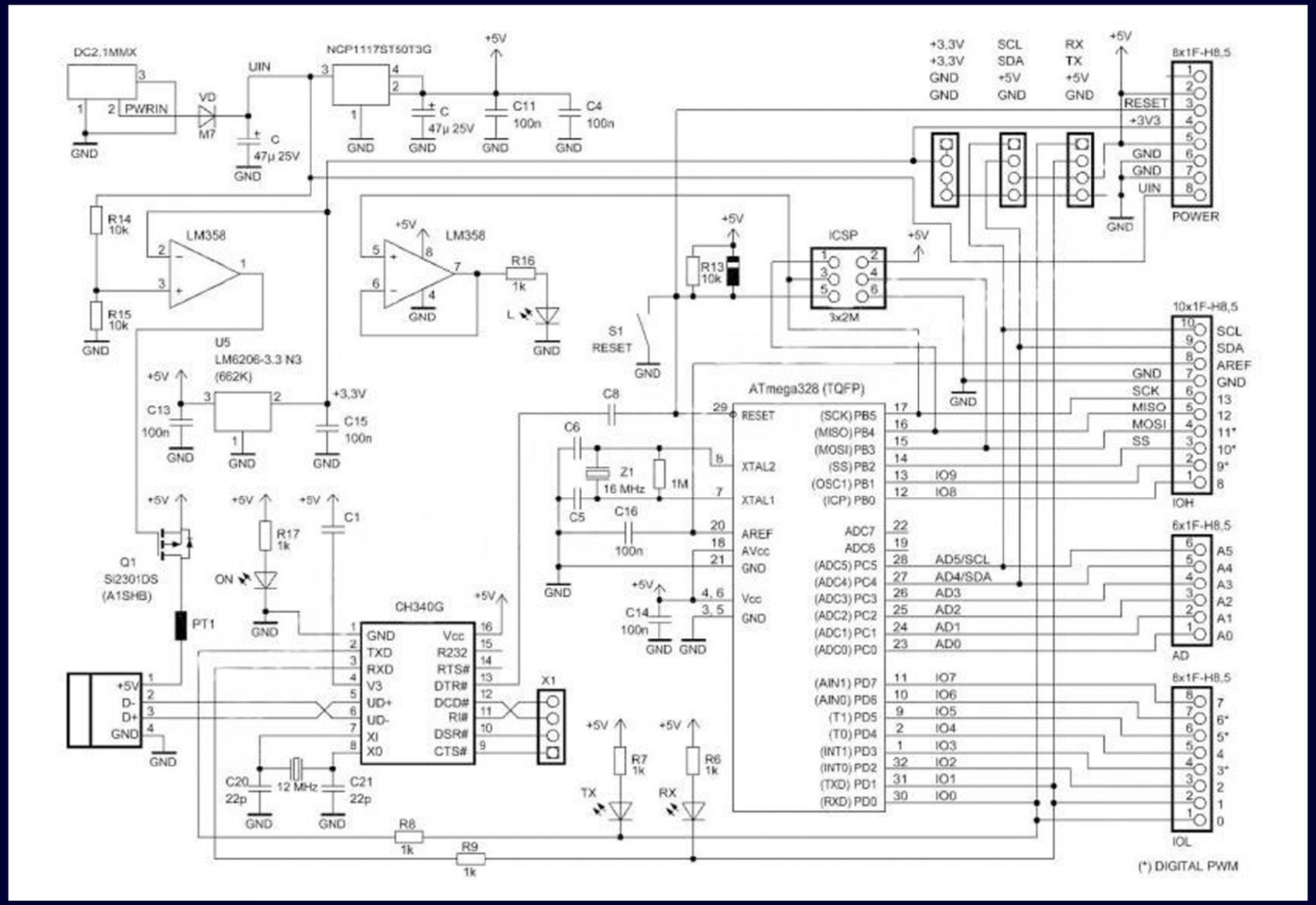
- If we want to share our projects with others, we need to split them up into parts.
  - The code (for microcontrollers this is sometimes called firmware)
  - The physical parts, these can be 3D models and files for the 3D printer and Laser Cutter.
  - The electrical wiring, this will be a schematic!

