LECTURE IV

ESP32 Programming Pt. 2

Group Activity

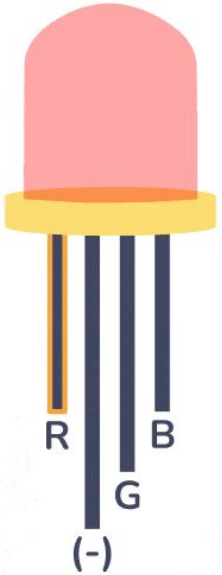
- Find your instructor
- Questions/Concerns
- If you could put anything into the ultrasonic trashcan project, what would it be?

SECTION I

Project III Review

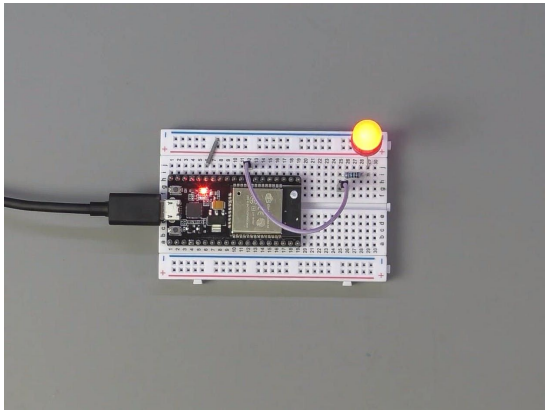
Project 3 Review

- Official Due Date: **Monday, 11/24 by 11:59PM**
- Ask questions and seek help (online or @Lab Hours), so you can finish it on time!



Learning Concepts:

- Embedded Systems
- Microcontrollers
- ESP32
- Arduino IDE
- Pulse Width Modulation (PWM)



New Deadlines

- For the full $20 refund for Fall Quarter, Final Project Deadlines will be:
  - Project 1: By the end of Winter Break (1/5/2025 at 11:59PM)
  - Project 2: By the end of Winter Break (1/5/2025 at 11:59PM)
  - Project 3: By the end of Winter Break (1/5/2025 at 11:59PM)
  - Project 4: By the end of Winter Break (1/5/2025 at 11:59PM)

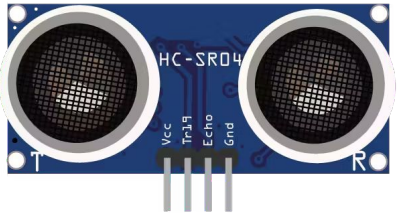
- Ask project-specific questions in the **#ops-help** channel on the Discord. We'll answer during the break!

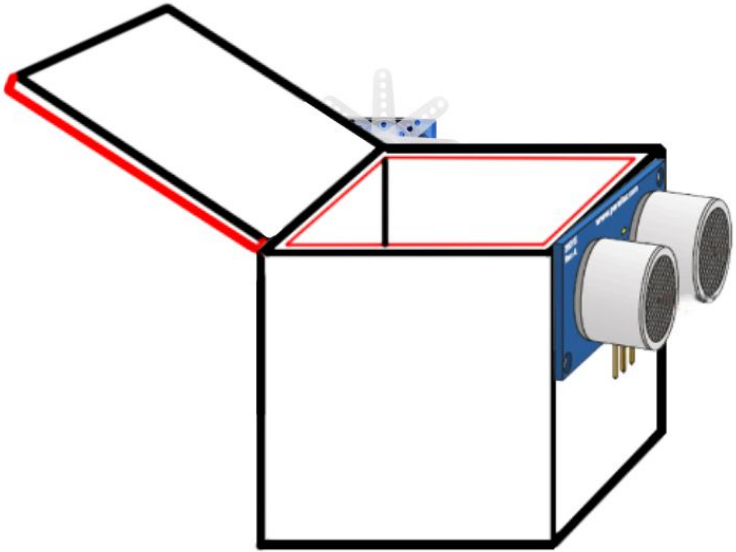
SECTION II

Project IV

Project 4 Overview

- Build a mini trash can that uses an **ultrasonic sensor, micro servo,** and **ESP32**





You will learn:

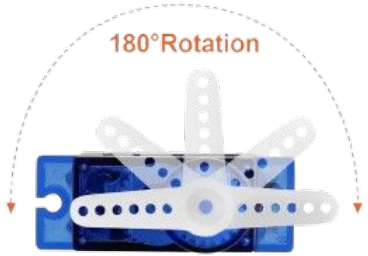
- ESP32 (Continued)
- Servos
- Ultrasonic Sensor
- Arduino Libraries
- Tips for Programming in Arduino IDE

SECTION III

Servos

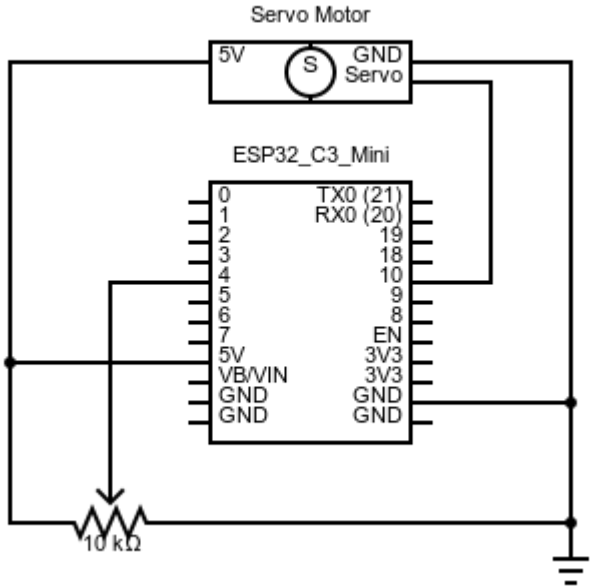
Servo

- A **servo motor** is a type of motor that converts electrical energy into mechanical energy to achieve precise control between 0°-180°
- Has three different wires: power (red), ground (brown), and pulse-width modulation (orange)
    - Insert jumper wires into the female end of the micro servo and to connect it to the ESP32 on a breadboard
- The servo will be controlled based on readings from the **ultrasonic sensor**



Servo Pinout

- To the ESP32, connect:
- Power (red) → 5V pin
- PWM (orange) → any GPIO
  - Remember PWM = Pulse-Width Modulation!
- Ground (brown) → GND



SECTION IV

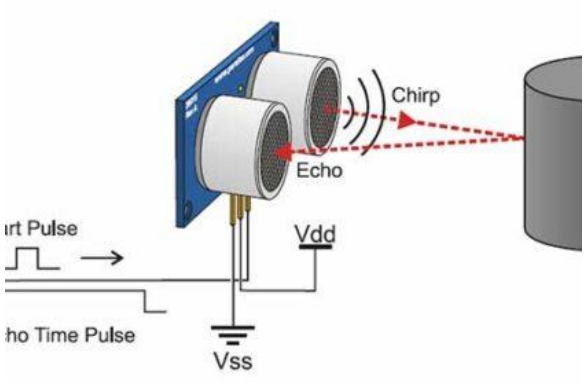
Ultrasonic Sensor

Ultrasonic Sensor

- A **ultrasonic sensor** is an electronic device that measures distance via high-frequency waves
- Has four different pins; **power**, **ground**, echo, and trig

Sequence of events:

1) **TRIG** pin receives a trigger command, and the sensor sends a sound wave
2) Right when the sound wave is sent, the ECHO pin is flipped to HIGH
3) The sound wave is reflected back and the moment it is detected, the ECHO pin is flipped to LOW
4) Microcontroller measures duration of the ECHO pulse

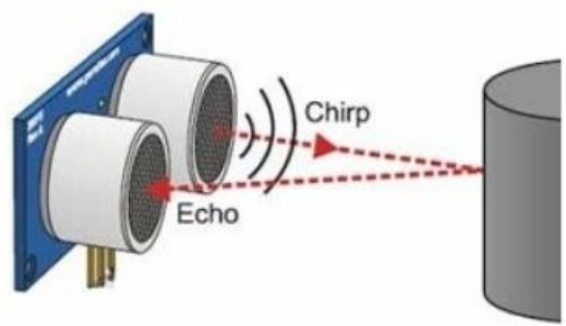


Ultrasonic Sensor

- A **ultrasonic sensor** is an electronic device that measures distance via high-frequency waves
- Four different pins: **power**, **ground**, **echo**, and **trig**

