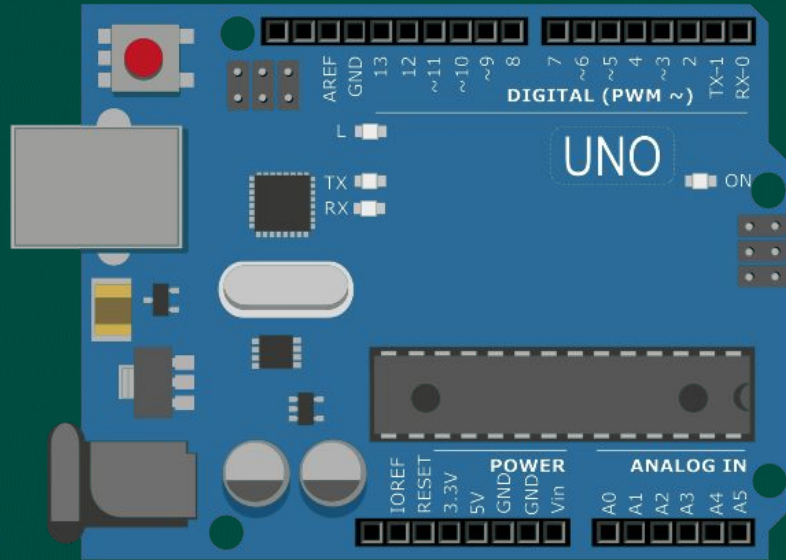


Lesson 1

Arduino Basics

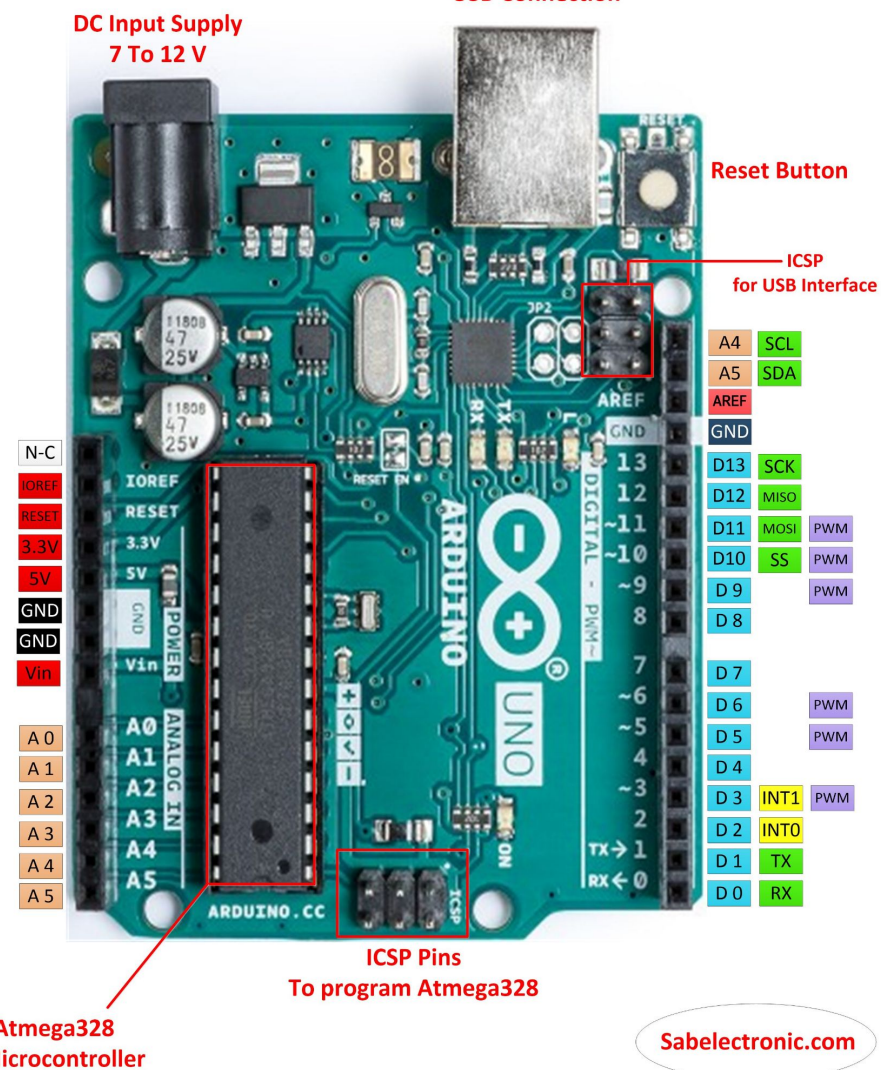


The background is a dark teal color. On the left side, there is a large, abstract shape composed of many small, bright green dots. These dots are arranged in a way that suggests a funnel or a large letter 'A' that tapers towards the bottom. A vertical beam of light, transitioning from purple at the top to a bright yellow-orange at the bottom, passes through the center of the dot formation. The text 'How Do They Work?' is positioned on the right side of the image, in a bold, white, sans-serif font.

