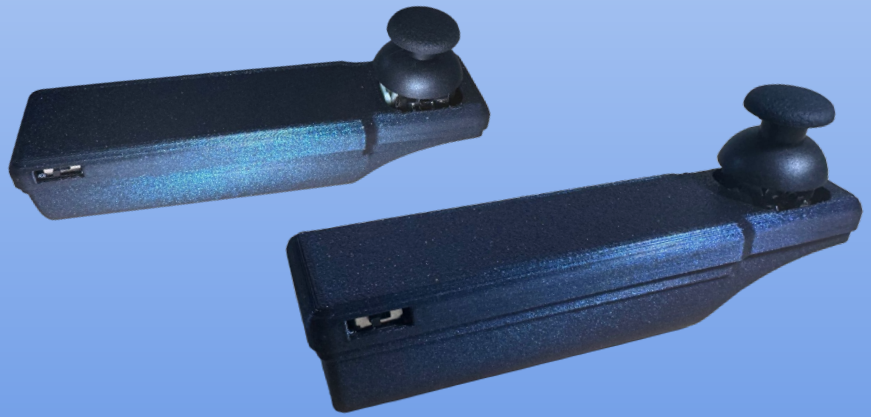
Mad Science 102: Remote Control

Mad Science 102: Remote Targeting

You will Learn:

- Power Systems: Batteries, Charging
- Reading Input: Buttons, Joysticks
- Wireless Communication: WiFi (ESP Now)
- More Programming Microcontrollers!



Mad Science 102: Remote Targeting

Schedule:

Date	Subject	Project
11/8/25	Power Up!	Battery Power for the Tower
11/15/25	Take Control!	Remote Control Transmitter
11/22/25	Transmission Online!	Remote Control Transmitter

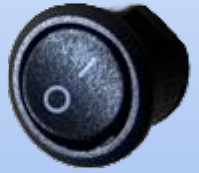
Mad Science 102: Remote Targeting

Part 1: Battery Systems

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Project: Battery Power for the Tower



You will Learn:

- Reading Voltage with Multimeters
- Switches
- Batteries

Mad Science 102: Remote Targeting

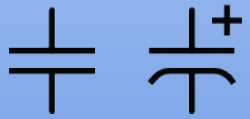
Part 1: Battery Systems

Components and their symbols:

Battery (provides power)

Resistor (bleeds off power)

LED (Light Emitting Diode)



Capacitor (a thing that oscillates)

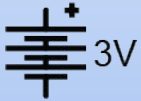
Transistor (a switch)

Switch (Also a switch!)

Mad Science 102: Remote Targeting

Part 1: Battery Systems

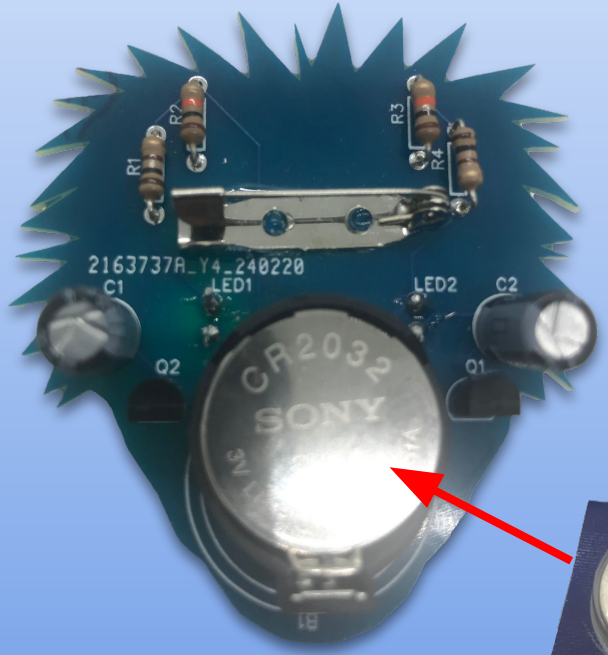
Batteries:



Batteries provide power for your circuit. They have a set amount of voltage, and can provide current up to a rated maximum amount.

In 101, we used a coin cell, 3 Volt CR2032 lithium battery

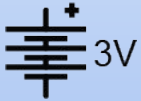
Batteries have polarity - positive is the flat side of the symbol



Mad Science 102: Remote Targeting

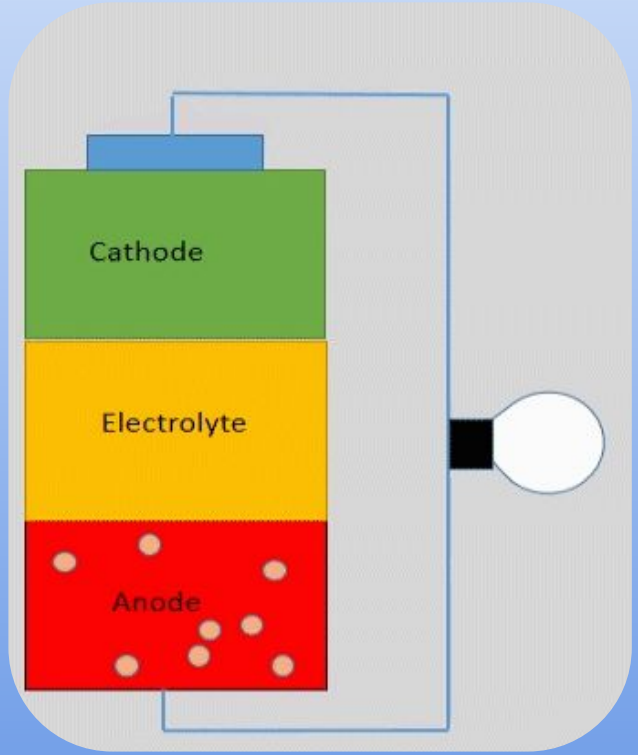
Part 1: Battery Systems

Batteries:



But how do they actually work???

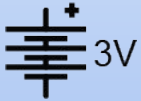
- Batteries are chemical “cells” where one side has a lot of electrons, and the other side has not very many electrons.
- When there is a connection, the electrons naturally try to balance out on both sides.
- When there is not a connection, they just sit there.



Mad Science 102: Remote Targeting

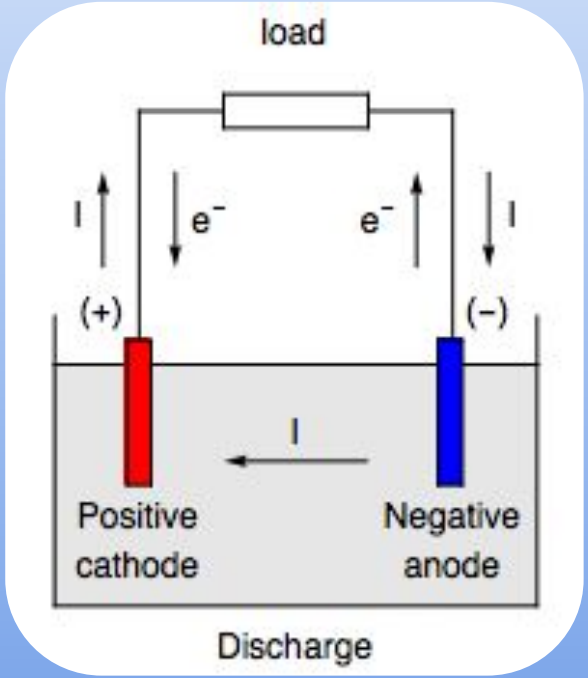
Part 1: Battery Systems

Batteries:



Rechargeable Batteries

- Some Batteries can move the electrons back to the other side when they're dead.



Mad Science 102: Remote Targeting

Part 1: Battery Systems

What is Power?

- "Knowledge is power."

  —Francis Bacon

- "With great power there must also come great responsibility!"

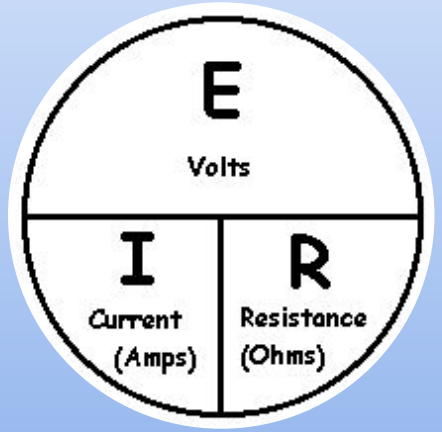
  —Stan Lee

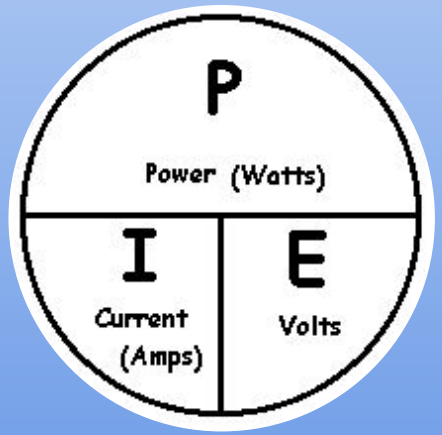
- "Power is the ultimate aphrodisiac."

  —Henry Kissinger

- "The object of power is power."

  —George Orwell

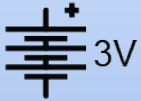




Mad Science 102: Remote Targeting

Part 1: Battery Systems

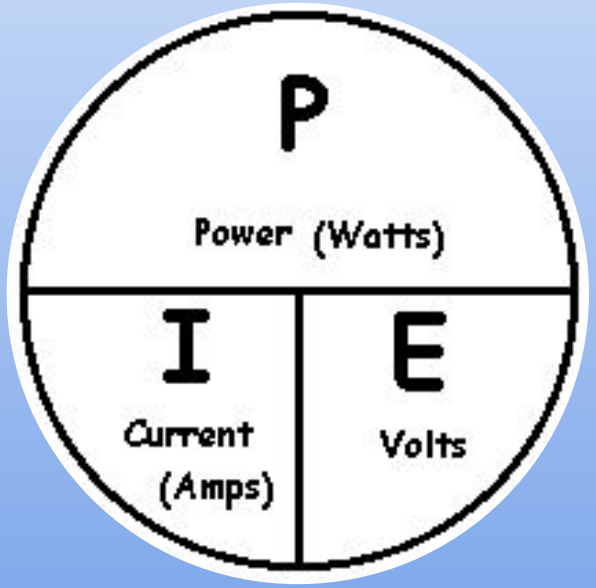
Batteries:



How much Power is in a Battery:

Power of a Battery is rated in Amp-Hours or Watt Hours:

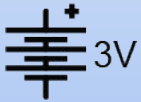
- Amp-Hours: How many hours can the battery run if it is running at 1 amp.
- Watt-Hours = Amp-Hours/Battery Voltage



Mad Science 102: Remote Targeting

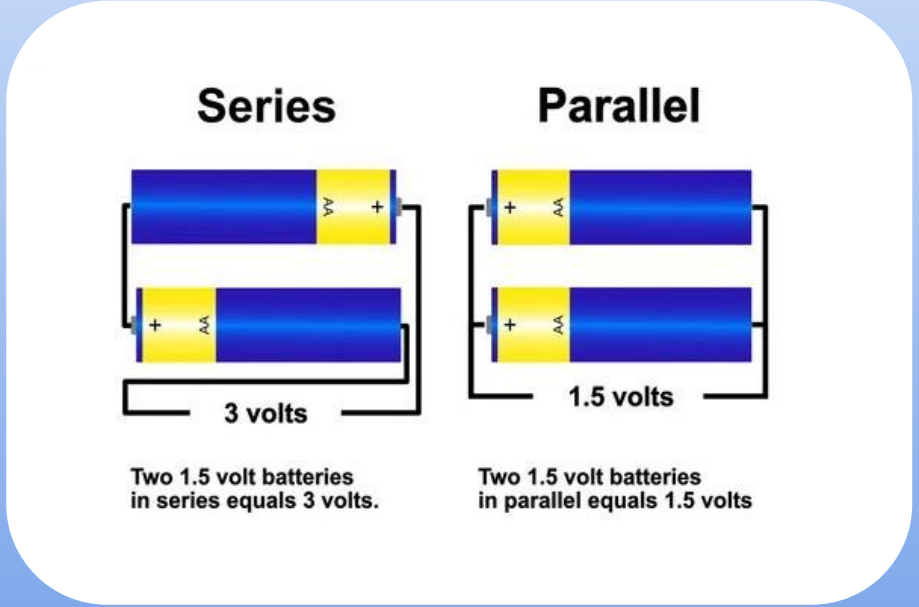
Part 1: Battery Systems

Batteries:



Multiple Batteries in Series or Parallel

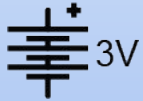
- In Series, Voltages from multiple batteries are added together
- In Parallel, Amp-Hours (capacity) are added together
- Example: 2x 1.5V batteries, with 1000 mAh
  - Series: 3V, 1000mAh
  - Parallel: 1.5V, 2000mAh



Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



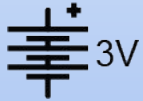
Types of Batteries

- Lead-Acid: Cars

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



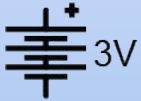
Types of Batteries

- Lead-Acid: Cars
- Alkaline: like most AA, AAA, C, D

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



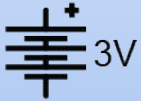
Types of Batteries

- Lead-Acid: Cars
- Alkaline: like most AA, AAA, C, D
- Nickel Metal Hydride: Early Rechargeables

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



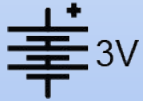
Types of Batteries

- Lead-Acid: Cars
- Alkaline: like most AA, AAA, C, D
- Nickel Metal Hydride: Early Rechargeables
- Lithium Ion: Modern Rechargeables

Mad Science 102: Remote Targeting

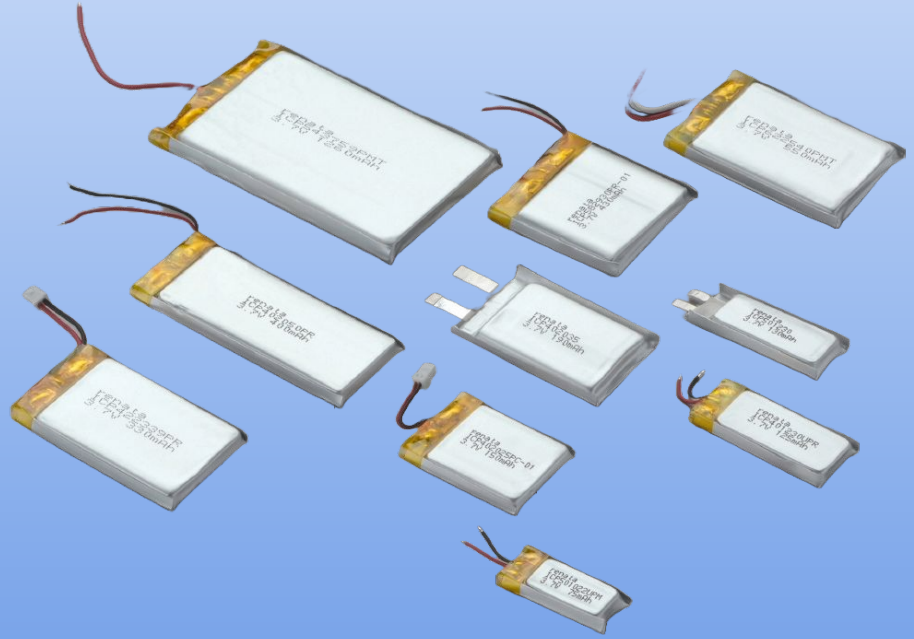
Part 1: Battery Systems

Batteries:



Types of Batteries

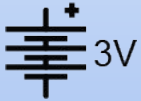
- Lead-Acid: Cars
- Alkaline: like most AA, AAA, C, D
- Nickel Metal Hydride: Early Rechargeables
- Lithium Ion: Modern Rechargeables
- Lithium Polymer: A very common type of Lithium Ion



Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



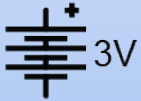
Types of Batteries

- Lead-Acid: Cars
- Alkaline: like most AA, AAA, C, D
- Nickel Metal Hydride: Early Rechargeables
- Lithium Ion: Modern Rechargeables
- Lithium Polymer: A very common type of Lithium Ion
- Lithium Iron Phosphate: Safer

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



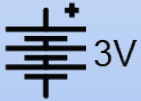
Types of Batteries

- Lead-Acid: Cars
- Alkaline: like most AA, AAA, C, D
- Nickel Metal Hydride: Early Rechargeables
- Lithium Ion: Modern Rechargeables
- Lithium Polymer: A very common type of Lithium Ion
- Lithium Iron Phosphate: Safer
- Sodium Ion: Good in cold. Better for environment.