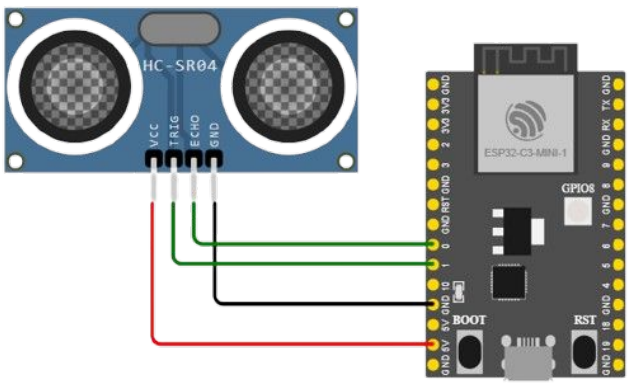
Sequence of events:

1) **TRIG** pin receives a command, and the sensor sends a sound wave
2) Sensor receives echo from sound wave and stores it as a pulse on the ECHO pin
3) Microcontroller measures duration of the pulse



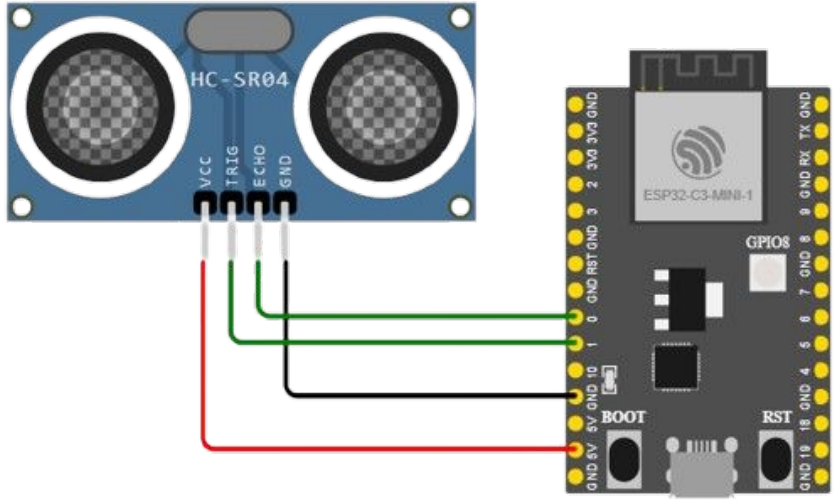
Ultrasonic Sensor Pinout

- Connect the **VCC** pin of the ultrasonic sensor to the **5V** pin of ESP32
- Connect the TRIG and ECHO pins to any GPIO pin of the ESP32
- Connect the **GND** of the ultrasonic sensor to the **GND** pin of the ESP32



Ultrasonic Sensor Pinout

- Connect the **VCC** pin of the ultrasonic sensor to the **5V** pin of ESP32
- Connect the TRIG and ECHO pins to any GPIO pin of the ESP32
- Connect the **GND** of the ultrasonic sensor to the **GND** pin of the ESP32