**How Do They
Work?**

The Insides of an Arduino

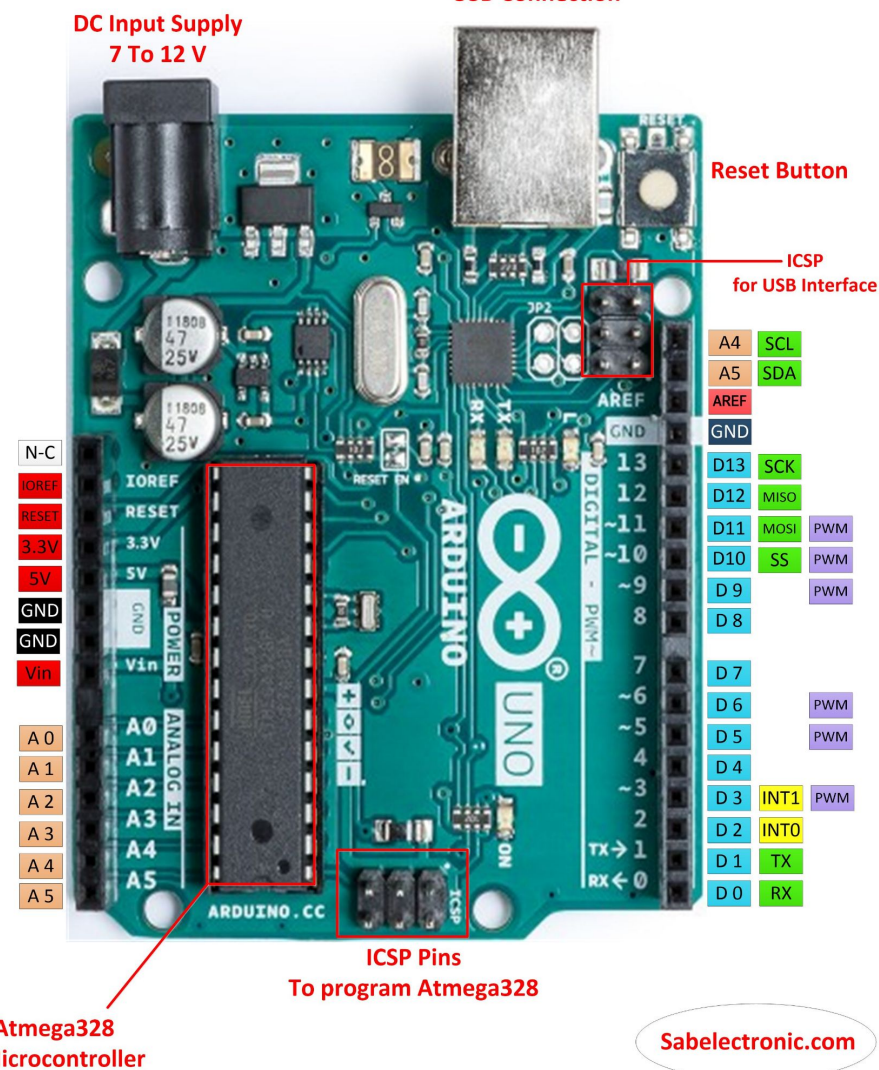
- Arduinos operate just like simple computers
- They are run by a *microcontroller*, which is a simple version of a CPU you might find in a computer at home
- This microprocessor is responsible for running single lines of code at a time
- It outputs or takes in signals at its pins
- You can use these signals to light up LEDs, play sounds, turn motors, or even use displays!
- The signals depend on the code that you *upload* to the Arduino