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



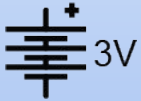
Types of Batteries

- Special Shout-Out to 18650 Batteries:
  - The most common industrial battery
  - Bigger
  - 3.7V
  - 2200mAh

Mad Science 102: Remote Targeting

Part 1: Battery Systems

Batteries:



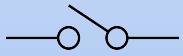
We are using: 18500 Batteries

- Rechargable
- Similar to the 18650, which is the most common rechargeable battery
- 3.7V or so (goes from 4.2V to 3.2V)
- 2000 mAh
- Name is Dimensions

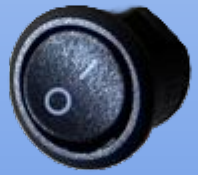
Mad Science 102: Remote Targeting

Part 1: Switches

Switches!



This one's simple, just turns stuff on and off. Opening or closing the circuit. Allowing electrons to flow or not.

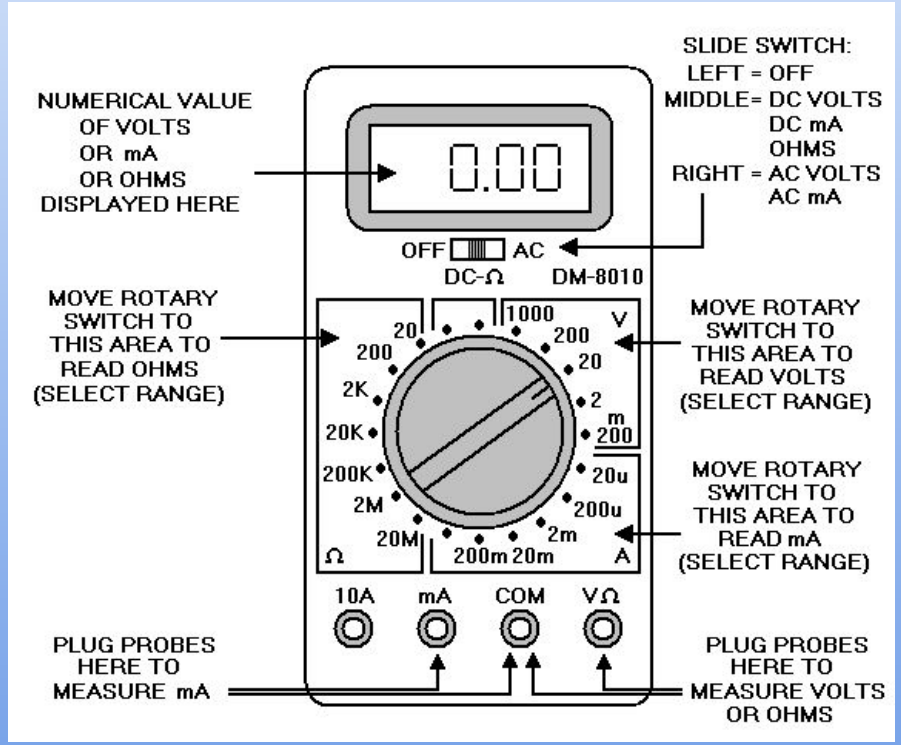


Mad Science 102: Remote Targeting

Part 1: Multimeters

Anatomy of a Multimeter

- Multimeters can test Voltage (Volts), Current (Amps), Resistance (Ohms), etc.
- We will only be testing voltage.



Mad Science 102: Remote Targeting

Part 1: Multimeters

How to test Voltage

- Black Wire goes to Com
- Red wire goes to V
- Dial goes to V, with the number just above what you expect the voltage to be. e.g. 20, if you are expecting 10.
- Put the red lead on the positive side of the item you are testing, and the black lead on the negative side of what you're testing.

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

Test the various power sources and find their voltage

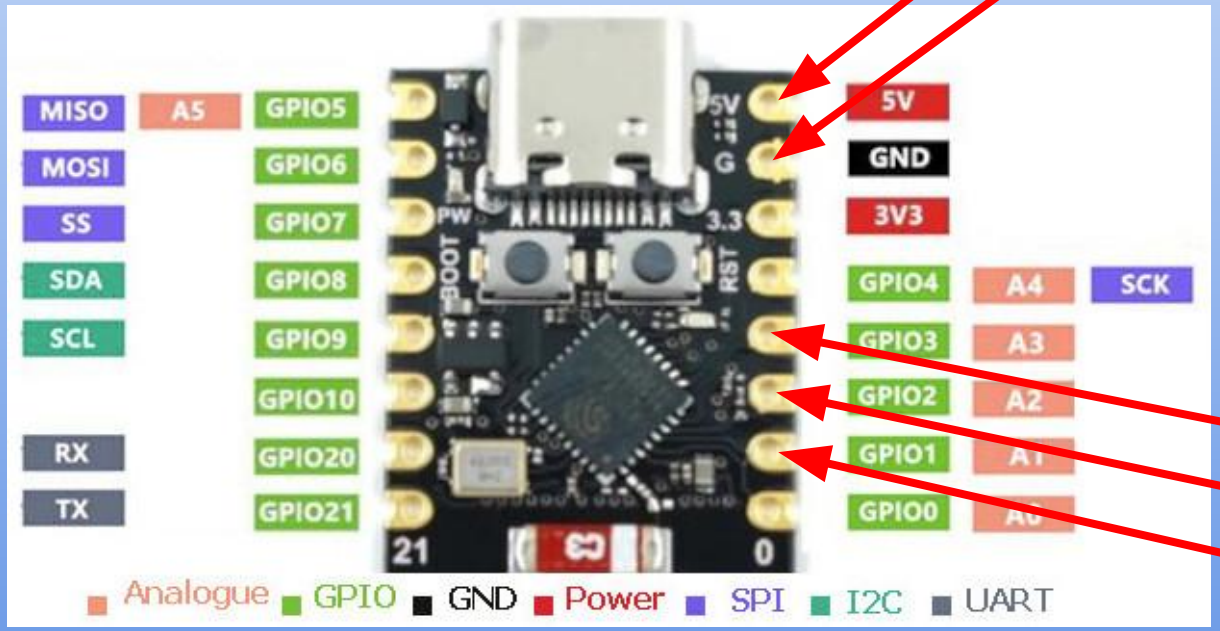
Battery Type	Nominal Voltage	Full Voltage	Dead Voltage
Alkaline (Disposable)	**1.5 V**	1.5V – 1.65V	~0.9V – 1.0V
NiMH (Rechargeable)	**1.2 V**	~1.4V	~1.0V
NiCd (Rechargeable)	**1.2 V**	~1.3V	~1.0V
Lithium (Li-FeS_2) (Disposable)	**1.5 V**	~1.7V – 1.8V	~0.8V – 1.1V
Lithium-Ion (14500) (Rechargeable)	**3.6 V – 3.7 V**	~4.2V	~2.7V – 3.0V

Mad Science 102: Remote Targeting

Part 1: Batteries

Powering the ESP32-C3

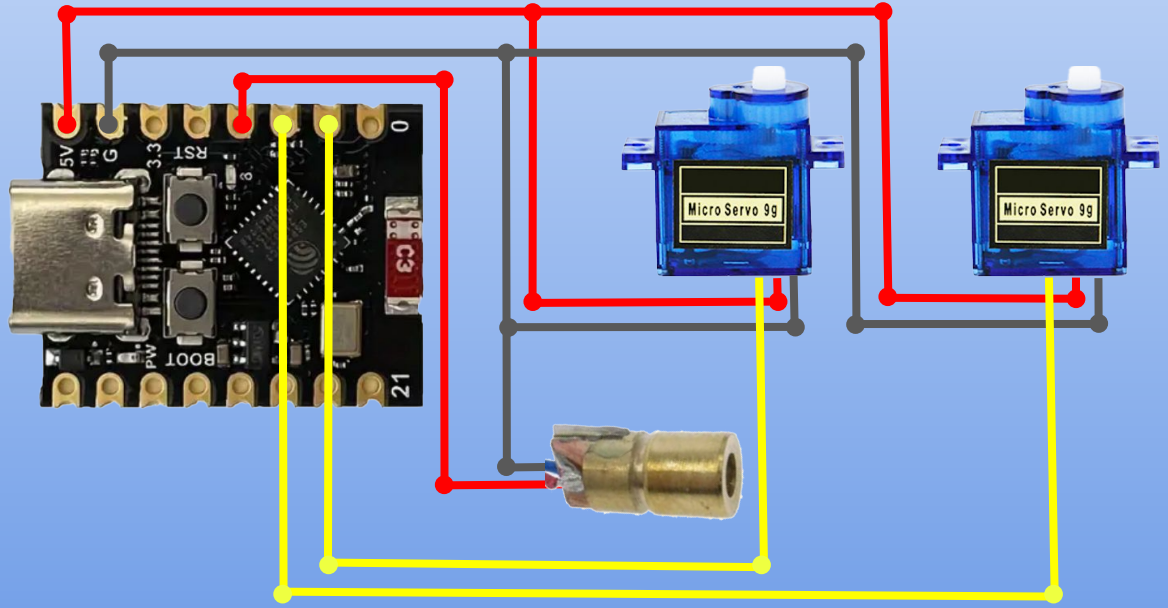
Remember Our ESP32-C3 Connection Map?



Mad Science 102: Remote Targeting

Part 1: Batteries

Wiring Diagram

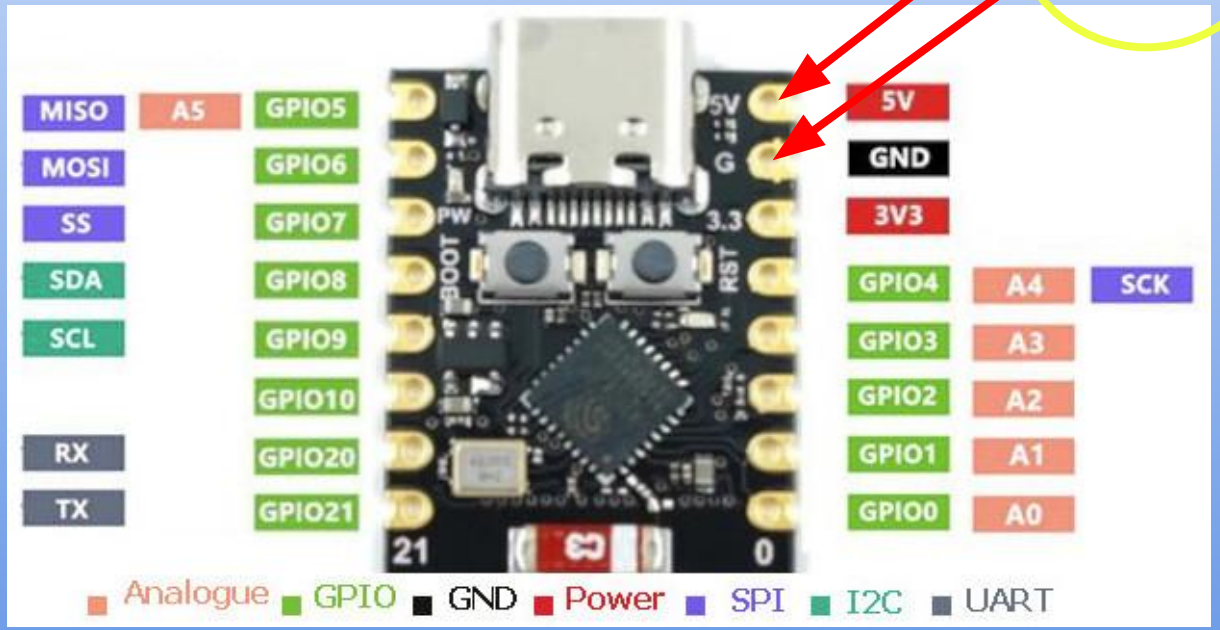


Mad Science 102: Remote Targeting

Part 1: Batteries

Powering the ESP32-C3

Our ESP32-C3 Connection Map:

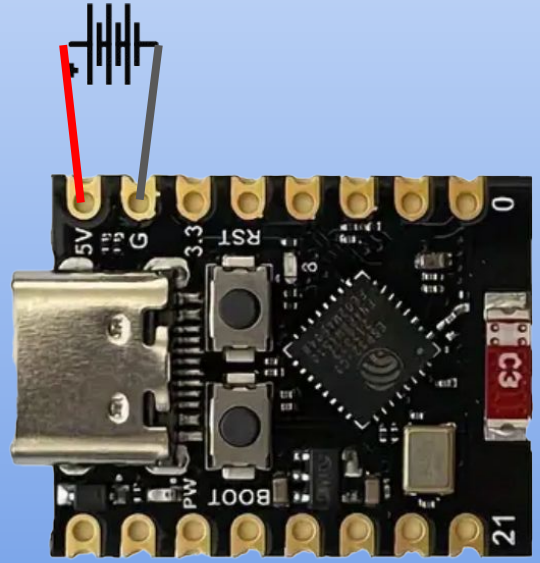


Mad Science 102: Remote Targeting

Part 1: Batteries

Powering the ESP32-C3

Attach a Battery to 5V and G!



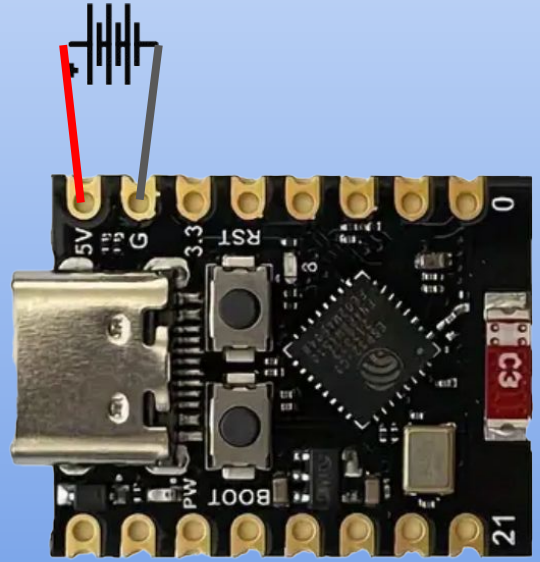
Mad Science 102: Remote Targeting

Part 1: Batteries

Powering the ESP32-C3

Attach a Battery to 5V and G!

Great! except, now it's on all the time.
How do we turn it off????



Mad Science 102: Remote Targeting

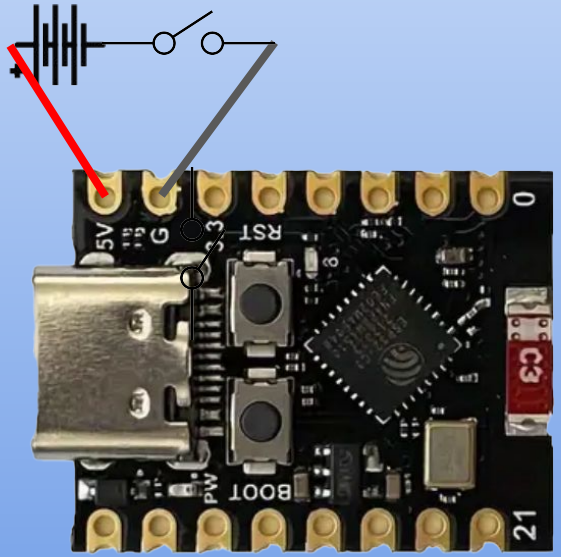
Part 1: Batteries

Powering the ESP32-C3

Attach a Battery to 5V and G!

Great! except, now it's on all the time.
How do we turn it off????

Add a Switch on either Positive or
Negative side of battery!