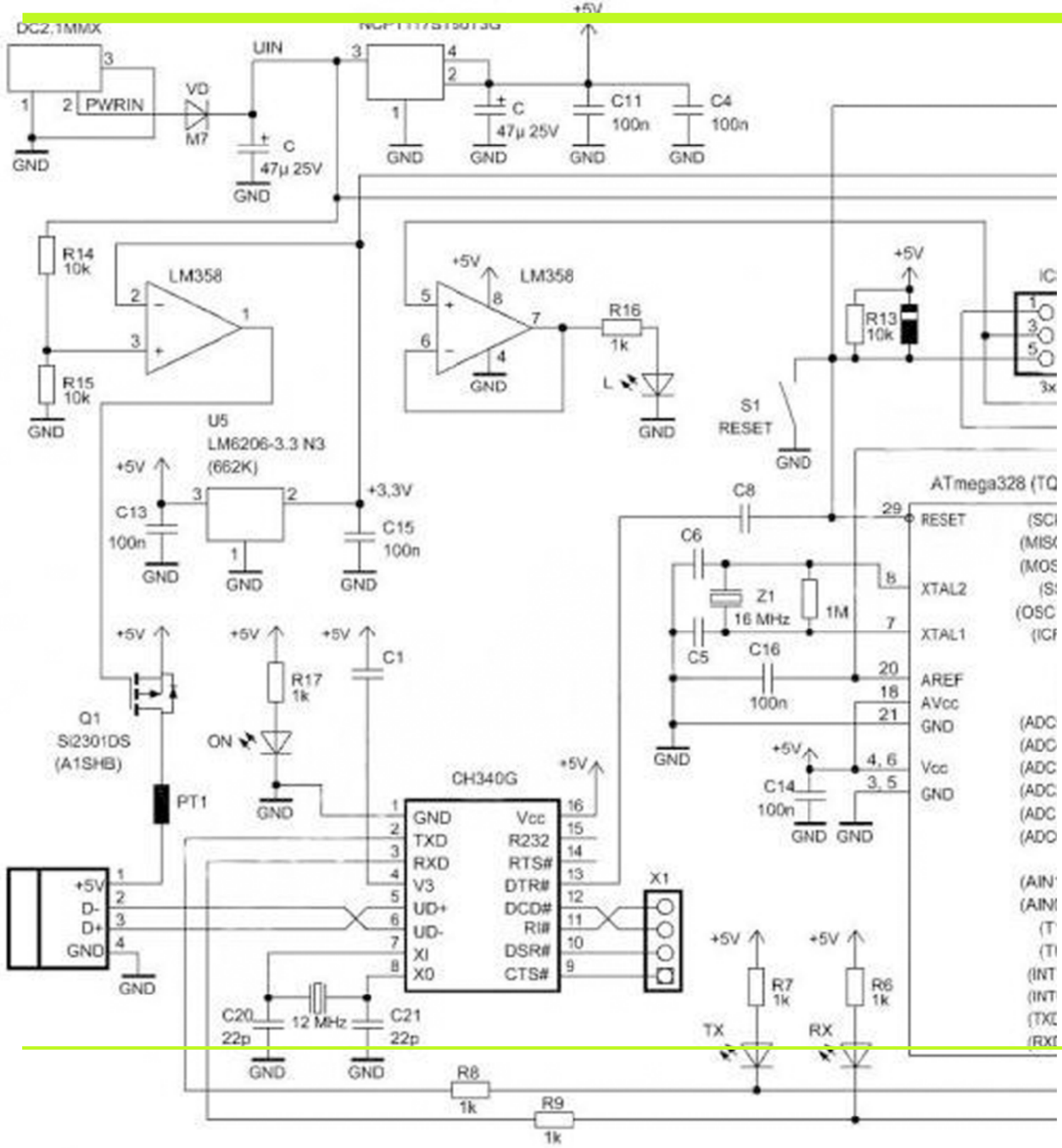


Schematics

- A map of the connections between the components in your project.
- Components are represented by Symbols.
- Wires/connections represented by straight lines.