Important Pins

There are several pins that are important to know (they are labeled to the right):

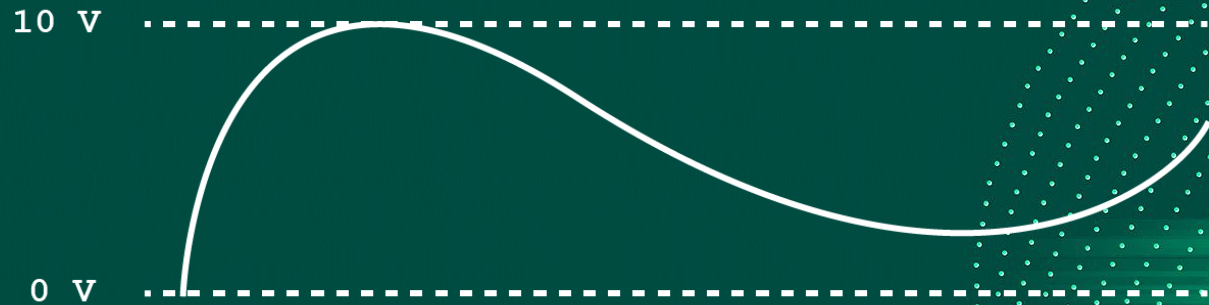
- 5V: Outputs a constant 5V signal (think “+” on a battery)
- 3.3V: Outputs a constant 3.3V signal
- GND: Is connected to ground, think “-” on a battery
- Vin: Voltage in, you can provide a voltage as an input to the Arduino
- A0-A5: Analog pins, input and output waves (for example, sound waves)
- D0-D13: Input and output digital signals, think 1 and 0



Digital




ANALOG SIGNAL



The background is a dark teal color. On the left side, there is a large, abstract shape composed of many small, bright green dots. These dots are arranged in a way that suggests a funnel or a large letter 'A' that tapers towards the bottom. A vertical beam of light, transitioning from purple at the top to a bright yellow-orange at the bottom, passes through the center of the dot formation. The text 'But How do We Code?' is positioned on the right side of the image, in a bold, white, sans-serif font.

**But How do
We Code?**

Preparing to Program an Arduino




Sign in to Arduino


[Forgot your password?](#)


SIGN IN


Don't have an account yet? [Create one.](#)

Or sign in with

 Google

 GitHub

 Facebook

 Apple

CHOOSE YOUR USERNAME

Username

☒

 I confirm to have read the [privacy policy *](#) and to accept the [Terms of service *](#)

☐

 I confirm my consent to receive your newsletter

☐

 I confirm my consent to the processing of my personal data for marketing purposes, consisting in commercial offers sent via email

☐

 I confirm my consent to automated processing of my personal data, by means of profiling, in order to receive commercial offers customized on the basis of my browsing and purchasing behavior

CREATE ACCOUNT

Preparing to Program an Arduino

WELCOME TO THE ARDUINO CREATE AGENT INSTALLATION!

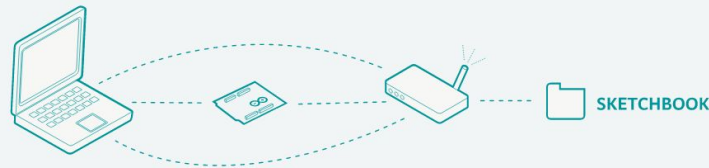


You're about to begin the process of downloading and installing the Arduino Create Agent. The agent will provide you with several features:

- Recognize Arduino boards and other supported devices connected to your computer via USB;
- Upload sketches from your web browser to your boards via USB or through a network;
- Read data from serial monitor, as well as write to it.