Mad Science 102: Remote Targeting

Part 1: Batteries

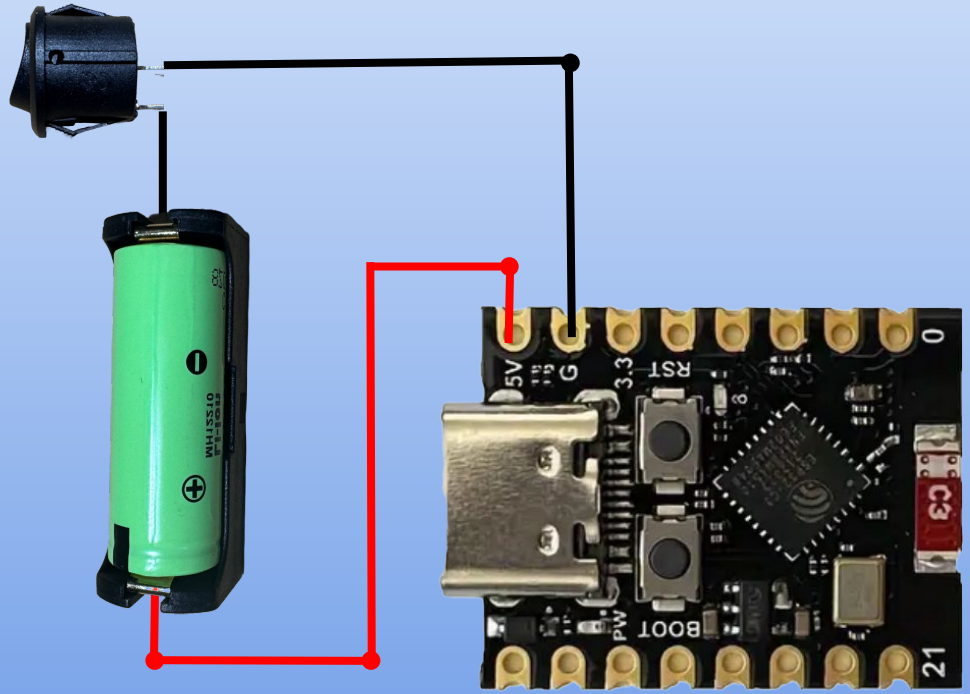
Powering the ESP32-C3

Attach a Battery to 5V and G!

Great! except, now it's on all the time.
How do we turn it off????

Add a Switch on either Positive or
Negative side of battery!

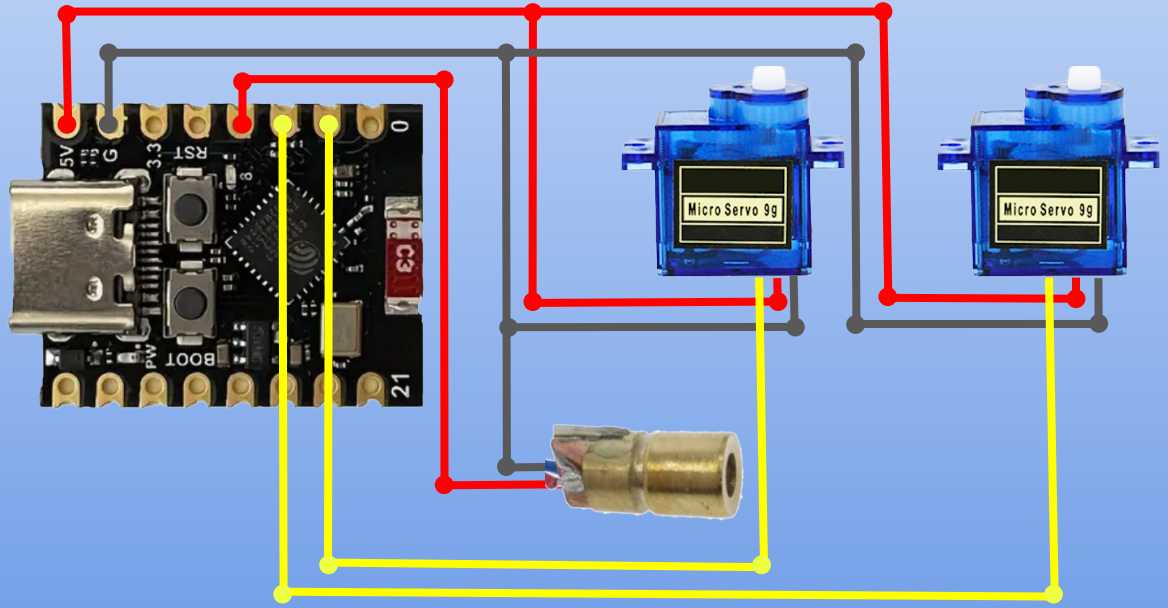
What it really looks like



Mad Science 102: Remote Targeting

Part 1: Batteries

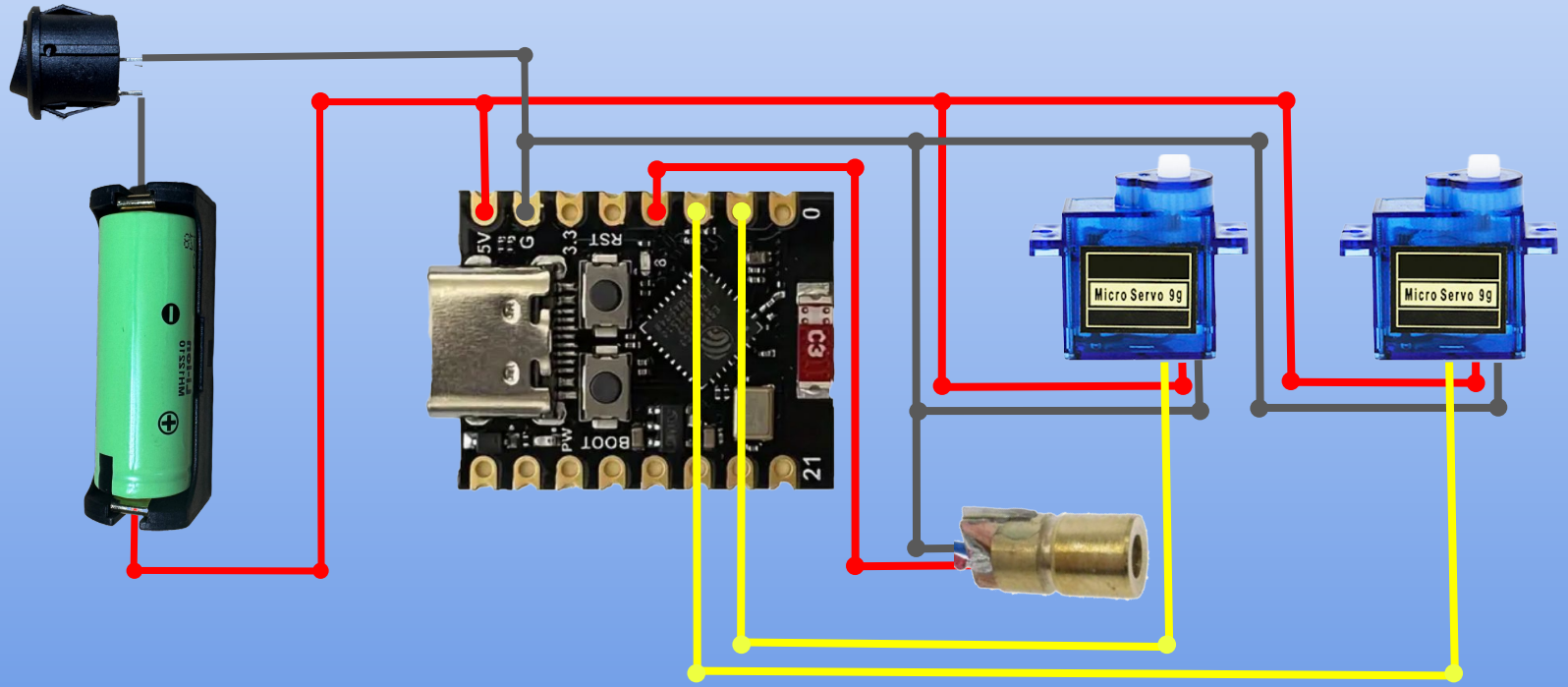
Except, Remember this? How do we make it fit together?



Mad Science 102: Remote Targeting

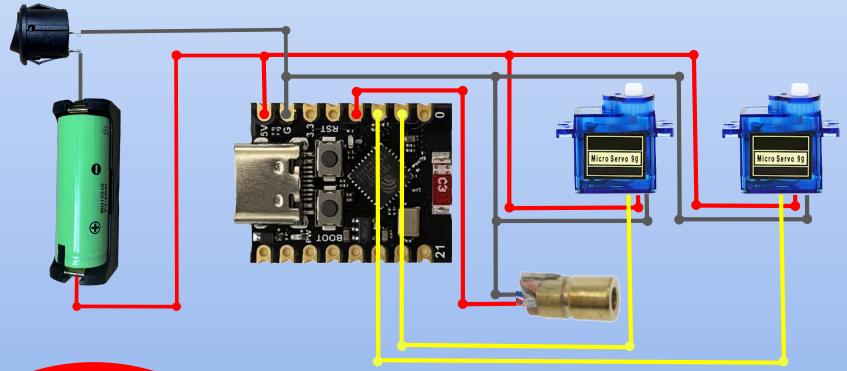
Part 1: Batteries

Join and You Shall Have Power!!



Mad Science 102: Remote Targeting

Part 1: Batteries



VERY IMPORTANT

Do Not Plug USB Into Your Board While Switch is ON!!

(OK while Switch is off)

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

BOM:

- 6" Red Silicone Wire
- 6" Black Silicone Wire
- 18500 Battery
- 18500 Battery Holder
- 18500 Battery Charger
- Switch
- New 3D Printed Body
- New 3D Printed Base
- Optional: New 3D Printed Laser Holder
- Optional: New 3D Printed Servo Holder

Mad Science 102: Remote Targeting

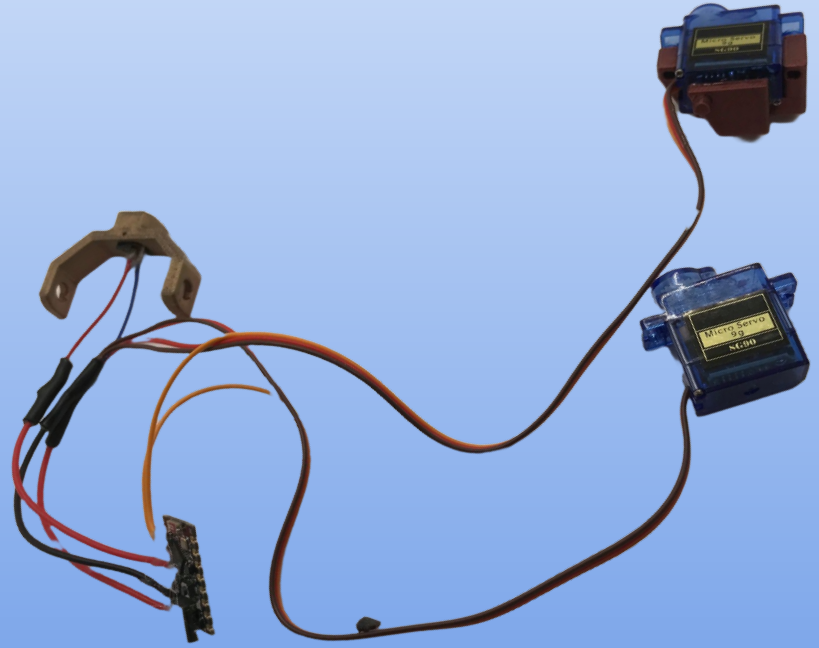
Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

Step 1:

Pull your electronics out of your tower body.

You can keep your laser holder and servo holder if you wish.



Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

Step 2: Switch Wires

Cut your black wire into two parts: one 2" and the other 4" long

Strip and tin red and black wires

Solder short black wire onto one leg of switch

Solder long black wire onto other leg of switch

Heat Shrink each leg of switch with a small length of heat shrink tubing (~1 or 2 cm)

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

Step 3: Battery Wires

MAKE SURE YOU DON'T HAVE THE BATTERY IN THE BATTERY HOLDER

Solder the other end of the short black wire to the Negative (-) side of the battery holder

Solder long red wire onto the Positive (+) side of the battery holder

TIP: Use lots of solder!

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

Step 4: Test Your Power Circuit

Put Battery in Battery Holder - Be mindful of +/-

Turn Switch On

Test Voltage with Multimeter (should be full)

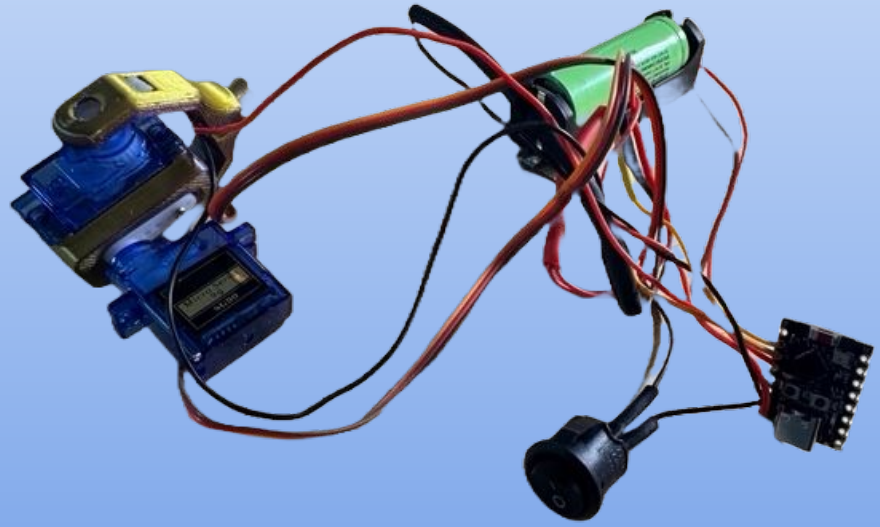
Turn Switch Off

Test Voltage with Multimeter (should be 0)

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!



Step 5: Power the Microcontroller

REMOVE THE BATTERY FROM THE HOLDER AGAIN

Solder the other end of the long black wire to the Ground (G) pin of the board. You may have to desolder and redo the servo ground.

Solder the other end of the long red wire to the 5V pin of the board. You may have to desolder and redo the servo power.

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

Step 6: Test the Power to the Microcontroller

MAKE SURE SWITCH IS OFF

Put Battery back in Battery Holder - Be mindful of +/-

Turn Switch On

Does the laser turn on? Servos?

Test 5V/G with Multimeter

Mad Science 102: Remote Targeting

Part 1: Batteries

Enough Talk, Let's Do Some Stuff!

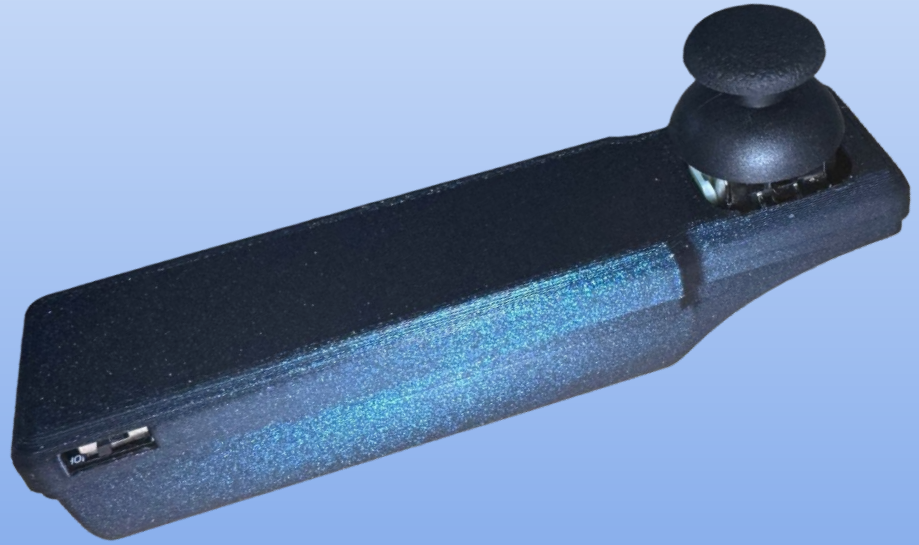
Step 7: Rebuild Tower

Turn Switch Off

Replace electronics in 3D Printed Case

Mad Science 102: Remote Targeting

Part 2: Remote Control!



