

# Arduino Programming Fundamentals

Week 4: Microcontroller Programming

# Programming Basics

- Languages are made up of several fundamental elements like verbs, nouns, adjectives, etc.
- Programming languages are the same way, except with:
  - Data types
  - Variables
  - Basic operations
  - Conditional statements
  - Loops
  - Functions

# Data types

- Tells of the type of data

Data Type	Example
Int	333
Float	0.003
Long	3333333333333
Char	K
String	Hello, World!
Bool	TRUE, FALSE

# Variables

- Names that you give the microcontroller to store values in
- Variables must be declared before they are used
- Variables can be reassigned many times, but only need to be declared once
- Variables should have names that describe their content
- You need to declare the data type before the variable name



2 yr



6 yr



20 yr

**Int CAT = 2**

**Int DOG = 6**

**Int CRAB = 20**

**String CAT\_SOUND = "MEOW"**

**String DOG\_SOUND = "WOOF"**

# Basic Operations

- Operations tells the microcontroller to perform some mathematical, relational, or logical operation
- Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder of Division)
++	[plus] +1 (ex: x = 1; x++; now x=2)
--	[minus] -1 (ex: x = 1; x--; now x=0)
+=	Increment by some # (x+=5 → x=x+5)
-=	Decrement by some # (x-=5 → x=x-5)

# Basic Operations

- Relational Operators (useful for conditional statements!)

Operator	Meaning
==	Is equals to
!=	Is not equals to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

= (1 equals sign) is an assignment operator. It assigns values to variables

# Basic Operations

- Logical Operators (useful for conditional statements!)

Operator	Meaning
&&	AND
	OR
!	NOT

- Helps make decisions.

# Conditional Statements

- Let you take actions based on if a condition is met or not
- You can also have nested conditional statements
- Pseudo code example:

if (button == ON)

    turn LED on

else

    turn LED off

Else is all the other conditions that aren't mentioned. In this case it is button == OFF



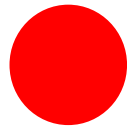
# Conditional Statements

- You can have multiple 'if' checking statements, with else if!

Pseudocode example:

if (button1 == ON)

    make led red



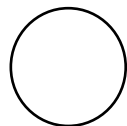
else if (button2 == ON)

    make led blue



else

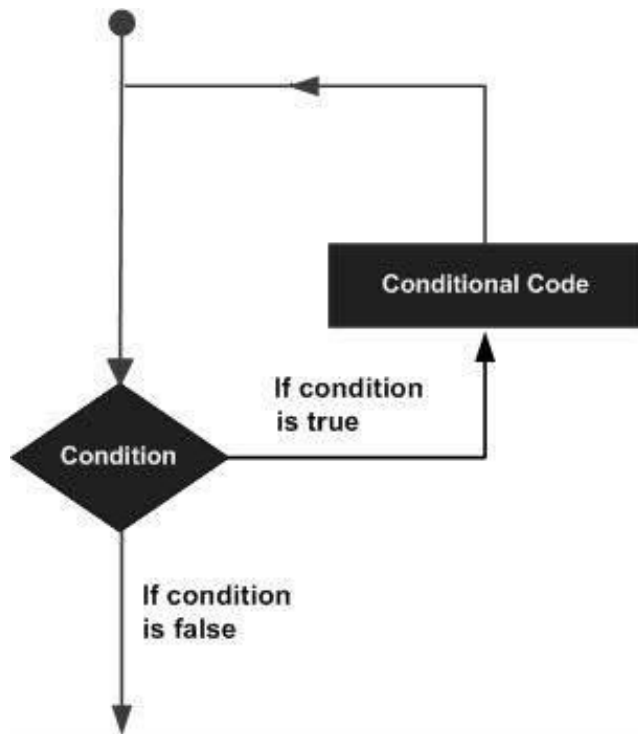
    turn led off



# Loops

- Loops are useful for executing lines of code multiple times
- Say I want to add the number 1-9 to 10.
- Hard coding it:
  - 10+1
  - 10+2
  - 10+3
  - ... boring
- With a loop
  - For number = 1 thru 9
  - 10 + number

# For loop



```
for (int x=1; x<10 ;x++){  
    Serial.println(x);  
}
```

For ( my iterative variable; my condition; go to the next case) {

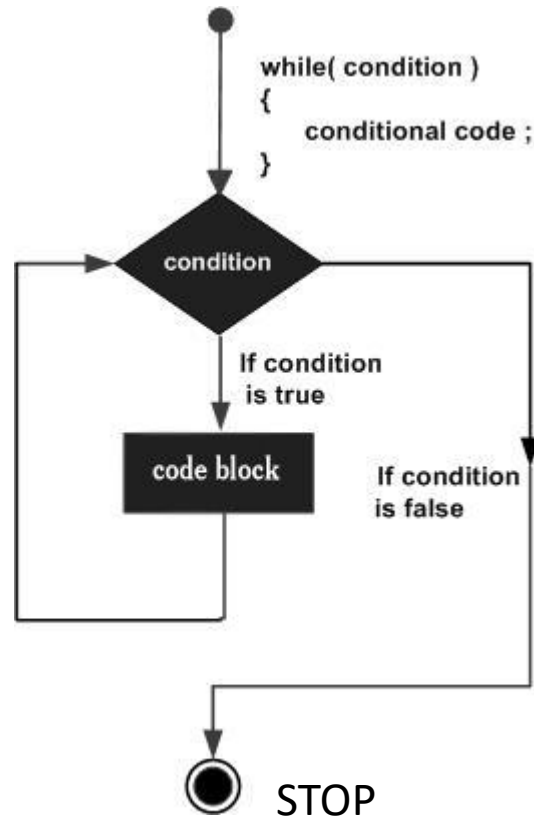
DO SOMETHING

}

OUTPUT

1  
2  
3  
4  
5  
6  
7  
8  
9

# While loop

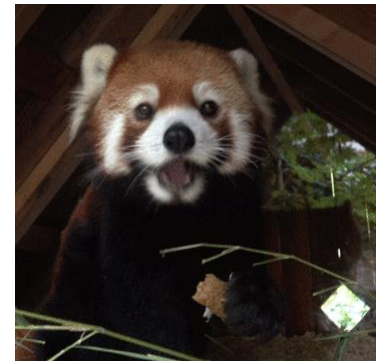


```
int x = 0;
while(x<5){
    Serial.println("RED PANDAS ARE THE BEST");
    x++;
}
```

While (condition){  
DO THIS UNTIL THE  
CONDITION IS NO LONGER TRUE  
}

OUTPUT

RED PANDAS ARE THE BEST  
RED PANDAS ARE THE BEST  
RED PANDAS ARE THE BEST  
RED PANDAS ARE THE BEST  
RED PANDAS ARE THE BEST



# Functions

- A block of reusable code
- Allows for non-redundant code

```
1  int slope = 2;
2  int time = 3;
3  int intercept = 1;
4  int value;
5
6  void setup() {
7      // put your setup code here, to run once:
8      Serial.begin(9600);
9      value = slopeCalc(slope,time,intercept);
10     Serial.println(value);
11 }
12
13 void loop() {
14     // put your main code here, to run repeatedly:
15 }
16
17 int slopeCalc(int m,int x, int b){
18     int y;
19     y = m*x + b;
20     return y;
21 }
```



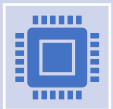
# What is Arduino?



Open source platform of hardware and software



Arduino UNO → accessible microcontroller



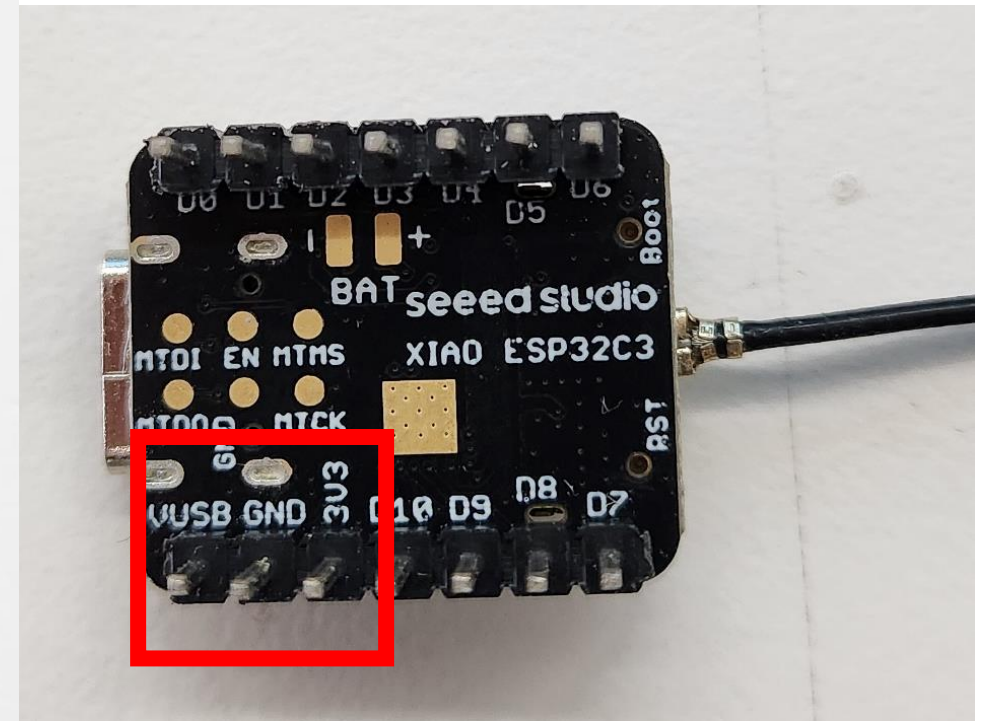
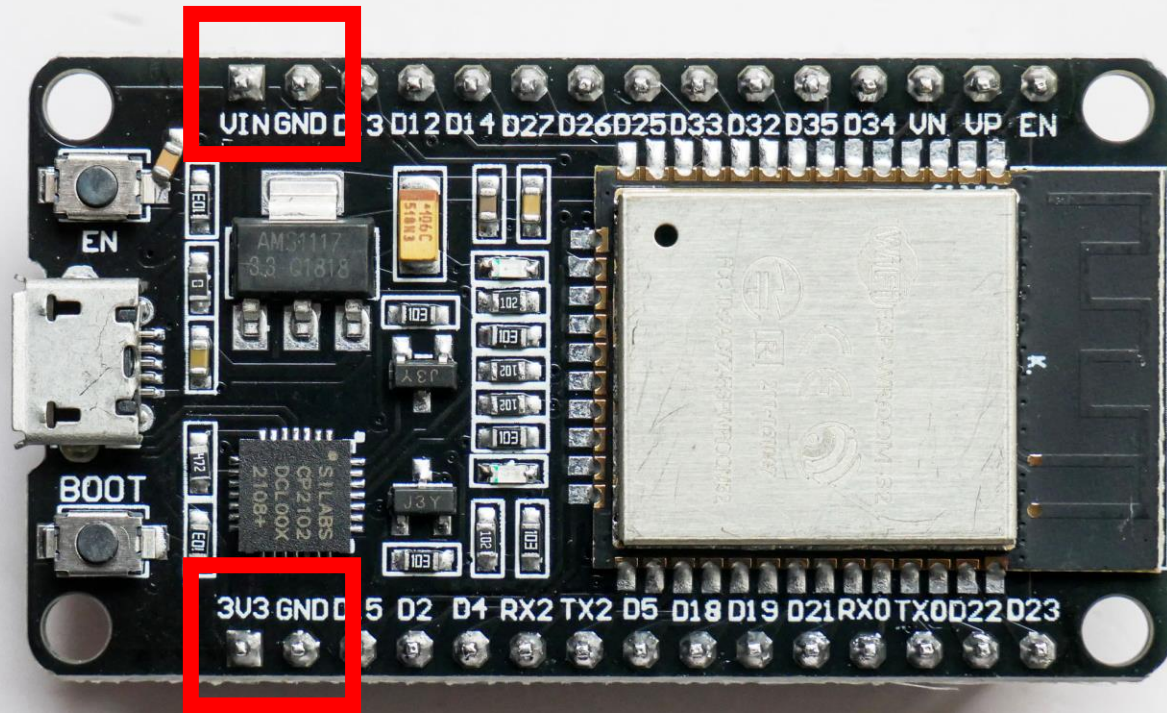
Microcontroller → like a mini-computer that can take inputs, perform outputs, store a bit of data



Uses a modified version of C/C++

## Power

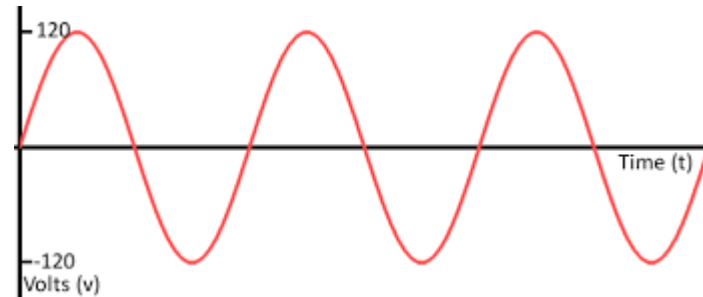
Get into the habit of color-coding voltage as Red and GND as Black. Will help you debug circuits.



