
@SOMEPUNKART
MAD SCIENCE 101
BUILD YOUR OWN
LASER!

Mad Science 101: Mechatronics for World Domination!

Mechatronics = Electronics

+ Mechanics

+ Programming

You will Learn:

- Basic Electronics: Circuits, Components, Soldering
- Controlling LEDs, Lasers, and Servos!
- Programming Microcontrollers!
- Build a Laser Turret to defend your Secret Base!

Mad Science 101: Mechatronics for World Domination!

Schedule:

Date	Subject	Project
10/4/25	Electronics	Blinking LED Badge
10/11/25	Mechanics	Laser Tower
10/18/25	Programming	Laser Tower

Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Project: Blinking LED Badge

You will Learn:

- Basic Electronics:
  - Circuits
  - Components
  - Soldering

Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

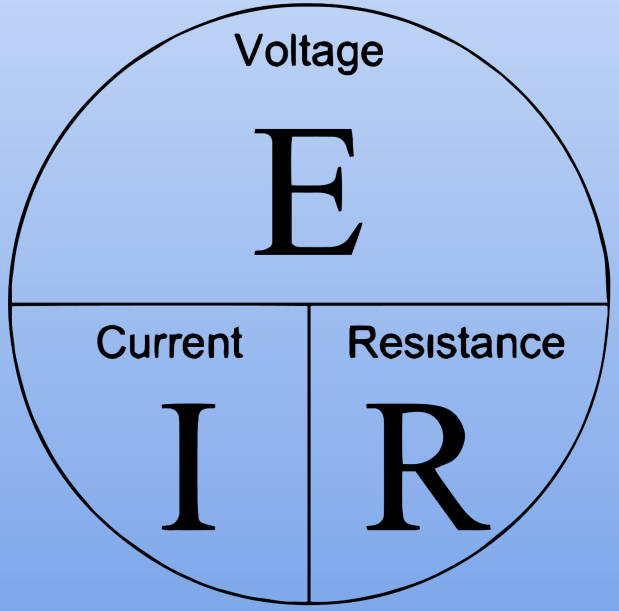
Basic Electronics

Convenient Fiction: Electricity goes from Positive to Negative

Electricity has Voltage (amount), Current (speed), and Resistance (load).

They are related by Ohm's Law: V=C*R: Voltage = Current * Resistance.

Note: ***You do not need to remember this!***



Ohm's Law Circle

$E = I \times R$

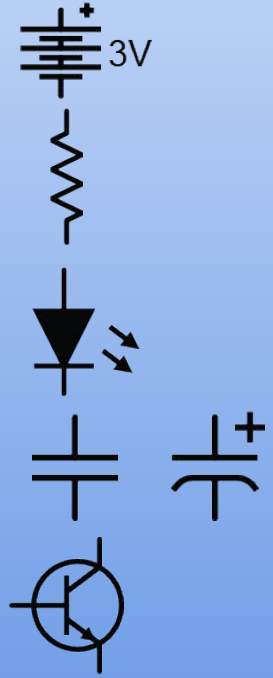
$I = E / R$

$R = E / I$

Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Components and their symbols:



Battery (provides power)

Resistor (bleeds off power)

LED (Light Emitting Diode)

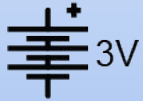
Capacitor (a thing that oscillates)

Transistor (a switch)

Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

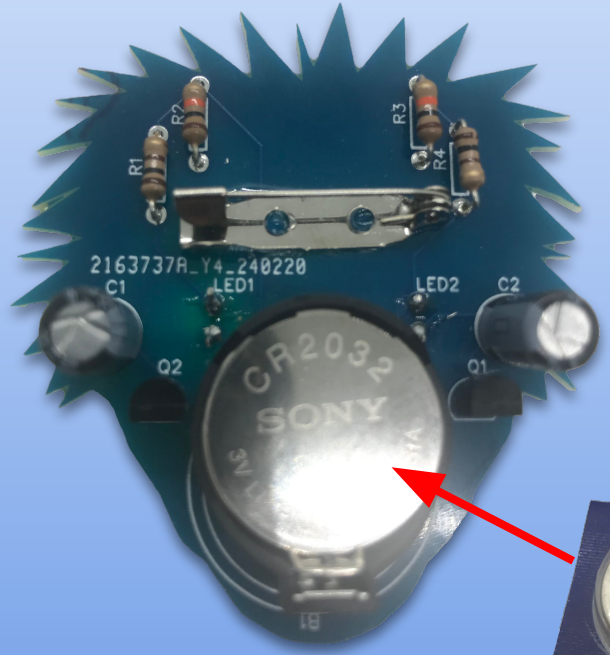
Batteries:



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

We will use a coin cell, 3 Volt CR2032 battery and will solder a battery holder onto the board.

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



Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

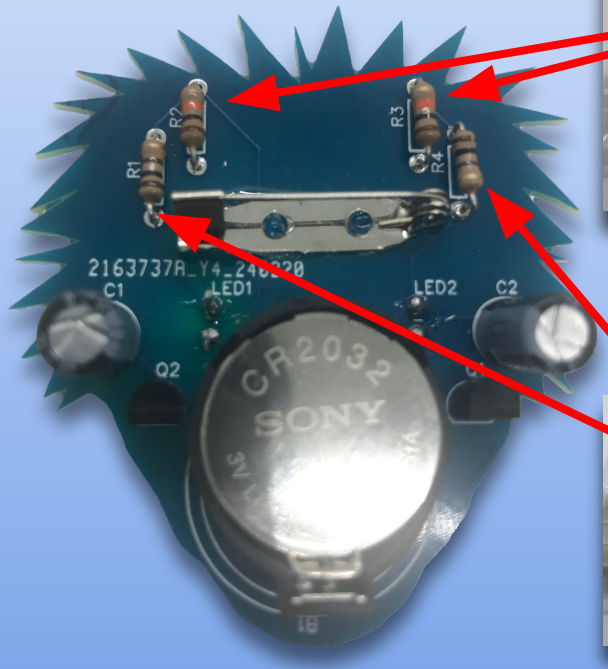
Resistors: 

Resistors decrease (resist) current or voltage. They dissipate the energy in the form of heat.

Resistance is measured in Ohms, which is denoted Ω (Greek Omega)

Resistors are NOT polarized (Doesn't matter which way you solder them in)

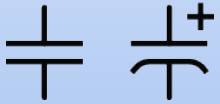
We use two different kinds of resistors: R2 and R3 are 100Ω
R1 and R4 are 10KΩ (10,000Ω)



Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Capacitors

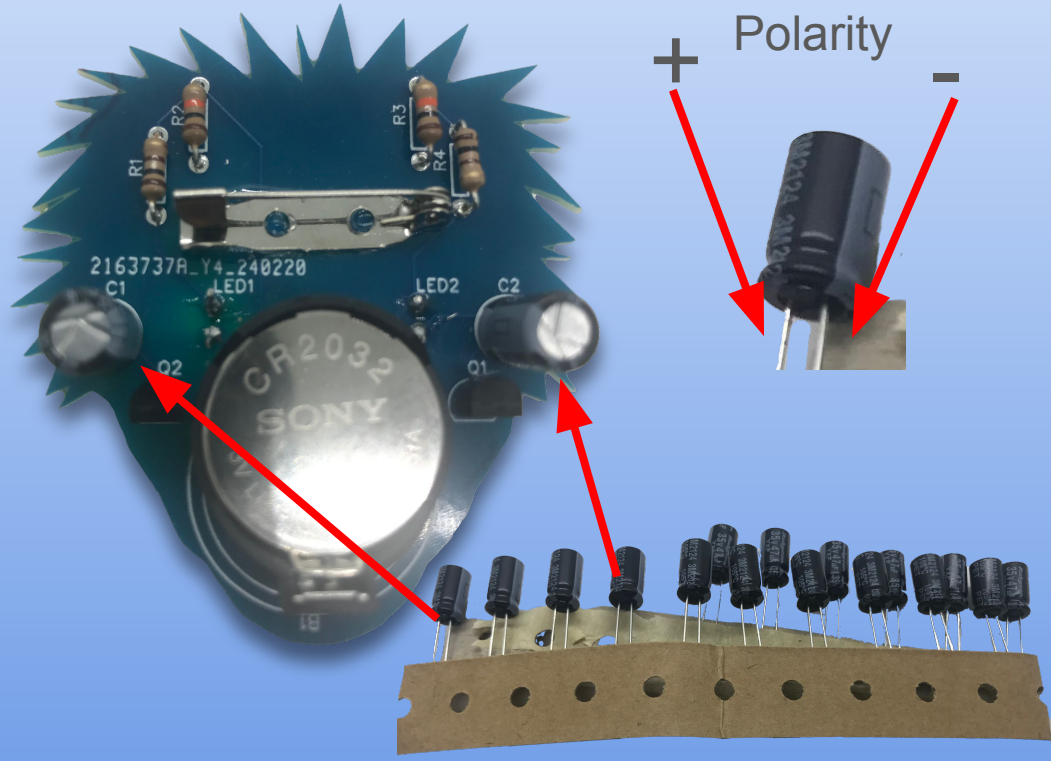


Capacitors provide an oscillation to the circuit. The electric field builds up and then jumps across.

Capacitors are rated in Farads, usually small, like µF (Micro Farads)

Capacitors are SOMETIMES polarized, and the ones we are using ARE polarized. The side with grey is negative.

We use two 47µF Capacitors



Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Transistors

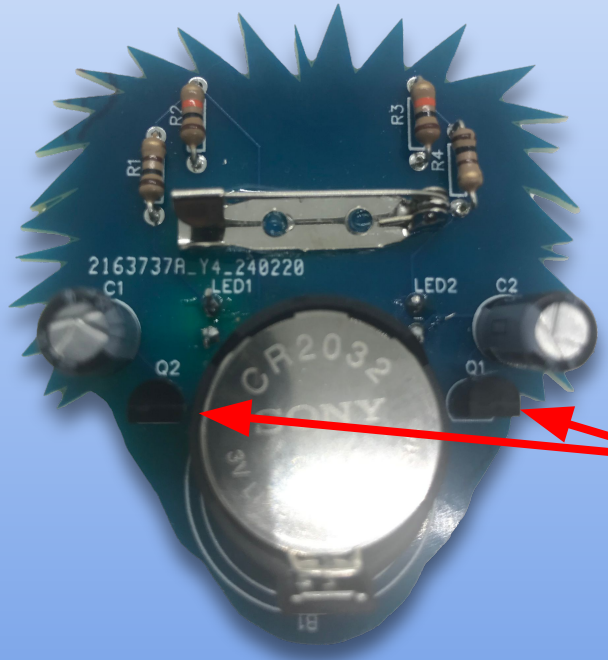
Transistors are switches that allow or deny the flow of current.

Transistors are Semiconductors, meaning that they only conduct sometimes.

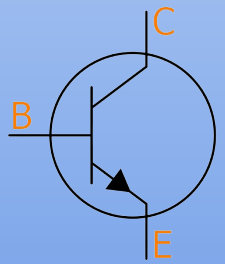
They have three pins: Base, Collector, and Emitter.

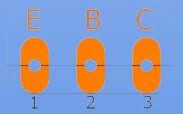
If electricity is applied to the base, then current will flow from collector to emitter.

The ones we are using are 2n3904 - NPN



2n3904





2N3904-NPN	
1	EMISOR
2	BASE
3	COLECTOR

Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

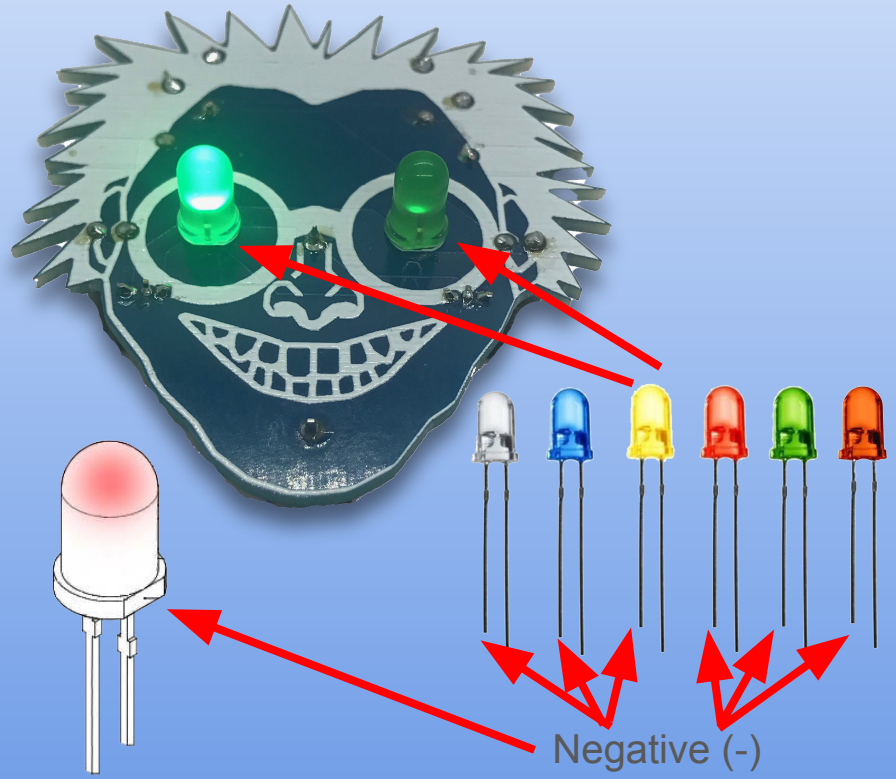
LEDs (Light Emitting Diodes)

LEDs are lights that are made from diodes

Diodes are semiconductors that only allow current to flow in one direction

LEDs come in many colors, and even some that can have programmable colors.

Diodes, including LEDs ARE Polarized. The shorter leg is negative (-). Also, the negative side is indicated by the flat side of the LED.



Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

PCBs

PCBs, or Printed Circuit Boards, are boards that have conducting and non-conducting surfaces. Circuits are etched into them so that they act as wires between components.

PCBs have multiple sides, and you can (sometimes) solder your components into either side.

Ours is *Through-Hole*, meaning that the components have wires that you solder into the board.

Mad Science 101: Mechatronics for World Domination!