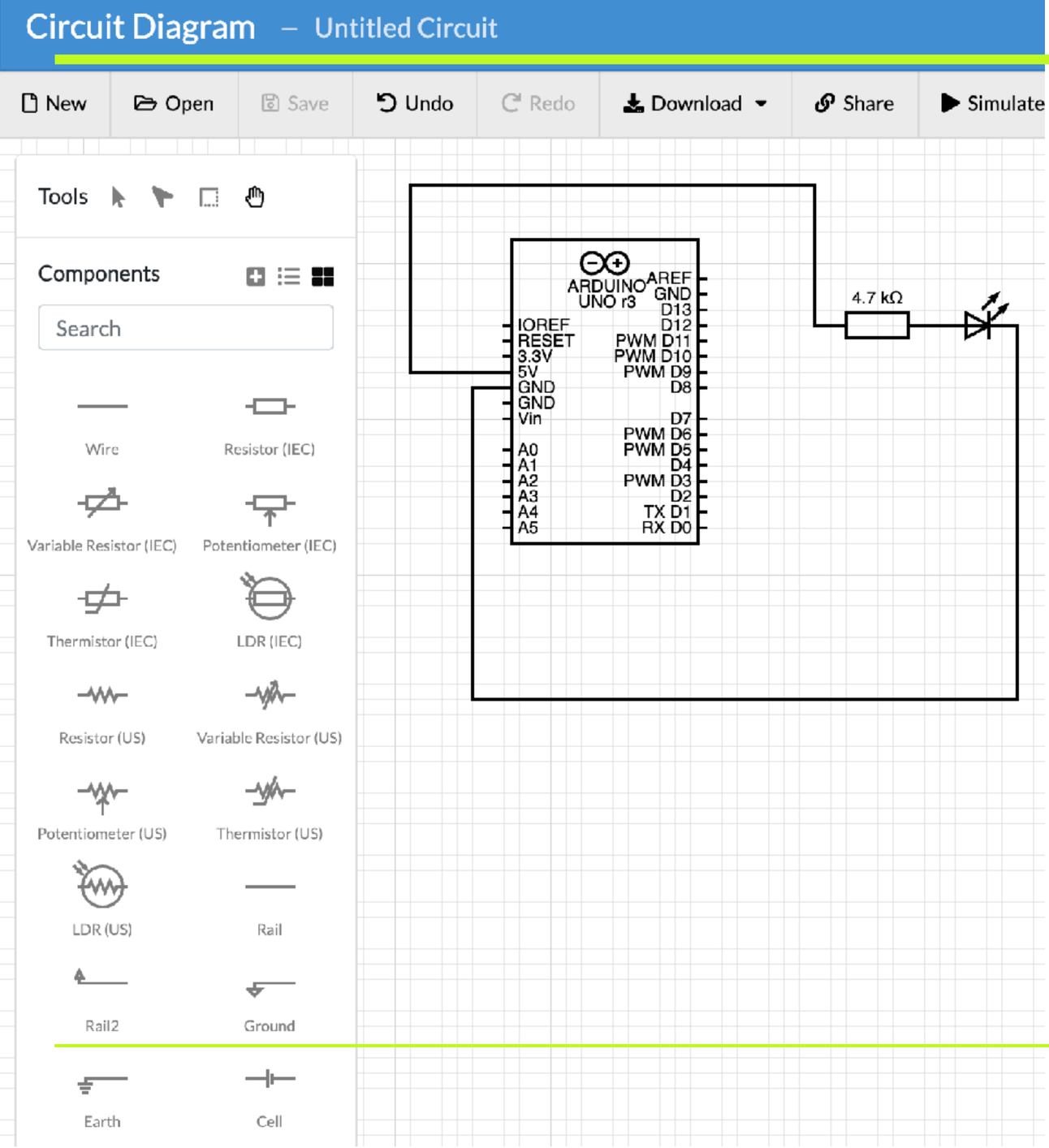
Can you guess what this Schematic is for?