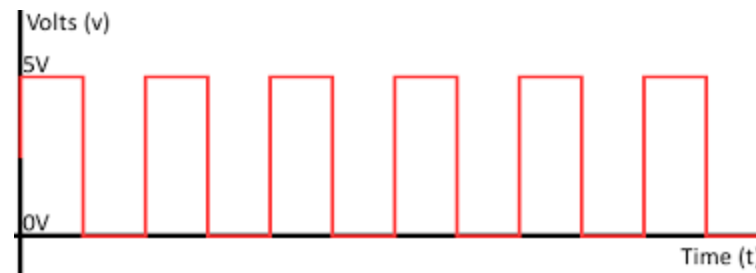


# Analog vs. Digital Signals

- Analog → data that continuously and infinitely varies over time
- Ex: radio waves, sound waves.



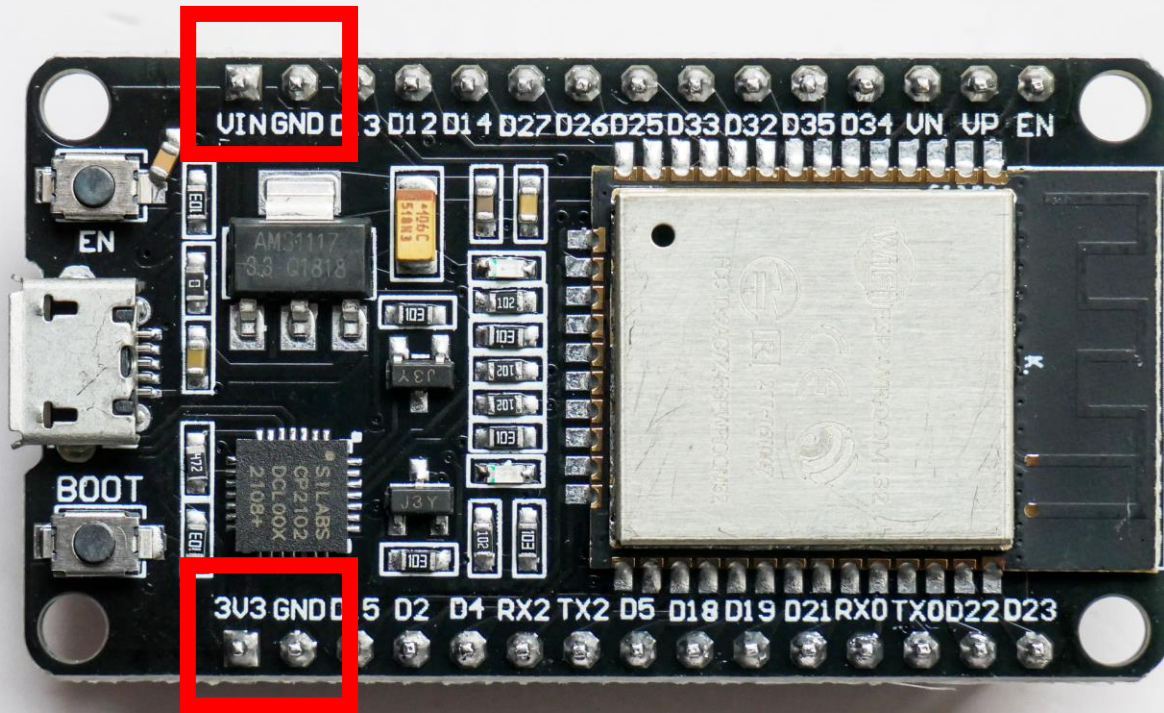
- Digital → data that has discrete values, can only take one value from a finite set of possible values at a given time
- Ex: Binary signal

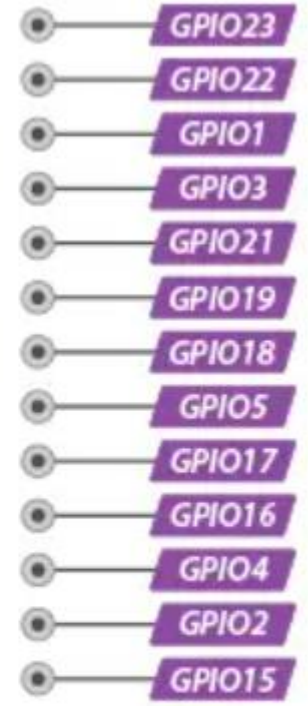
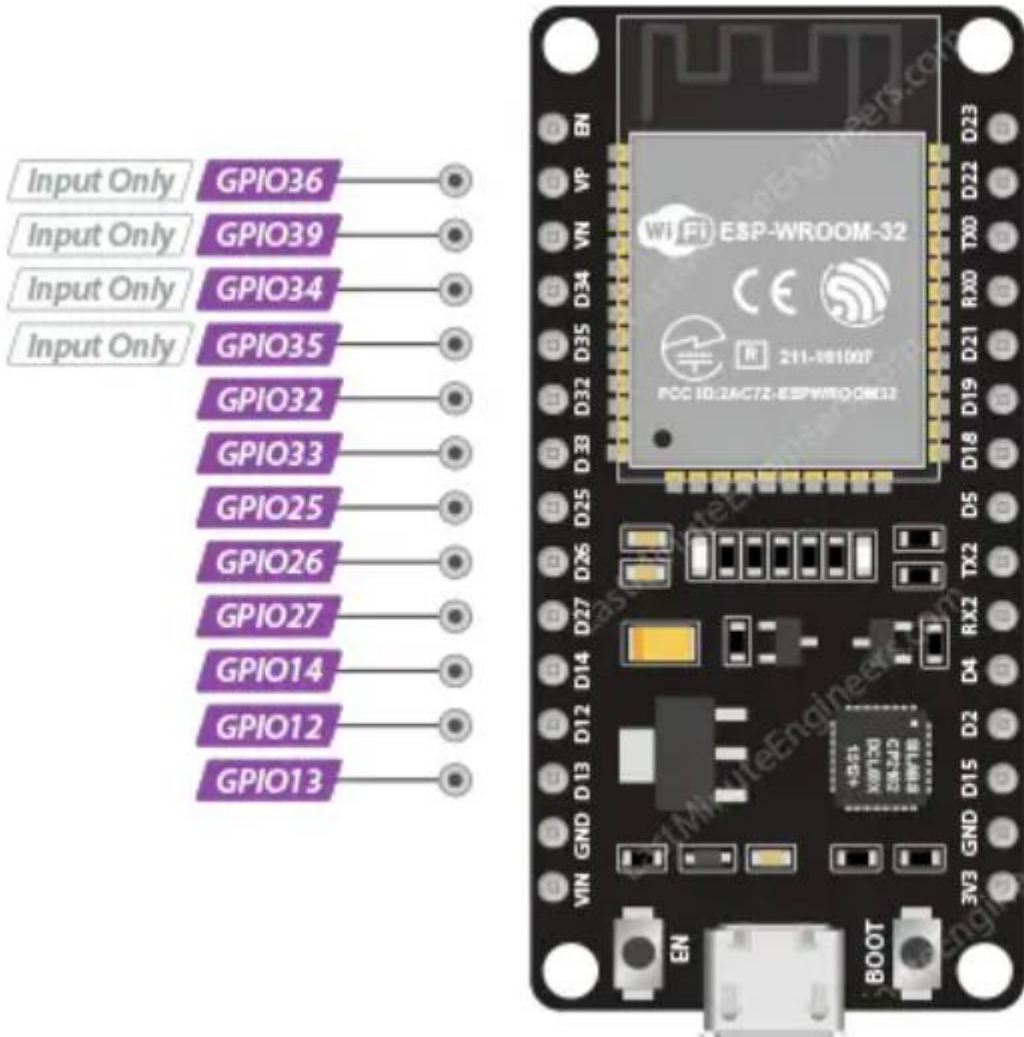


## Power

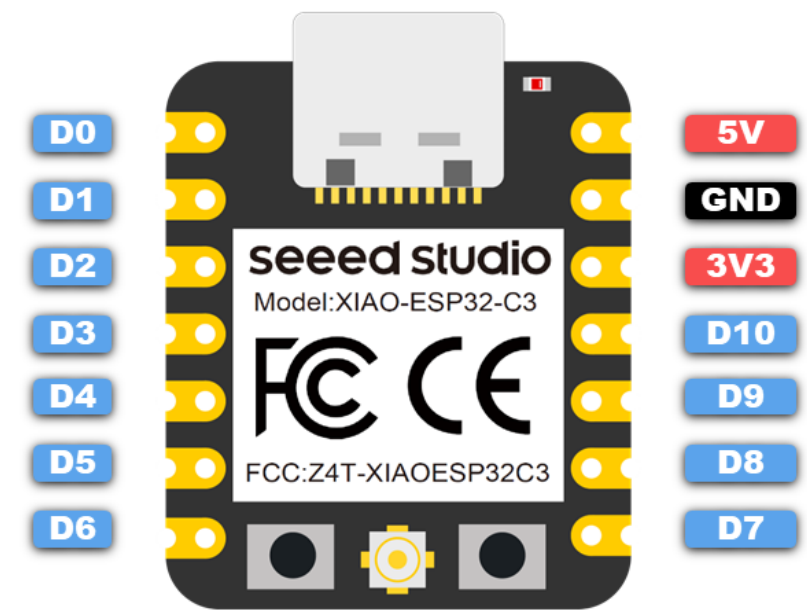
Get into the habit of color-coding voltage as Red and GND as Black. Will help you debug circuits.

Get into the habit of color-coding voltage as Red and GND as Black. Will help you debug circuits.