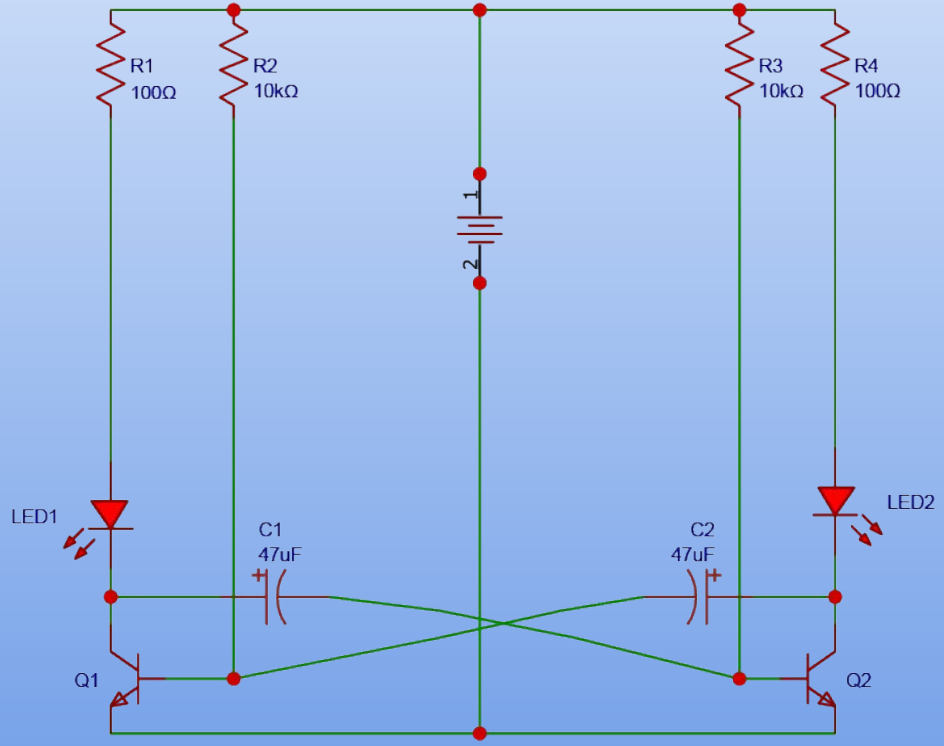
Part 1: Electronics for Mad Scientists

Circuits:

Circuits are components connected together so that the electricity flows from Positive (+) to Negative (-).

Our circuit is called an *Astable Multivibrator* and is a common way to make things oscillate.

Formula for period:
F=1/T=1/1.38*R*C
[Astable Multivibrator Simulator]
[Astable Multivibrator Tutorial]



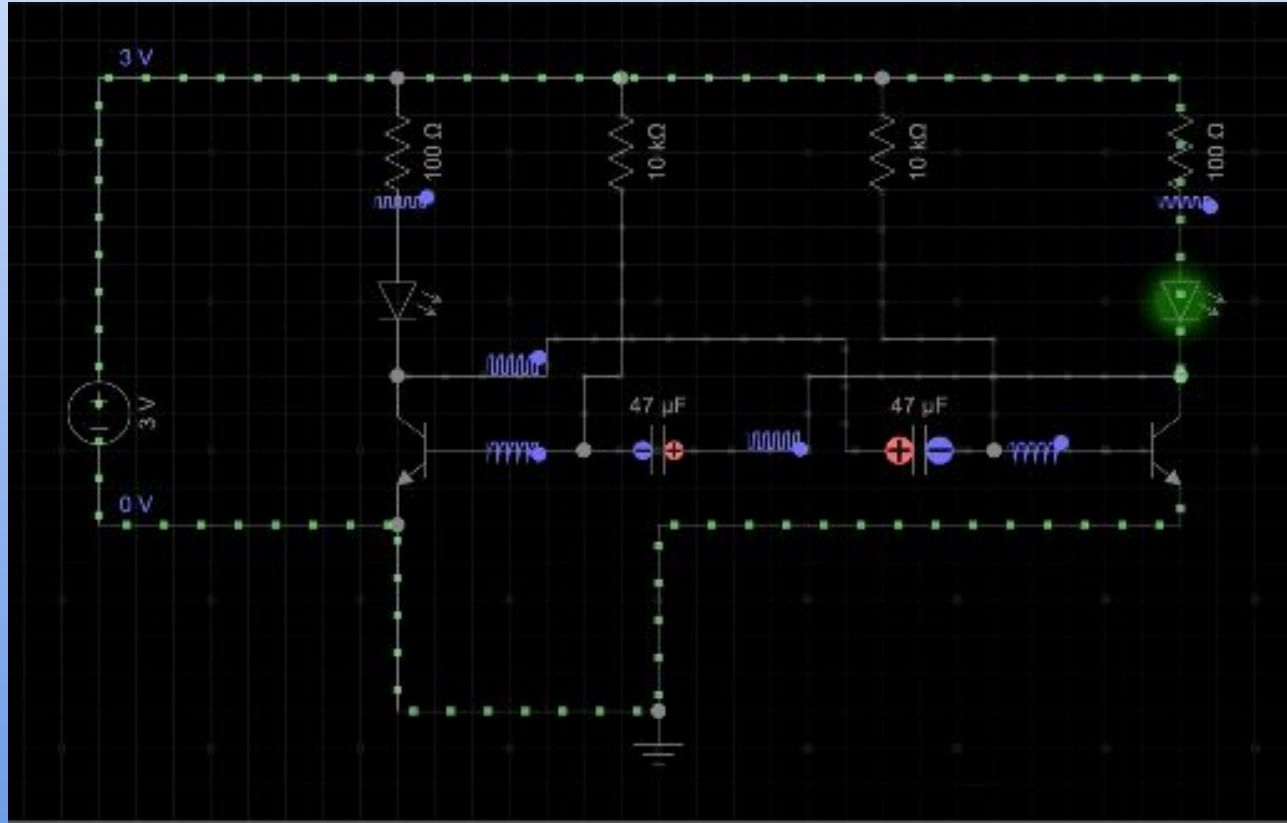
Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

How it works:

The oscillation of the capacitors cause the base of the transistors to turn on and off periodically, which causes current to flow, lighting the LED for a while, and eventually triggering the other transistor to turn on, turning off the first transistor.

Larger values of middle resistors will slow down the oscillation.



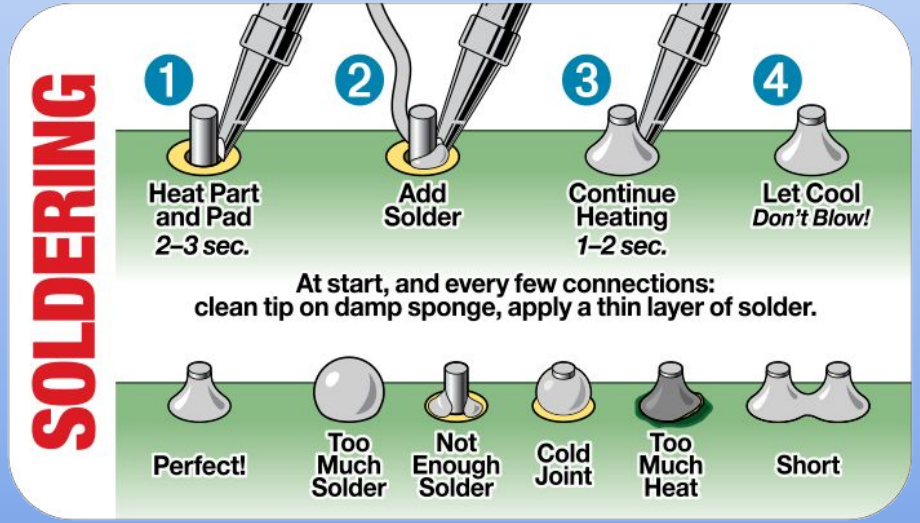
Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Enough Talk! Let's Get Making!

Soldering: Solder is usually a metal with a low melting point. Melting the solder over the wire and the PCB allows current to flow between the two.

A. Place the component in the hole and flip over. You may need to bend the leads of the component so it stays in the hole.
B. Place the tip of the soldering iron on the lead touching the PCB surface
C. Place the solder against both the wire and the PCB
D. Once it melts, bring solder and iron tip away from board and let it cool.
E. Be careful not to connect two or more pins together. This creates a short.



Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Enough Talk! Let's Get Making!

Project 1 - Blinky Lights: The goal of the project is to solder together a blinky Mad Scientist badge.

Components Needed
(AKA BOM - Bill of Materials):

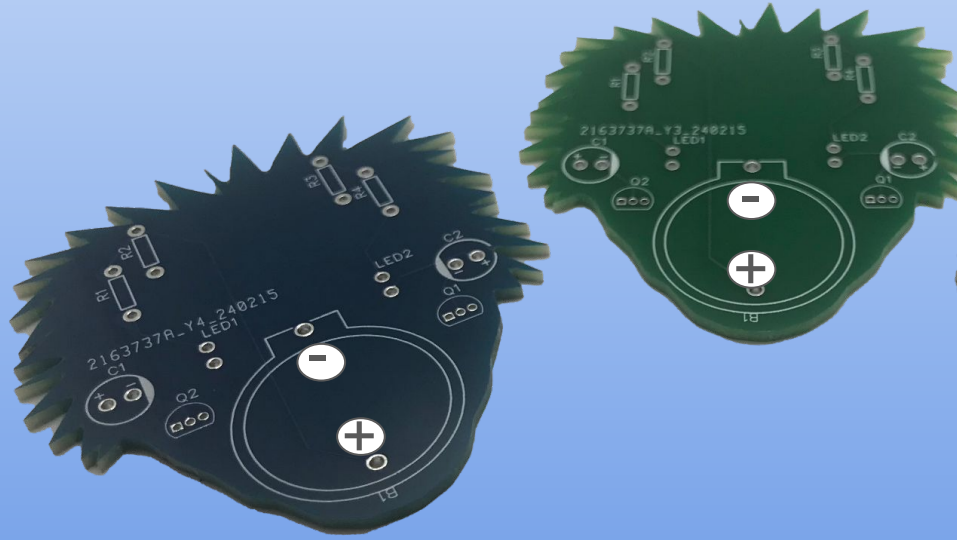
- 1 PCB (any color)
- 2x LEDs (any color)
- 2x 47µF Capacitors
- 2x 100Ω Resistors
- 2x 10KΩ Resistors (or bigger for slower)
- 2x 2n3904 Transistors
- 1x CR2032 Battery
- 1x CR2032 Battery Holder

Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Enough Talk! Let's Get Making!

Project 1 Important Note:
The green and blue PCBs have the battery connector reversed. (SORRY!!)
This means that you will need to solder in the battery connector upside down! I will mark them on the PCBs.



Mad Science 101: Mechatronics for World Domination!

Part 1: Electronics for Mad Scientists

Summary

You have Learned:

- Basic Electronics:
    - Circuits
    - Components
    - Soldering

You have built:

- Blinking LED Badge

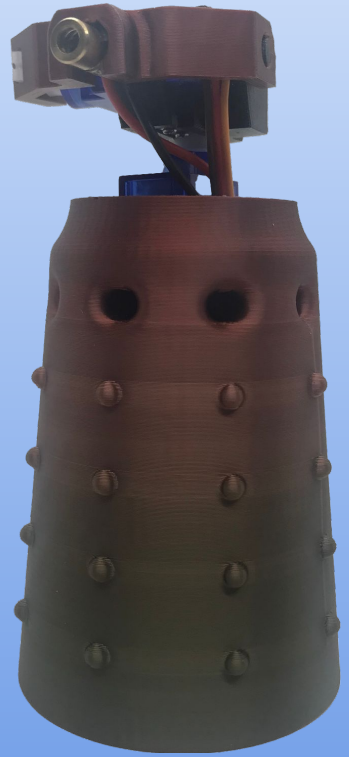
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Project: Laser Sentry Tower

You will Learn:

- Lasers
- Servos and Motors
- Microcontrollers
- Assembly of Components
- Soldering Part 2
  - Wire Stripping
  - Heat Shrink
  - Wire Tinning
  - Wire Splicing

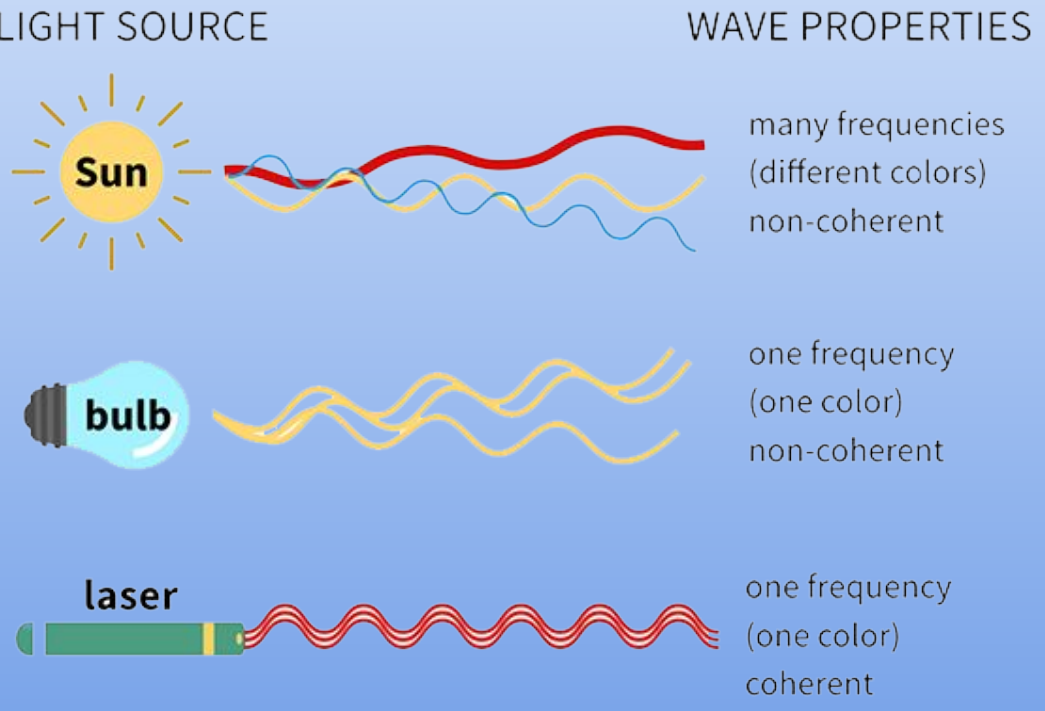


Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Lasers

- Light happens when atoms vibrate faster and then slow down, releasing energy.
- Lasers are when all the atoms release exactly the same amount of their energy into light at exactly the same time.
- Single wavelength/frequency
- Same Phase
- High Amplitude = High Power



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

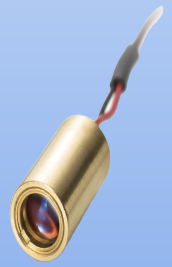
Lasers

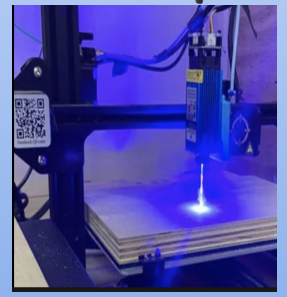
Lasers are categorized by Type, Wavelength, and Wattage.