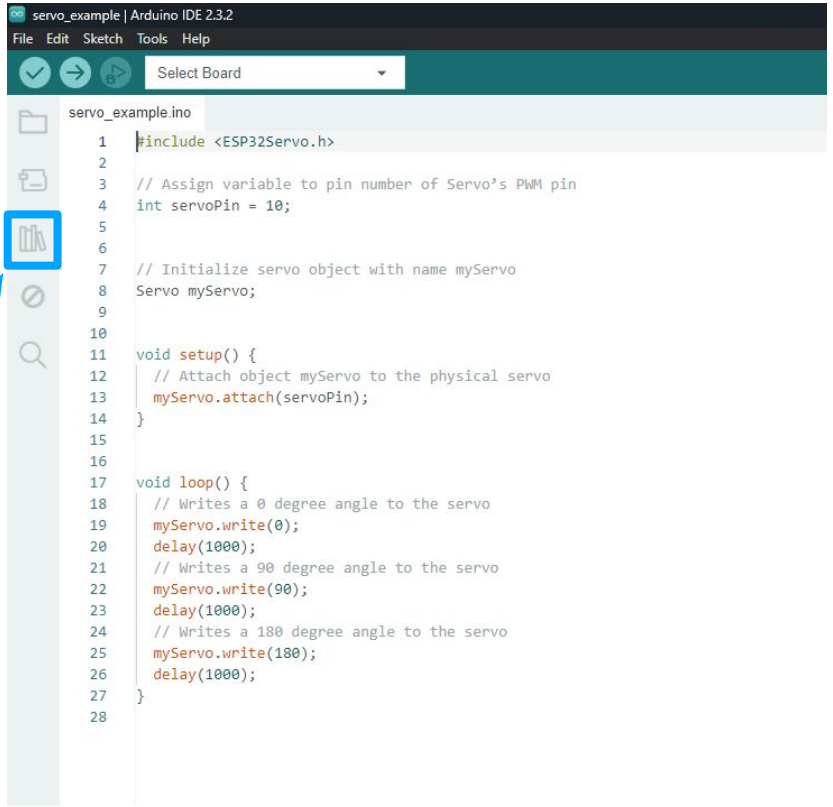


SECTION V

Arduino Libraries

Arduino Library

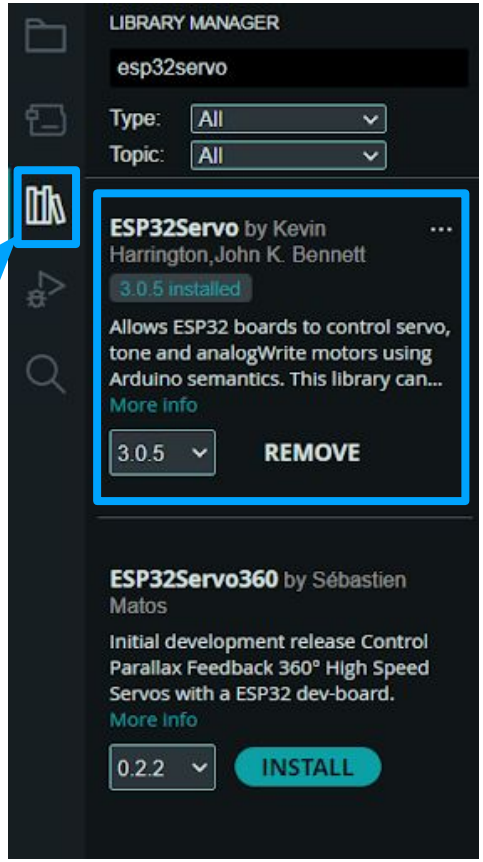
- A collection of pre-written code that makes it easier to interface with hardware or perform specific tasks
- Libraries made by Arduino's development team or Arduino community members
- Libraries can be downloaded from the library manager



Servo Library

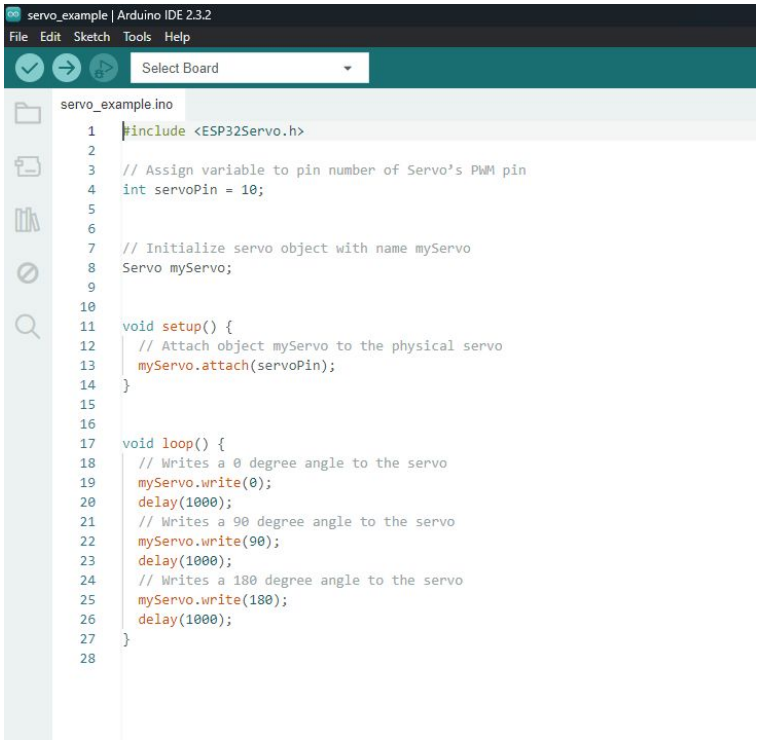
- The **ESP32Servo** library will need to be installed to control the Servo in Project 4