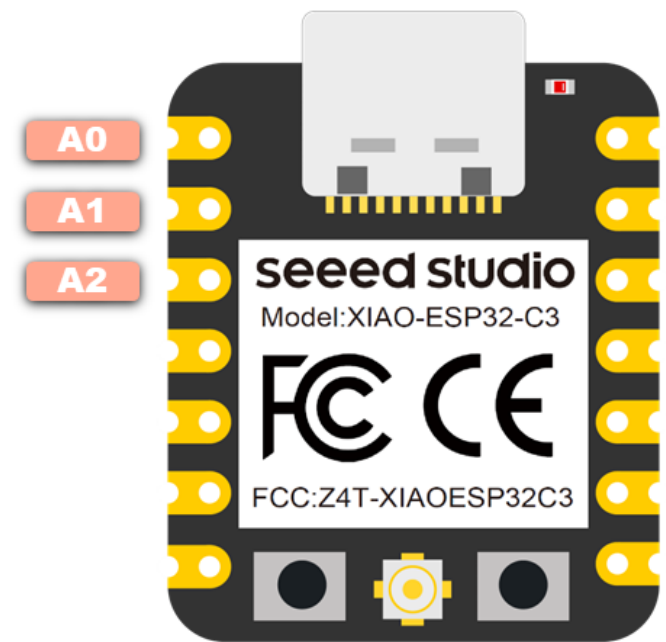


Digital  
Writes and reads HIGH or  
LOW, 1 or 0 data.  
Ex: blinking a light, HIGH is  
turn on, LOW is turn off





Analog In  
Reads in time-varying data.  
Ex: potentiometer

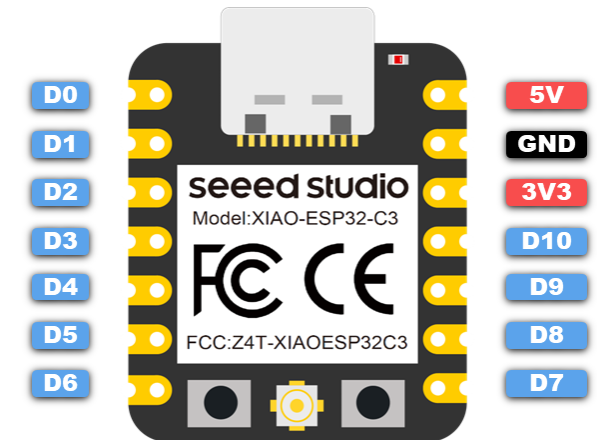
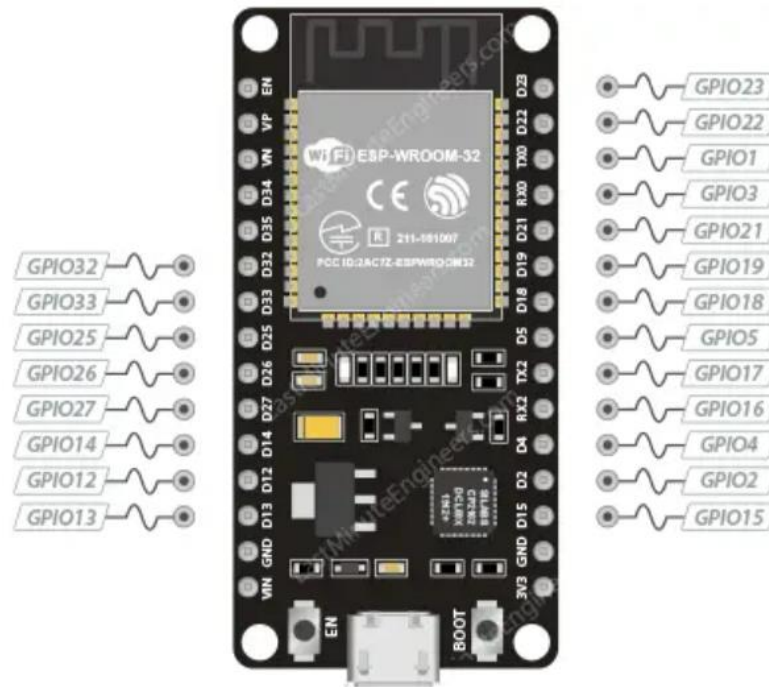
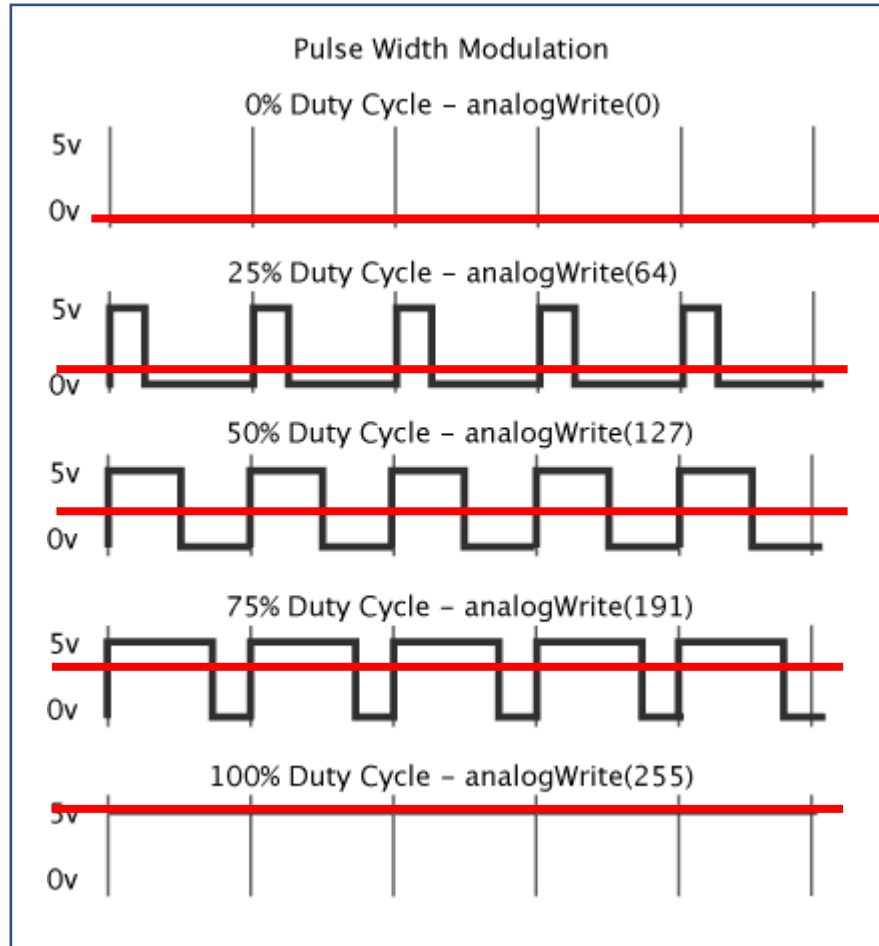




PWM, Pulse Width Modulation  
Technique to get analog results  
from digital signal  
By manipulating the time the  
signal spends in HIGH an LOW

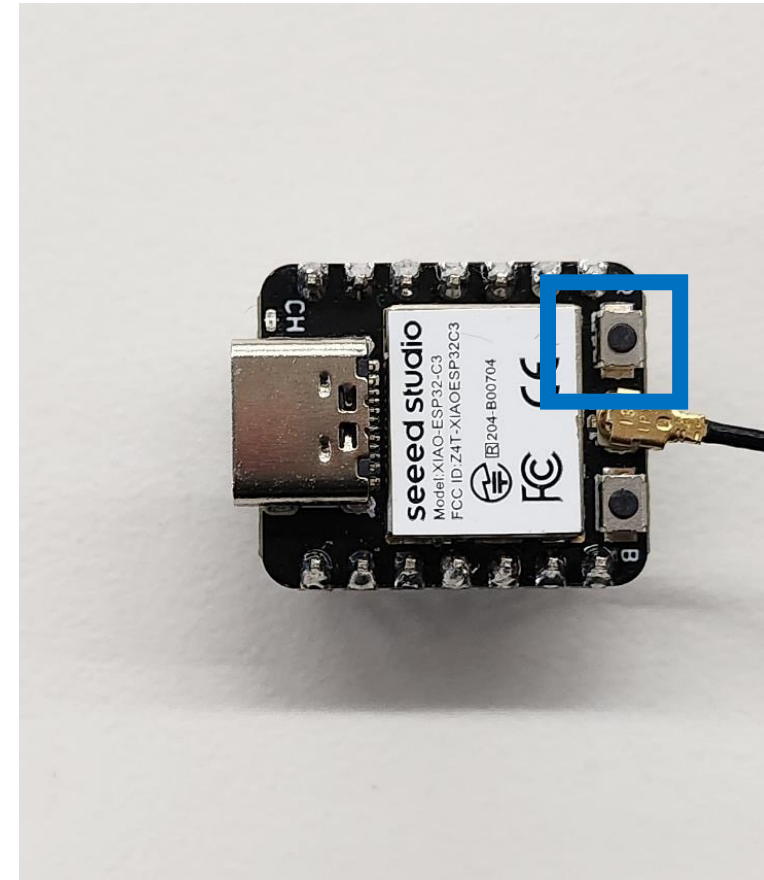
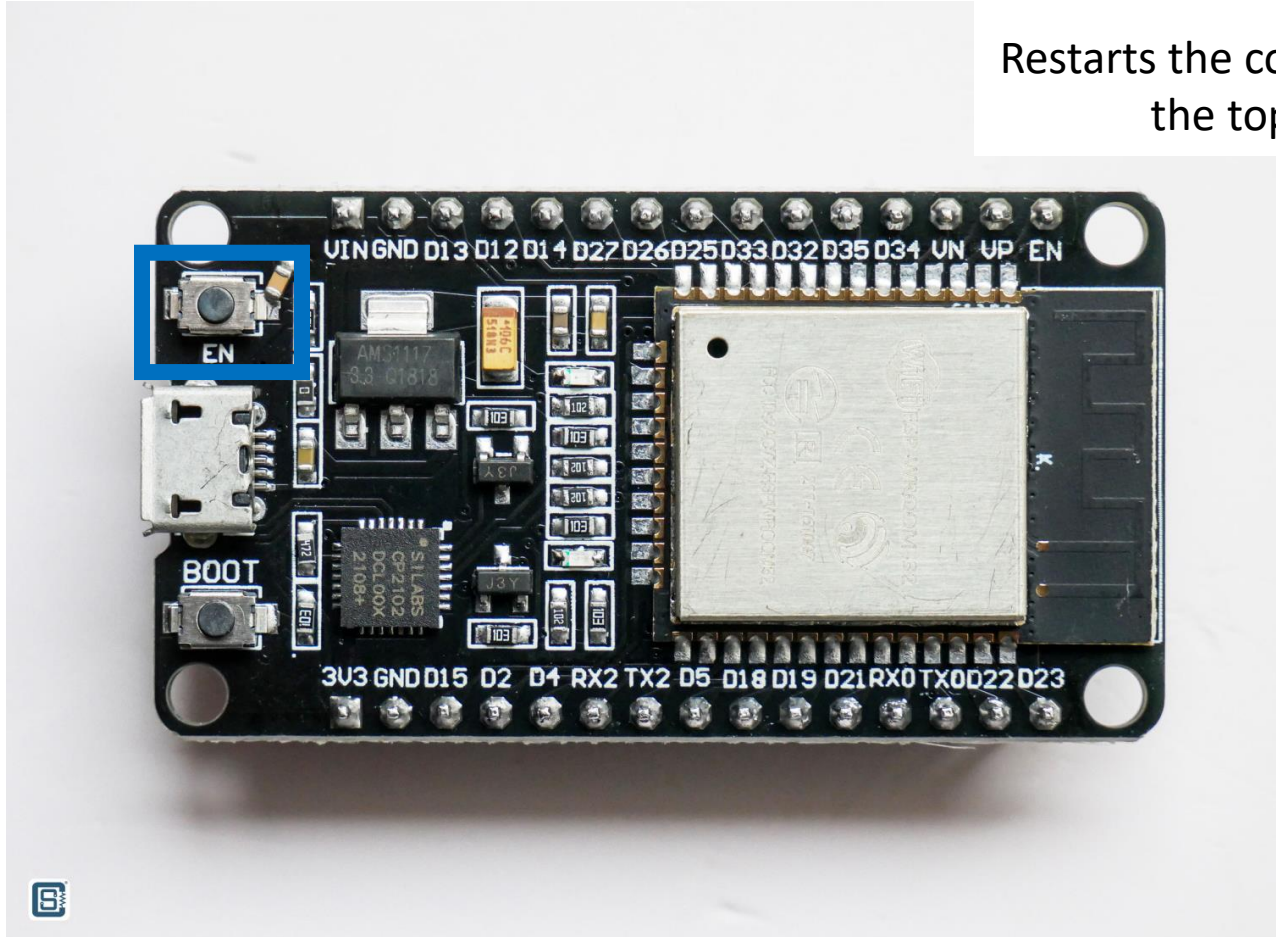
Examples of use:

- Dimming a LED
- Variable speed control



## Reset Button

Restarts the code from  
the top



Only way to stop the code is to  
unplug the microcontroller :[  
Or code it in some way that only  
runs when a button is pressed

# Arduino IDE

Environment where you write your code, upload code, and monitor outputs and inputs with a Serial Monitor

etch\_sep26a | Arduino IDE 2.1.1

Edit Sketch Tools Help



sketch\_sep26a.ino

```
1 void setup() {  
  // put your setup code here, to run once:  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
}
```

Whatever you code, the 'sketch' has to have the **void setup and void loop**; AND ONLY ONE OF EACH  
If you don't, Arduino will be angry

# Usual Code Structure

Initialize libraries and  
variables

What's a library?