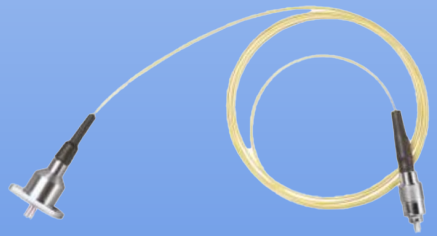
Typical (engraver) wattage varies from 5W-120W

Wavelength can be red, green, blue, UV, IR, etc.

- CO2 Lasers: For engraving Organic materials, e.g. Wood and Acrylic. 10,600nm wavelength (IR)
- Diode Lasers: For marking on anything, but slow and typically low power. 1000nm (IR) to 455nm (the very low end of the visible light spectrum, near UV)
- Fiber Lasers: For marking on Metals 1064nm (IR)







Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Lasers

Powerful Lasers:

Smallest Laser weapon: 10 kW (enough to power 10 houses)

Largest Laser Weapon: 300 kW (enough to power 300 houses)

Most powerful laser (for fusion experiments): 10 PW (10 PetaWatts = 10 million billion Watts)

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Lasers

Our Lasers:

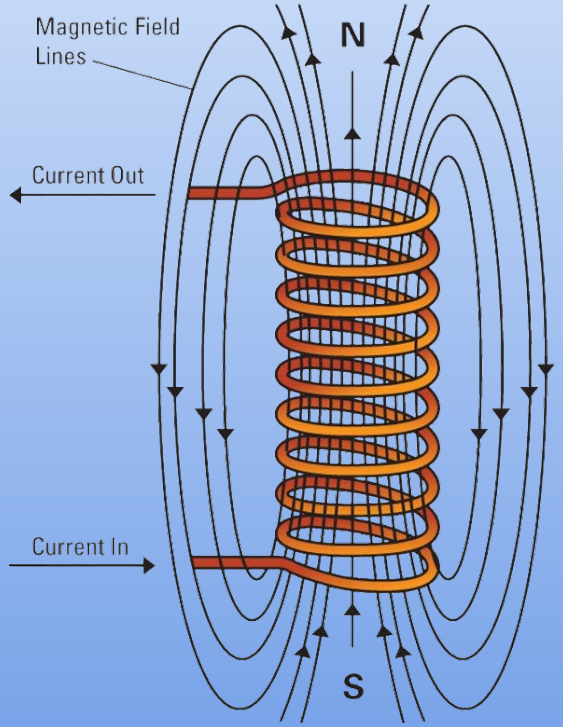
- Diode Laser Module
- 3 V
- Power: 5 mW (0.005 Watts)
- Wavelength: 650 nm (0.000000650 Meters)
    - Red Visible Light!
- About Laser Diodes

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Electromagnets

- Sending current through a wire produces a small magnetic field.
- Twisting the wire into a coil, creates a much stronger field.
- Add an iron core, and you have a usable magnet.
- You can observe this with just some wire, a battery, and a nail.

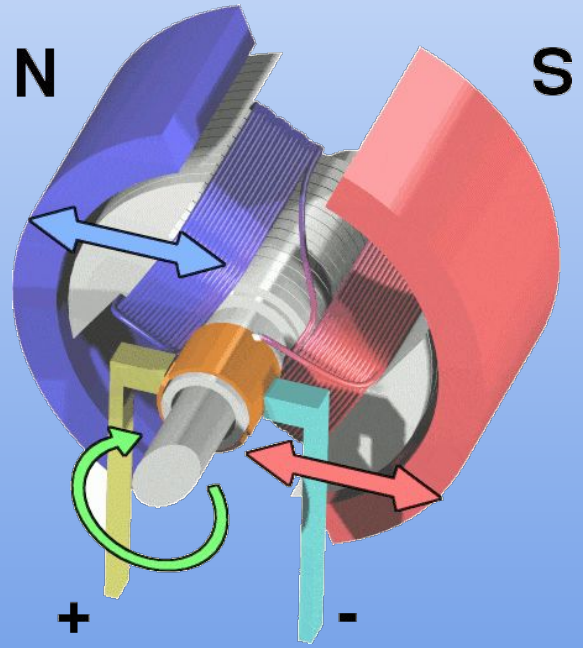


Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Motors

Motors move because magnetic fields rotate in relation to magnets!



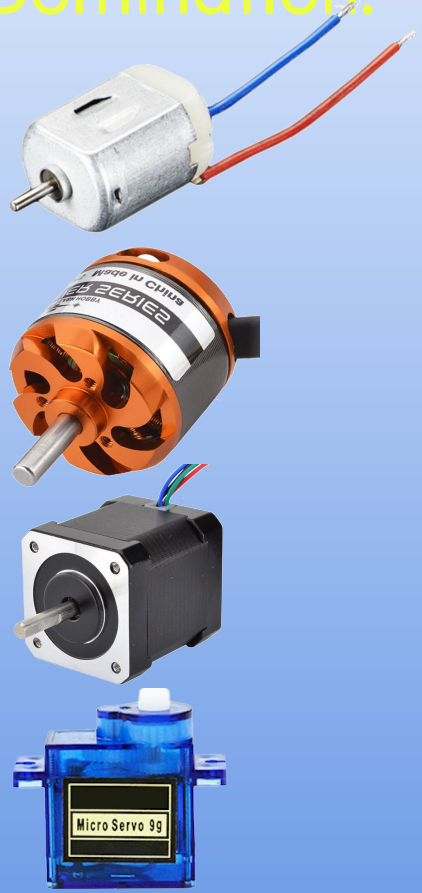
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Motors

Types of Motors:

- Brushed DC Motors
  - Simplest motor with direct contact of spinning part
- Brushless DC Motors
  - Like an AC motor but with a DC driver and feedback mechanism to control speed
- Stepper Motors
  - A brushless dc motor with precise control of motion
- Servos
  - Ours is a brushed dc motor with gears



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Motors

Our Motors

- SG90: common, low-torque servo
- 9 Gram Servos - 9g weight
- 1 kg torque or power
- Brushed motor with gears to control torque and speed
- 0° to 180° (Half of a Circle)
- Three Wires:
  - Red: +, 3.3V
  - Brown/Black: -. 0V (ground)
  - Yellow/Orange: Signal - pulses from 0V to 3.3V
- Signal is a PWM (Pulse Width Modulation) signal

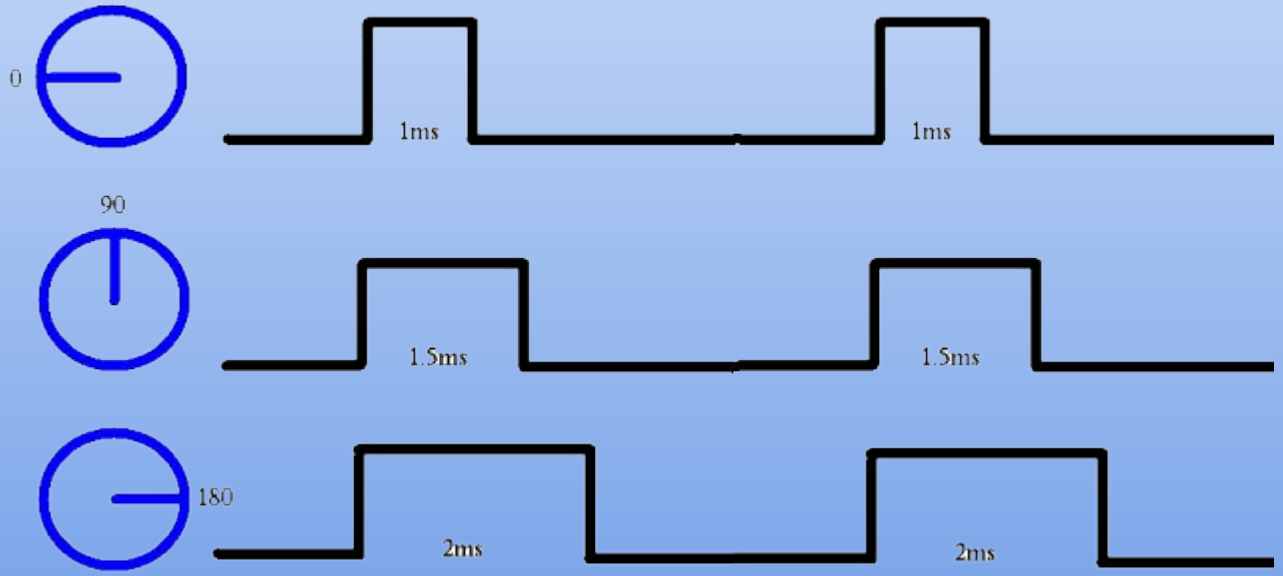
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Motors

Pulse Width Modulation (PWM)

- Off (0V), then On (3.3V) for a short time, then Off again
- Servo moves based on how long the pulse is on



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Microcontrollers

Microcontrollers (MCUs) are tiny computers that typically run a single, very simple program.

The earliest personal computers are many times less powerful than the microcontroller we are using!

Pins: There are often connectors on the sides of the MCU boards that have things like power, ground, and Input/Output

Digital IO: Microcontrollers typically have Digital Input/output connections and some other special connection types to connect to other devices (Peripherals).

Power and Programming: There are power connectors in the pins, but also they can be both powered and programmed from the USB connector

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Microcontrollers

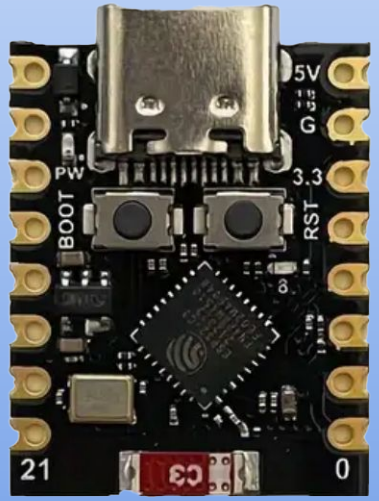
Our microcontroller:

ESP32-C3: an awesome microcontroller made by Espressif: 160mHz speed, single core, with WiFi and Bluetooth built-in, and 11 Digital IO pins

32-bits: each instruction is 32 bits. There are 8 bits in a byte. A byte is enough to store a number from 1-256 or a single english letter - so 32 bits is the equivalent of 4 english letters.

Instruction set: Microcontrollers have different native languages for programming them. Ours is called RISC-V, a very new, low-power instruction set.

Memory: 400 kB RAM, 384 kB ROM, 1-4 MB Flash

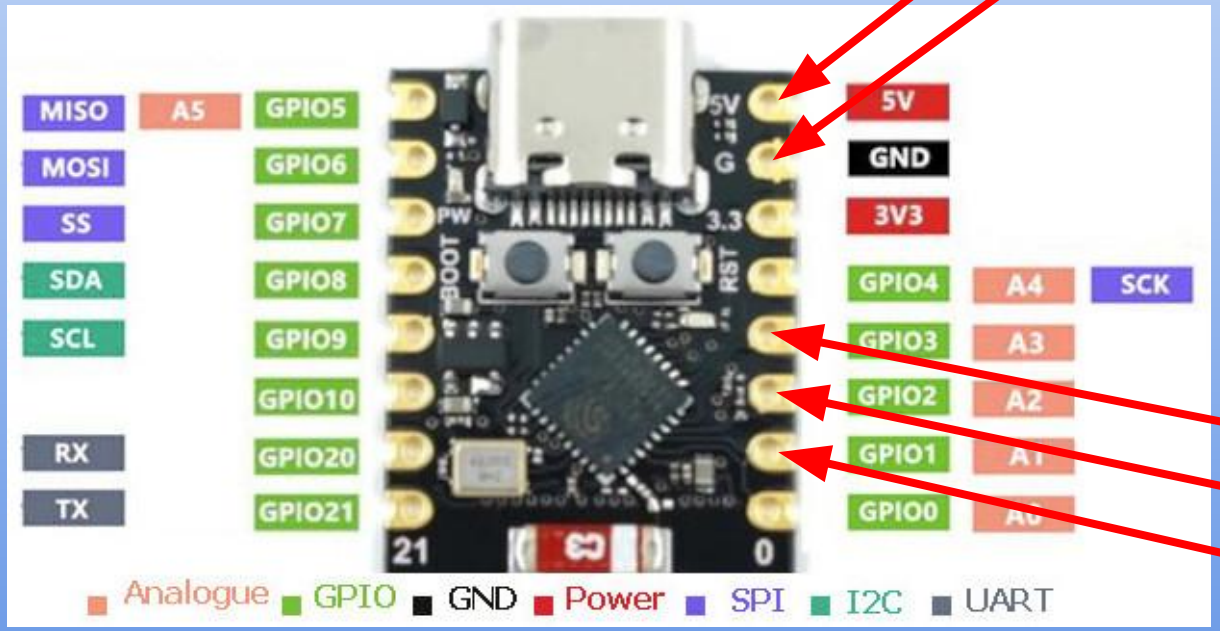


Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Microcontrollers

Our ESP32-C3 Connection Map:

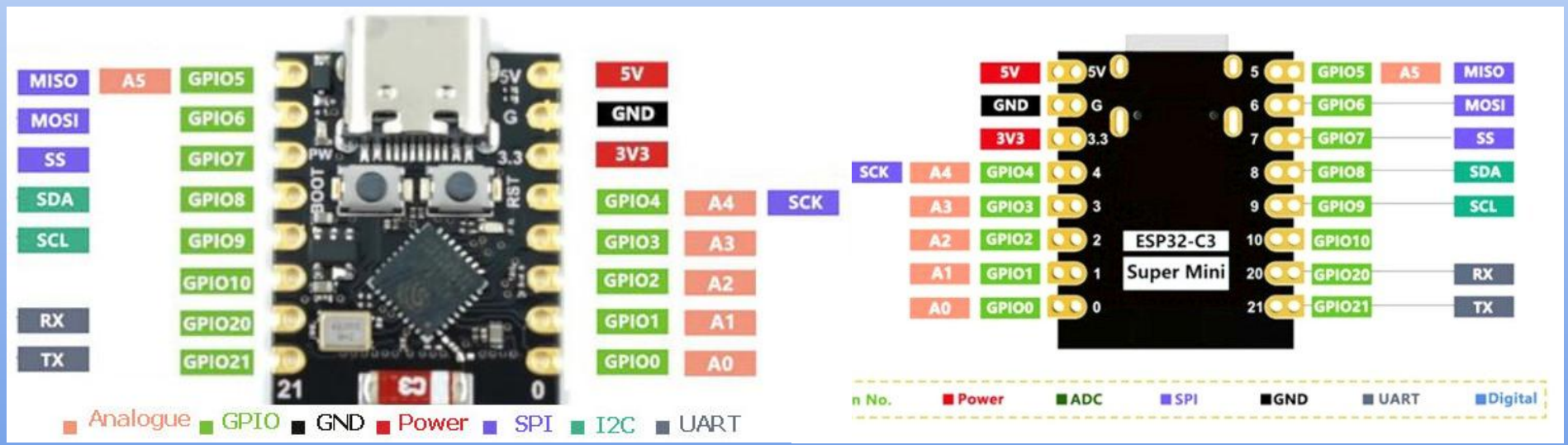


Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Microcontrollers

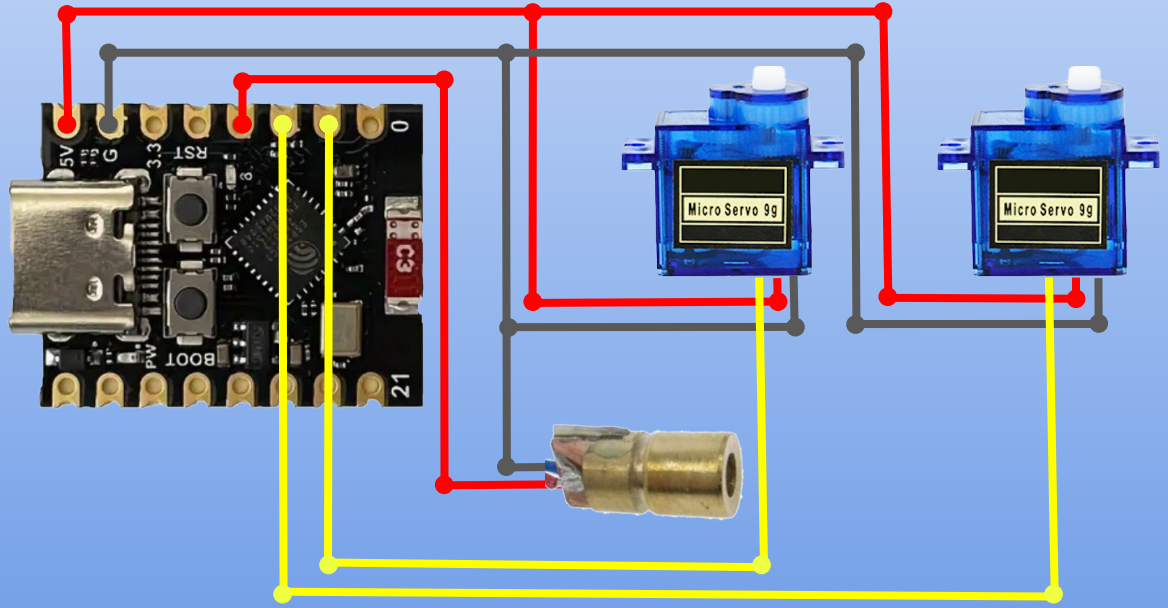
Our ESP32-C3 Pinout:



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Wiring Diagram



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

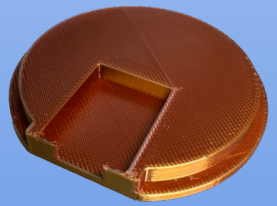
BOM (Bill of Materials):

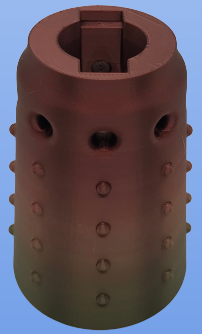
Parts:

- SG90 Servos (X2)
- Laser Module
- ESP32-C3 Microcontroller Board

3D Printed Parts:

- 3D Printed Turret Base
- 3D Printed Turret Base Bottom
- 3D Printed Vertical Servo Platform
- 3D Printed Laser Holder





Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Full BOM (Bill of Materials):

Parts:

- SG90 Servos (X2)
- Laser Module
- ESP32-C3 Microcontroller Board
- Power Plug
- USB -> USBC Power Cord
- 3D Printed Turret Base
- 3D Printed Turret Base Bottom
- 3D Printed Laser Holder
- 3D Printed Vertical Servo Platform

Misc Assembly Items

- 18” red silicone wire
- 18” black silicone wire
- 3” red heat shrink tubing
- 3” black heat shrink tubing

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Wire Stripping: Remove outer sheath from wire. Usually about 6mm (¼")

Can use:

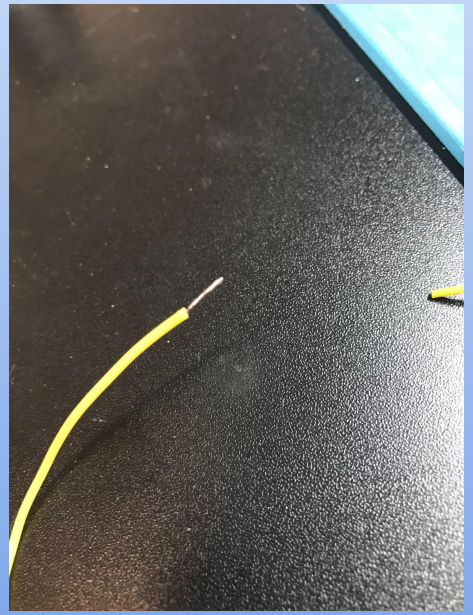
- Wire Strippers
- Fingernails (easy for the softer, silicone wire)
- Knife or side cutters:
    - Be careful - just cut the outside of the sheath, not the wires

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Wire Tinning: Melt solder into the end of a wire to make it solid

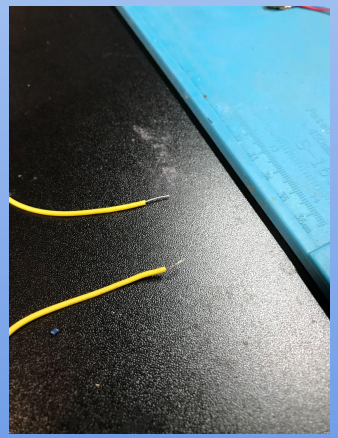
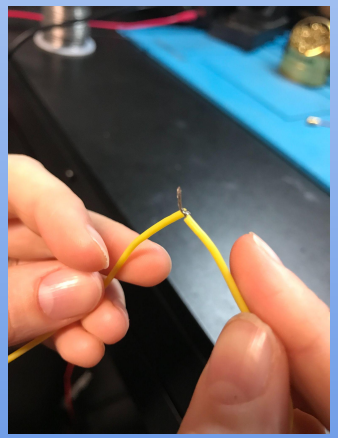
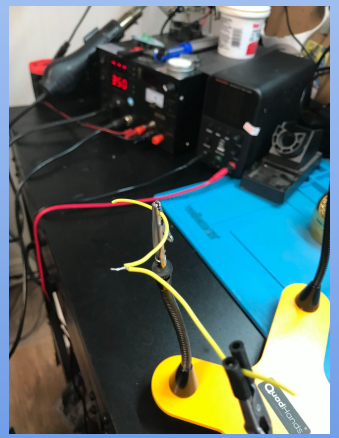
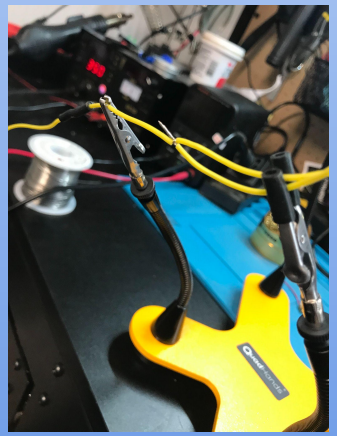
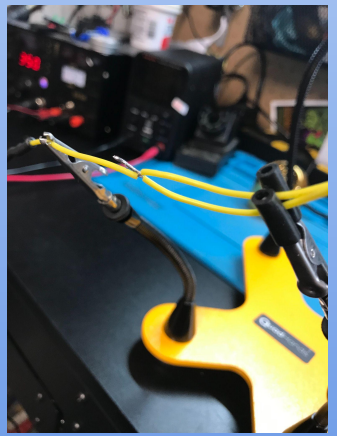


Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Wire Splicing: Twist wires together, and then melt solder through the joint. Snip off spikey bits, and then cover with heat shrink tubing.

Mad Science 101: Mechatronics for World Domination!

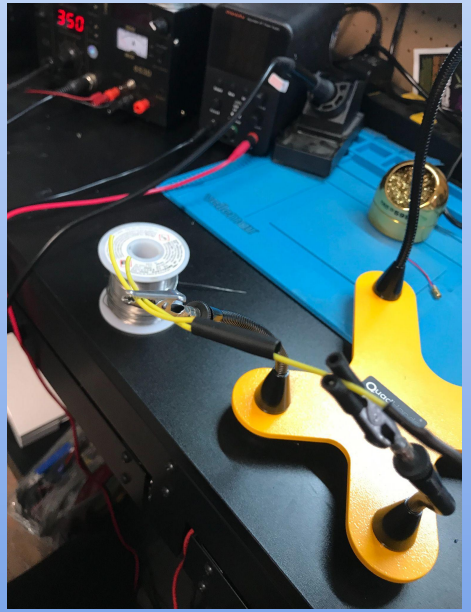
Part 2: Mechanical Control

Enough Talk, let's get Making!

Heat Shrinking: Cover wires in heat shrink tubing and apply heat

Can use:

- Heat Gun
- Lighter
- Butane Torch



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Overview:

1. Make 3 Splices and put heat shrink over them ← this is the hardest part of the project!
2. Tin 5 wires
3. Solder 5 wires onto Microcontroller Board
4. Put servos and laser into 3d-printed housing to form servo assembly
5. Put servo assembly into top of tower-base

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 1: Prep Wires

1. Cut Black silicone wires into 8" lengths (you should have 2 of them)
2. Cut Red silicone wires into 8" lengths (you should have 2 of them)
3. Cut servo wires near the plug (so they are long) and separate them from each other
4. Strip all wires to about 3mm

Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 2: Prep Laser Module

1. Replace wires on Laser Module with longer silicone wires.
   a. Original **Red** -> Silicone **Red** (8")
   b. Original **Blue** -> Silicone **Black** (8")
2. Use a Hot Glue Gun to put glue onto the soldered end of laser module to keep wires steady

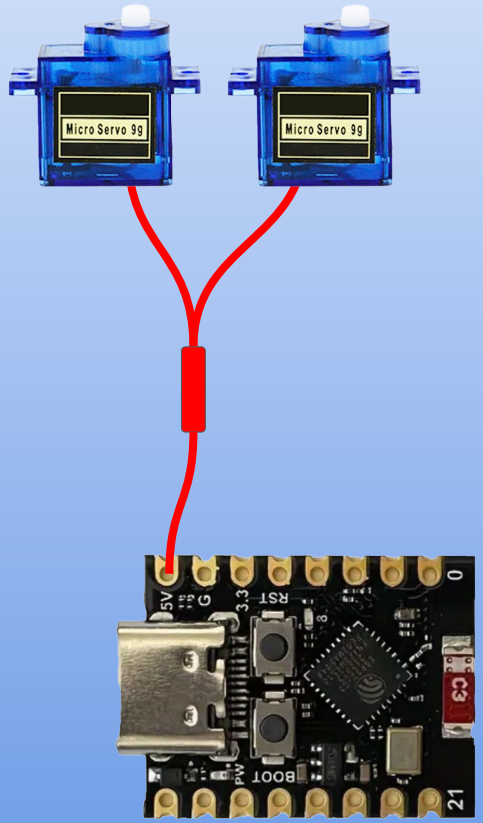
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 3: Servo Power

1. Splice red wires from each servo together with a single red silicone wire
2. Cover with Heat Shrink
3. Tin the other end of the red silicone wire
4. Solder end of red silicone wire from servos into 5V pin of MCU



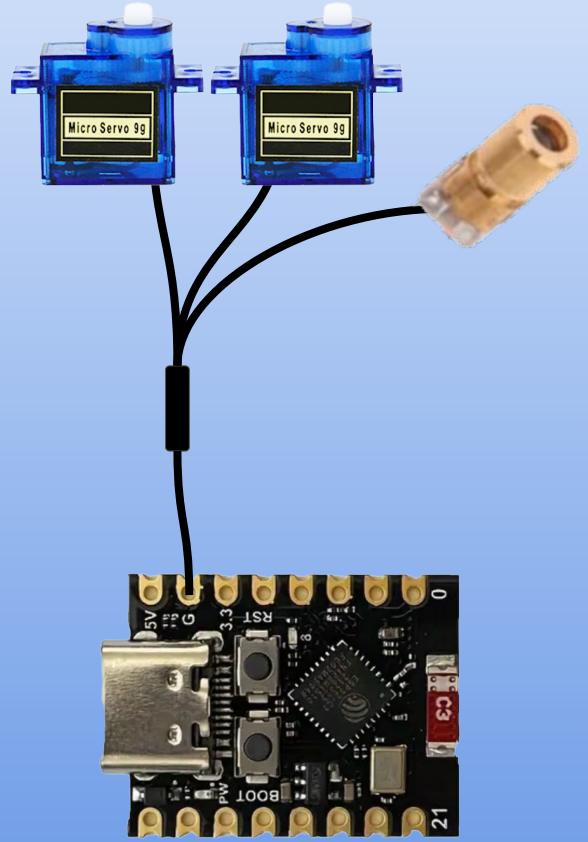
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 4: Servo and Laser Ground