SETUP STEPS

1. DOWNLOAD AGENT
2. INSTALL AGENT
3. CONGRATULATIONS!



You need to download and install the Create Agent to be able to upload sketches from Arduino Cloud to your board. Please note that you have to be Administrator of your system to install the Agent. Administrative privileges aren't required for MacOS El Capitan or an earlier version.

macOS Ventura: if you are using macOS Ventura you need to download a specific version. Please follow the instructions here: [support document](#)

Source code for the Create Agent is available on [GitHub](#).

DOWNLOAD



**What Language
Does the Arduino
Use?**

Programming an Arduino

- Arduinos understand instructions from a universal serial bus connection, also known as USB (which you may recognize)
- So you can just plug right into your computer, “compile” the code (text -> 0’s and 1’s), and upload it to the Arduino!
- Arduinos are coded in a language known as C++
- It is an easy to learn language first made in 1985 by Danish computer scientist Bjarne Stroustrup

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

Programming an Arduino

- Each Arduino program contains a setup, and a loop, and is a “.ino” file
- The setup runs once when the program is first run, and the loop runs continuously
- The below setup segment contains a line of code that sets a pin to be the output for an LED
- The loop then turns the LED on and off with a 1000 millisecond, or 1 second delay between each “on” and “off” state.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```




**What Are We
Going to Do?**

Our Plan:

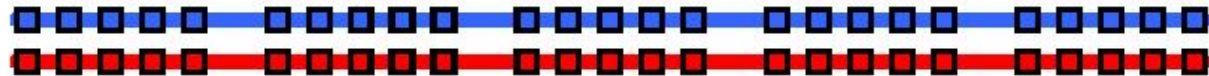
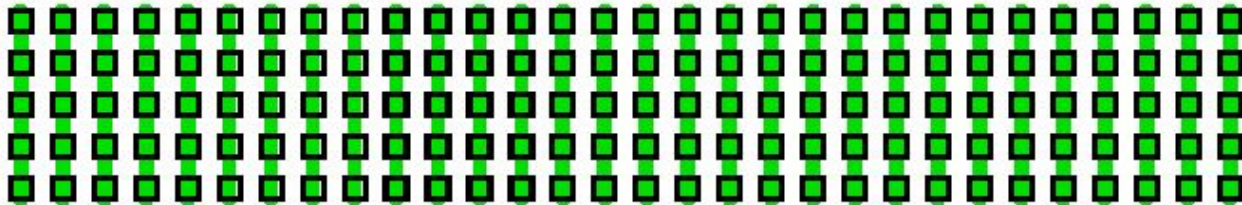
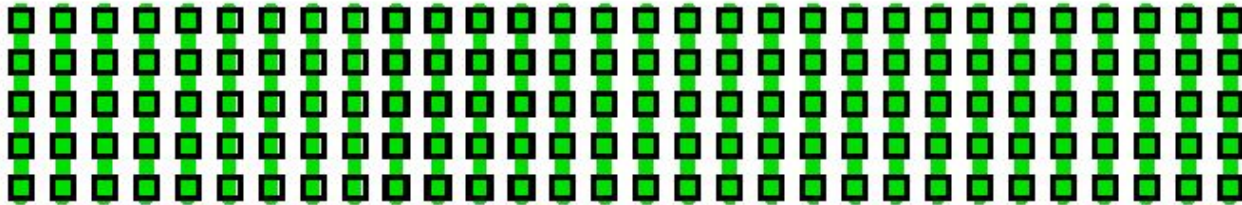
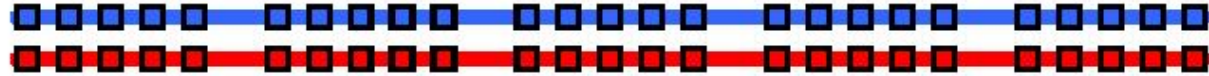
- Our goal is to make a buzzer “buzz” at a certain frequency.
- Using this we will send out “Hello World!” in Morse Code!
- First, let’s use the default “Blink” example found on the GitHub page to check if our setup works.
- Use this link to access the GitHub: <https://github.com/dgkaminski/Millbury-Electrical-Engineering>
- Once we confirm everything works, we can begin working on the circuit