Software designed to add  
functionality to your programs

Code that only gets run  
once. Start the serial  
monitor, declare pins, etc.

Code that you want to run  
over and over. Reading  
input sensors, outputting  
actions, etc.

```
1  #include library
2
3  int value;
4  int buttonPin = 3;
5  String message = "Hello World";
6
7  void setup() {
8      // put your setup code here, to run once:
9      Serial.begin(9600);
10     pinMode(buttonPin, INPUT);
11     Serial.println(message);
12 }
13
14 void loop() {
15     // put your main code here, to run repeatedly:
16     value = digitalRead(buttonPin);
17     Serial.println(value);
18 }
```

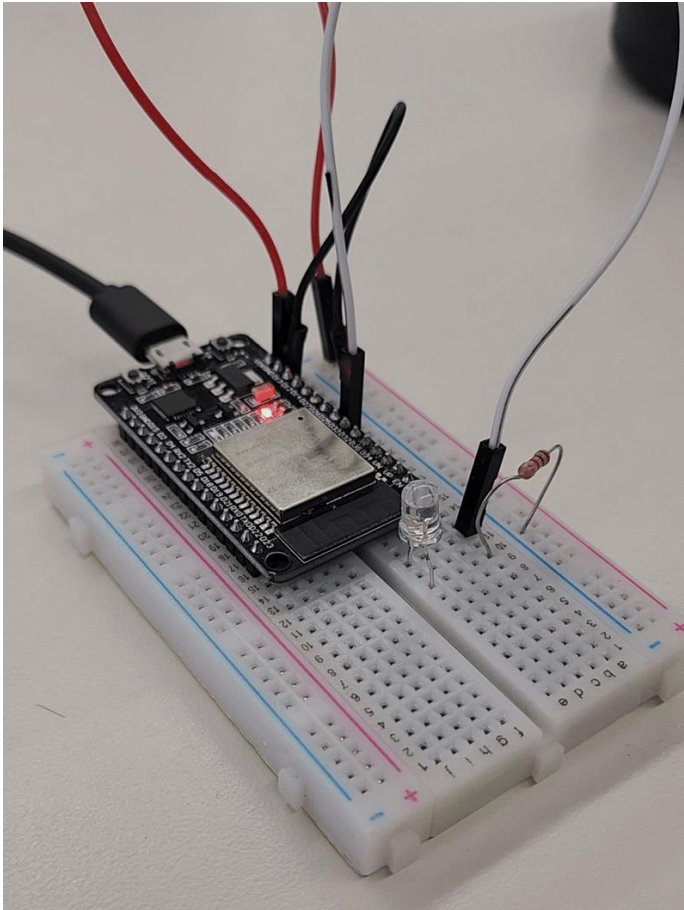


LET'S CODE

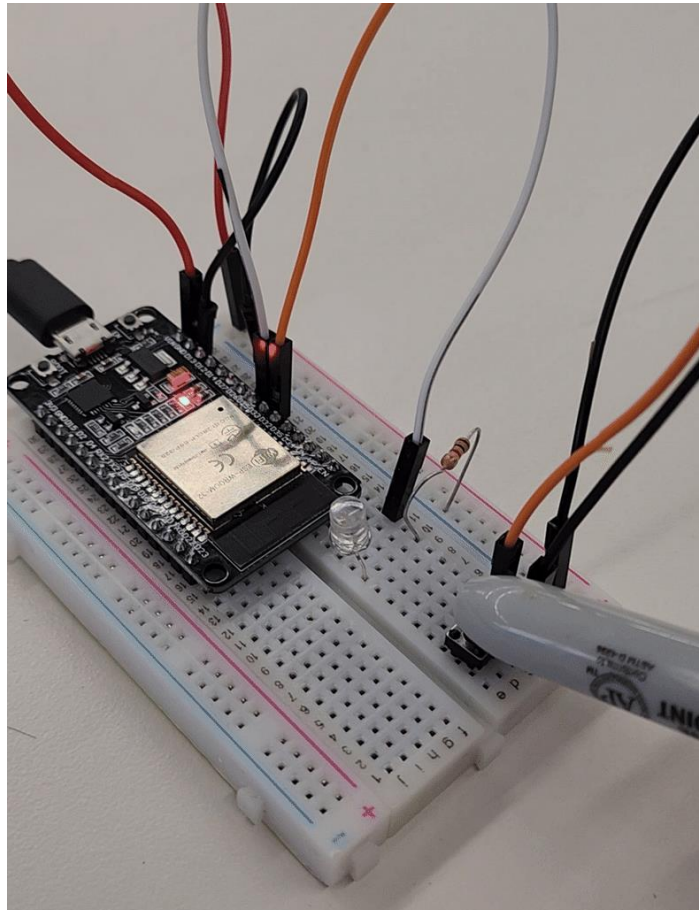


# Coding Examples

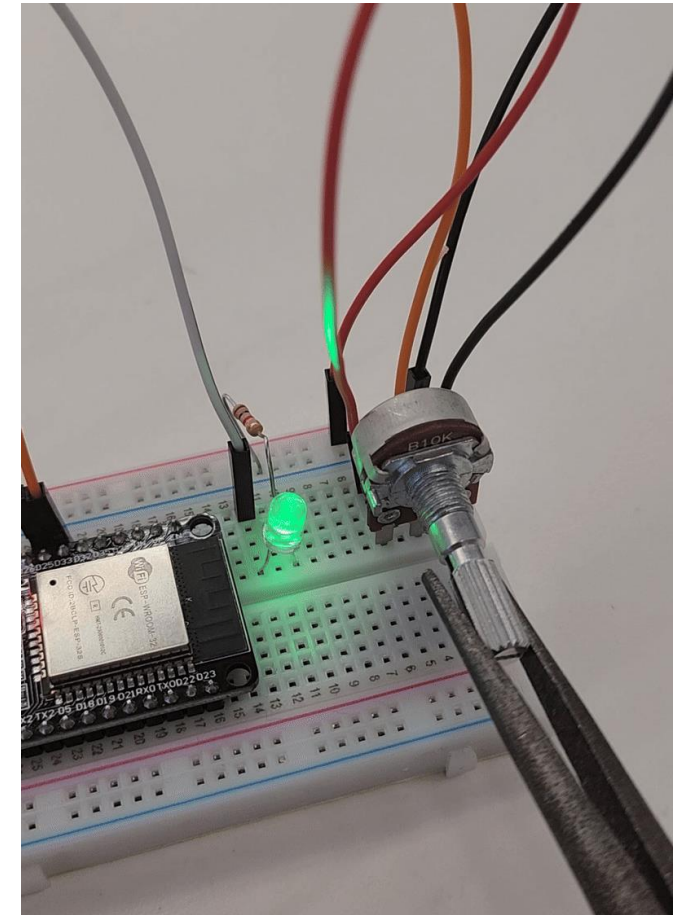
## Blinking LED



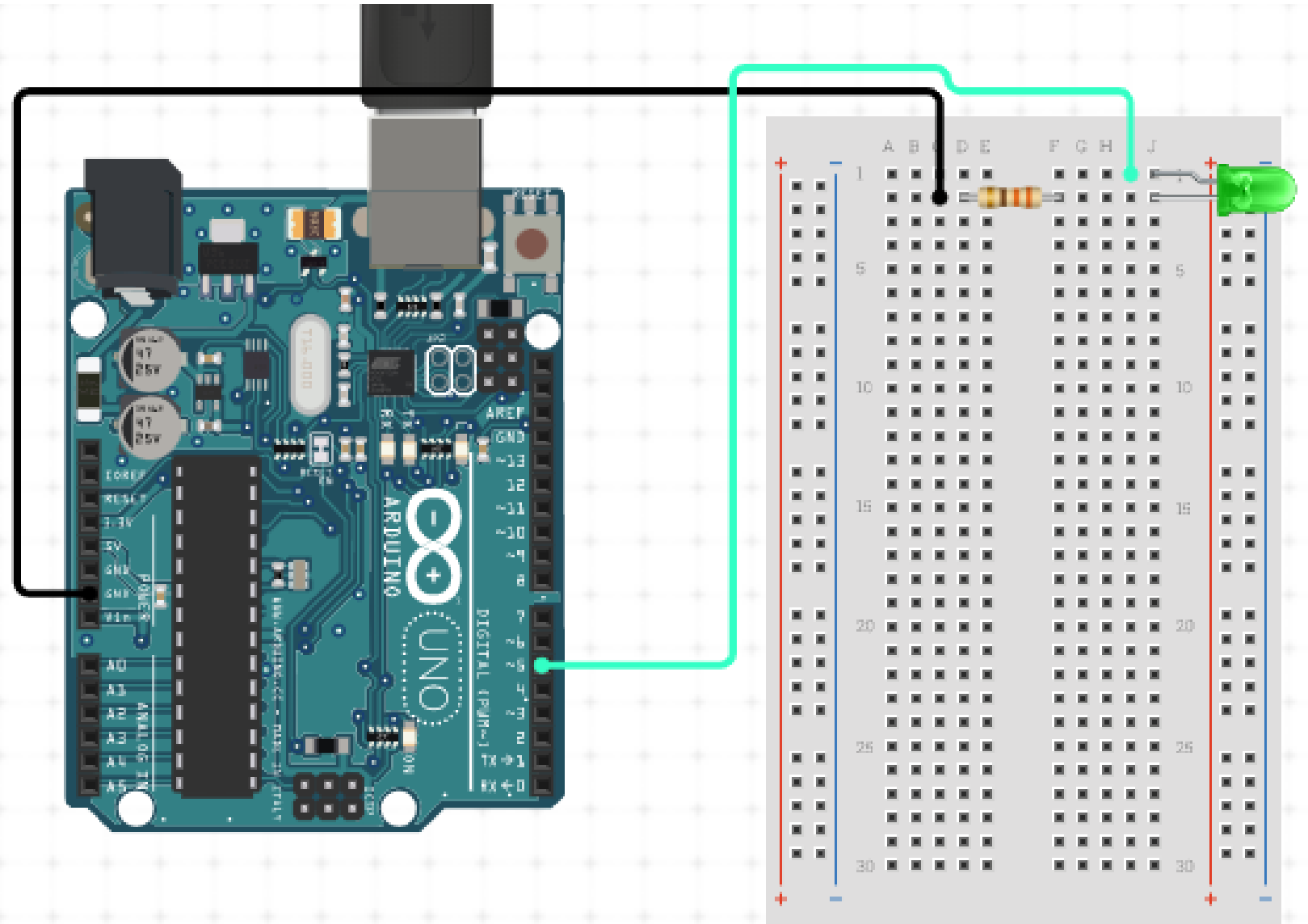
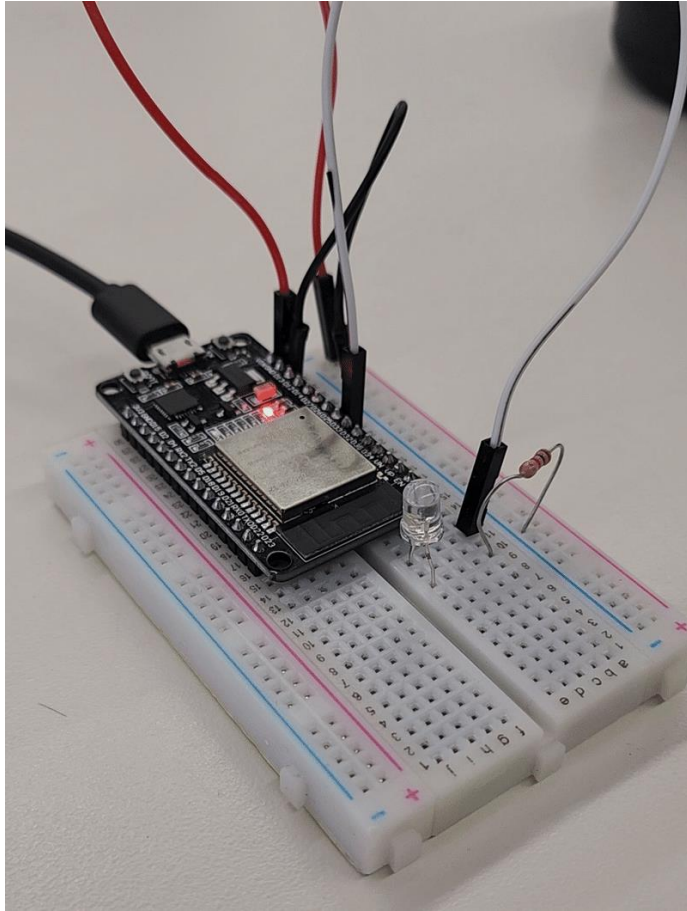
## Button Control