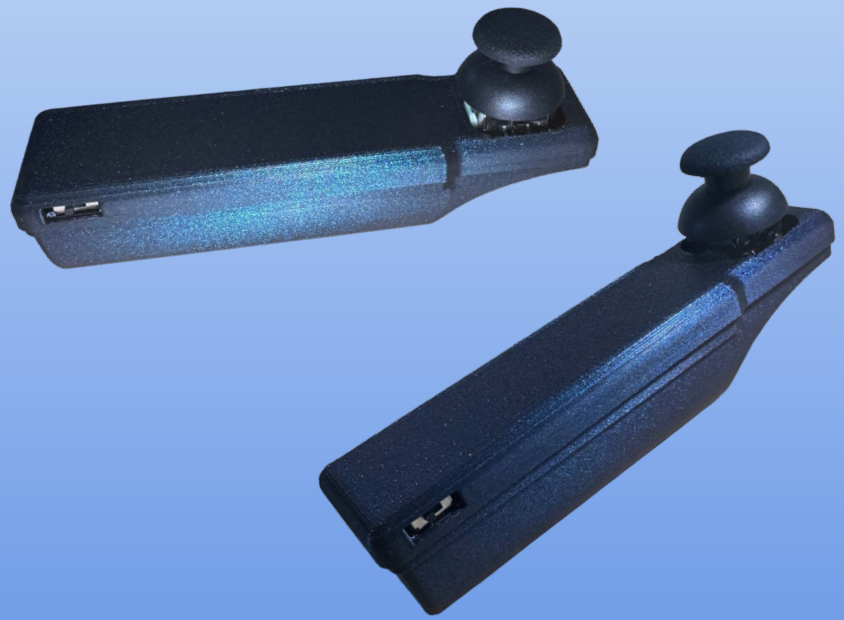
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Project: Remote Control Transmitter

Today you will Learn:

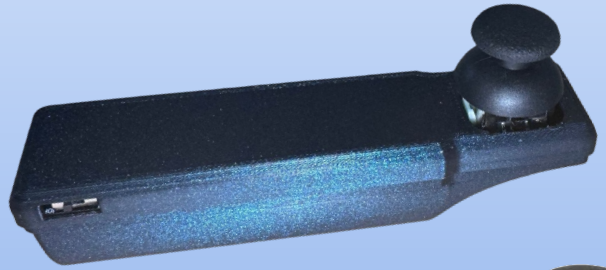
- Reading Digital Input
- Sending Data with ESPNow
- Arduino ESP32 Serial Debugging

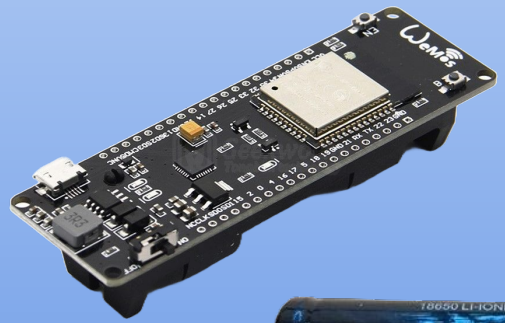


Mad Science 102: Remote Targeting

Part 2: Remote Control!

Project: Remote Control Transmitter





Remote Controller

- Uses Original ESP32 (Not C3)
- Has built-in Battery Charger for 18650!
- Connects to Joystick + Button
- Annoyingly: Has older Micro-USB

Connector is FRAGILE!! Be careful.

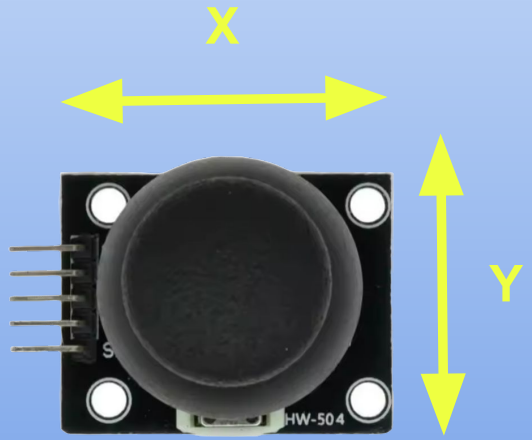
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Project: Remote Control Transmitter

Joystick:

- Has two Potentiometers to measure offset
- One for X
- One for Y
- These are read via Analog Input. We will read these next class.



Mad Science 102: Remote Targeting

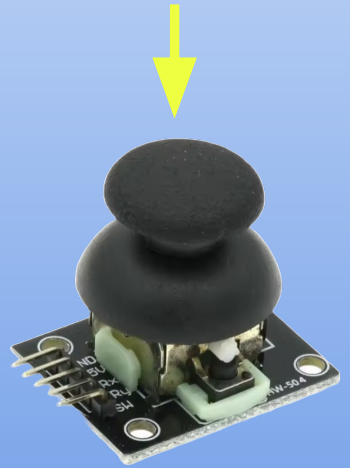
Part 2: Remote Control!

Project: Remote Control Transmitter

Joystick:

- Has a push button
- This is read via Digital Input
- Almost exactly the same as how you turned on the laser and LED from MS101
- We will read this today!

Button



Mad Science 102: Remote Targeting

Part 2: Remote Control!

Project: Remote Control Transmitter

Joystick:

Five Pins:

- Ground
- Power
- X
- Y
- Button

Mad Science 102: Remote Targeting

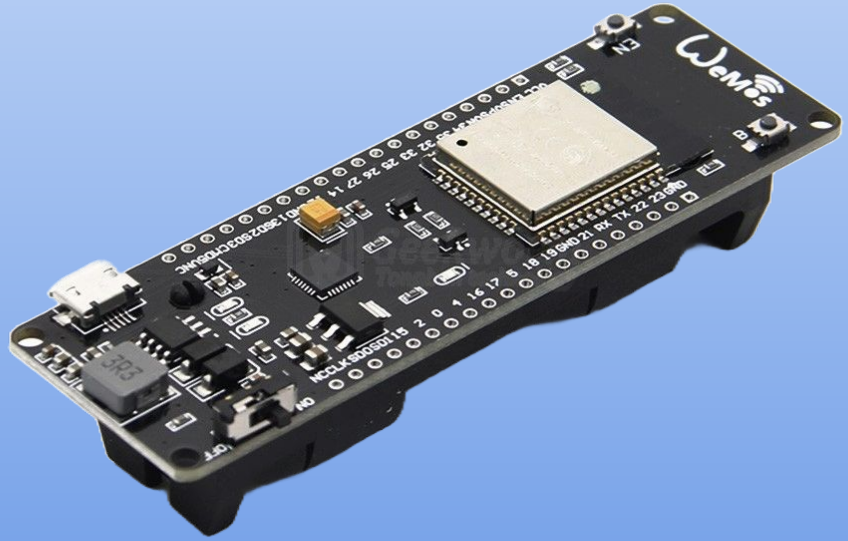
Part 2: Remote Control!

Project: Remote Control Transmitter

Microcontroller + Battery Charger

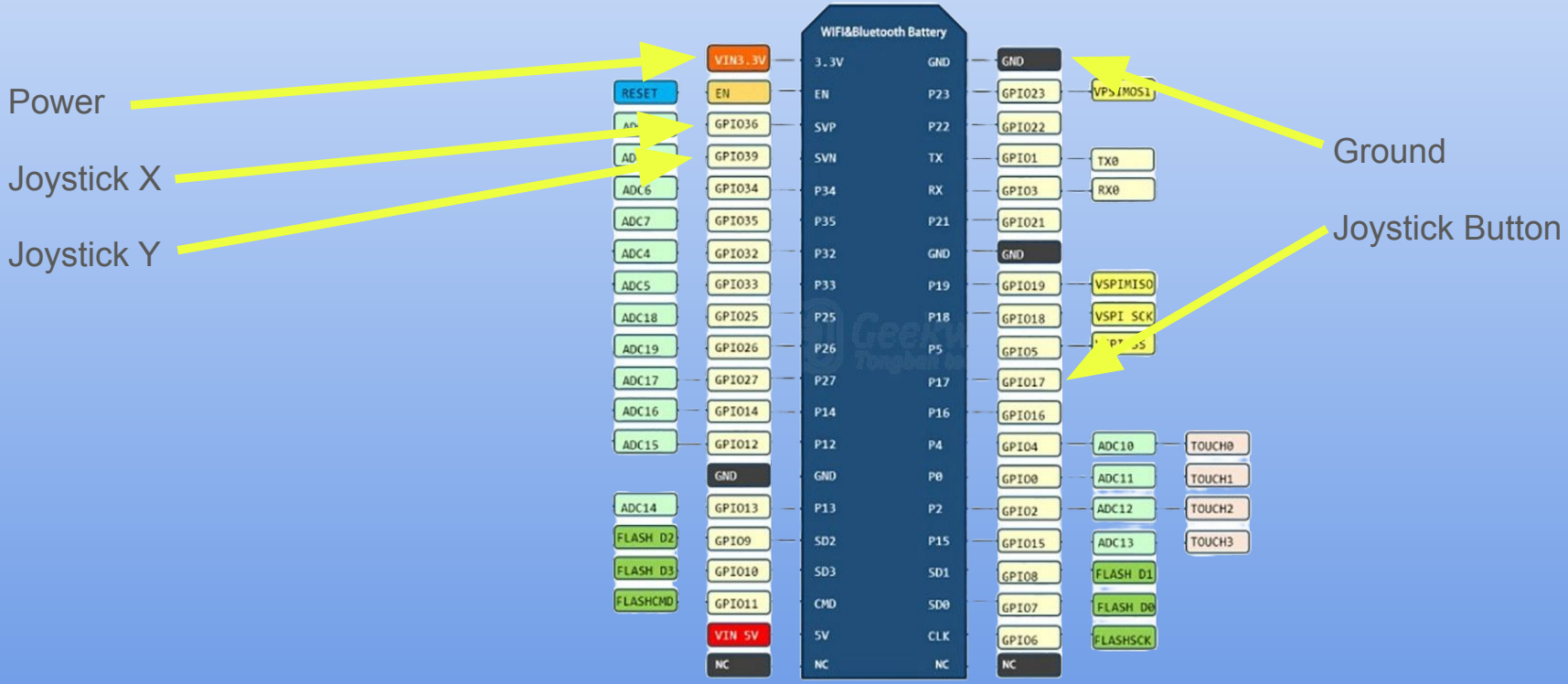
WeMos ESP32 + Battery Charger

- Pins Labeled on Board
- Same as using your other board Except:
- Must use different board profile in Arduino IDE
- If you plug in the Micro-USB, it will charge the battery!



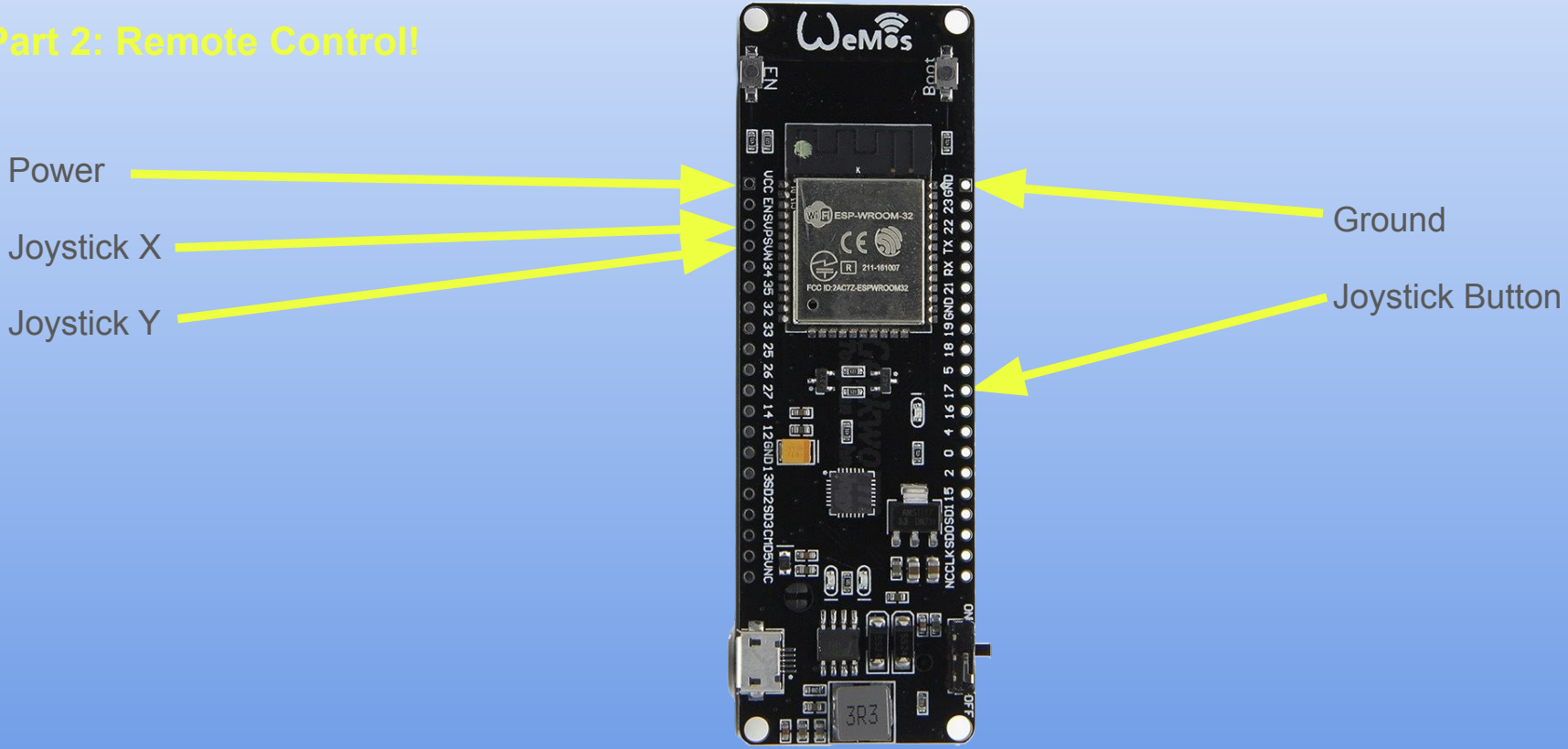
Mad Science 102: Remote Targeting

Part 2: Remote Control!



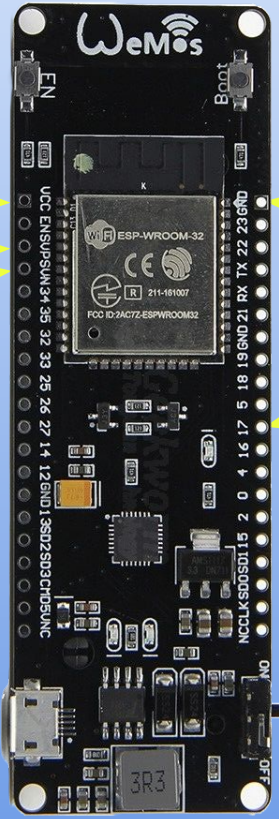
Mad Science 102: Remote Targeting

Part 2: Remote Control!



Mad Science 102: Remote Targeting

Part 2: Remote Control!



Power

Joystick X

Joystick Y

Ground

Joystick Button

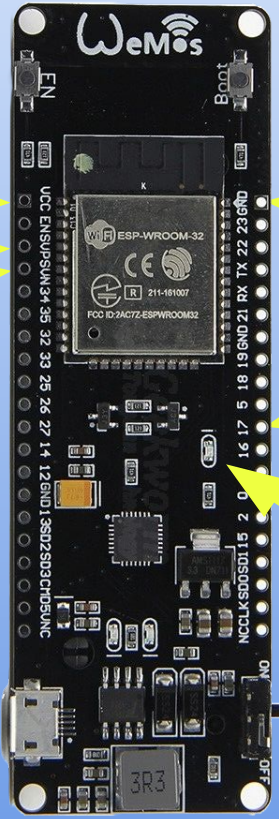
Micro USB port for charging and programming:

Very Fragile!

Power Switch: ESP32 is only on when this switch is on.

Mad Science 102: Remote Targeting

Part 2: Remote Control!