How to send a program to an Arduino

- First, check the code with me (with the online development tool, you are limited to 25 code uploads a day)
- Then, click the upload button



Breadboard Basics

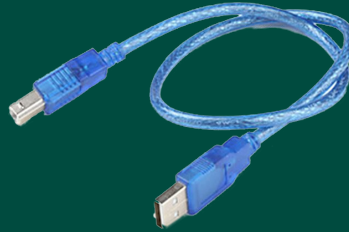


What We Need

1 x Arduino Uno
(Elegoo brand)



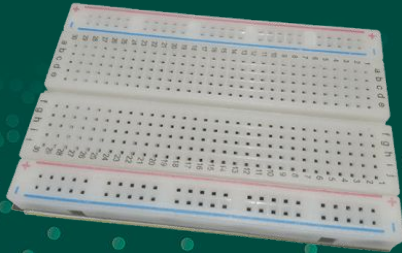
1 x USB Type B to
USB Type A Cable



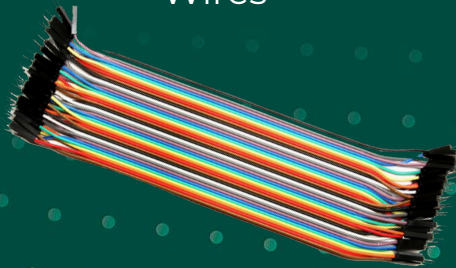
1 x Push Button
w/ Cap



1 x Breadboard



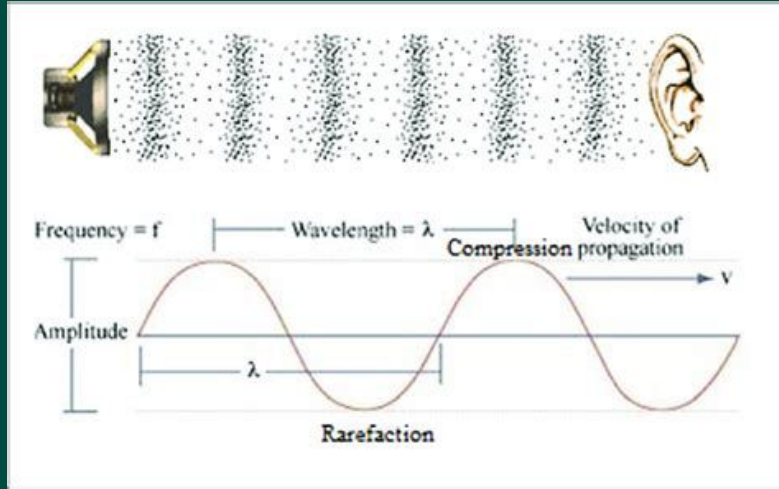
4 x Male-Male
Wires



1 x Passive Buzzer

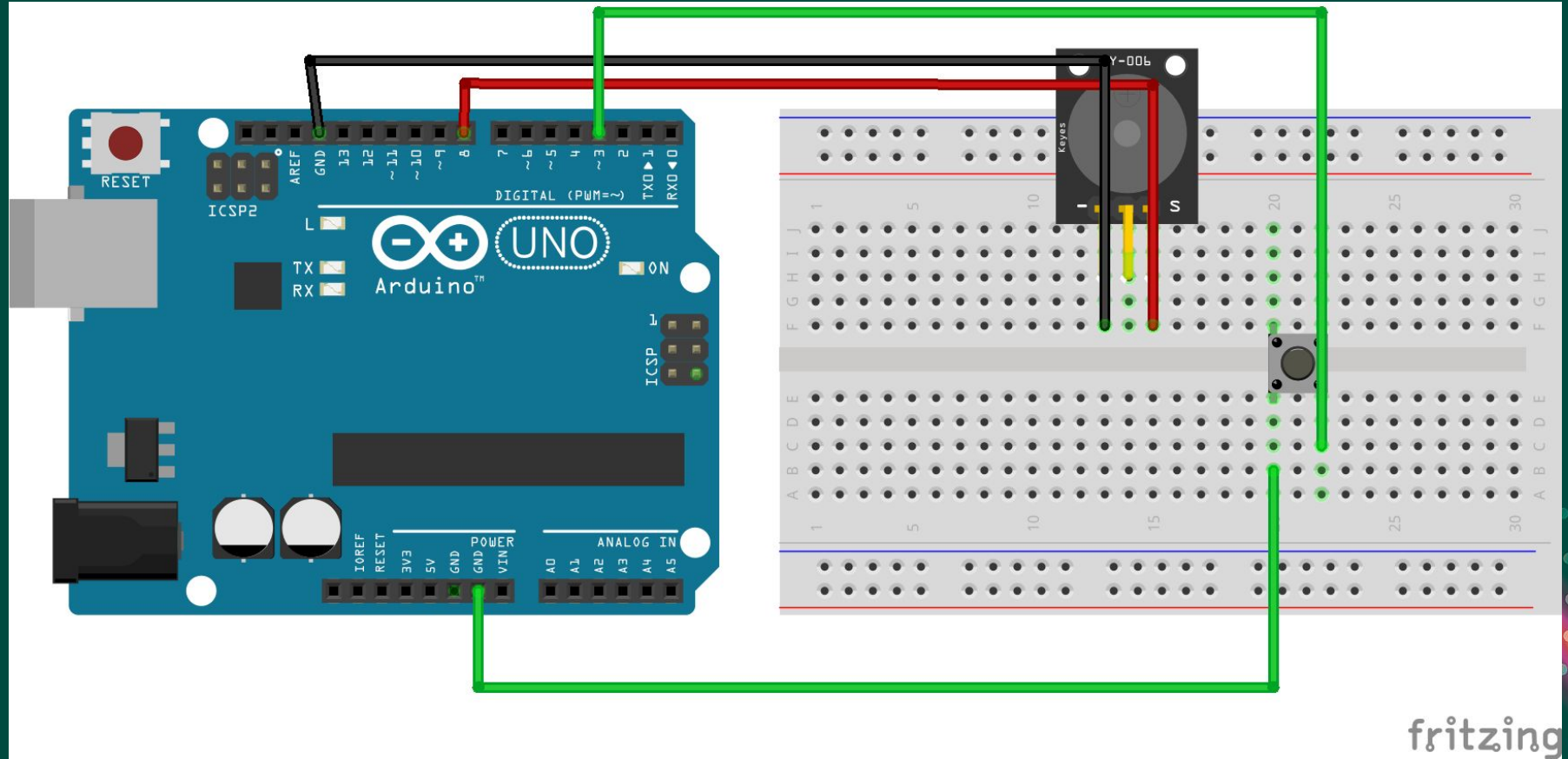


What is a Passive Buzzer?