1. Click on the Library manager icon
2. Type in ESP32Servo
3. Download the first result by Kevin Harrington, John K. Bennet



Servo Library (Continued)

- Must #include <ESP32Servo.h> to use the library
- `Servo your_servo_name`
  - Initializes servo object with whatever you want to name the servo object
- `your_servo_name.attach(int pin)`
  - Links the servo object to physical pin `int pin`
- `your_servo_name.write(int angle)`
  - Moves the servo to a specified degree (0°-180°)

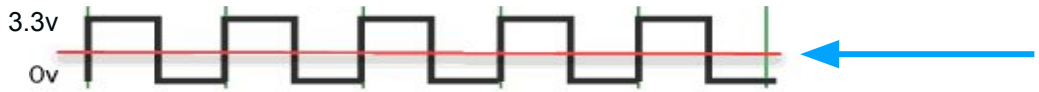


SECTION VI

ESP32 Concepts

Pulse Width Modulation

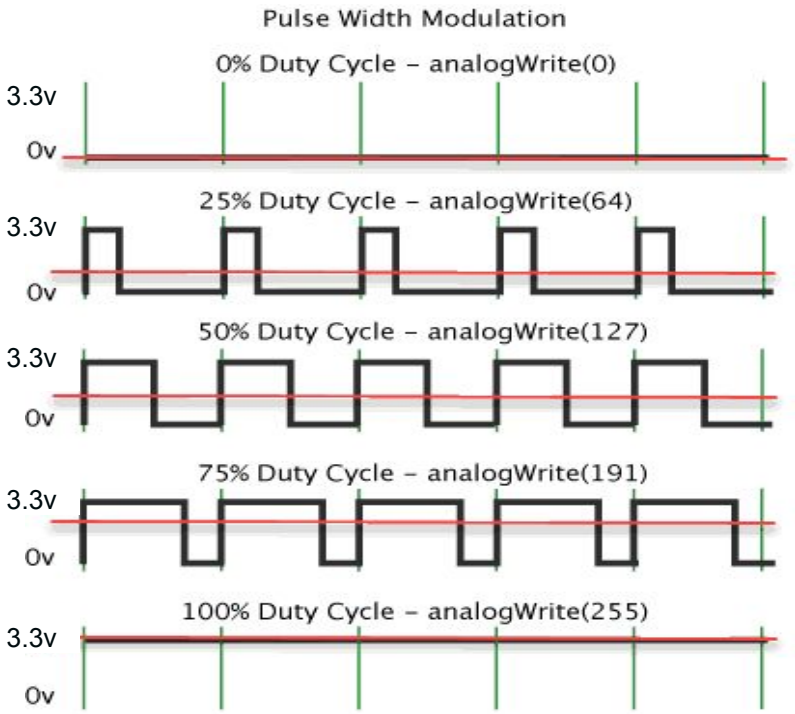
- The ESP32 board is a **digital source**, meaning it can only output a `HIGH` (3.3V) or `LOW` (0V) voltage
  - Then how does `analogWrite` output analog signals?
- **Pulse Width Modulation** (**PWM**) is an oscillating digital waveform that emulates an analog output
  - By oscillating a signal from `HIGH` to `LOW` quickly, the average voltage over time will be *between* `HIGH` and `LOW` - an analog value



The average value, the *analog value*, of this waveform is **1.65V**

PWM Duty Cycle

- The **duty cycle** of the PWM wave is the percentage of time where the signal is `HIGH`
  - For example, a 50% duty cycle translates to an average value of 1.65V (50% of 3.3V)
  - Allows us to output a continuous range of voltages between `HIGH` and `LOW`
  - Duty cycle is **controlled by a timer** inside the ESP32 microcontroller