Power

Joystick X

Joystick Y

Ground

Joystick Button

LED at pin 16

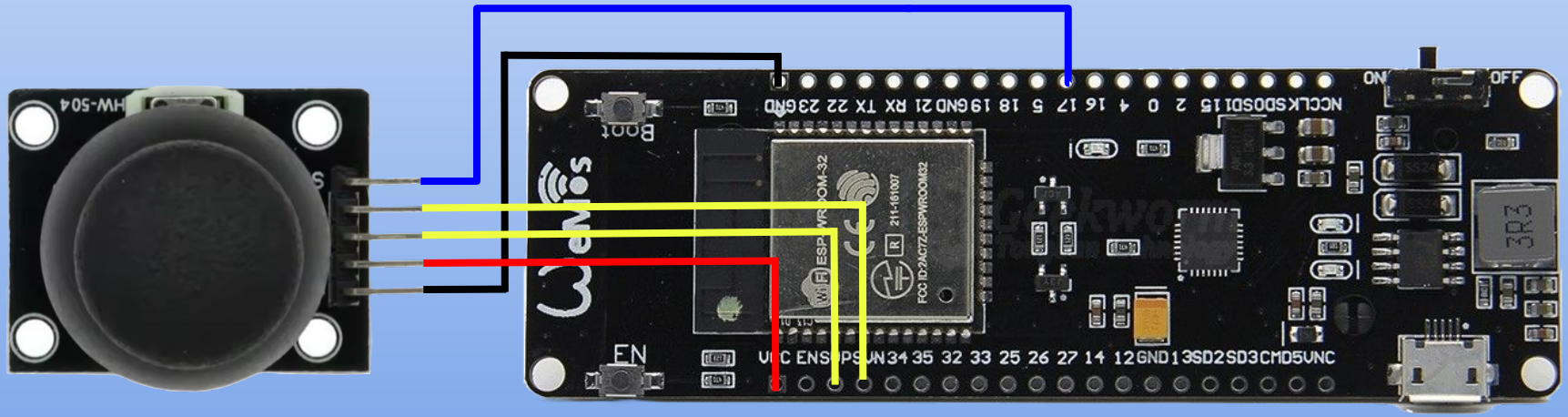
Micro USB port for charging and programming:

Very Fragile!

Power Switch: ESP32 is only on when this switch is on.

Mad Science 102: Remote Targeting

Part 2: Remote Control!



Mad Science 102: Remote Targeting

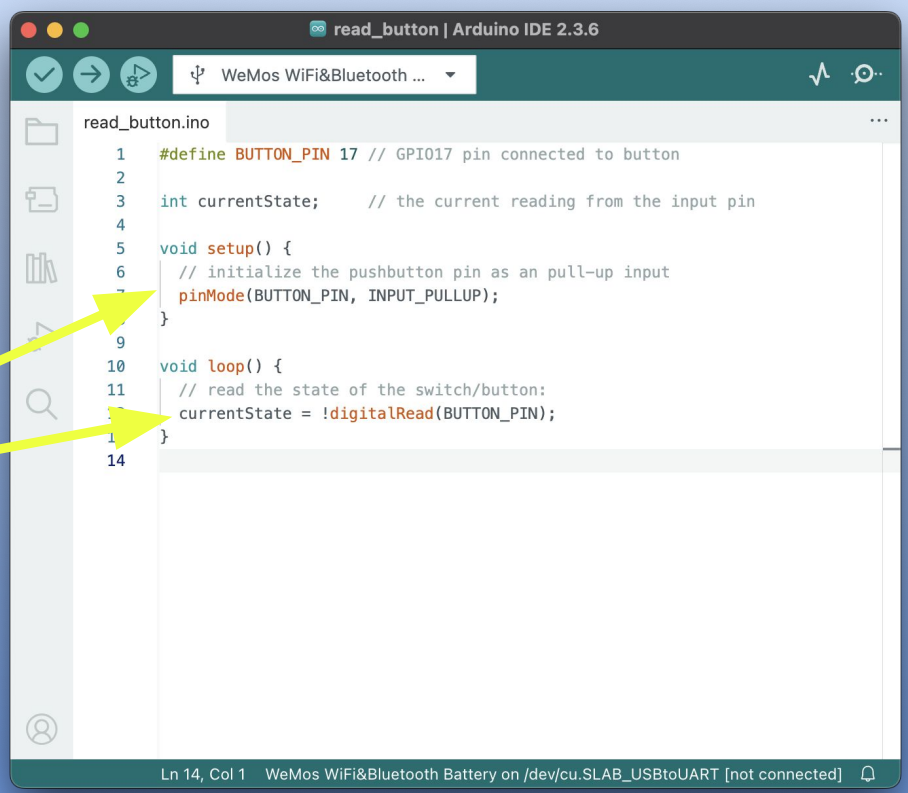
Part 2: Remote Control!

Project: Remote Control Transmitter

Reading the Button State

Similar to Writing to LED or Laser.

- Set up Pin for Input in Setup()
- Read from Pin in Loop()



Mad Science 102: Remote Targeting

Part 2: Remote Control!

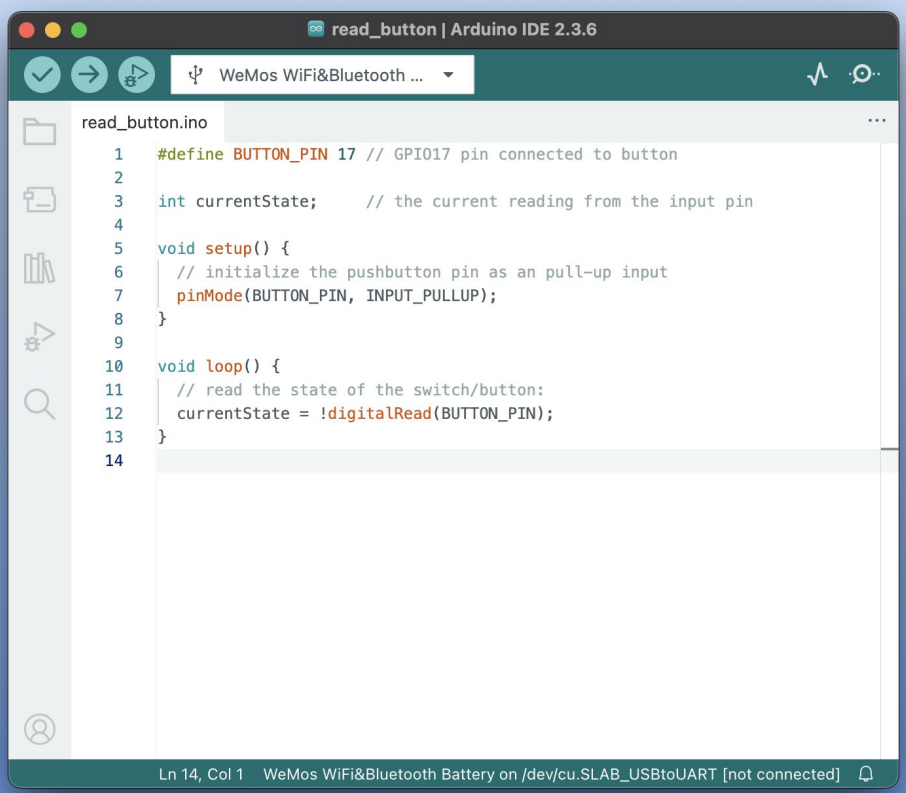
Project: Remote Control Transmitter

Reading the Button State

Similar to Writing to LED or Laser.

But how do I know if I've done it

correctly?????



Mad Science 102: Remote Targeting

Part 2: Remote Control!

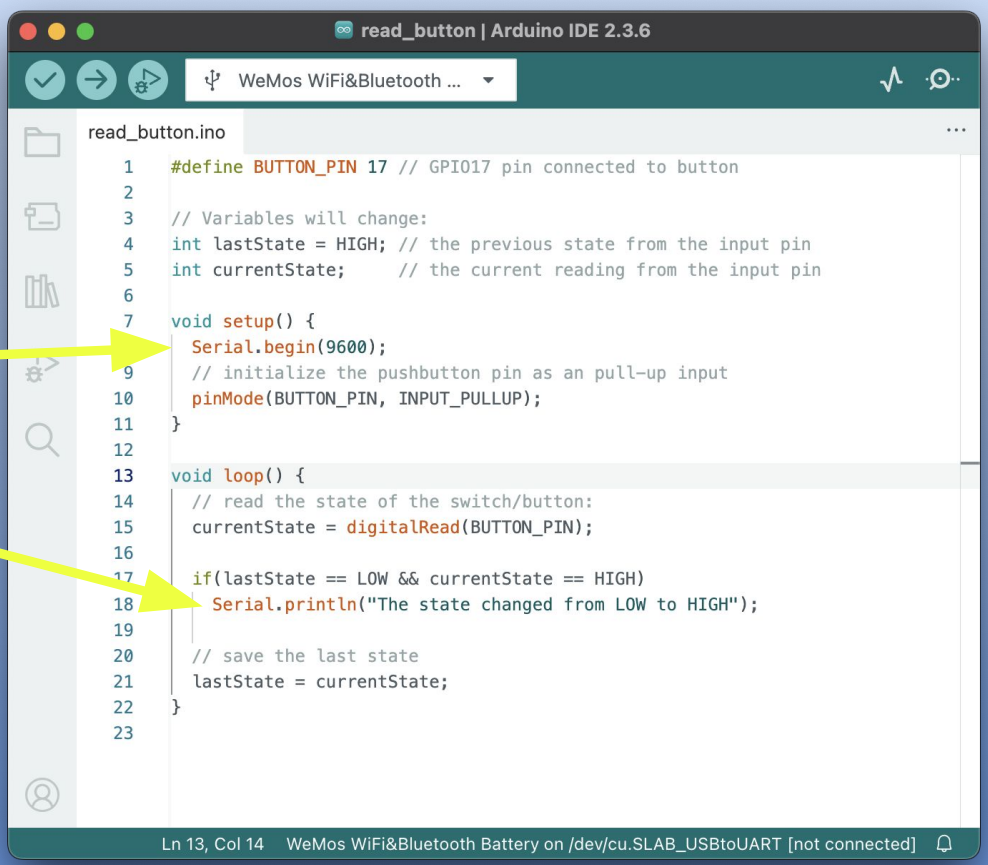
Project: Remote Control Transmitter

Serial Debugging in Arduino!!

Set up Serial Debugger

Write to Serial Debugger

When you run go to Tools->Serial Monitor and you will see your output at the bottom of the IDE



Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

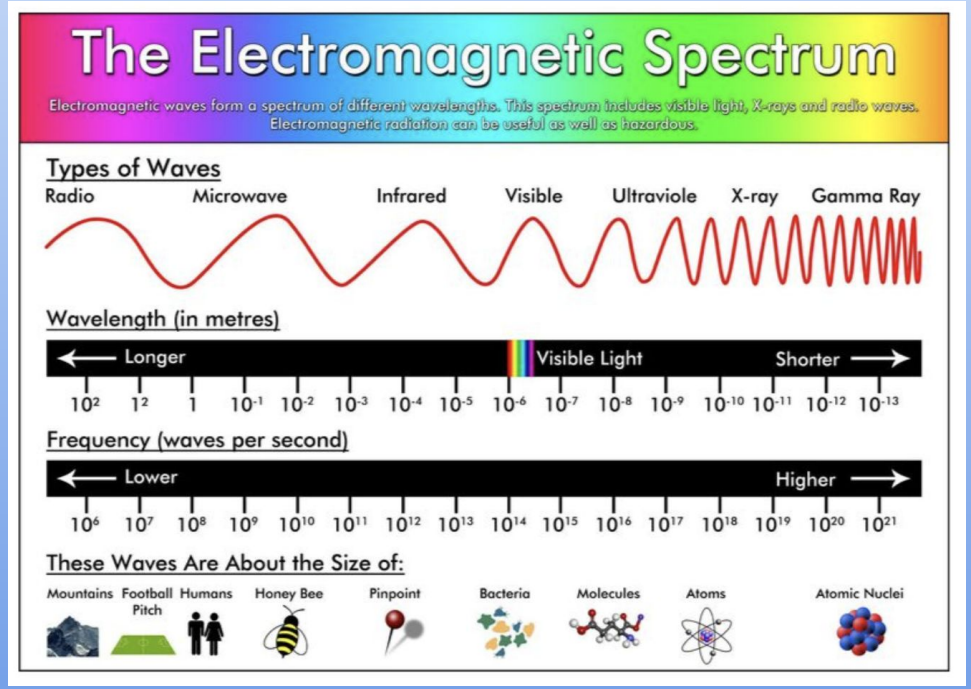
Everything from Radio to gamma rays are electromagnetic radiation

Signals are distinguished by Frequency or Wavelength (these are related and opposite)

Frequencies are measured in Hz (waves per second)

Wavelengths are measured in Meters

We care about Radio Waves!

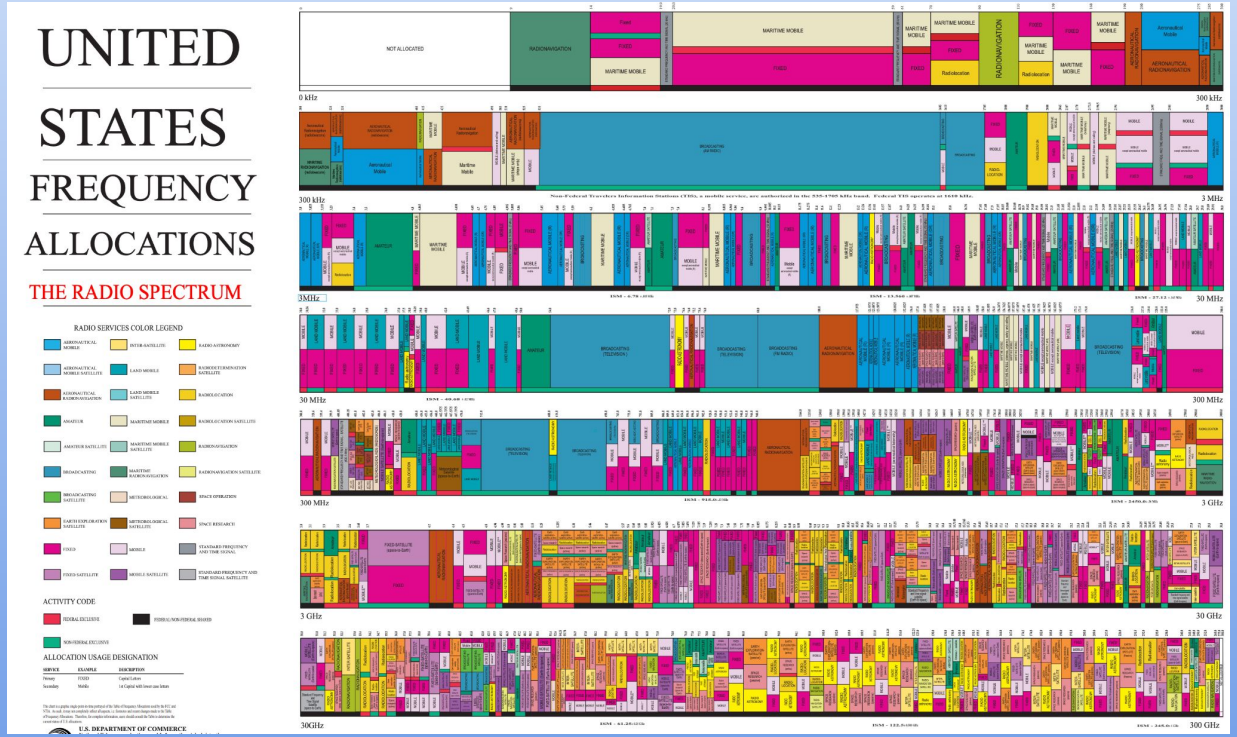


Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

But There Are a TON of Radio Waves Everywhere All the Time!!!!