1. Splice brown wires from each servo together with a single black silicone wire and the black lead from the laser
2. Cover with Heat Shrink
3. Tin the other end of the black silicone wire
4. Solder end of black silicone wire into G pin of MCU



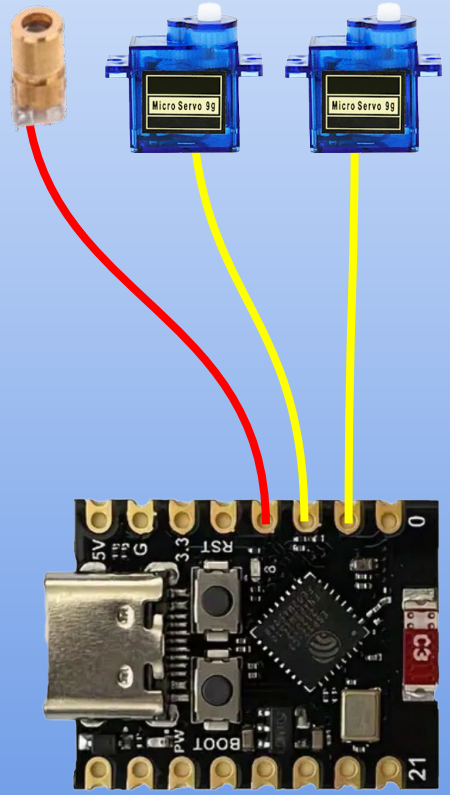
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 5: Solder to MCU

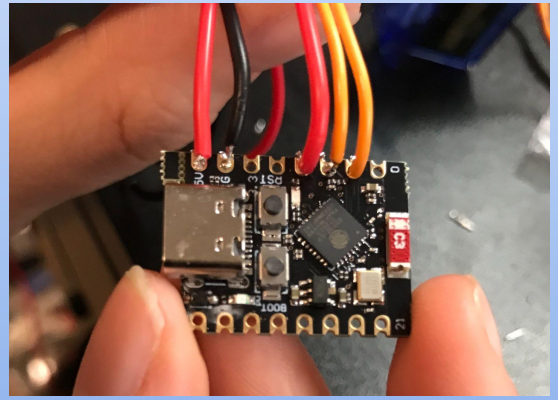
1. Tin the yellow servo wires and red laser wire
2. Solder end of yellow servo 1 into pin 1 of MCU
3. Solder end of yellow servo 1 into pin 2 of MCU
4. Solder end of red silicone wire from Laser into pin 3 of MCU



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!



Steps to assemble the Turret Step 5: Solder to MCU

1. Trim off ends of excess wires
2. Take project to me to get it programmed!

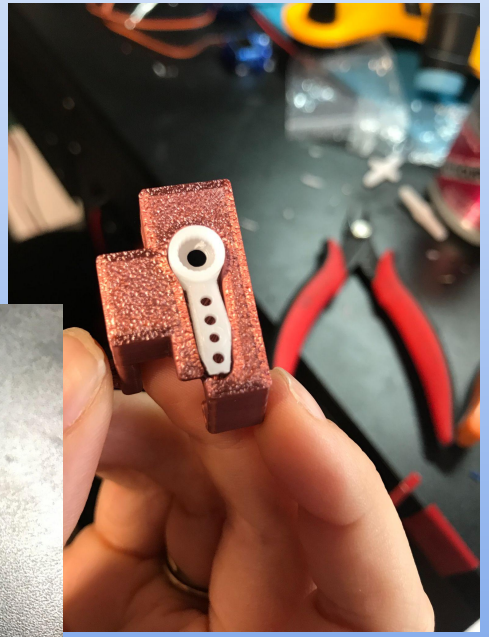
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 6: Vertical Servo Platform

1. Attach servo arm to vertical servo platform
2. Attach vertical servo platform to vertical servo
3. Attach servo arm to horizontal servo



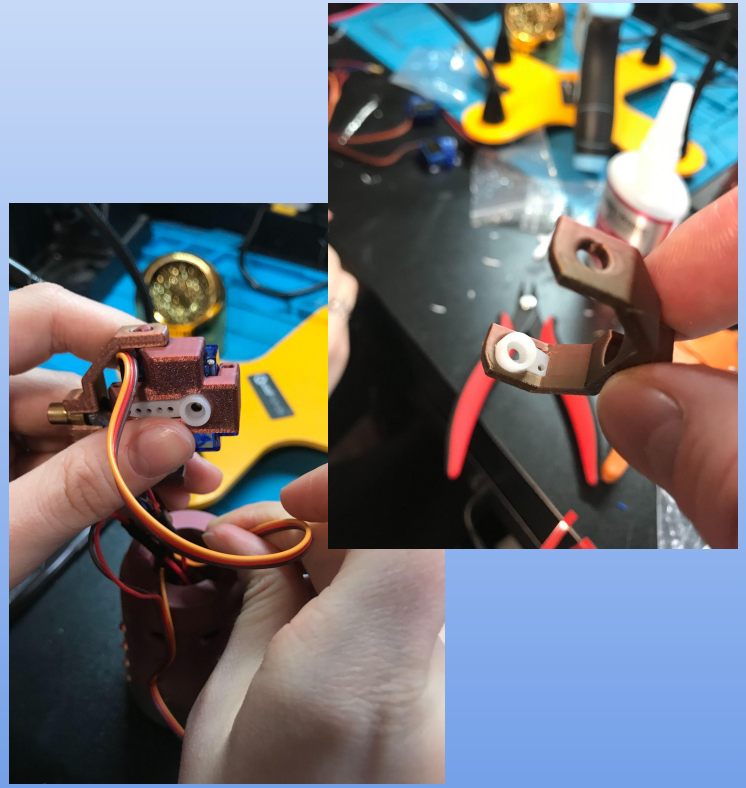
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 7: Laser Holder

1. Attach Laser Diode into Laser Holder with double sided tape
2. Attach servo arm to Laser Holder
3. Attach servo arm to vertical servo and vertical servo platform



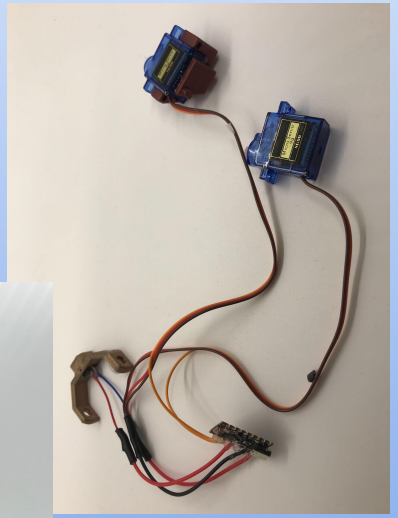
Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Enough Talk, let's get Making!

Steps to assemble the Turret Step 8: Final Assembly

1. Insert MCU from top down into the Turret Base
2. Insert Horizontal Servo (bottom one) into turret base from the top
   a. It will be tight - push hard
3. Attach MCU board to inside of turret base with double sided sticky tape
4. Plug in servo and pew pew!



Mad Science 101: Mechatronics for World Domination!

Part 2: Mechanical Control

Summary

You have learned

1. Lasers
2. Electromagnets
3. Motors
4. Microcontrollers
5. Skills:
    a. Wire Tinning
    b. Wire Stripping
    c. Wire Splicing
    d. Heat Shrink tubing
    e. Mechanical Assembly

Mad Science 101: Mechatronics for World Domination!

Part 3: Mechatronics Programming

Summary

Prep:

1. Who does not have a laptop that they can bring for the next class?
2. For those who do, if you can download the Arduino IDE onto your laptop it will speed things along, but it is not required.

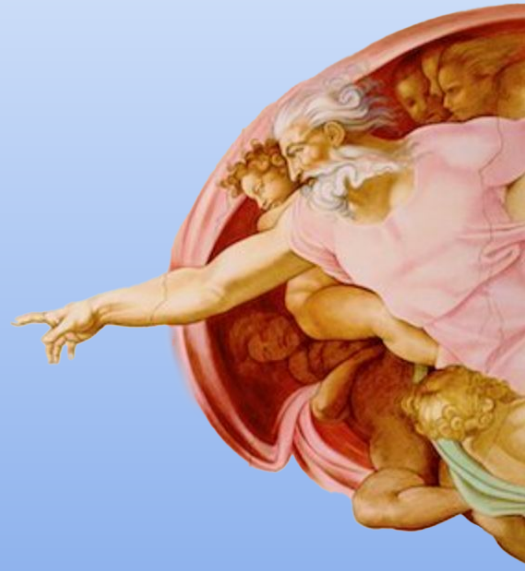
Mad Science 101: Mechatronics for World Domination!

Part 3: Programming - Breathing Life Into Your Creation!

Project: Laser Sentry Tower

You will Learn:

- Programming Languages
- Programming Constructs
- Programming Microcontrollers with Arduino IDE
- Practical Skills:
  - Turning the laser on and off
  - Moving the servos

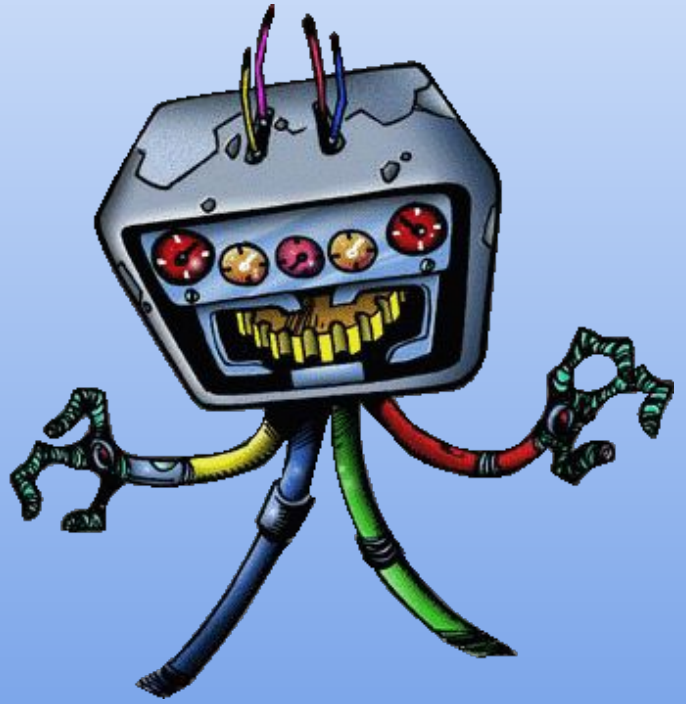


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Programming/Coding Tips :

- Practical coding is about pattern recognition. It uses the same part of your brain as language. (not math!)
  - Look to see how the pattern is in the code
  - Manipulate the pattern to do what you want it to do
- Fiddle around until it works
  - Don’t worry about efficiency
  - Don’t worry about breaking things
- Google is your friend but be specific about your hardware: e.g. *How to move a **servo** on an **ESP32** in **Arduino***
- Stack Overflow
- Github



Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Programming Languages

Computers “Speak” Binary. All signals are 0 (0V) or 1 (3.3V).
One of these is called a **Bit**.

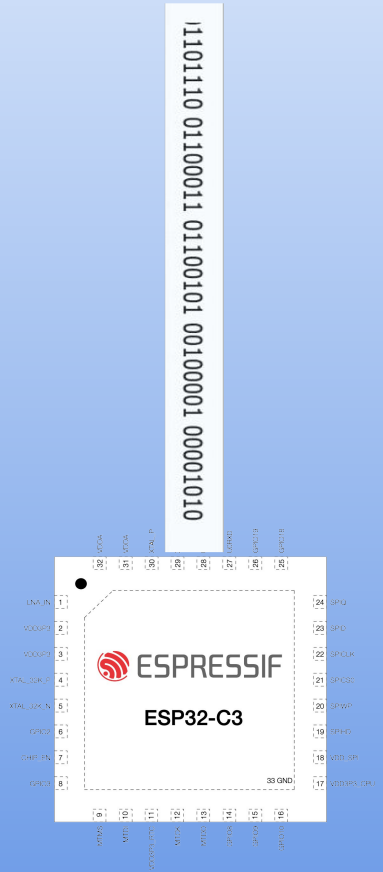
8 Bits come together to make a **Byte.**

Some number of Bytes come together make a **Word**

In our microcontroller’s case, **4 Bytes** become a **32-bit Word**

Those **Words** can hold instructions for the processor, such as **turn on the GPIO pin connected to laser.**

Remember that all actual communication with the processor is in binary, even though we talk about it other ways.



Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Programming Languages

- There are various languages to communicate with computers. Each language has domains of maximal efficacy.
- Programming Languages can be roughly classified by **Level**
- **High-Level**: Far away from the hardware
- **Low-Level**: Close to the hardware
- **Embedded programming** is programming for low-level applications.