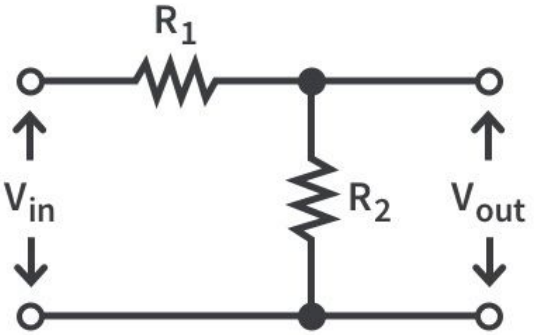


ADC and `analogRead`

- `analogRead` **utilizes the analog-to-digital converter** inside the microcontroller to **measure the real-world, analog signal** and **convert it to a digital signal**
    - The measurement resolution is 12 bits, which is why the function returns values from `0-4095`

Voltage Divider

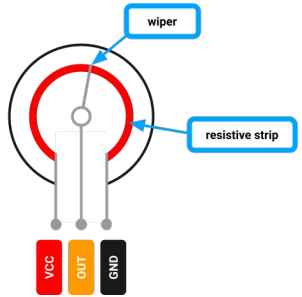
- A circuit configuration with **two resistors** that is used to scale down a voltage (**Vout**).
- Creates a ratio of two resistors to achieve a desired output voltage
  - **R1** and **R2** are connected in series
  - **Vin** is the input voltage
  - **Vout** is the output voltage across **R2**



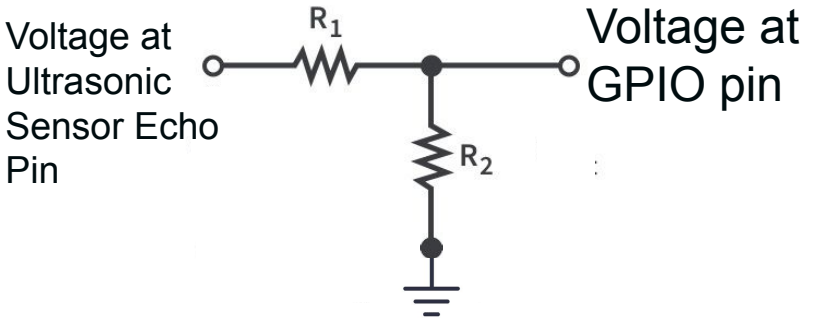
$$V_{out} = \left(\frac{R_2}{R_1 + R_2}\right) V_{in}$$

Voltage Divider in Project 4

- A **potentiometer** is an adjustable resistor divider
  - A wiper/slider divides the resistive element into two adjustable parts
- The Ultrasonic Sensor is rated for 5V, but the GPIO pins are rated for 3.3V. We need to reduce the voltage at the this pin
  - A Voltage divider solves this



$$V_{out} = \left(\frac{R_2}{R_1 + R_2}\right) V_{in}$$



SECTION VII

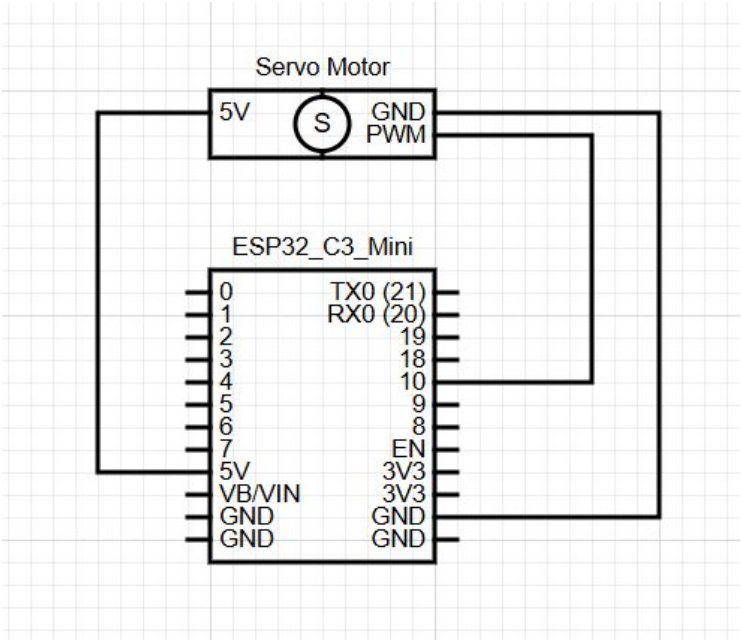
Servo Activity

Get into your groups!