Mad Science 102: Remote Targeting

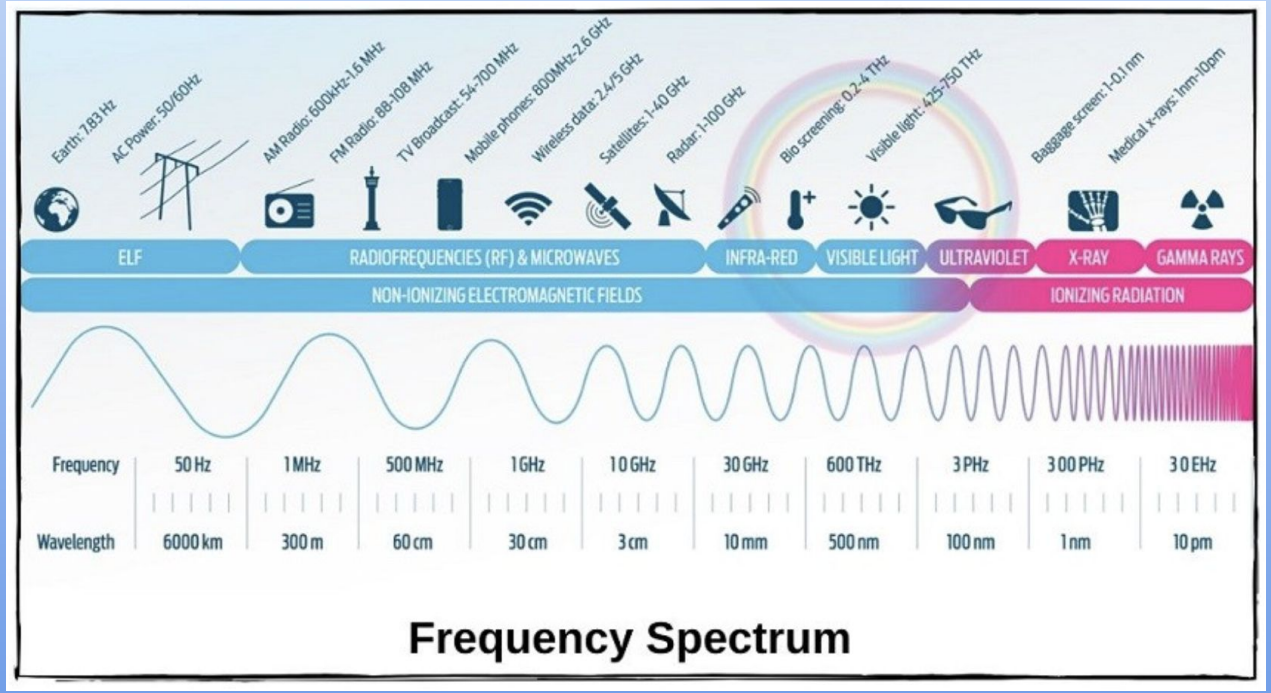
Part 2: Remote Control!

Wireless Communication

Simplified Radio Spectrum

WiFi is at 2.5 GHz or 5GHz or 6GHz (WiFi 7)

I highly recommend taking a ham radio class to learn about radio waves.



Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

WiFi: Wireless Network

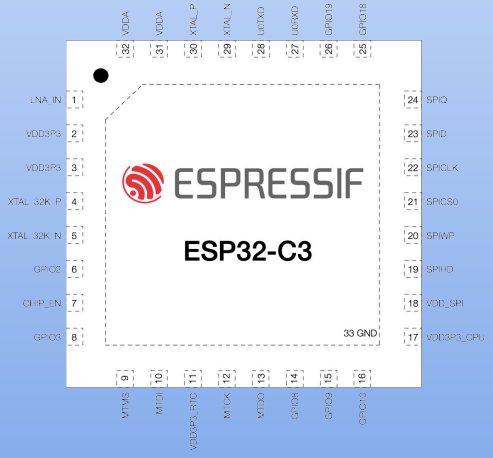
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

WiFi: Wireless Network

ESP32-C3 only uses 2.4GHz



Mad Science 102: Remote Targeting

Part 2: Remote Control!

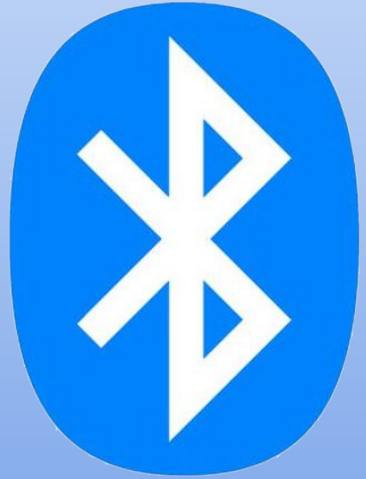
Wireless Communication

Bluetooth:

Wireless Devices

Wireless Headphones

ESP32-C3 does not have Bluetooth



Mad Science 102: Remote Targeting

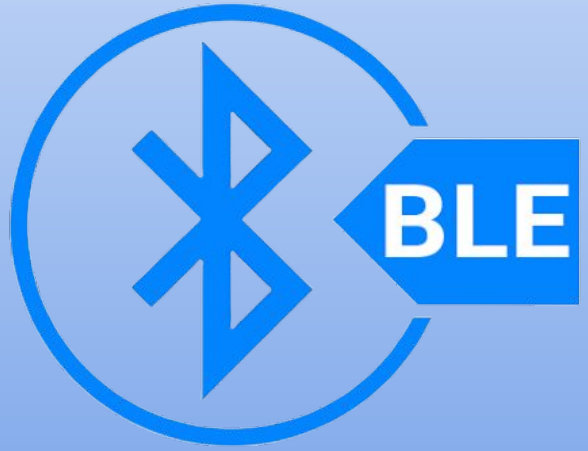
Part 2: Remote Control!

Wireless Communication

Bluetooth Low Energy (BLE)

Very easy low-bandwidth connections

ESP32-C3 has BLE!



Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

ESP Now

Uses WiFi in a special mode

Only works with Espressif (ESP32)

Mad Science 102: Remote Targeting

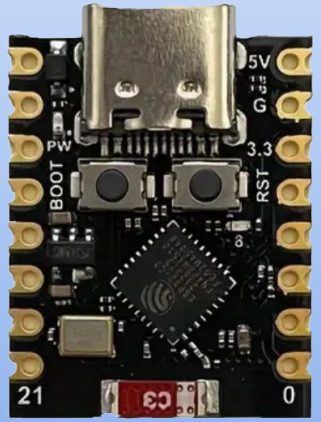
Part 2: Remote Control!

Wireless Communication

ESP NOW

On the ESP32-C3 SuperMini Board

- 2.4GHz BLE, WiFi, or ESP-Now
- Chip Antenna isn't awesome. ☹️



Chip Antenna

Mad Science 102: Remote Targeting

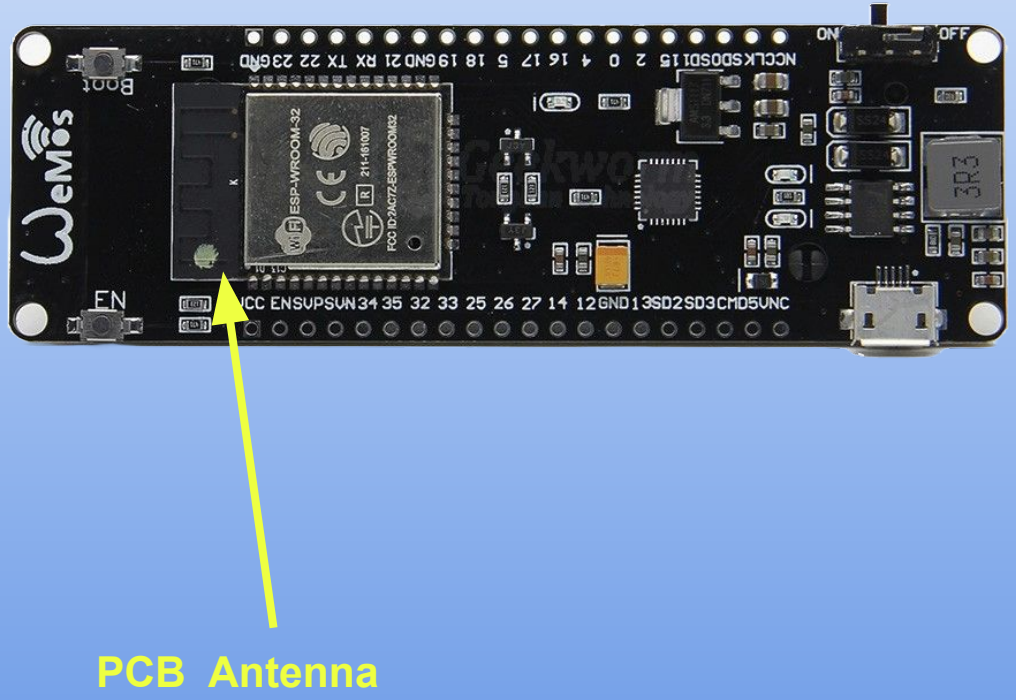
Part 2: Remote Control!

Wireless Communication

ESP NOW

On the ESP32 WeMos Board

- 2.4GHz BLE, WiFi, or ESP-Now
- PCB Antenna is pretty good



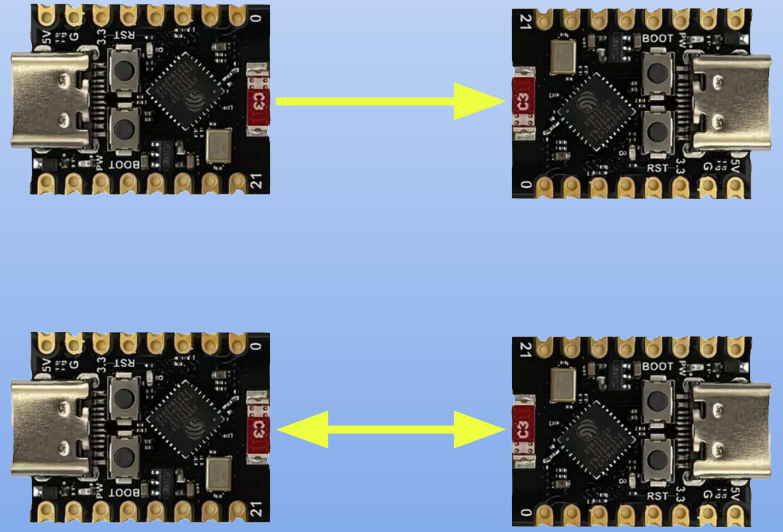
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

ESP NOW

One-Way or Two-Way



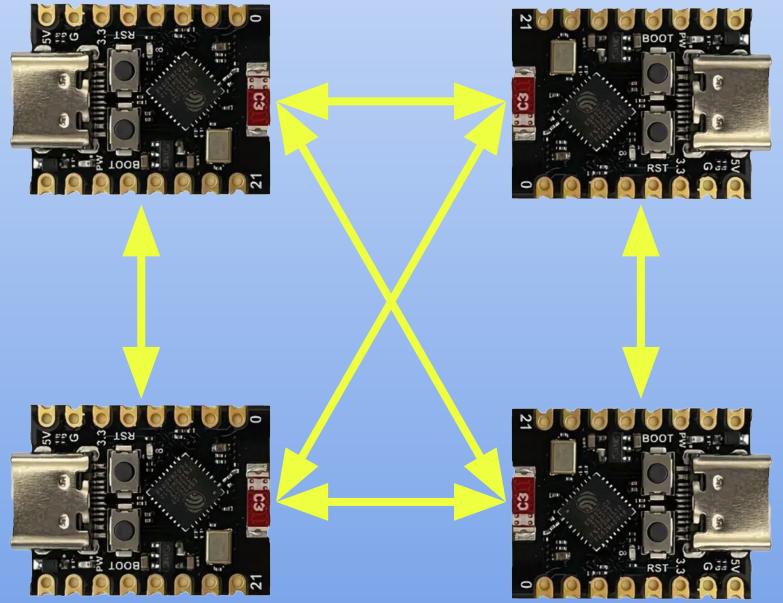
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Wireless Communication

ESP NOW

Mesh



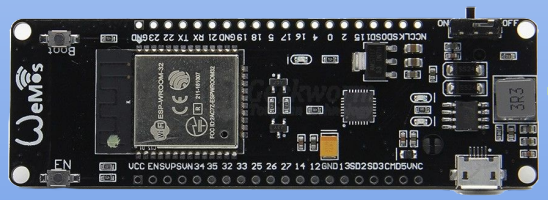
Mad Science 102: Remote Targeting

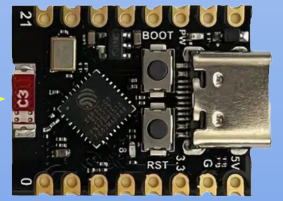
Part 2: Remote Control!

Wireless Communication

ESP NOW

We will use One-Way from Transmitter to Tower





Mad Science 102: Remote Targeting

Part 2: Remote Control with ESP NOW

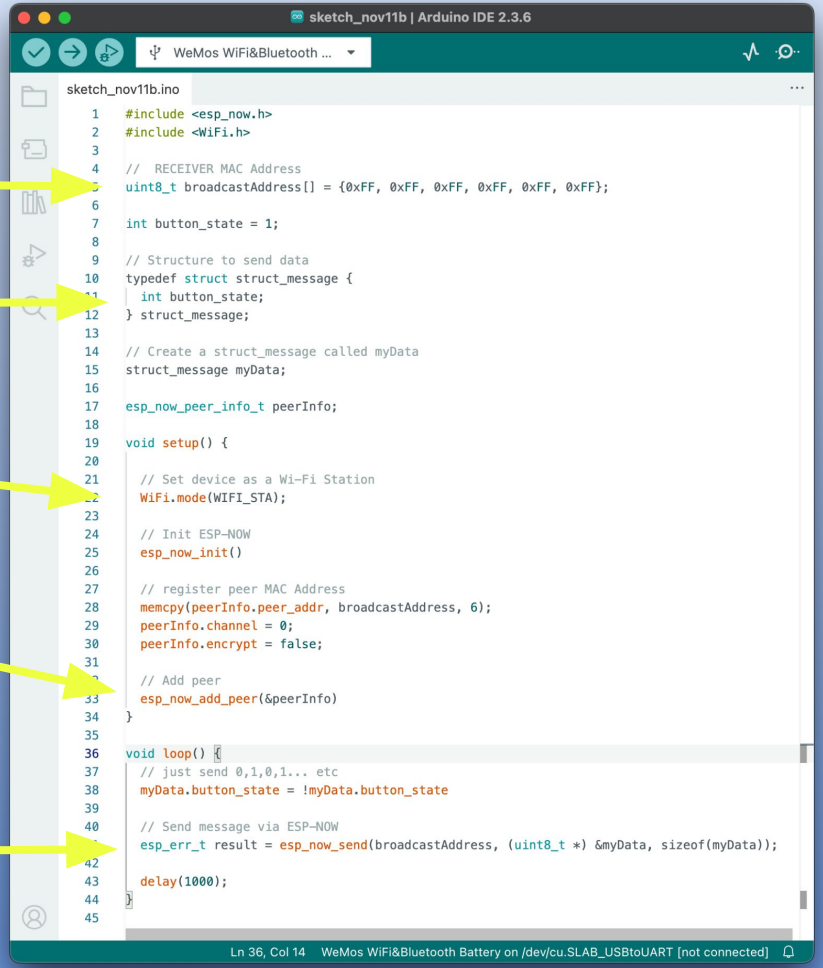
Setup the MAC address of tower

Say what data you are sending

Start WiFi

Setup and add Tower as Peer

Send Data



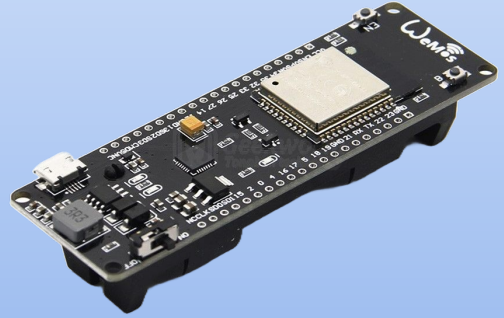
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Enough Talk, Let's Do Some Stuff!

BOM:

- 4" Red Silicone Wire
- 4" Black Silicone Wire
- 8" Yellow Silicone Wire
- 4" Blue silicone Wire
- 18650 Battery
- Joystick
- Wemos Battery Charger MCU
- Transmitter Case Bottom
- Transmitter Case Top



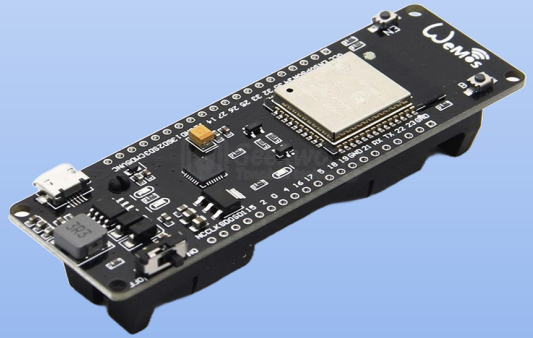
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Enough Talk, Let's Do Some Stuff!