## Potentiometer Control



# Blinking LED



# Code Blinking LED

## DEVKIT

```
int ledPin = 25;
int onTime = 500;
int offTime = 1000;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(onTime);
    digitalWrite(ledPin, LOW);
    delay(offTime);
}
```

## XIAO

```
int ledPin = D7;
int onTime = 500;
int offTime = 1000;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(onTime);
    digitalWrite(ledPin, LOW);
    delay(offTime);
}
```



# Buttons/Switches

## PULL\_UP Resistor vs PULL\_DOWN Resistor

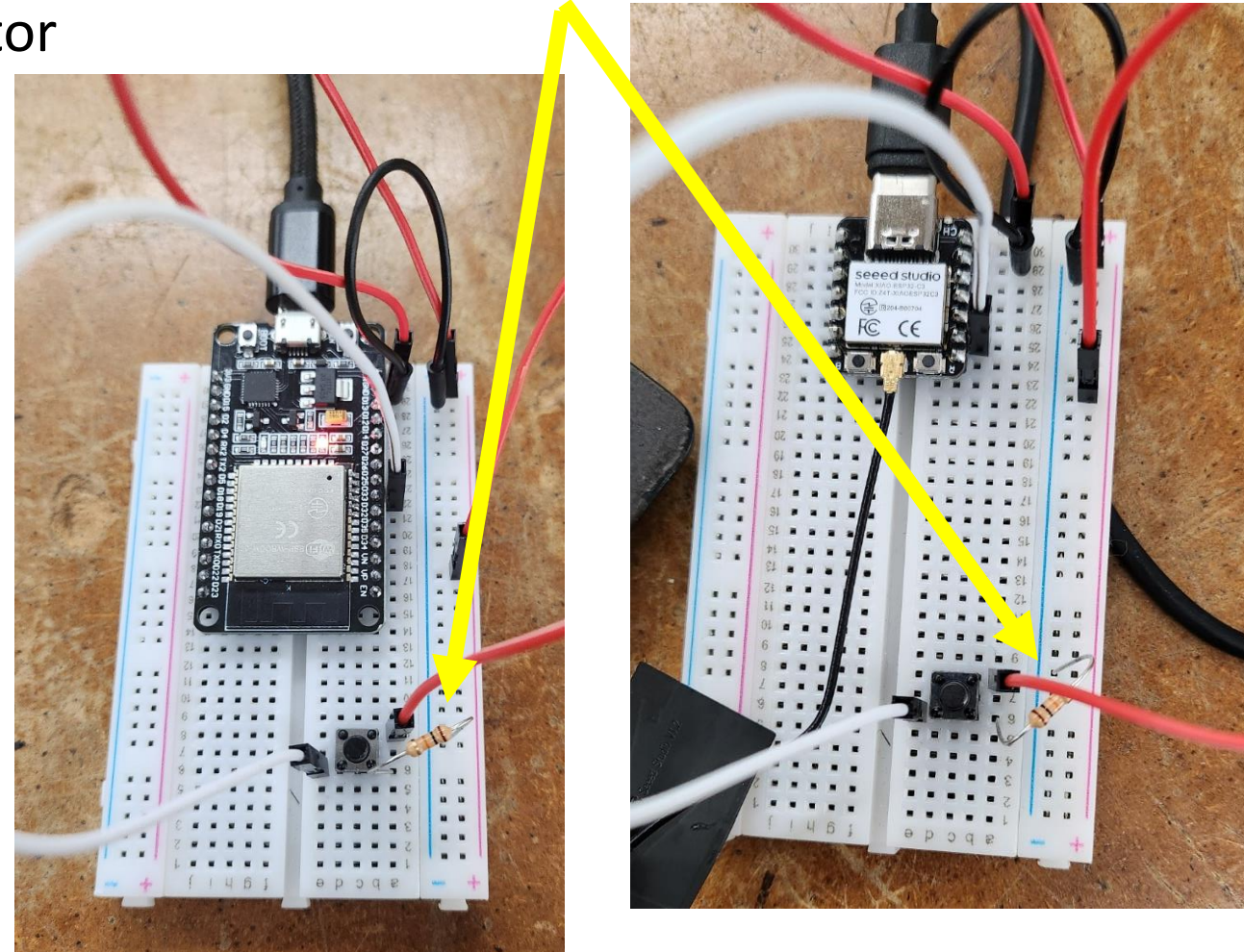
Let's first make a pull-down resistor

When the button is NOT pressed, the value that the pin reads is LOW

The 10kOhm resistor is in between the reading pin and GND

Let's code!

Note that the resistor is on the GND side



# Code Button

## DEVKIT

```
int buttonPin = 33;
int val;
void setup() {
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
}
void loop() {
    val =
digitalRead(buttonPin);
    Serial.println(val);
}
```

## XIAO

```
int buttonPin = D8;
int val;
void setup() {
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
}
void loop() {
    val =
digitalRead(buttonPin);
    Serial.println(val);
}
```

# Button/Switches

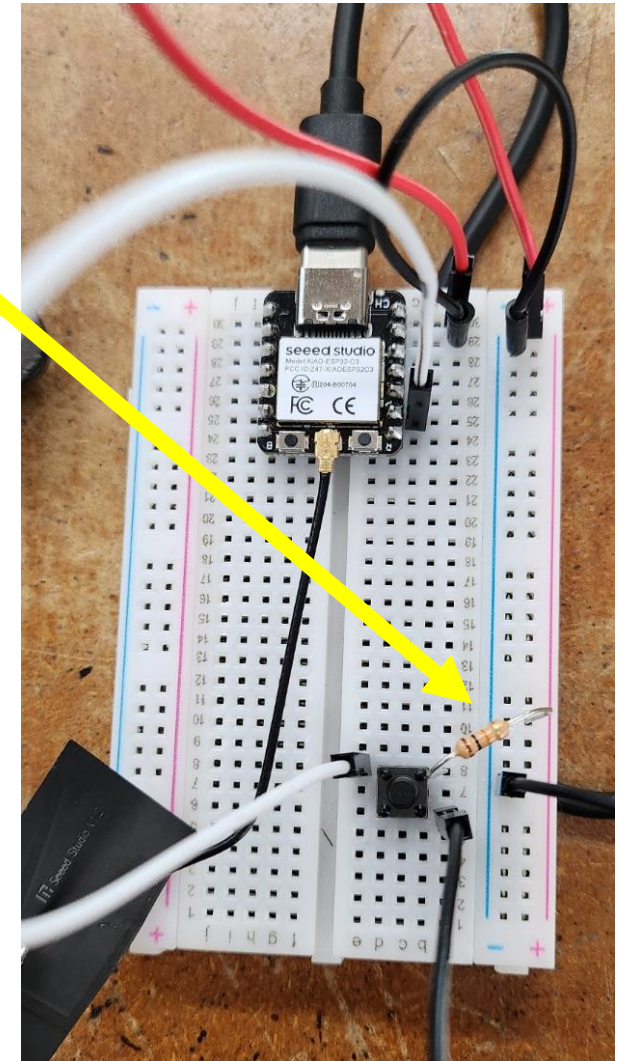
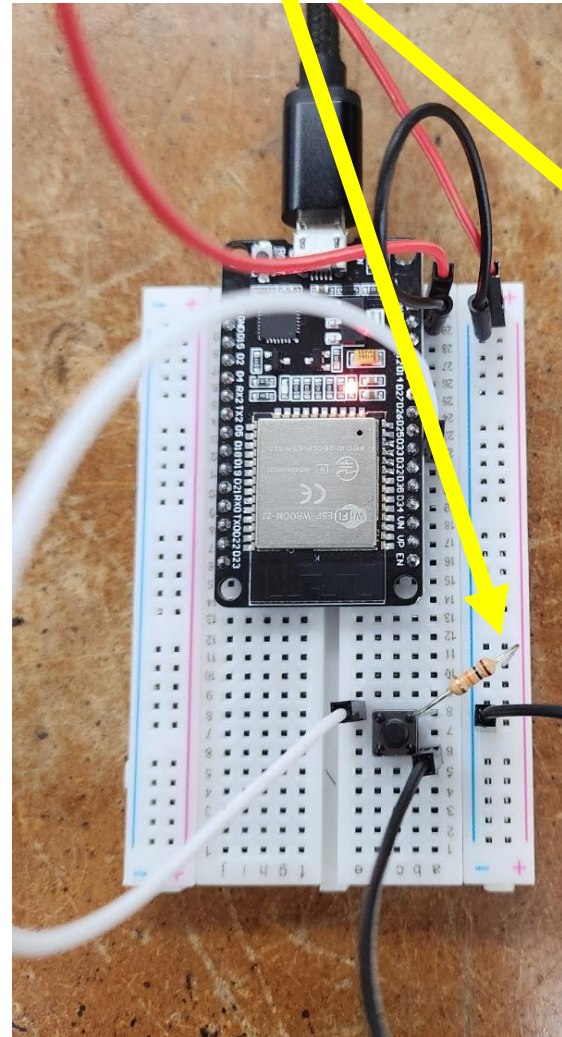
Now make a pull-up resistor

When the button is NOT pressed, the value that the pin reads is HIGH

The 10kOhm resistor is in between the reading pin and POWER (5V/3.3V)