- For this activity, you'll be doing it alongside your groups!
- Find your instructor, and then they will direct you where to sit
- If your instructor is not here today, come up to the front and we will merge you with another group!

Servo Activity

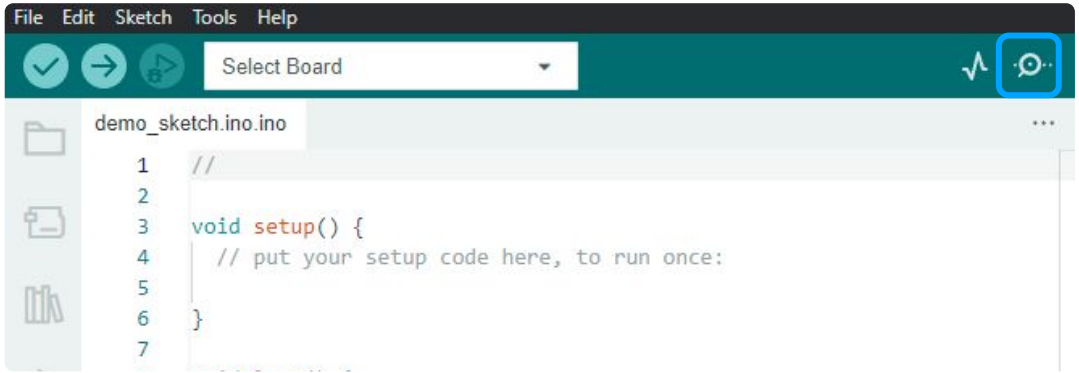
- We will create a circuit where you can set the servo's angle in code and have it turn there! Build the schematic on the right →
- Here are some hints:
  - Don't forget to include the **ESP32Servo** library in the code
  - Use the library functions to attach the servo and write to it
  - You can also add more write commands with delay between them to have more than one servo angle



SECTION VIII

Tips for Project 4

Using the Serial Monitor

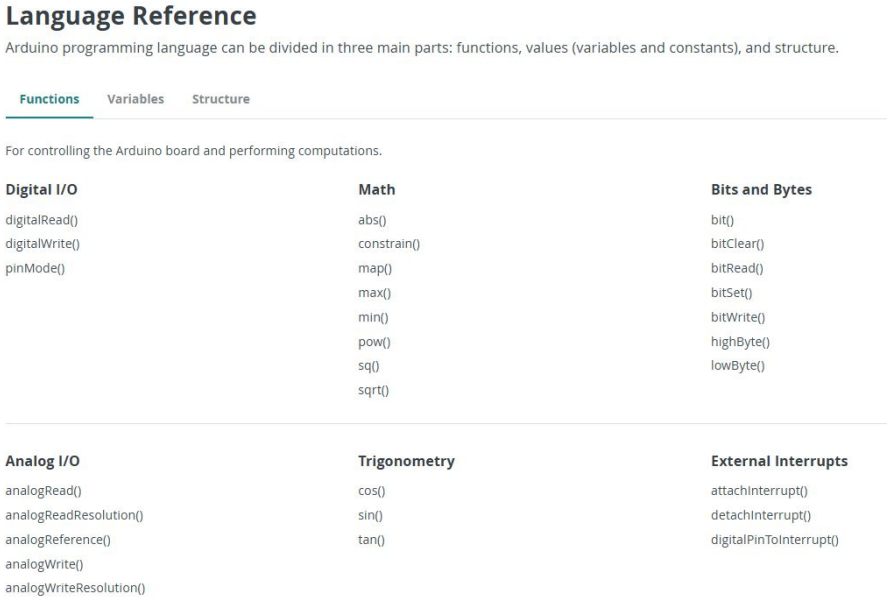


- `Serial.print` is an excellent tool to **help debug programs** and **tune sensors**
    - Print values to track the ultrasonic sensor's readings
        - Find the sensor's value at different distances

Arduino Functions

- Understand the purpose of each function in a code (Don’t just blindly copy and paste code!)