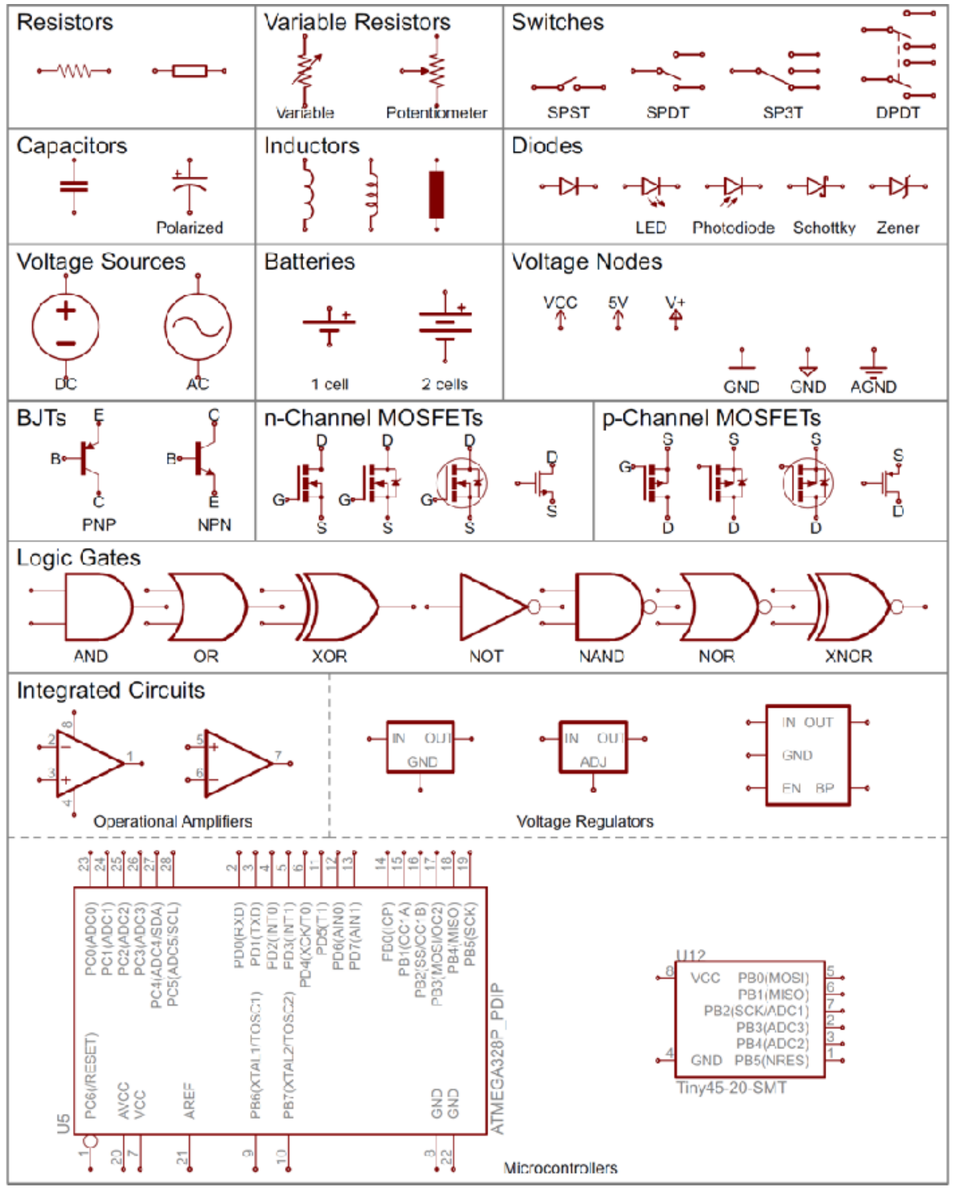
Schematics

- Schematics DON’T show how components are physically connected, so breadboards, pin headers solder points etc.
- Schematics DO show the simplified path the electricity takes.



Schematics

- There are lots of online tools to help us draw schematics, a very simple one is :
- https://www.circuit-diagram.org/editor/
- A more complex designer is EasyEDA :
- https://easyeda.com/
- EasyEDA also lets you use your schematics to design custom circuit boards, to make it really easy to assemble your project.



Schematics

- You can also use symbol charts like this one to draw your own schematics by hand!

Demo

Idea Generation!

Roll the Dice!

Roll the Dice using the Website!

https://robhallart.github.io/PhysCompIdeaDice/

Get into groups of 3 people

- Discuss prompts from the Digital Dice.
- Would this make a good project?
- What might this project be about?

First Assignment

- Please make the most interesting button you can.
- You can ONLY USE BUTTONS as your input but you can USE LEDs or the Serial Monitor as your output.
- Think about the meaning of what you make, how it will be used, the colours you choose, the design of your device.
- Can you make it tell a story? Can you make it surprise the user?
- Deliverables:
  - Upload this assignment to the VLE and include: A short video of the project (1 minute or less) , your code, a schematic of your circuit. (This can be hand drawn or done on a computer)
  - Post just the video of your device to the class forum so we can all see everyone's work.
  - Please think of this brief as an opportunity to try things out, the result does not have to be perfect, feel free to try things and fail, to play, to experiment, to get things wrong.