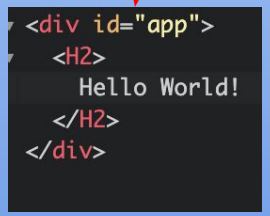
```
#define SERVO_REVERSE_X 1
#define SERVO_REVERSE_Y 0
#define SERVO_RANGE_X SERVO_MAX_ANGLE_X - SERVO_MIN_ANGLE_X
#define SERVO_RANGE_Y SERVO_MAX_ANGLE_Y - SERVO_MIN_ANGLE_Y


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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

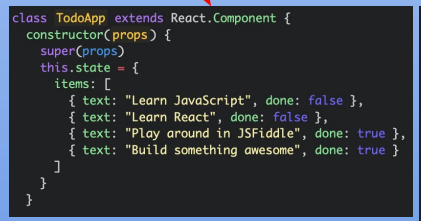
Mad Science 101: Mechatronics for World Domination!

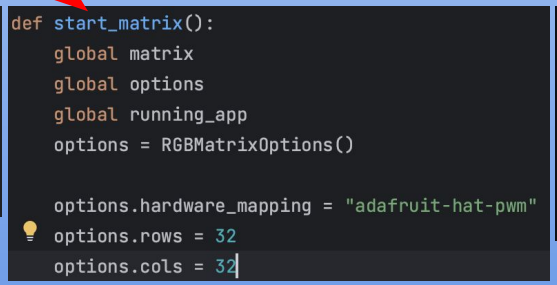
Part 3: Programming

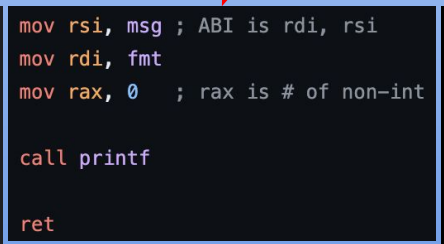
Programming Languages

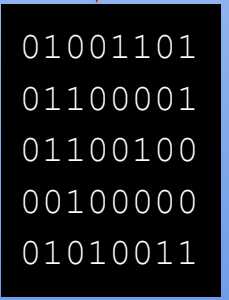
HTML	Javascript	Python	Java	C/C++	Assembly	Machine Code
Websites	Web Apps	Web Services Scripts AI	Web Services Local Apps	Local Apps Embedded High Performance Games	Drivers Kernels Embedded	Not really used by humans











Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Programming Languages

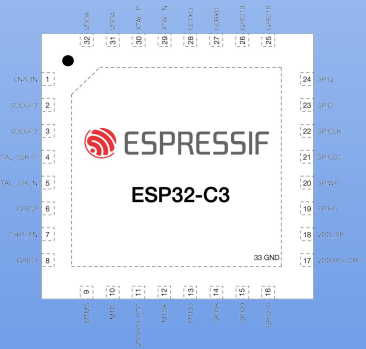
Lower-Level languages are good for talking to hardware. That's why we're going to use (a very simplified version of) **C++**

C++ is a **compiled** language, which means the text of the program is run through a translator to convert it to binary machine code prior to execution.

```
#define SERVO_REVERSE_X 1
#define SERVO_REVERSE_Y 0
#define SERVO_RANGE_X SERVO_MAX_ANGLE_X - SERVO_MIN_ANGLE_X
#define SERVO_RANGE_Y SERVO_MAX_ANGLE_Y - SERVO_MIN_ANGLE_Y


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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Compiler

1101110 01100011 01100101 00100001 00001010



Mad Science 101: Mechatronics for World Domination!

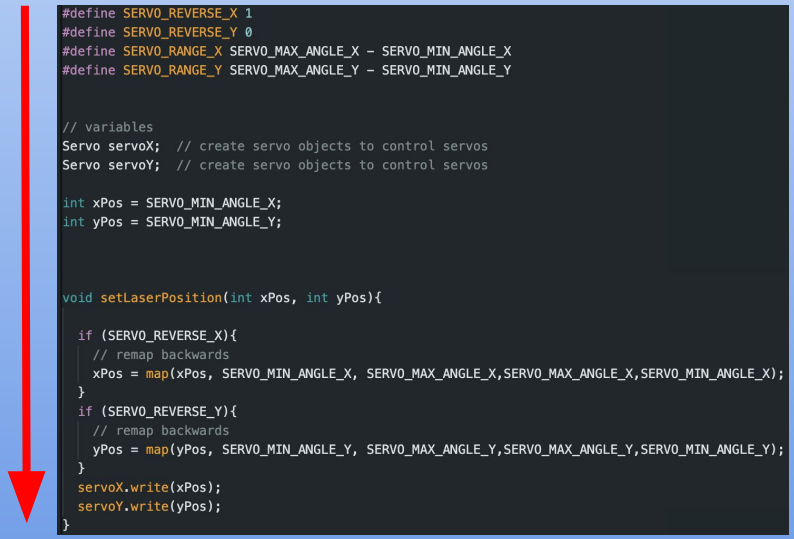
Part 3: Programming

C++ Programming Constructs

Every Programming Language Implements a set of Fundamental Programming Constructs. C++ has the following:

Construct 1:

Top to bottom execution: Program “runs” from top to bottom.



```
#define SERVO_REVERSE_X 1
#define SERVO_REVERSE_Y 0
#define SERVO_RANGE_X SERVO_MAX_ANGLE_X - SERVO_MIN_ANGLE_X
#define SERVO_RANGE_Y SERVO_MAX_ANGLE_Y - SERVO_MIN_ANGLE_Y


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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 2:

Variables: arbitrary words can be used as holders of values.

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 3:

Setting Values: The = operator sets the left side to the value of the right side as it executes the instruction for that line. Left side of = must always be a variable.

```
#define SERVO_REVERSE_X 1
#define SERVO_REVERSE_Y 0
#define SERVO_RANGE_X SERVO_MAX_ANGLE_X - SERVO_MIN_ANGLE_X
#define SERVO_RANGE_Y SERVO_MAX_ANGLE_Y - SERVO_MIN_ANGLE_Y


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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 4:

Line Termination: Every single-line instruction terminates with a semicolon ";"

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

I'm lying to you here, but it's for a good cause!

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 5:

Strong Typing: Variables must be created with a type. Some standard types are things like **int** (integer for a number), **string** (string of characters, words). Declaration syntax is:

```
Type variableName;
```

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 6:

Blocks/Scope: multi-line expressions must be enclosed in curly braces e.g.

```
{
    int a = 1;
    a = a + 1;
}
```

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 7:

Conditionals, AKA if-then: keyword `if` followed by condition in regular parentheses, and then it will execute the next expression if the condition is true:

```
if (a!=1) { int a=1 };
```

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 8:

Loops - While Loops - Loops while a condition is true: keyword **`while`** followed by condition in regular parentheses, and then it will execute the next expression while the condition is true:

```
while (a<6) { a=a+1 };
```

```
loopServos();
// a dumb while loop just to sleep for a second (10 x 100ms delays)
int i = 0;
while (i < 10){
  delay(100);
  i+=1;
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 9:

Loops - For Loops - Loops a set number of times: keyword `for` followed by an idiomatic expression:

```
for (int a; a<10; a=a+1)

{

    printf("hello!");

}
```

```
void drawSquare() {
  Serial.println("Drawing Square");
  // loop x and y servos from min to max.  should make a square with the laser

  int slowdown = 2;
  // start lower left
  xPos = SERVO_MIN_ANGLE_X;
  yPos = SERVO_MIN_ANGLE_Y;
  setLaserPosition(xPos, yPos);

  // bottom left to bottom right
  for (int pos=0; pos < 180; pos++ ) {
    // this converts the value in the range 0->180 to the value ranges defined
    xPos = map(pos, 0, 180, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X);
    setLaserPosition(xPos, yPos);
    delay(slowdown);
  }
  xPos = SERVO_MAX_ANGLE_X;
  setLaserPosition(xPos, yPos);
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 10:

Functions - Programs can define and call functions, passing variables to the function while calling.

Definition:

```
void printHelloName(string name){

        printf("hello %s", name);

}
```

Call (invocation):

```
printHellowName("Thomas");
```

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 11:

Functions on Objects (AKA Methods) - Functions can be defined as part of a variable type called a Class. These Classes are created into Objects when a variable of that type is declared. Functions on the object can be called with the "." operator after the Object name and before the function name.

- `Servo` is a Class
- `servoX` is an Object
- `servoX.write()` is a function on `servoX`

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

int xPos = SERVO_MIN_ANGLE_X;
int yPos = SERVO_MIN_ANGLE_Y;



void setLaserPosition(int xPos, int yPos){

  if (SERVO_REVERSE_X){
    // remap backwards
    xPos = map(xPos, SERVO_MIN_ANGLE_X, SERVO_MAX_ANGLE_X,SERVO_MAX_ANGLE_X,SERVO_MIN_ANGLE_X);
  }
  if (SERVO_REVERSE_Y){
    // remap backwards
    yPos = map(yPos, SERVO_MIN_ANGLE_Y, SERVO_MAX_ANGLE_Y,SERVO_MAX_ANGLE_Y,SERVO_MIN_ANGLE_Y);
  }
  servoX.write(xPos);
  servoY.write(yPos);
}
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 12:

Preprocessor Directives:

#define is like a variable declaration except it doesn't need a type and it has no equals sign

#include reaches out and loads another file. Any time we want to use something in the standard c/c++ library or a library to control something (e.g. a servo) we need to include the header file (libraryname.h) for that library.

Preprocessor Directives don't end with ;

```
#include <ESP32Servo.h>
#include <math.h>

// define pins for digital io
#define ESP_C3_LED_BUILTIN_PIN  8  // Builtin LED (blue) on front.  Also
#define SERVO_Y_SIGNAL_PIN      1  // servoY's signal pin is at pin #1
#define SERVO_X_SIGNAL_PIN      2  // servoX's signal pin is at #2
#define LASER_POWER_PIN         3  // laser power is at pin #3


// set the max angles of the servo turret to 45 degrees
#define SERVO_MIN_ANGLE_X 0
#define SERVO_MAX_ANGLE_X 45
#define SERVO_MIN_ANGLE_Y 0
#define SERVO_MAX_ANGLE_Y 45
#define SERVO_REVERSE_X 1
#define SERVO_REVERSE_Y 0
#define SERVO_RANGE_X SERVO_MAX_ANGLE_X - SERVO_MIN_ANGLE_X
#define SERVO_RANGE_Y SERVO_MAX_ANGLE_Y - SERVO_MIN_ANGLE_Y
```

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Programming Constructs

Construct 13:

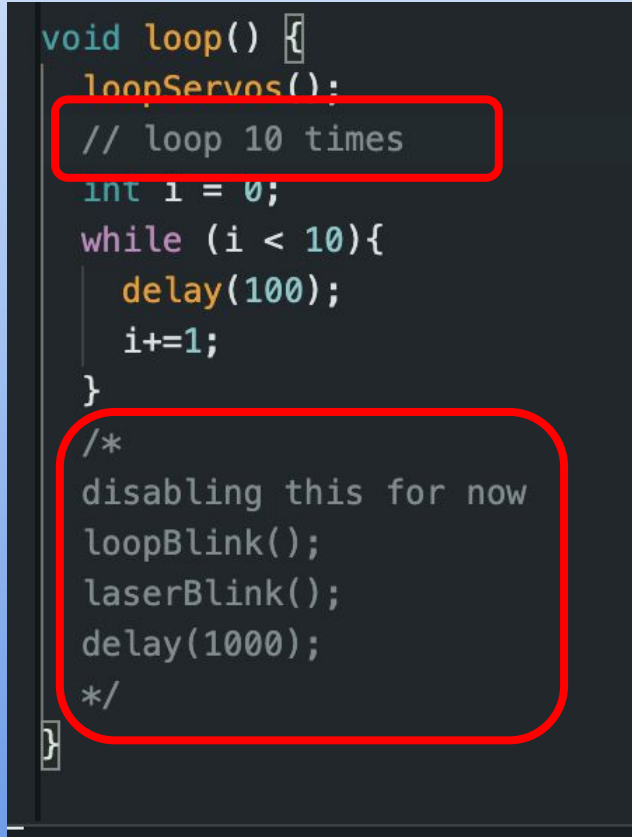
Comments:

Any line that starts with // is a comment

Any block that starts with /* and ends with */ is a comment

Comments are not executed or compiled into machine code

Comments don’t end with ;

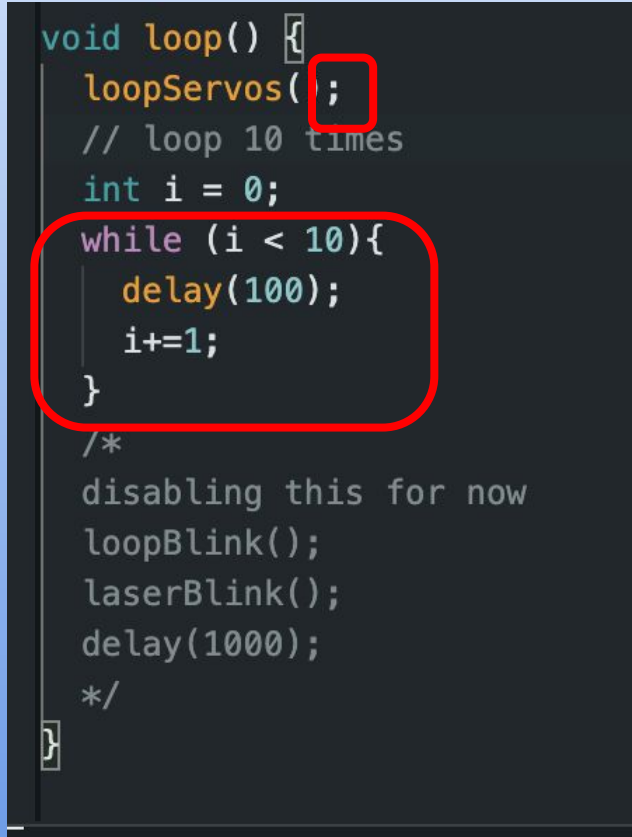


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

C++ Gotchas

1. Check to make sure your line ends with “;”
2. C++ is **Case Sensitive**
3. Make sure you match your curly braces { .. }
4. Indenting is not mandatory but it helps you debug your program. I advise Indenting anything inside of curly braces { .. } (Use tab)
5. The = operator is actually read “Set Equal To”. If you want to determine if something “Is Equal To” something else, the proper operator is ==. This is like the < operator which can be read as “Is Less Than”



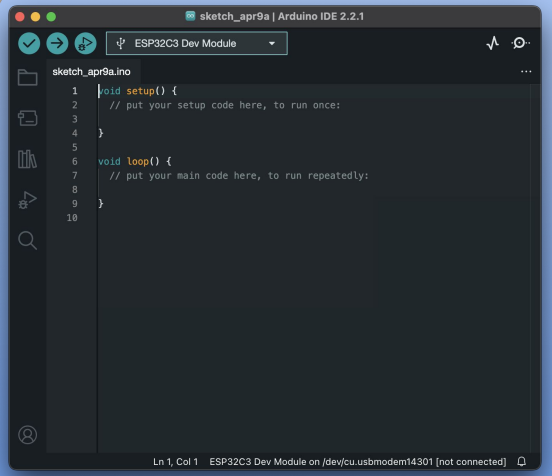
Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Arduino

Arduino is multiple things:

- **Arduino Boards** - Arduino Uno was the first Maker board
- **Arduino IDE** - Integrated Development Environment, where we will edit our code and upload into our board



We are only using the Arduino IDE, not the boards.