Notice no need to change the code

Note that the resistor is on the Power side



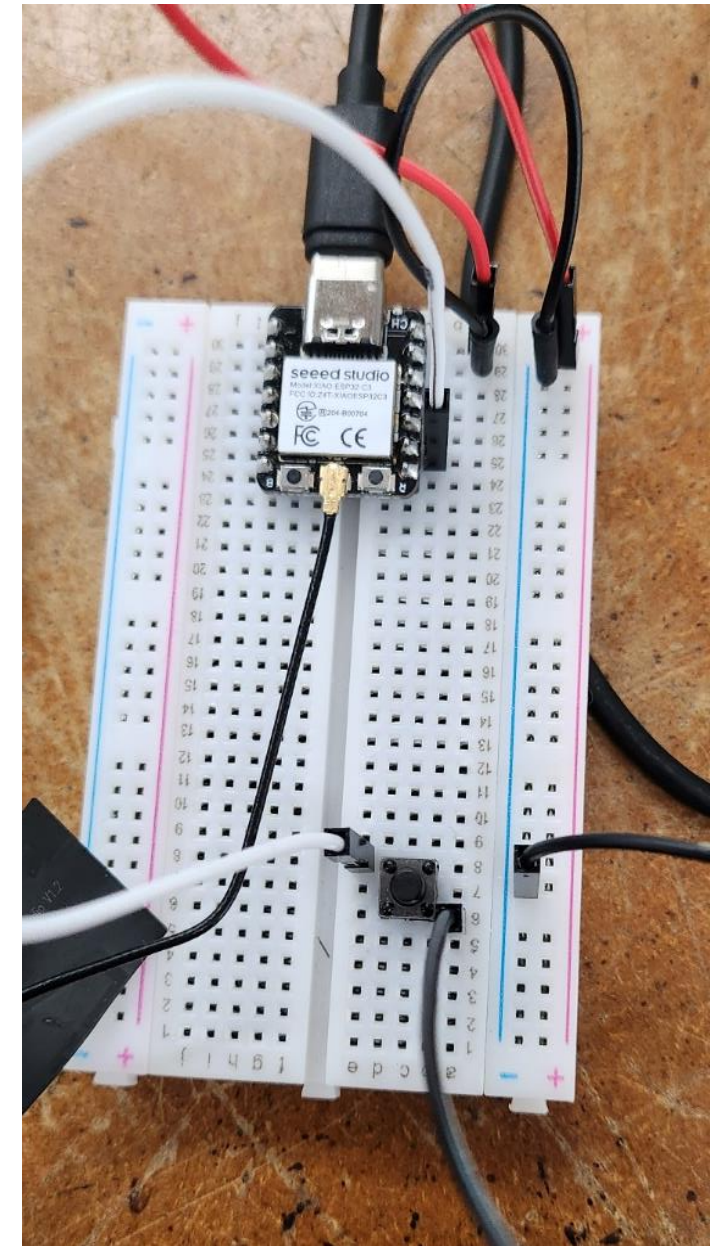
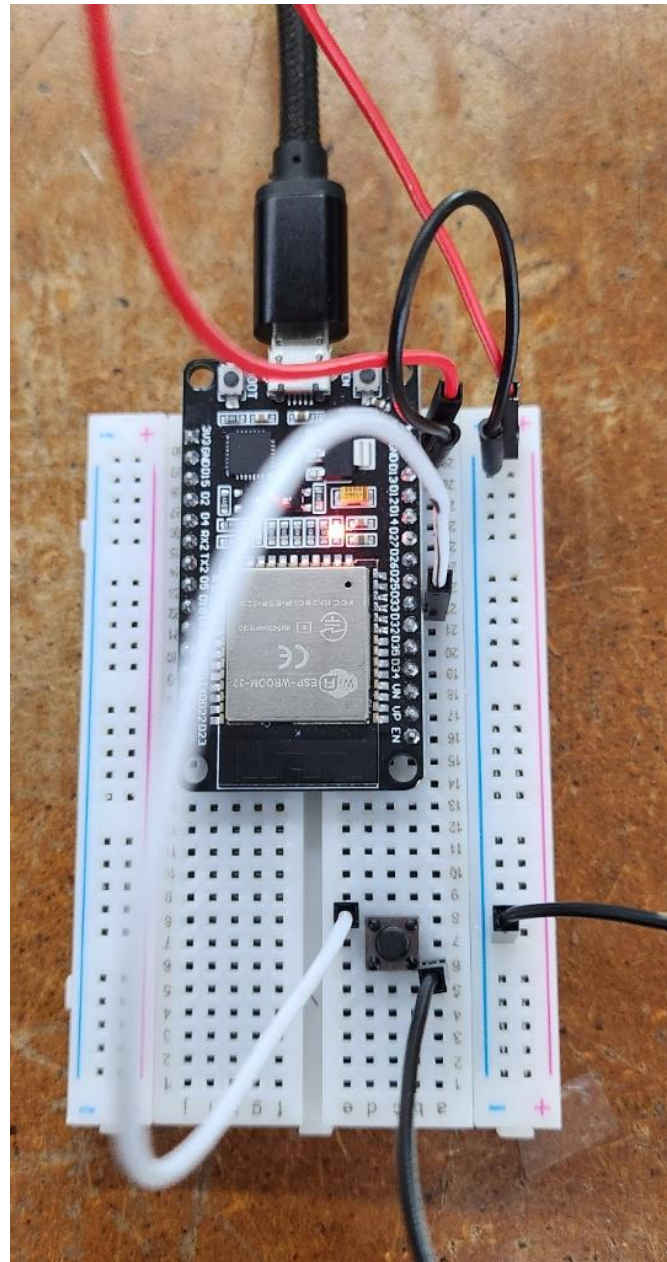


# Button/Switches

Arduino pins have an internal pull-up resistor.

```
pinMode(buttonPin, INPUT_PULLUP);
```

So, we can make a button circuit without the physical resistor!





# Code Button Input Pull Up

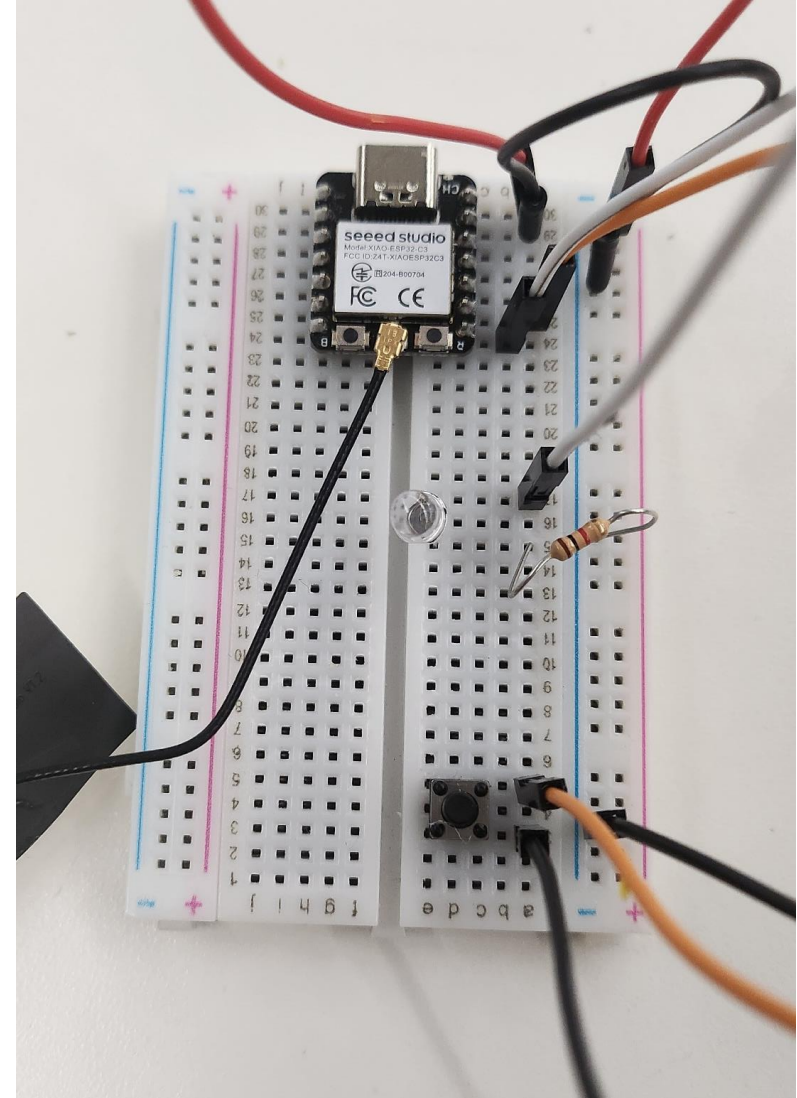
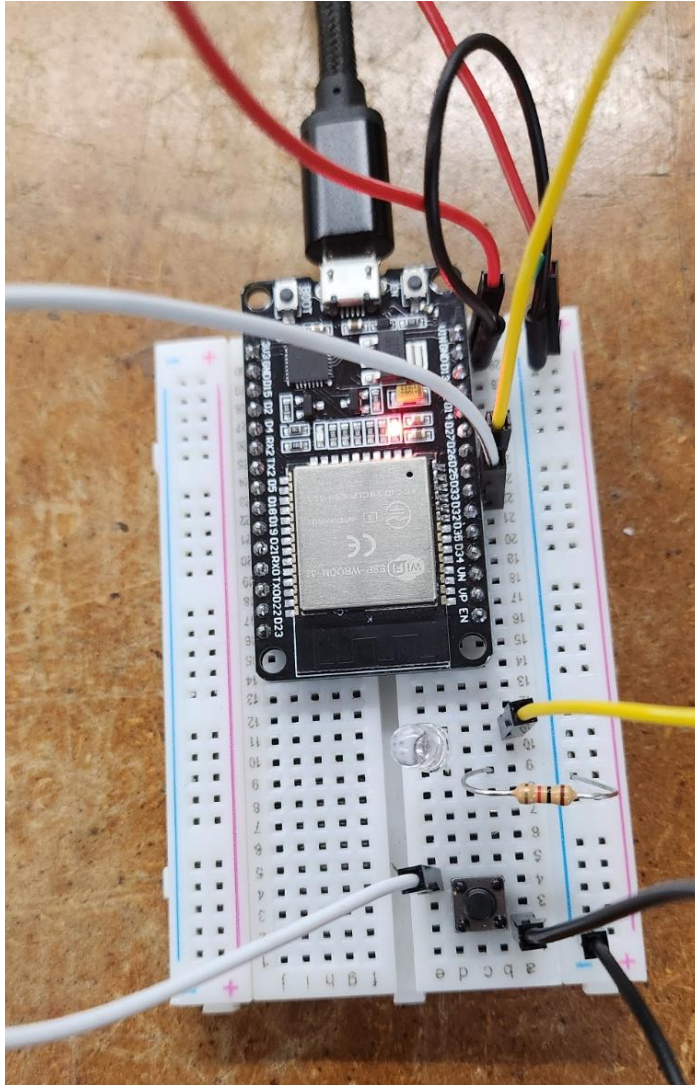
## DEVKIT

```
int buttonPin = 33;
int val;
void setup() {
    Serial.begin(9600);
    pinMode(buttonPin, INPUT_PULLUP);
}
void loop() {
    val = digitalRead(buttonPin);
    Serial.println(val);
}
```

## XIAO

```
int buttonPin = D8;
int val;
void setup() {
    Serial.begin(9600);
    pinMode(buttonPin, INPUT_PULLUP);
}
void loop() {
    val = digitalRead(buttonPin);
    Serial.println(val);
}
```

# Button Control LED



# Code Button Control LED DEVKIT

```
int ledPin = 25;
int buttonPin = 33;
int val;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
    val = digitalRead(buttonPin);
    if(val == LOW){
        digitalWrite(ledPin, HIGH);
    }
    else{
        digitalWrite(ledPin, LOW);
    }
}
```

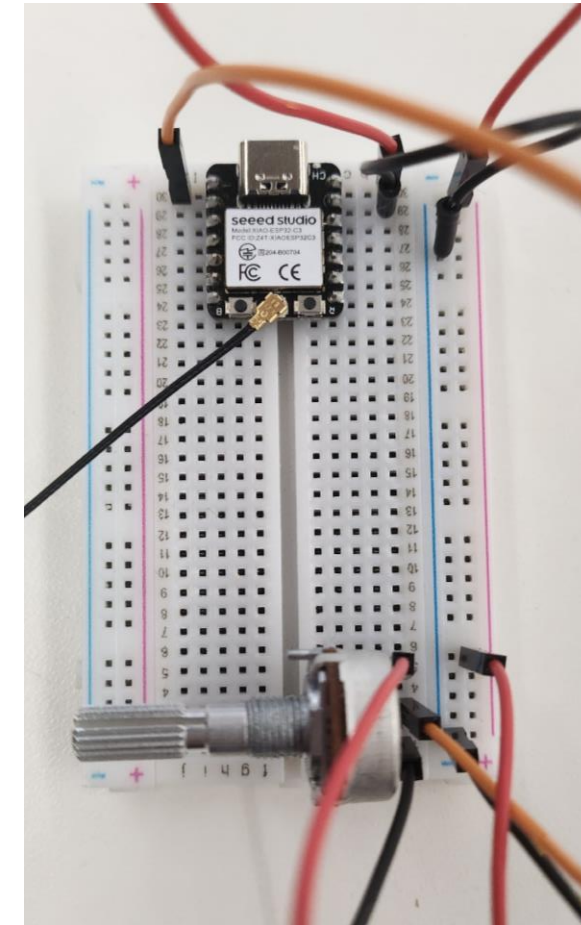
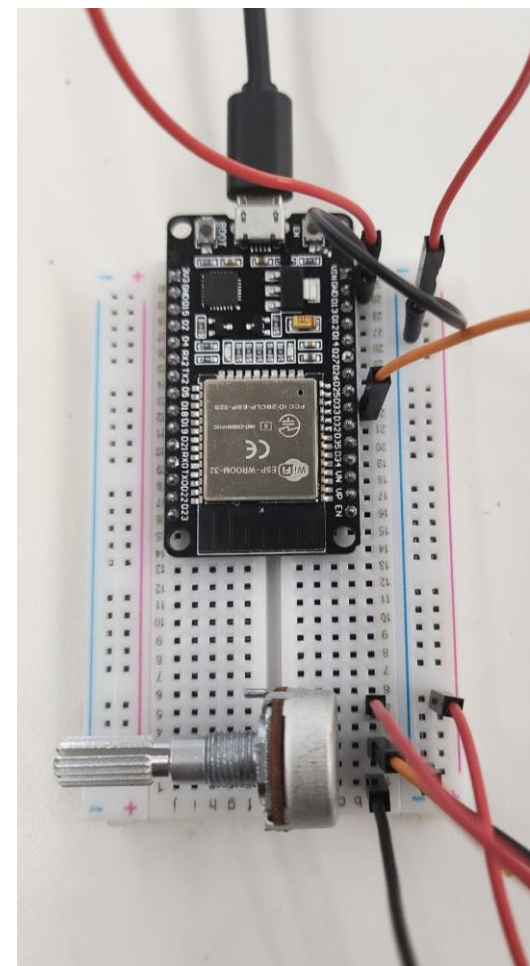
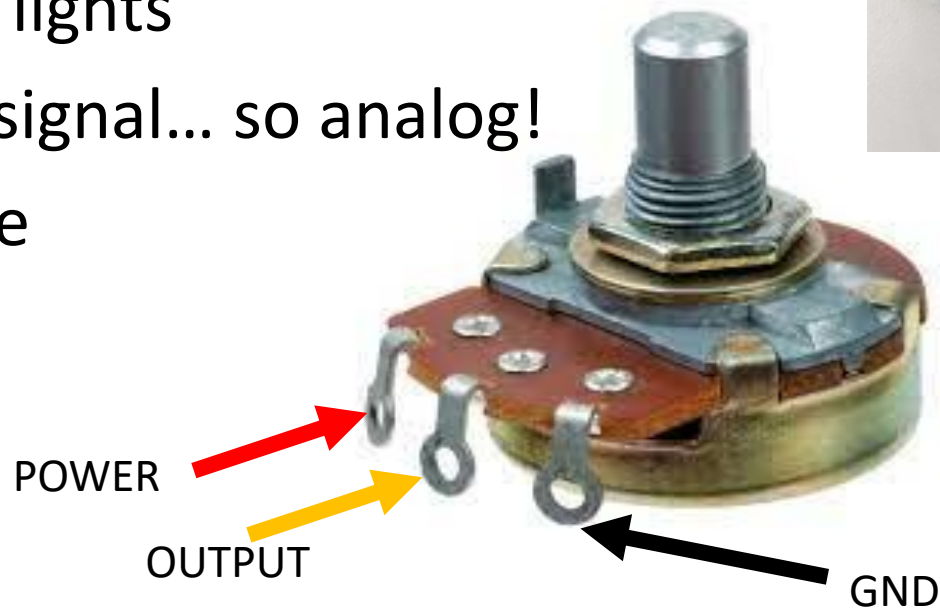
```
int ledPin = D7;
int buttonPin = D8;
int val;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
    val = digitalRead(buttonPin);
    if(val == LOW){
        digitalWrite(ledPin, HIGH);
    }
    else{
        digitalWrite(ledPin, LOW);
    }
}
```