- Visit the Arduino Language Reference for an explanation of the most common Arduino functions



Language Reference

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

Functions | Variables | Structure

For controlling the Arduino board and performing computations.

Digital I/O

digitalRead()
digitalWrite()
pinMode()

Math

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

Bits and Bytes

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

Analog I/O

analogRead()
analogReadResolution()
analogReference()
analogWrite()
analogWriteResolution()

Trigonometry

cos()
sin()
tan()

External Interrupts

attachInterrupt()
detachInterrupt()
digitalPinToInterrupt()

Map Function

- Re-maps a number from one range to another
- `map(int inputValue, int min_range_1, int max_range_1, int min_range_2, int max_range_2)`
  - `min_range_1` and `max_range_1` are the min and max values of the first range, and `min_range_2` and `max_range_2` are the min and max values of the second range

Map function in Project 4 Checkpoint 1:

- You will need to map the range of the ADC (0-4095) to the servo range (0-180)

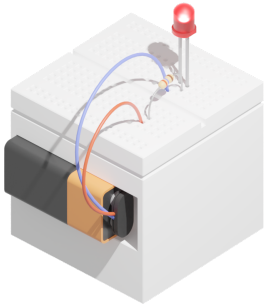
RGB LED Ex:

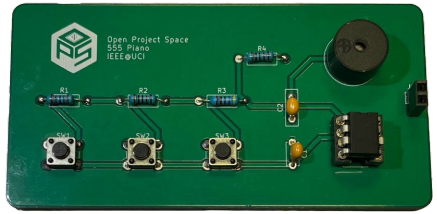
```
potValue = analogRead(potPin)

map(potValue, 0, 4095, 0, 255)
```

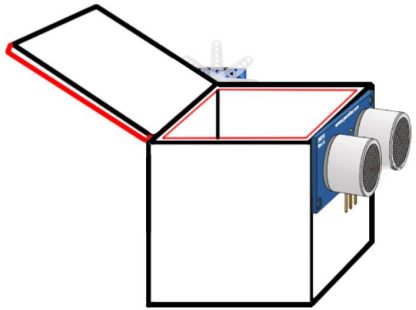
Fall Quarter Recap

- Completed 4 Projects in total!
  - LED there be Light
  - 555 Piano
  - LED RGB Wizard
  - Mini Trash Can





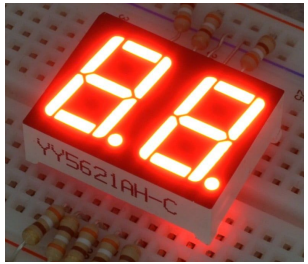
- Main Learning Concepts:
  - Ohm's Law
  - Breadboarding & Soldering
  - Basic Circuit Analysis
  - ESP32
  - Programming in Arduino IDE

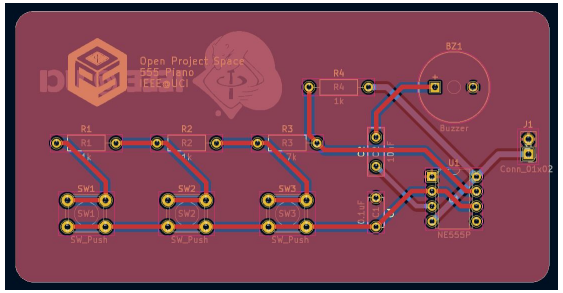


Winter Quarter

- 4 New Projects!
  - iPoduino 4.0
  - Weather Station
  - Digital Stopwatch
  - Capstone PCB Design
- **Challenge Project! More details to come** 👀

- Main Learning Concepts:
  - Communication Protocols (UART, SPI, I^2C)
  - Interrupts, Timers
  - WiFi/Radio
  - PCB Design





T Shirt Contest

- Submit a Tshirt Design via Custom Ink
- If your Design is selected, you will receive a **prize**!!!
- Look out for a Canvas announcement soon for submission requirements

Last Year's design

FAIR USE DISCLAIMER

Copyright Disclaimer under section 107 of the Copyright Act 1976, allowance is made for “fair use” for purposes such as criticism, comment, news reporting, teaching, scholarship, education and research.

Fair use is a use permitted by copyright statute that might otherwise be infringing.

Non-profit, educational or personal use tips the balance in favor of fair use.

CC BY-NC-SA 4.0