Step 1: Prep

- Cut your yellow wire in half
- Strip and Tin all wires
- Make sure switch is off
- Make sure you don't have a battery in the WeMos

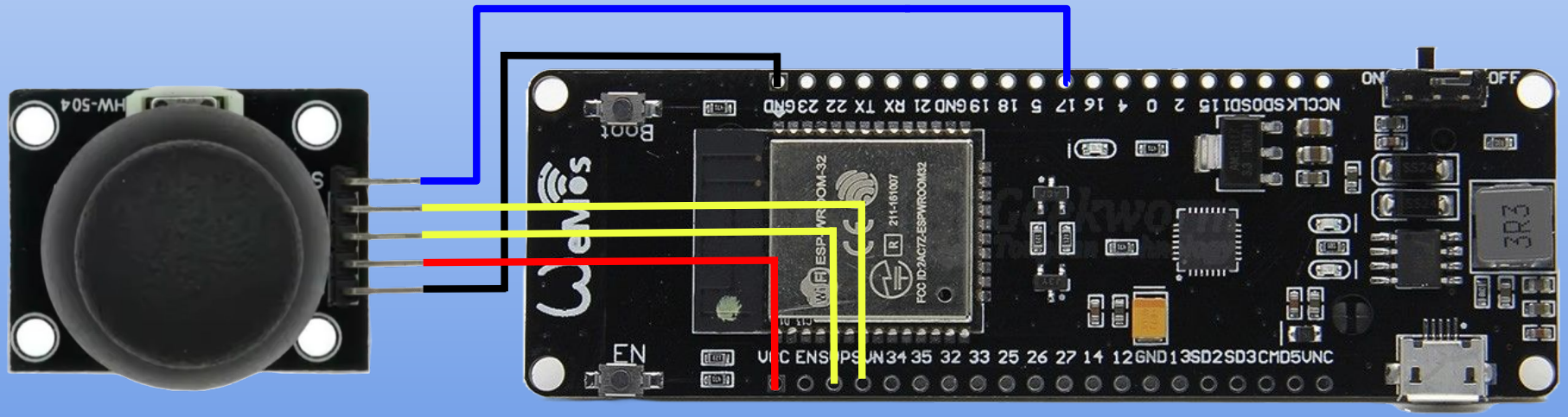


Mad Science 102: Remote Targeting

Part 2: Remote Control!

Enough Talk, Let's Do Some Stuff!

Step 2: Solder wires as indicated on wiring diagram



Mad Science 102: Remote Targeting

Part 2: Remote Control!

Enough Talk, Let's Do Some Stuff!

Step 3 Assemble the Controller

- Make sure you have the battery inserted in the correct way (+/-)
- Take off the joystick handle before putting on the top.
- Add screws if you like.
- NOTE: You may want to leave the top off for now so that you can see the LEDs

Mad Science 102: Remote Targeting

Part 2: Remote Control!

Enough Talk, Let's Do Some Stuff!

Step 4 Read the Joystick Button

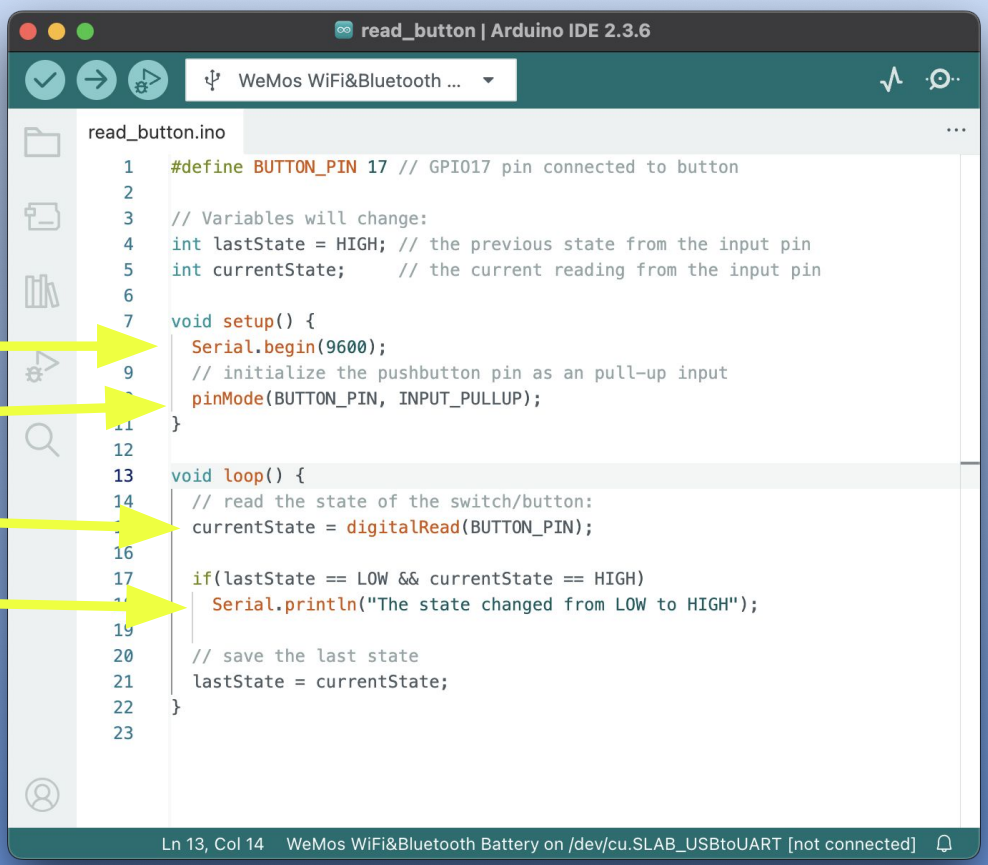
Set up Serial Debugger

Initialize your Input Pin

Read your Input Pin

Write to Serial Debugger

When you run go to Tools->Serial Monitor and you will see your output at the bottom of the IDE



Mad Science 102: Remote Targeting

Part 2: Remote Control with ESP NOW

Enough Talk, Let's Do Some Stuff!

Step 5 Sending data!

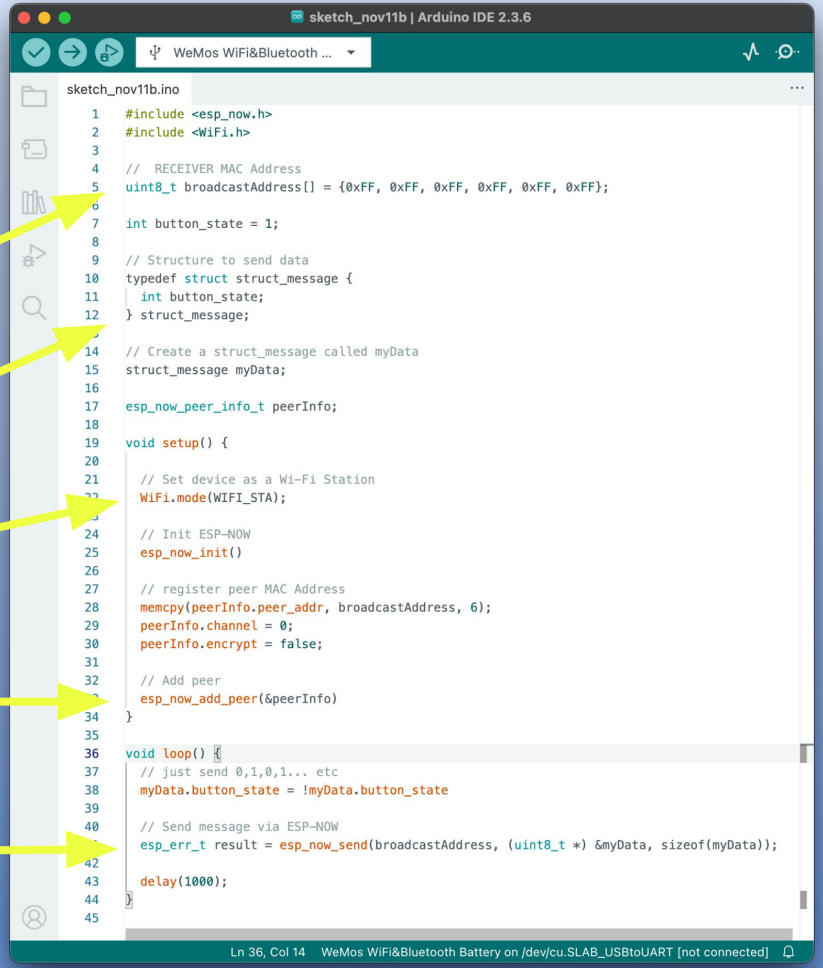
Setup the MAC address

Say what data you are sending

Start WiFi

Setup and add Tower as Peer

Send Data



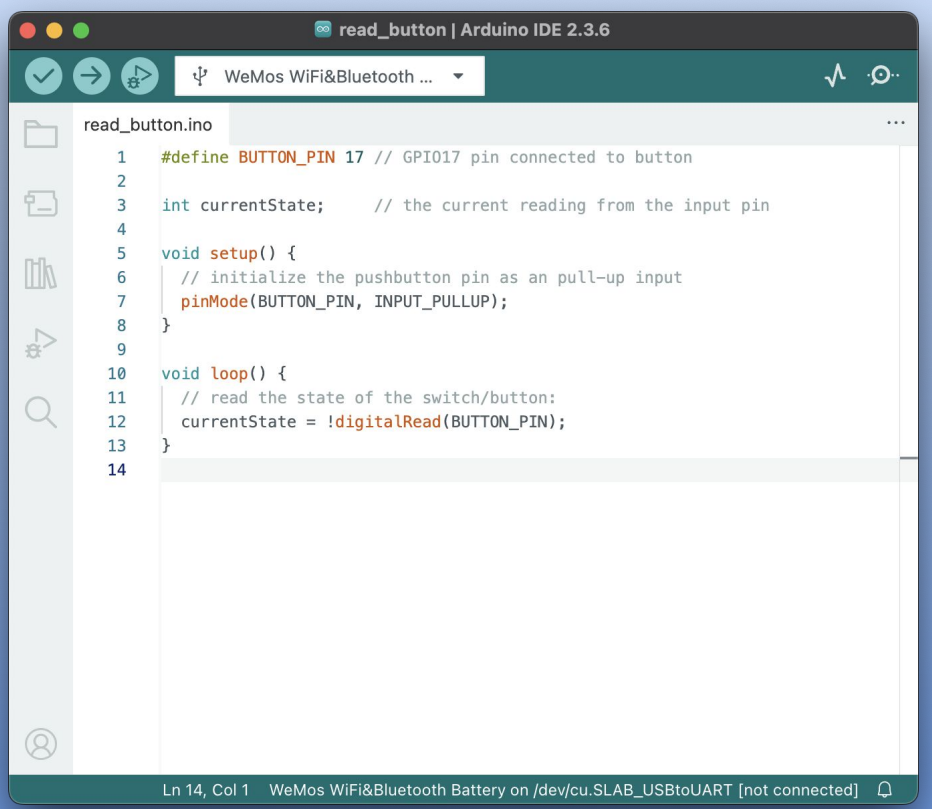
Mad Science 102: Remote Targeting

Part 2: Remote Control!

Enough Talk, Let's Do Some Stuff!

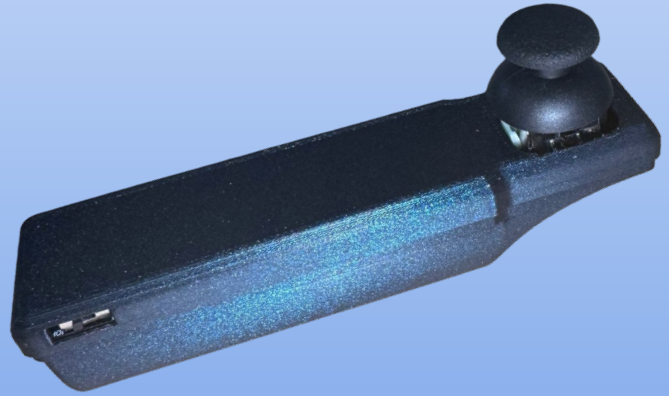
Step 6 Put it all together!

- Read the Joystick Button
- Send value out over ESP Now to my MAC address at:
  - **48:27:E2:16:CF:24**



Mad Science 102: Remote Targeting

Part 3: Laser Control!



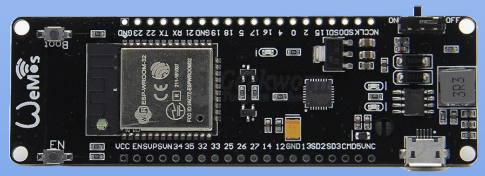
Mad Science 102: Remote Targeting

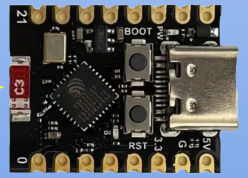
Part 3: Laser Control!

Project: Remote Control Transmitter

Today you will Learn:

- Reading Analog Input from Joysticks
- Simple C++ Data Structures
- Receiving Data with ESP-Now





Mad Science 102: Remote Targeting

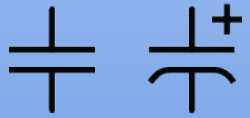
Part 3: Laser Control

Components and their symbols:

Battery (provides power)

Resistor (bleeds off power)

LED (Light Emitting Diode)



Capacitor (a thing that oscillates)

Transistor (a switch)

Switch (Also a switch!)

Potentiometer

Mad Science 102: Remote Targeting

Part 3: Laser Control

Potentiometers:

A Type of Variable Resistor

- Many Different Shapes and Sizes
- Always have a dial or a way to set resistance
- Dial resistance
- Can also Dial Voltage
    - Voltage Divider

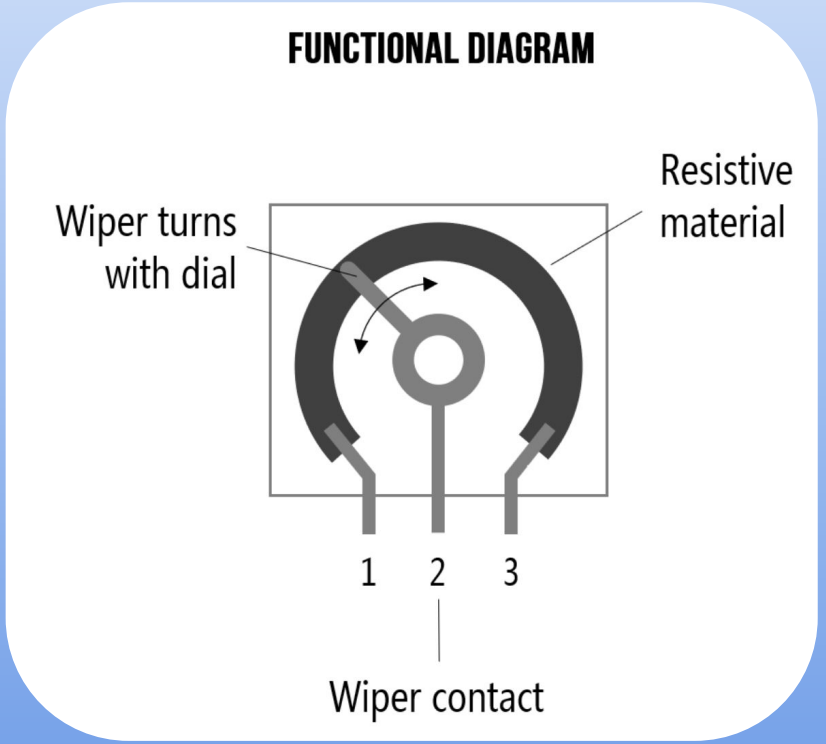
Mad Science 102: Remote Targeting

Part 3: Laser Control

Potentiometers:

A Type of Variable Resistor

- Has a Wiper and Resistive Material
- Three Leads are
  1. Ground
  2. Output V
  3. Reference V
     - +5V for us
- The higher you turn the wiper knob, the higher the voltage between 2&3



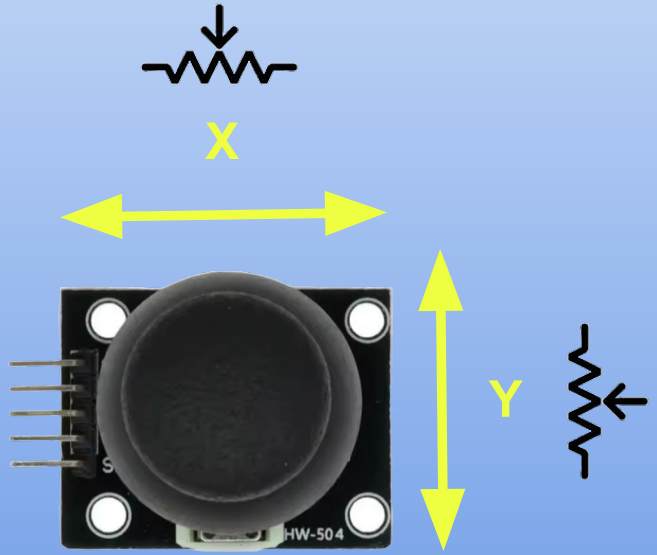
Mad Science 102: Remote Targeting

Part 3: Laser Control

Joystick:

Two Potentiometers

- X and Y are Potentiometers
- Between X or Y Lead and Ground is like measuring from Wiper and Ground
- Output is Voltage between 0 and 5V



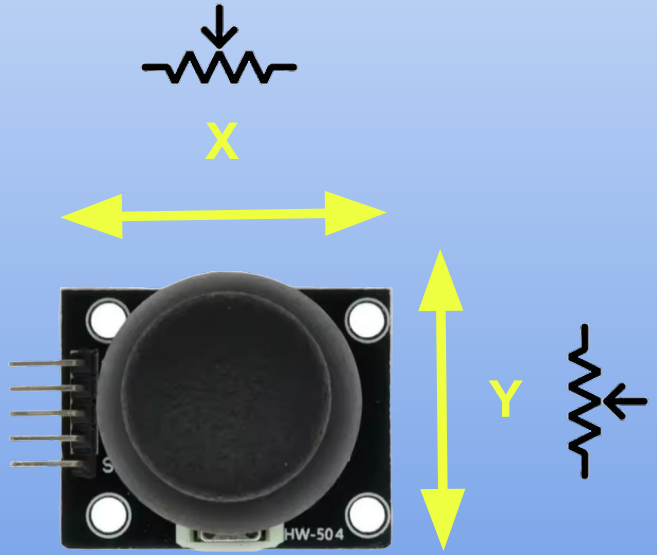
Mad Science 102: Remote Targeting

Part 3: Laser Control

Reading Analog Values:

Analog Input

- Output from Joystick Axes is Voltage between 0 and 5V
- We only know how to read Off or On type inputs like the button.
- How do we read an actual value??