# Potentiometer

- A variable resistor. By turning the knob you vary the resistance which in turn varies the amount of voltage allowed through.
- Can be useful for speed control, dimming lights
- Variable signal... so analog!
- Let's code



# Code Potentiometer DEVKIT

```
int potPin = 33;
int potVal;

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

void loop() {
  potVal = analogRead(potPin);
  Serial.println(potVal);
}
```

# XIAO

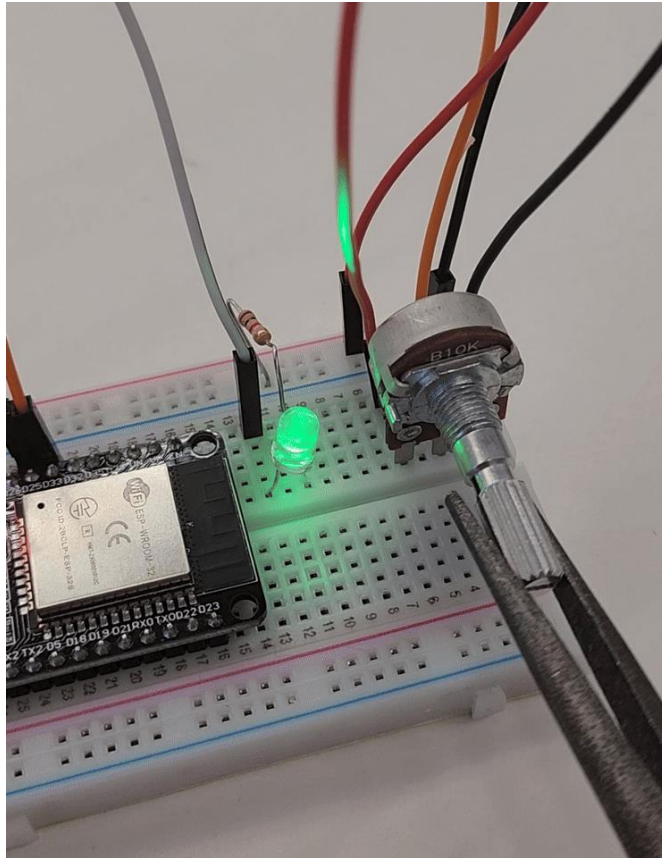
```
int potPin = A0;
int potVal;

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

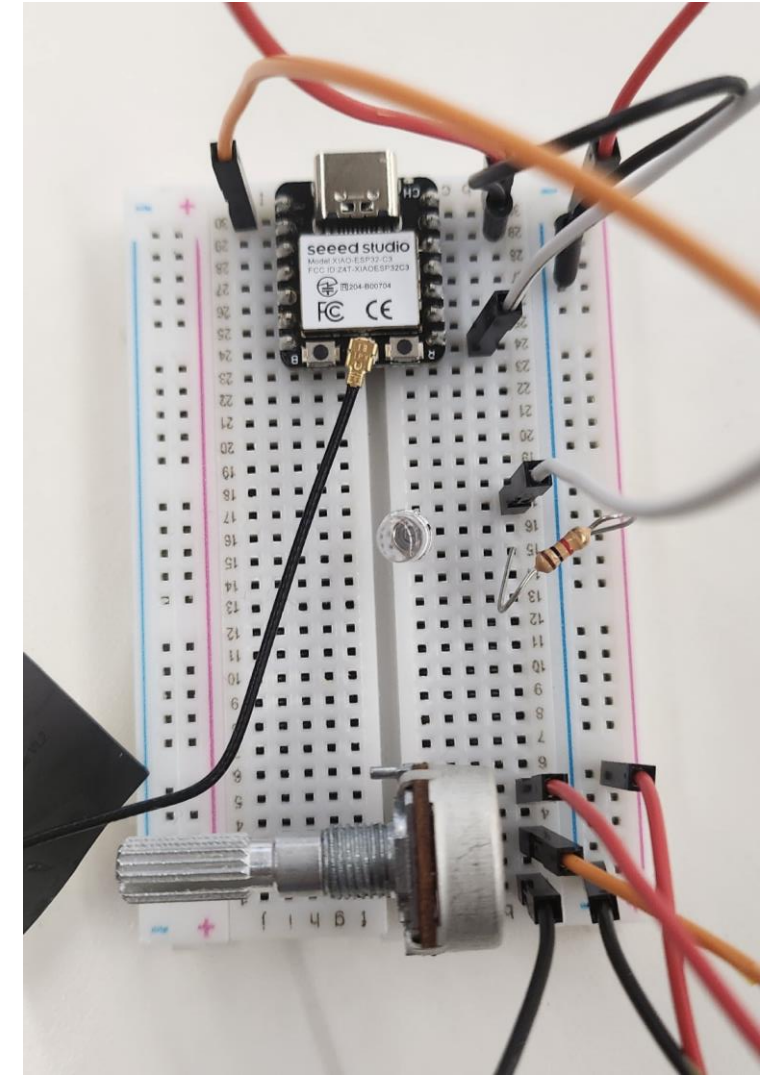
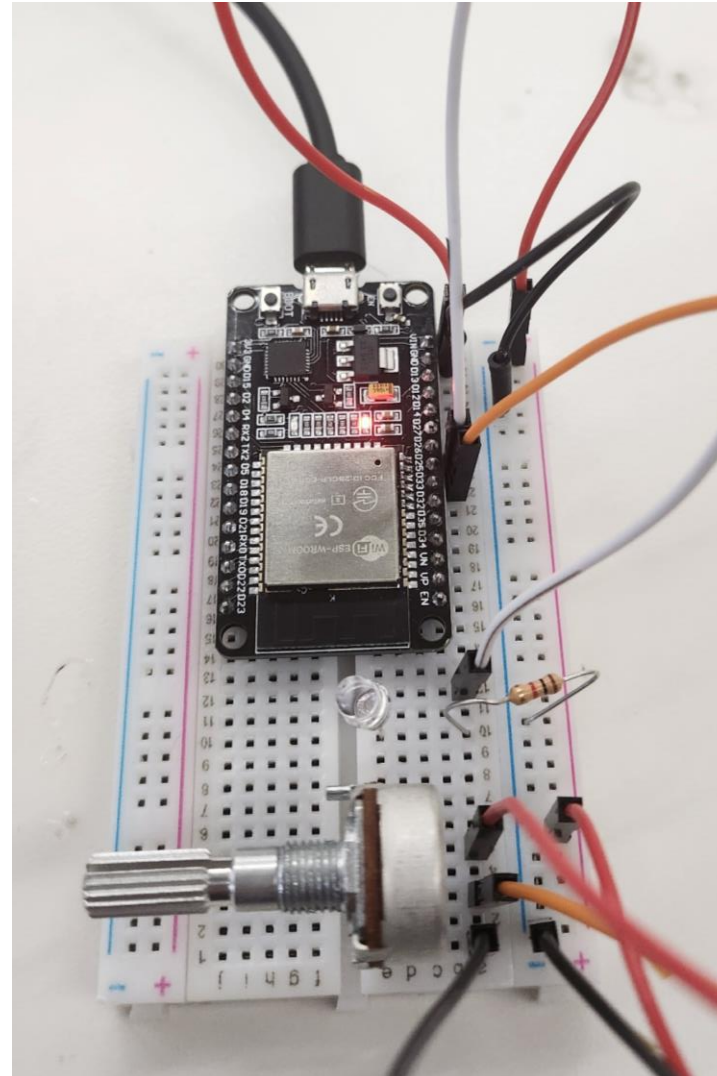
void loop() {
  potVal = analogRead(potPin);
  Serial.println(potVal);
}
```