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

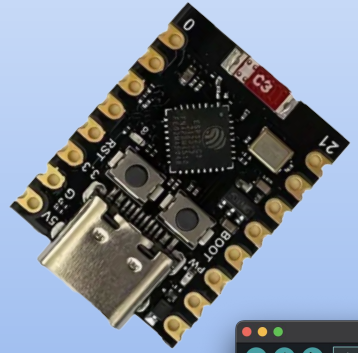
Arduino

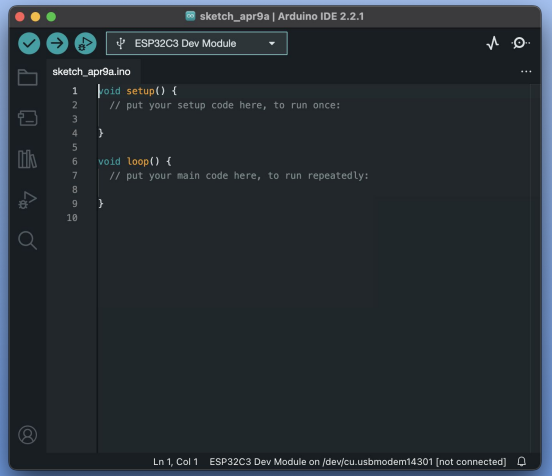
Arduino IDE Can be used to program other boards.
It is perfect for our **ESP32-C3** Super Mini Board

There is an **ESP32 Board Manager** to add additional boards to the Arduino IDE.

This allows you to plug in your ESP32-C3 and immediately start running your programs on it.

There are MANY example programs and libraries for the ESP32 which show up automatically in the Arduino IDE.





Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

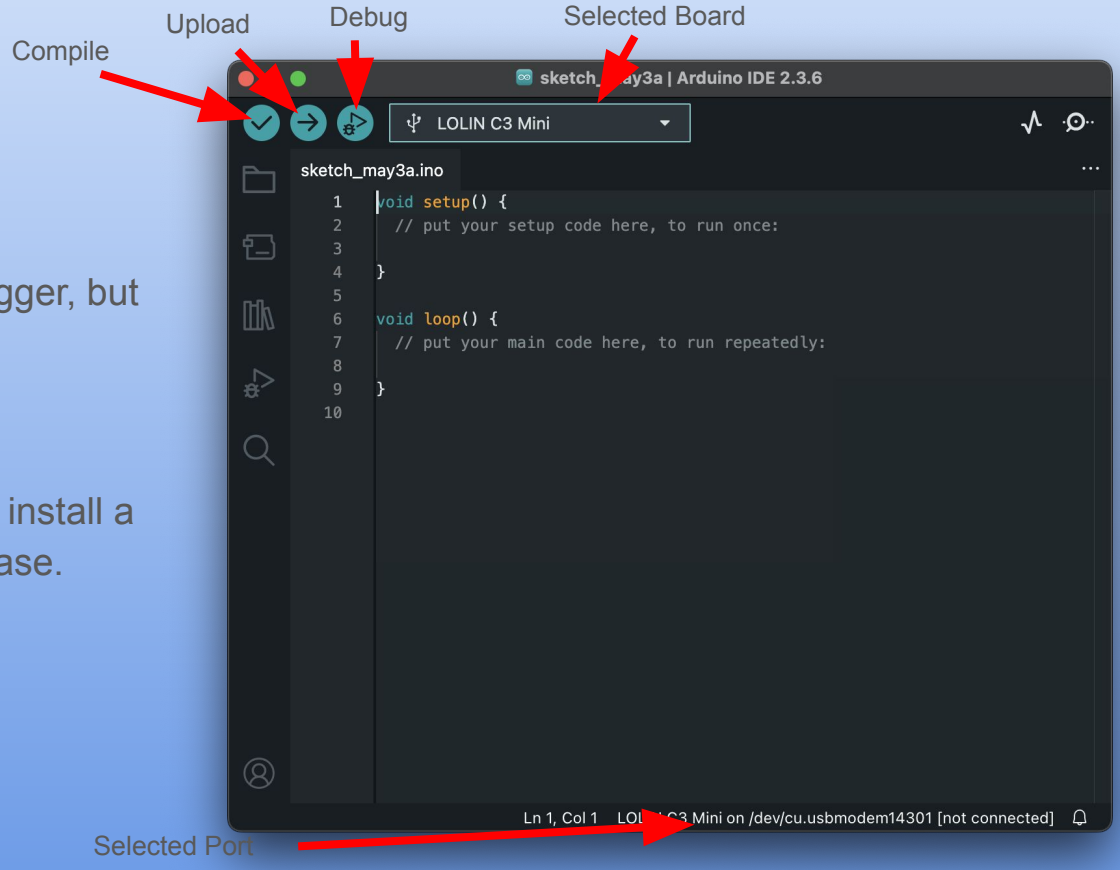
Arduino

Arduino IDE has built-in compiler and programmer for your board. Also a debugger, but we will not be using that.

Board must be set to **LOLIN C3 Mini**

Port must be selected - you may need to install a driver. I'll help if this ends up being the case.

Arduino calls a program a **sketch**



Mad Science 101: Mechatronics for World Domination!

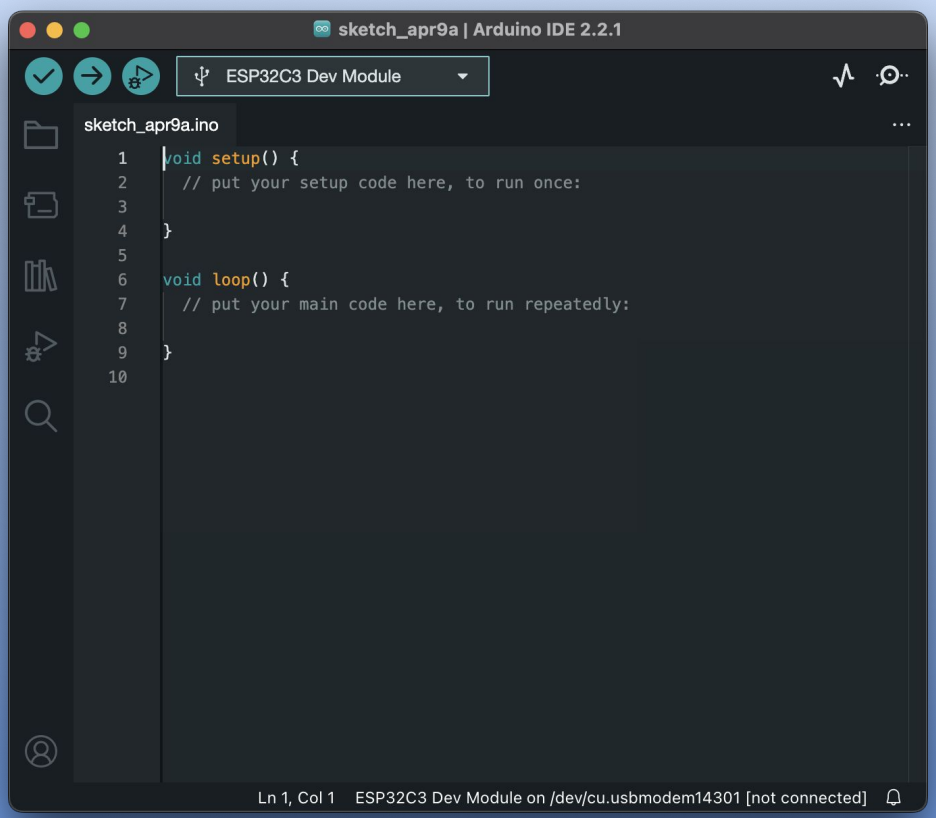
Part 3: Programming

Arduino

Arduino Sketches have two basic functions that are called automatically by the board:

`setup()`: Called once at startup. All the setup code should go here.

`loop()`: Called over and over. The processor infinitely loops this function. The main functionality of your program goes here.



Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Arduino

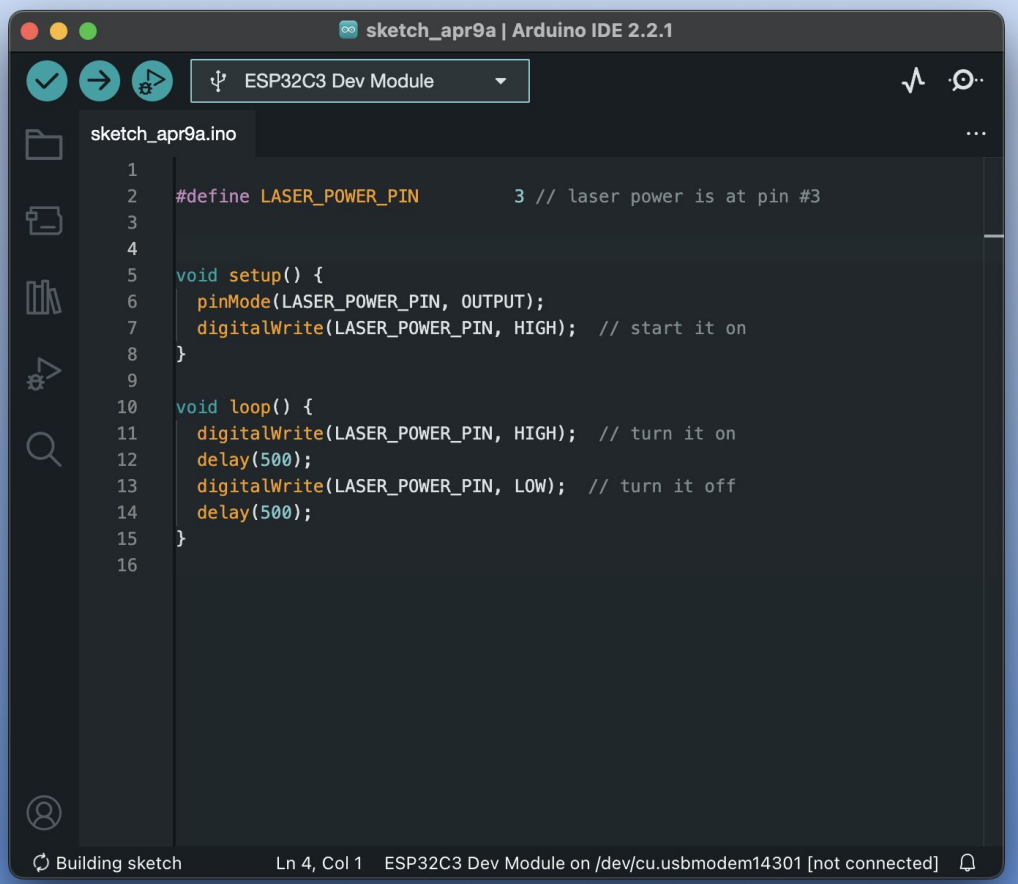
Turning on and off the Laser:

Setup:

1. Initialize the pin mode
2. Set power to HIGH or LOW to start

Loop:

1. Set power to HIGH
2. Delay (ms)
3. Set power to LOW
4. Delay (ms)



Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Arduino

Moving a Servo:

Import:

```
#include <ESP32Servo.h>
```

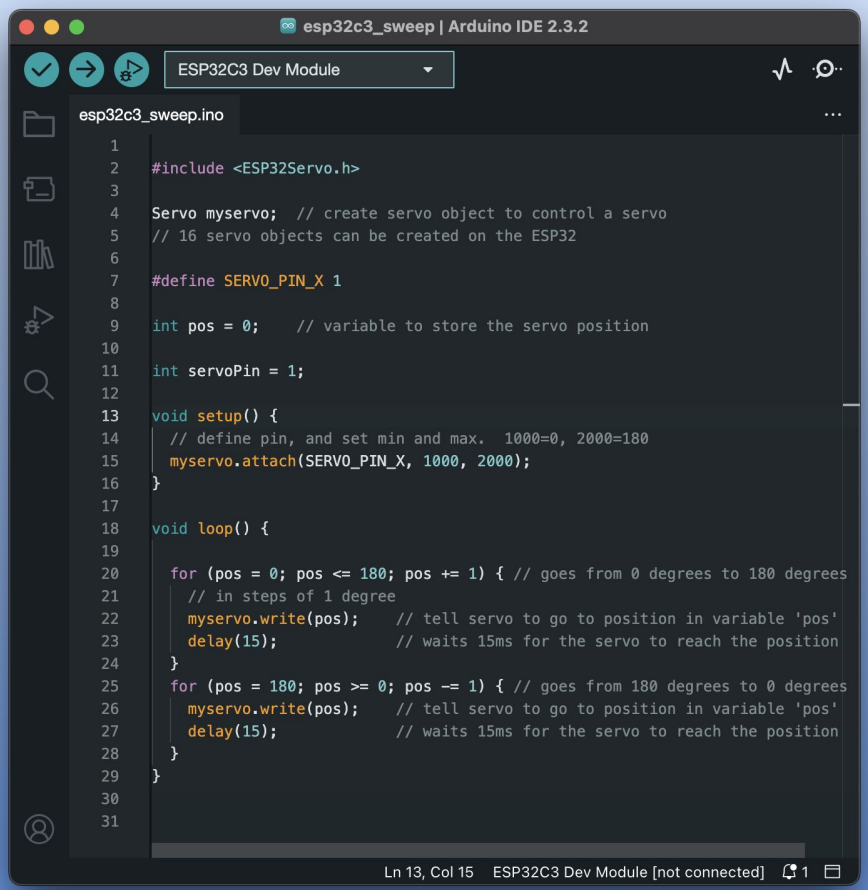
Setup:

Attach to pin:

```
myservo.attach(SERVO_PIN_X, 1000, 2000);
```

Loop:

```
myservo.write(pos); // 0 to 180 for position

delay(15);
```



Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Bringing Life to your Creation!