Mad Science 102: Remote Targeting

Part 3: Laser Control

Reading Analog Input:

Arduino Analog Input

- Built into arduino
- Do not need to define pins like with digital
- Just read the pins directly

```
1
2   void setup() {
3     Serial.begin(115200);
4   }
5
6   void loop() {
7     // Read the analog value from the potentiometer
8     int analogValueX = analogRead(36);
9     int analogValueY = analogRead(39);
10
11    // Print the read value to the Serial Monitor
12    Serial.print("Analog Value X: ");
13    Serial.print(analogValueX);
14    Serial.print(", Analog Value X: ");
15    Serial.println(analogValueY);
16    // Add a small delay to make the output readable
17    delay(100);
18  }
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Network Addressing:

How do I know who I am talking to?

- Each piece of hardware usually has a built-in address called a MAC Address
- Stands for Media Access Control
- It is 6 bytes long
- A byte can be represented by two Hexadecimal digits. Each digit is called a “nibble”
- Often, first part of MAC identifies manufacturer

Mad Science Matrix

48:27:E2:16:CF:24

Thomas’ Demo Tower

20:6E:F1:6D:1B:14

Mad Science 102: Remote Targeting

Part 3: Laser Control

Aside: Hexadecimal

- **Binary** is 0-1
  - Two possibilities
- **Decimal** is 0-9
  - Ten possibilities
- **Hexadecimal** is 0-15
  - Sixteen possibilities
- A Byte can be represented by two Hexadecimal digits, and is 0-255
  - 256 possibilities
- Upper case / Lower case does not matter
- Literal hexadecimal digits are written with 0x at the beginning, e.g. 0x3A or 0x11

Denary	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Mad Science 102: Remote Targeting

Part 3: Laser Control

Network Addressing:

How do I find my own MAC address?

- Turn on Wifi Mode
- Print MAC via serial monitor

```
1   #include <WiFi.h>
2
3   void setup() {
4     Serial.begin(115200);
5
6     WiFi.mode(WIFI_STA);
7     delay(2000);
8     Serial.print("MAC Address: ");
9     Serial.println(WiFi.macAddress());
10  }
11
12  void loop() {
13
14  }
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Structures

Structs (Structures)

- A holder for a set of variables
- Use `typedef` to give it a name
- Instantiate by declaring like any other variable
- Reference the member variables with variable name and `.` or `->` operators

```
21  typedef struct struct_message {
22      int x;
23      int y;
24      bool button;
25  } struct_message;
26
27  // Create a struct_message called myData
28  struct_message myData;
29  int a = myData.x;
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Structures

Classes

- Classes are Structs with functions embedded in them
- Annoyingly, functions in a class are called **methods**.
- They have keywords **public** and **private** to say who can access members. You can only access public functions in code outside of the class.

```
1   class Led {
2     public:
3       // Constructor: Called when
4       // an object of the class is created
5       Led(byte pin);
6
7       // Member functions to control the LED
8       void turnOn();
9       void turnOff();
10      void toggle();
11
12    private:
13      // Private member variable to store
14      // the LED pin number
15      byte _ledPin;
16  };
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Structures

Arrays

- Arrays are just a number of other variables in a list.
- Has to be the same type of variable, but can be Classes or Structs

This array is a string of bytes

- Name is `thomasDemoMAC`
- `[]` tells it that it is an array
- `uint8_t` is a name for a byte

```
uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Structures

Arrays

- You can also declare an array with a set size without any data in it.
- The system will reserve that space for you to use later

```
uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};
uint8_t destinationAddress[6];
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Sizes

Each variable takes up a set amount of space in memory:

- `int` is 4 bytes
- `bool` is 1 byte
- `string` has one byte per character in the string +1 byte for a 0 at the end.

```
21  typedef struct struct_message {
22      int x;
23      int y;
24      bool button;
25  } struct_message;
26
27  // Create a struct_message called myData
28  struct_message myData;
29  int a = myData.x;
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Sizes

Each variable takes up a set amount of space in memory:

- `int` is 4 bytes
- `bool` is 1 byte
- `string` has one byte per character in the string +1 byte for a 0 at the end.

How Many Bytes are in the struct?

```
21  typedef struct struct_message {
22      int x;
23      int y;
24      bool button;
25  } struct_message;
26
27  // Create a struct_message called myData
28  struct_message myData;
29  int a = myData.x;
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Data Sizes

Here are two arrays

- First is a list of bytes
- Second is also a list of bytes, but it's empty

How Many Bytes are in each of these variables?

```
uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};
uint8_t destinationAddress[6];
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Moving Data

Memcpy

- You can copy the memory from one place to another with `memcpy(destination, source, length);`

```
uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};
uint8_t destinationAddress[6];
memcpy(destinationAddress, thomasDemoMAC, 6);
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Moving Data

Memcpy

- You can copy the memory from one place to another with `memcpy(destination, source, length);`
- **Important: Data flows from right to left!**

```
uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};
uint8_t destinationAddress[6];
memcpy(destinationAddress, thomasDemoMAC, 6);
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Moving Data

Memcpy

- You can copy the memory from one place to another with `memcpy(destination, source, length);`
- Important: Data flows from right to left!
- **Use `sizeof(variable);` to get size of source data**

```
uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};
uint8_t destinationAddress[6];
memcpy(destinationAddress, thomasDemoMAC, sizeof(thomasDemoMAC));
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

Extra Credit: Mapping Values for Smooth Servos

The arduino function `map()` will convert data from one range to another range.
For example, if you have input data range from 0-10, and output data range from 0-100, it will map a value from 0-10 to 0-100. so if you had 7, it would become 70.
Syntax looks like this:

map(value, fromLow, fromHigh, toLow, toHigh)

```
int output = map(7, 0, 10, 0, 100); // output here will be 70
```

You can also do it backwards from 100 to 0:

```
int output = map(7, 0, 10, 100, 0); // output here will be 30 (100-70)
```

Mad Science 102: Remote Targeting

Part 3: Laser Control

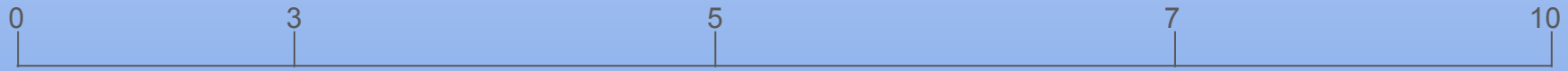
Extra Credit: Mapping Values for Smooth Servos

```
int output = map(7, 0, 10, 0, 100); // output here will be 70
```

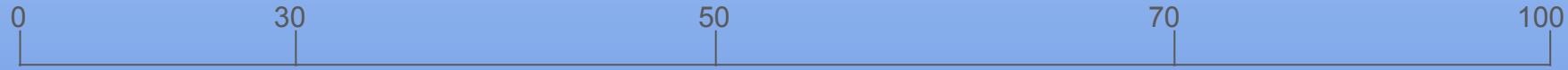
vs

```
int output = map(7, 0, 10, 100, 0); // output here will be 30 (100-70)
```

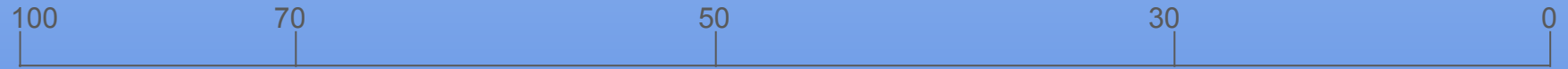
Input Range: 0-10



Output Range: 0-100



Output Range: 100-0



Mad Science 102: Remote Targeting

Part 3: Laser Control

Extra Credit: Mapping Values for Smooth Servos

Our joystick X and Y values are from 0µs to 4000µs (microseconds)
Our Servos go from 0° to 180° (degrees).
And we may want to reverse them!

```
servoX         = map(joystickX, 0, 4000, 0, 180);
servoXReversed = map(joystickX, 0, 4000, 180, 0);
```

Input Range: 0-4000

0 1000 2000 3000 4000

Output Range: 0-180

0 45 90 135 180

Reversed Output Range: 180-0

180 135 90 45 0

Mad Science 102: Remote Targeting

Part 3: Remote Control with ESP NOW

Enough Talk, Let's Do Some Stuff!

Step 1 Getting your MAC Address (Tower)

- Plug in your tower **MAKE SURE SWITCH IS OFF!**
- Write MAC to console and copy it down somewhere

```
1   #include <WiFi.h>
2
3   void setup() {
4     Serial.begin(115200);
5
6     WiFi.mode(WIFI_STA);
7     delay(2000);
8     Serial.print("MAC Address: ");
9     Serial.println(WiFi.macAddress());
10  }
11
12  void loop() {
13
14  }
```

Mad Science 102: Remote Targeting

Part 3: Remote Control with ESP NOW

Enough Talk, Let's Do Some Stuff!

Step 2 Sending data revisited! (Remote)

MAC address Array (Use your own Tower's MAC now!)

Struct with your data to send

Start WiFi

Here's your memcpy!

Setup and add Tower as Peer

Send Data

```
/*
  Thomas Marsh
*/
#include <esp_now.h>
#include <WiFi.h>

#define VRX_PIN  39 // ESP32 pin GPIO39 (ADC3) connected to VRX pin
#define VRY_PIN  36 // ESP32 pin GPIO36 (ADC0) connected to VRY pin
#define SW_PIN  17
#define LED_PIN  16

uint8_t thomasDemoMAC[] = {0x20, 0x6e, 0xf1, 0x6d, 0x1b, 0x14};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
  int x;
  int y;
  bool button;
} struct_message;

// Create a struct_message called myData
struct_message myData;

esp_now_peer_info_t peerInfo;
bool bSentCenter = false;

void setup() {
  // Init Serial Monitor
  Serial.begin(115200);
  // define LED pin
  pinMode(LED_PIN, OUTPUT);
  // define button for joystick switch
  pinMode(SW_PIN, INPUT_PULLUP);  // pullup because this is default high
                                  // pushing the button makes it go low
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  // Init ESP-NOW
  esp_now_init();
  // Register peer
  memcpy(peerInfo.peer_addr, thomasDemoMAC, sizeof(thomasDemoMAC));
  // Add peer
  esp_now_add_peer(&peerInfo);
}

void loop() {
  // get values from hardware
  // we are using !(not) because by default when pressed, it goes low.
  myData.button = !digitalRead(SW_PIN);
  myData.x = analogRead(VRX_PIN);
  myData.y = analogRead(VRY_PIN);

  esp_err_t result = esp_now_send(thomasDemoMAC, (uint8_t *) &myData, sizeof(myData));
  delay(10);
}
```

Mad Science 102: Remote Targeting

Part 3: Remote Control with ESP NOW

Enough Talk, Let's Do Some Stuff!

Step 3 Receiving a Button Click to turn on or off the laser with ESP Now (Tower)

MAKE SURE SWITCH IS OFF!

Struct with your data to receive

This is a callback function, called when you receive data. Everything important happens here!

Set Servo Position directly from data (Not Ideal but works)

```
#include <esp_now.h>
#include <esp_wifi.h>
#include <WiFi.h>
#include <ESP32Servo.h>
#include <math.h>
// define pins for digital io
#define SERVO_Y_SIGNAL_PIN      2 // servoY's signal pin is at pin #1
#define SERVO_X_SIGNAL_PIN      1 // servoX's signal pin is at #2
#define LASER_POWER_PIN         3 // laser power is at pin #3

// variables
Servo servoX;  // create servo objects to control servos
Servo servoY;  // create servo objects to control servos

// Must match the sender structure
typedef struct struct_message {
    int x;
    int y;
    bool button;
} struct_message;
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  servoX.writeMicroseconds(myData.x);
  servoY.writeMicroseconds(myData.y);
}

void setup() {
  Serial.begin(9600);
  Serial.println("Remote Control Laser Tower Starting up...");
  servoX.attach(SERVO_X_SIGNAL_PIN, 0, 4000);
  servoY.attach(SERVO_Y_SIGNAL_PIN, 0, 4000);
  pinMode(LASER_POWER_PIN, OUTPUT);
  digitalWrite(LASER_POWER_PIN, HIGH);  // start it on
  // Init ESP-NOW
  WiFi.mode(WIFI_STA);
  esp_now_init();
  esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
}

void loop() {
}
```

Mad Science 102: Remote Targeting

Part 3: Remote Control with ESP NOW

Enough Talk, Let's Do Some Stuff!

Extra Credit - Use `map()` for smoother servos (Tower)

Map your joystick microseconds PWM signal to degrees. Also reverse X channel by swapping 0 and 180

Set Servo Position from remapped data using `write()` instead of `writeMicroseconds()`

```
// callback function that will be executed
// when data is received
void OnDataRecv(const uint8_t * mac,
                const uint8_t *incomingData,
                int len) {

  memcpy(&myData, incomingData, sizeof(myData));
  int x = map(myData.x, 0, 4000, 180, 0);
  int y = map(myData.y, 0, 4000, 0, 180);
  servoX.write(x);
  servoY.write(y);
}
```

Mad Science 102: Remote Targeting

Part 3: Remote Control with ESP NOW

Enough Talk, Let's Do Some Stuff!

Extra Extra Credit: toggle the laser with the button (Tower)

Add a toggle to set button state

Toggle if the button has been pressed. If toggling on, set power to laser `HIGH`, else `LOW`

You will notice that this doesn't always work. This is because the button needs to be **debounced.** That means that we need to slow down when we receive this signal.

```
bool buttonState = true;
// callback function that will be executed when data is
received
void OnDataRecv(const uint8_t * mac, const uint8_t
*incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  int x = map(myData.x, 0, 4000, 180, 0);
  int y = map(myData.y, 0, 4000, 0, 180);
  if (myData.button){
    Serial.println("button click detected");
    buttonState = !buttonState;
    if (buttonState){
      digitalWrite(LASER_POWER_PIN, HIGH);
      Serial.println("Turning laser on");
    } else {
      digitalWrite(LASER_POWER_PIN, LOW);
      Serial.println("Turning laser off");
    }

  }
  servoX.write(x);
  servoY.write(y);
}
```

Mad Science 102: Remote Targeting

Links

[ESP32-C3 Datasheet]

[Arduino IDE]

[Adafruit]

[Sparkfun]

[Link to ESP32-C3 Super Mini Board]

[Unicornmafia's Mad Science github]

[ESP Now Tutorial]

[Potentiometers]

[Reading Joysticks from ESP32]

[VCP Drivers for Older ESP32]

[WeMos ESP32 Battery Board]