# Potentiometer Control LED



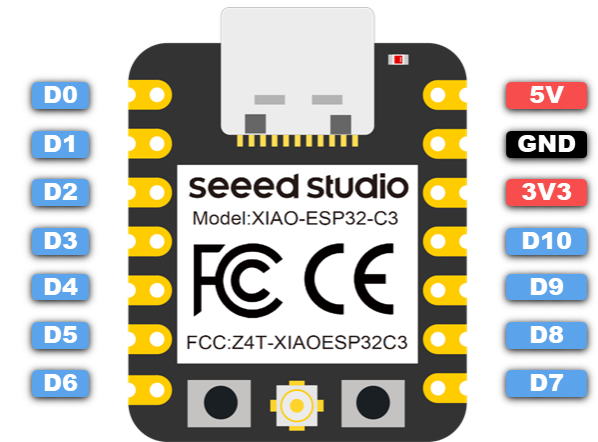
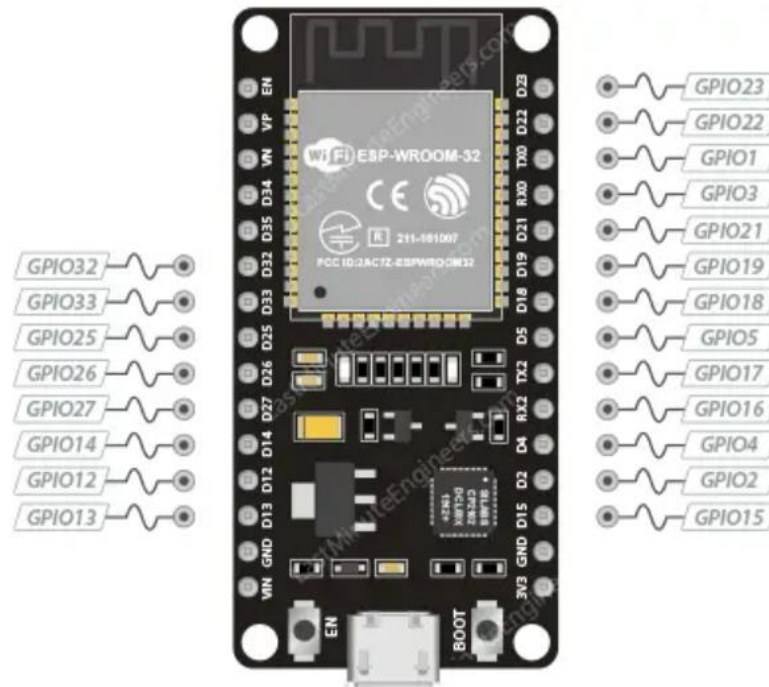
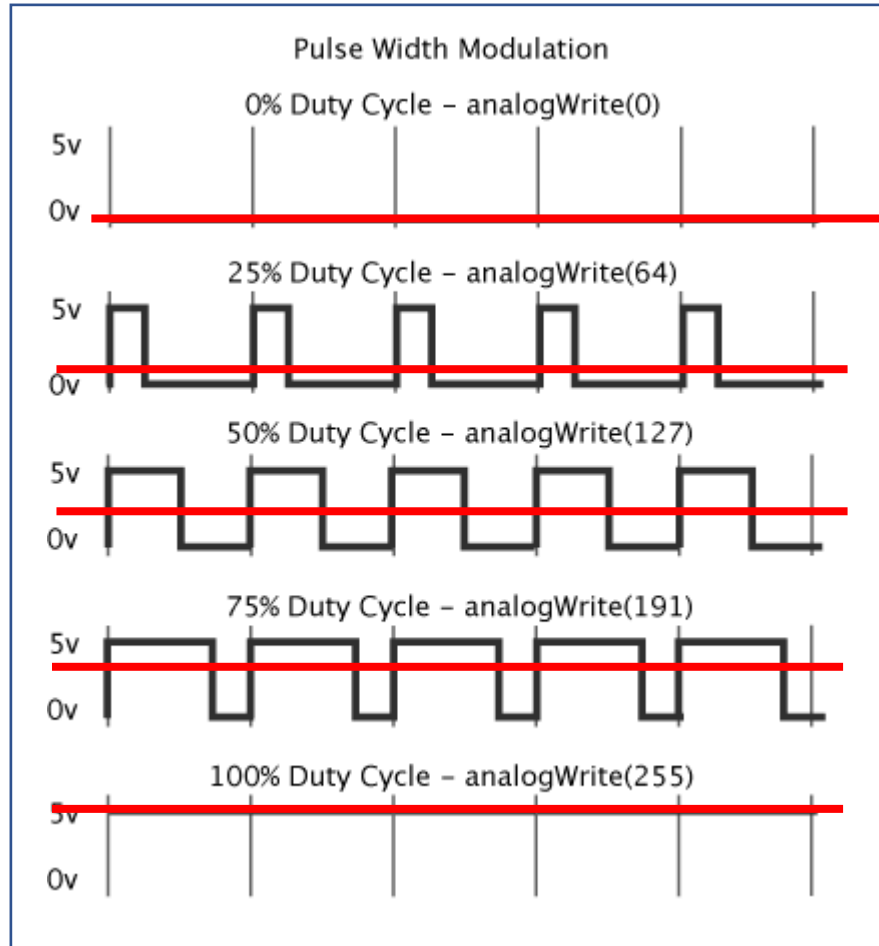
Remember PWM signals?



PWM, Pulse Width Modulation  
Technique to get analog results  
from digital signal  
By manipulating the time the  
signal spends in HIGH an LOW

Examples of use:

- Dimming a LED
- Variable speed control



# Mapping Values

Arduino has an `analogRead` range from 0 to 4095, and an `analogWrite` range only from 0 to 255

Potentiometer has range of 0 to 4095

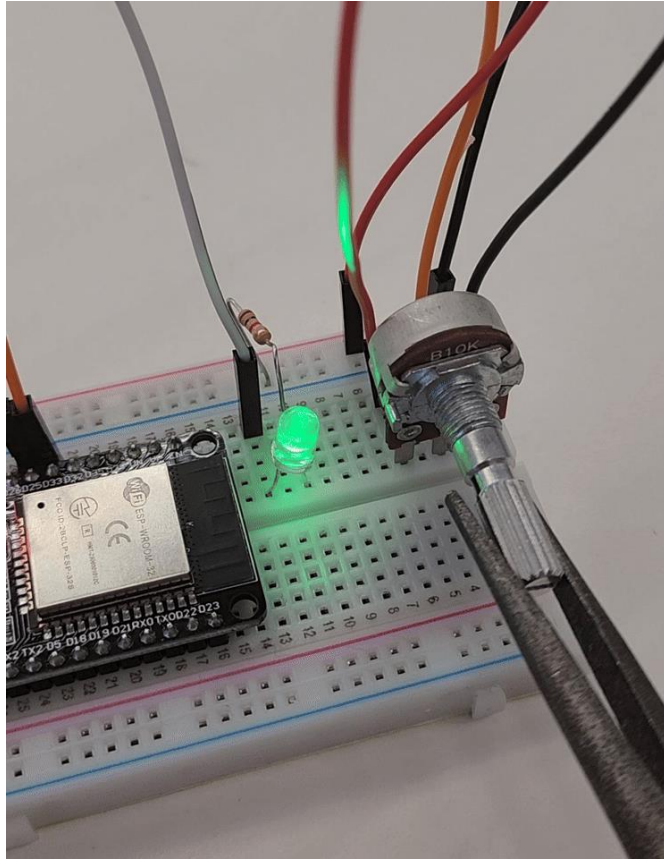
LED has range of 0 to 255

Need to map the values 0 to 4095 to 0 to 255

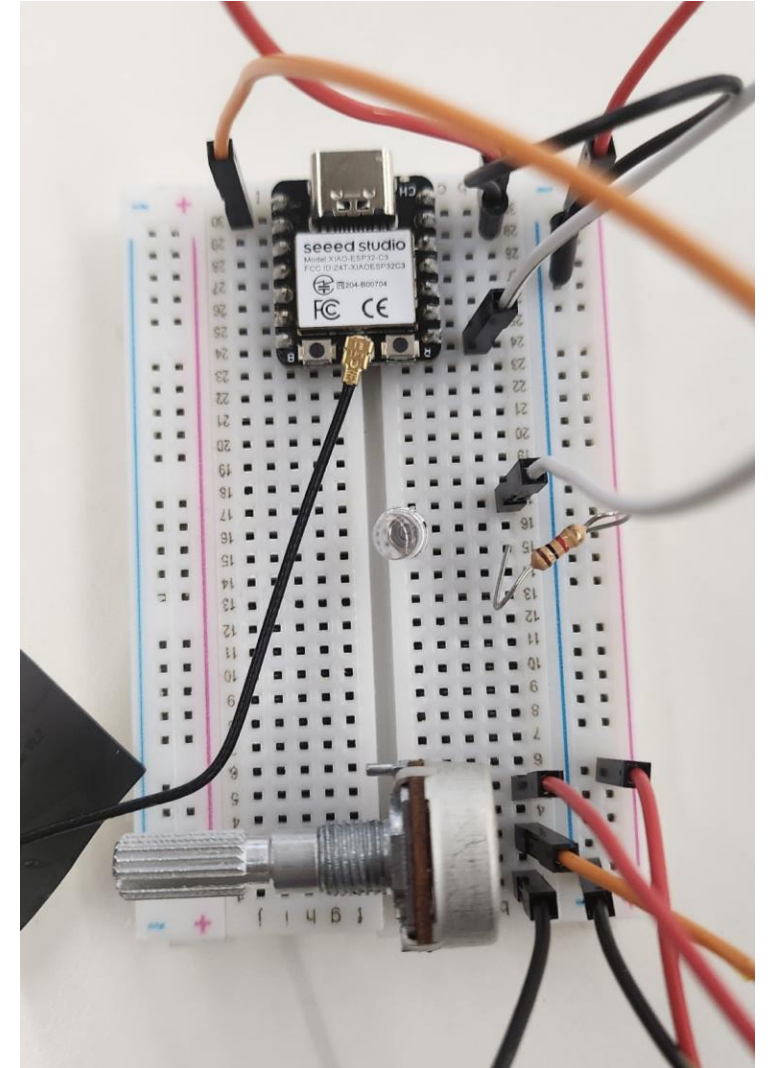
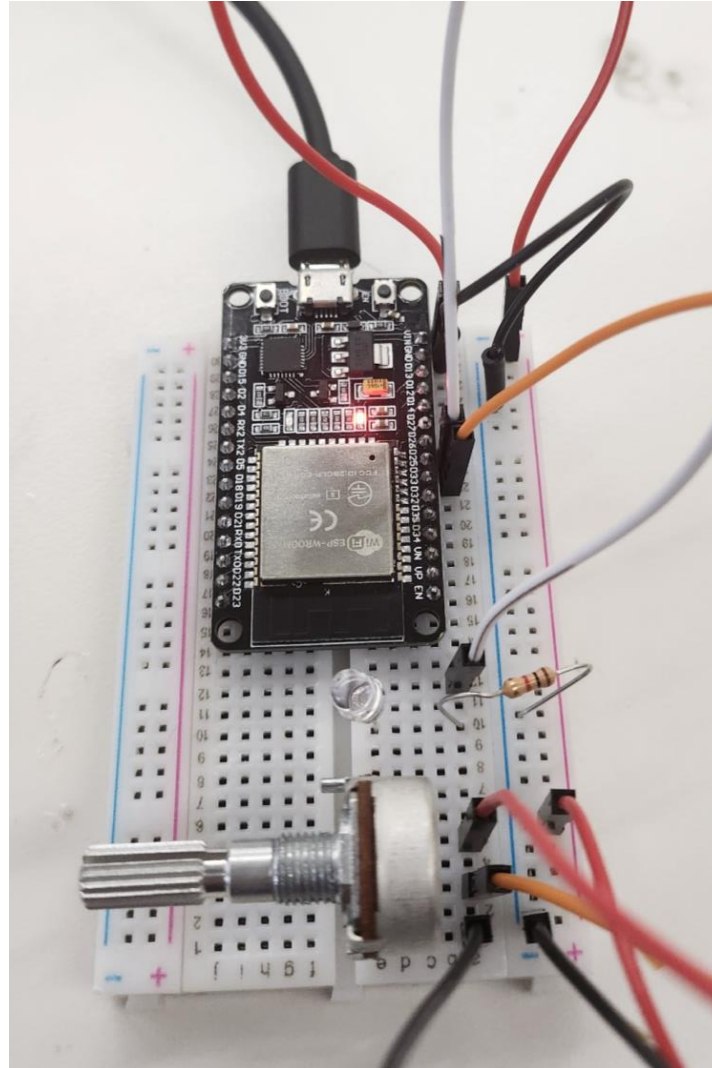
`map(val I want to map, [lowerB, upperB] of the initial, [lowerB, upperB] of the final)`



# Potentiometer Control LED



Let's code



# Code Potentiometer Control LED

## DEVKIT

```
int potPin = 33;
int potVal;
int ledPin = 25;
int ledVal;

const int freq = 5000;
const int resolution = 8;

void setup() {
  pinMode(potPin, INPUT);
  pinMode(ledPin, OUTPUT);
  ledcAttach(ledPin, freq, resolution);
}

void loop() {
  potVal = analogRead(potPin);
  ledVal = map(potVal, 0, 4095, 0, 255);
  ledcWrite(ledPin, ledVal);
}
```

## XIAO

```
int potPin = A0;
int potVal;
int ledPin = D7;
int ledVal;

const int freq = 5000;
const int resolution = 8;

void setup() {
  pinMode(potPin, INPUT);
  pinMode(ledPin, OUTPUT);
  ledcAttach(ledPin, freq, resolution);
}

void loop() {
  potVal = analogRead(potPin);
  ledVal = map(potVal, 0, 4095, 0, 255);
  ledcWrite(ledPin, ledVal);
}
```

# Sources

- [Computer Programming Tutorial \(tutorialspoint.com\)](http://tutorialspoint.com)
- [Learn | Arduino Documentation](#)
- [PS70: Introduction to Digital Fabrication \(nathanmelenbrink.github.io\)](http://nathanmelenbrink.github.io)