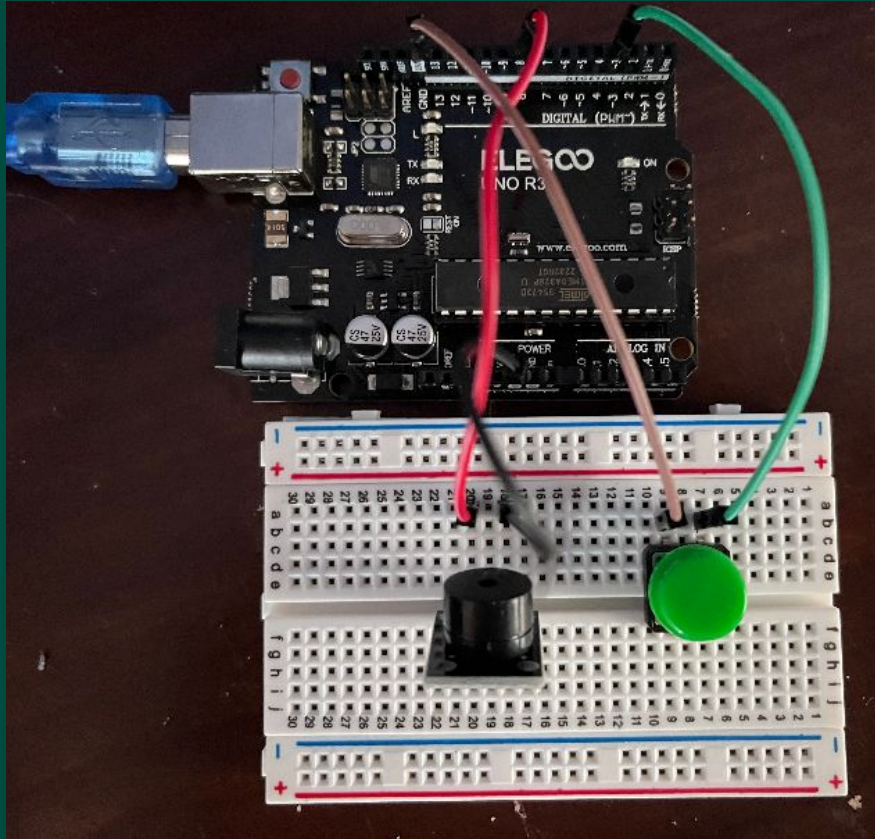


- There are two types of buzzers: passive and active.
- **Key Knowledge:** All speaker, buzzers, etc., make sound by oscillating a membrane, and thus compressing air, making a sound.
- **Active Buzzers:** Imagine them as buzzers that you turn on and leave running. They take in a simple digital signal, basically, a voltage, and have their own converter that turns this into a sound wave. They can only play one tone (think A, B, C, F, like in music).
- **Passive Buzzers:** These buzzers don't have the internal converter, meaning that they work exactly the same, but require an analog signal, aka., the electronic equivalent of your sound wave! Thus, unlike with active buzzers, you can play more than one tone.

What We're Building:



Everything Wired Up



Important Information:

- Make sure to push the button in all the way and to have it bridging the gap between the two sides of the breadboard.
- Your buzzer may look slightly different from mine, and you can place it diagonally across several rows so that you do not have to bend its pins, as they are fragile.
- You can use whatever colors for the wires that you want, just make sure to remember which ones are which.
- The buzzer has a "+" side and a "-" side, they should both be marked, connect the "+" to pin 8 and the "-" to a ground pin (labeled **GND**).

Testing the Circuit

To test the circuit, follow these steps:

1. Connect your Arduino to your Chromebook
2. Upload the code (also found on the GitHub in the Lesson 1 folder, called *BuzzerTest*)
3. Try pressing the button and see if the buzzer makes a sound.


If this all works, try sounding “Hello World!” on the Arduino using the push button. It should be done as follows (where “.” is a short press, “-” is a long press, and “/” is a break of a few seconds):

Morse Code:

.... . .-.. .-.. --- / .-- --- .- .-.. -.. -.-.-

If this all works:

1. Try to code in the lengths of the sounds to be automatic. The code for this is located on the GitHub if you get stuck.
2. Try to play one of the songs found on the GitHub page on your buzzer.

The background is a dark teal color. It features several abstract, glowing patterns. On the left and right sides, there are curved, particle-like structures made of small white dots, resembling a stylized 'S' or a double helix. These are accented with bright pink and magenta light streaks that radiate outwards. Diagonal green lines also cross the frame, adding to the dynamic, high-tech feel.

**Congratulations, you're
now an electrical
engineer!**



Thank you for coming to this lesson, and I hope you learned something!

Come back next week for more!

CREDITS: This presentation template was
created by Slidesgo, including icons by Flaticon,
and infographics & images by Freepik.

Please keep this slide for attribution.