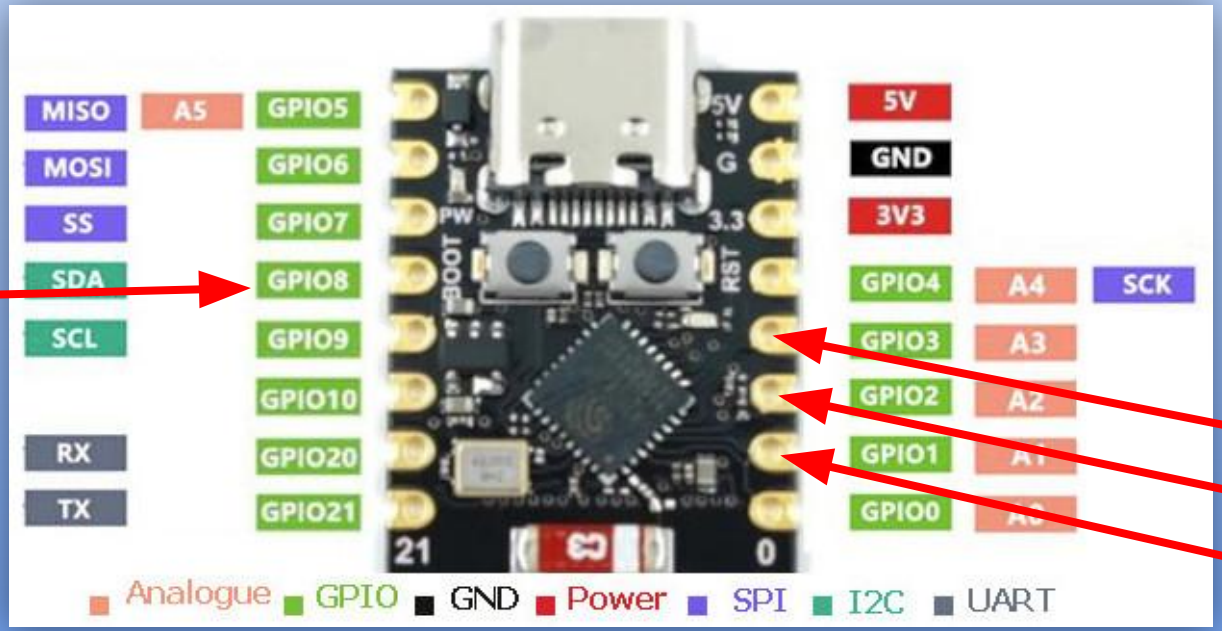
1. Download Arduino IDE
2. Install the ESP32 Board Manager
3. Plug in your ESP32-C3 board
4. Install and run the LED blink program
5. Make a Laser blink program
6. Make a servo sweep program
7. Put it all together

Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Laser Sentry Tower Pinout Reminder

NOTE: Builtin Blue LED is at GPIO **Pin 8** on this board. This lights the LED and also turns on and off the GPIO Pin 8 on the edge of the board



Mad Science 101: Mechatronics for World Domination!

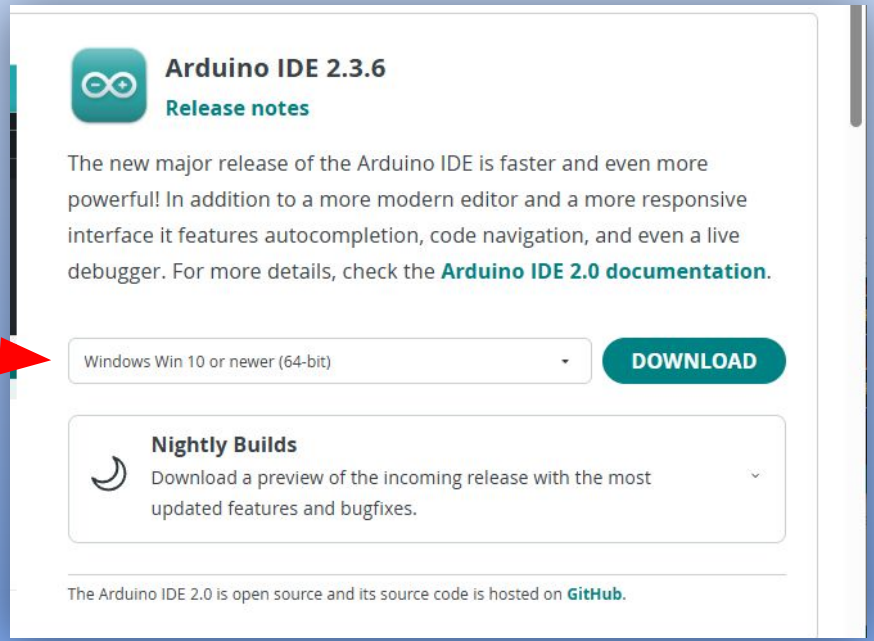
Part 3: Programming

Download Arduino IDE

Download the IDE from:

https://www.arduino.cc/en/software

Click on the operating system you are using to download and hit the DOWNLOAD button.

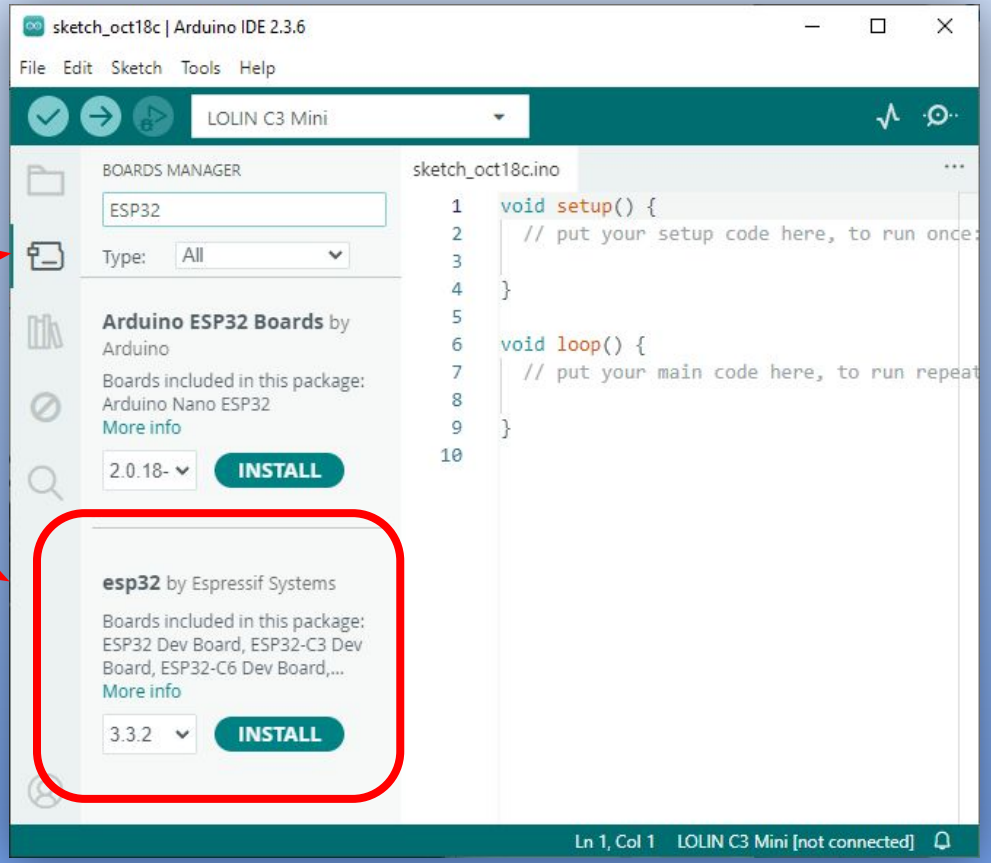


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Install the ESP32 Board Manager

- Open Boards Manager
- Search for esp32 (exactly)
- Select the one from Espressif
- click INSTALL

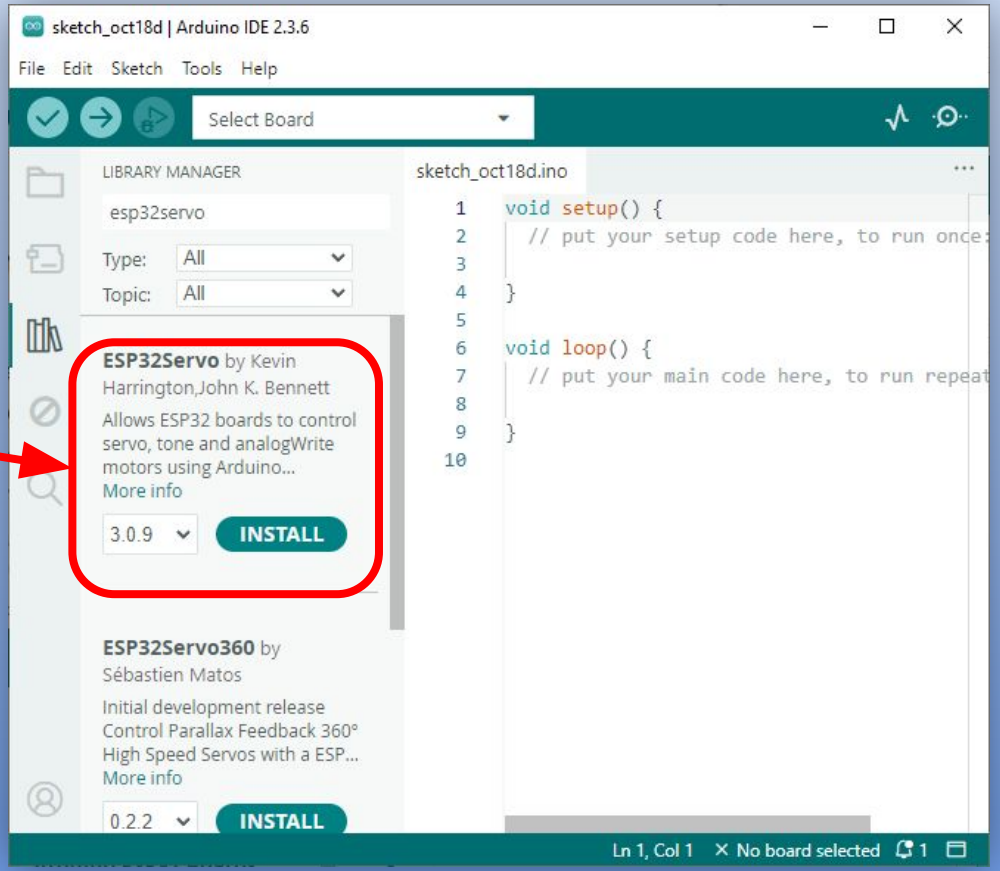


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Install ESP32Servo Library

- Open Library Manager
- Search for esp32servo
- Select this one
- click INSTALL



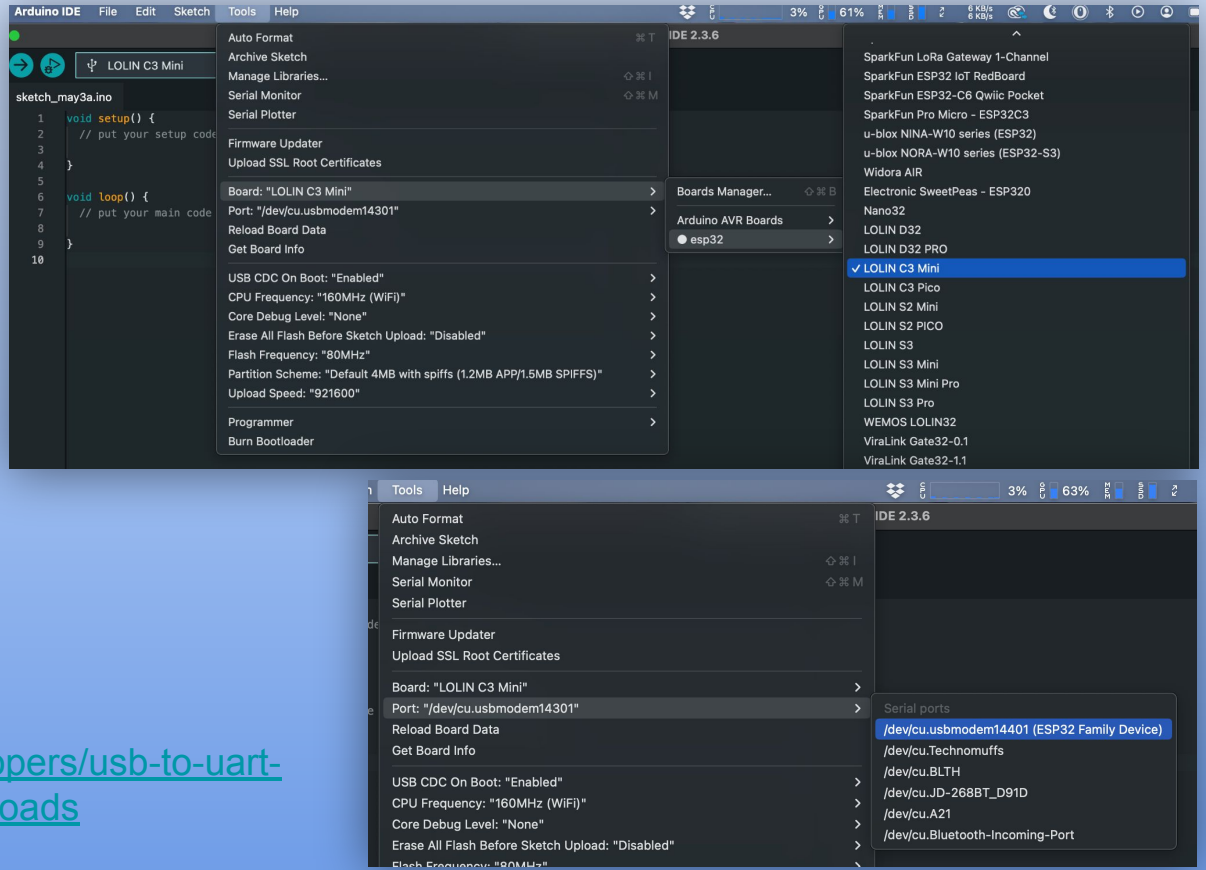
Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Connect to your ESP32-C3 board

- Plug in the MCU to computer with USB Cable
- Select **LOLIN C3 Mini** from Tools->Board->esp32
- Select the port from Tools->Port (e.g. COM3)
- If you don't see the port, talk to me and I'll help you install a VCP Driver:

https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads

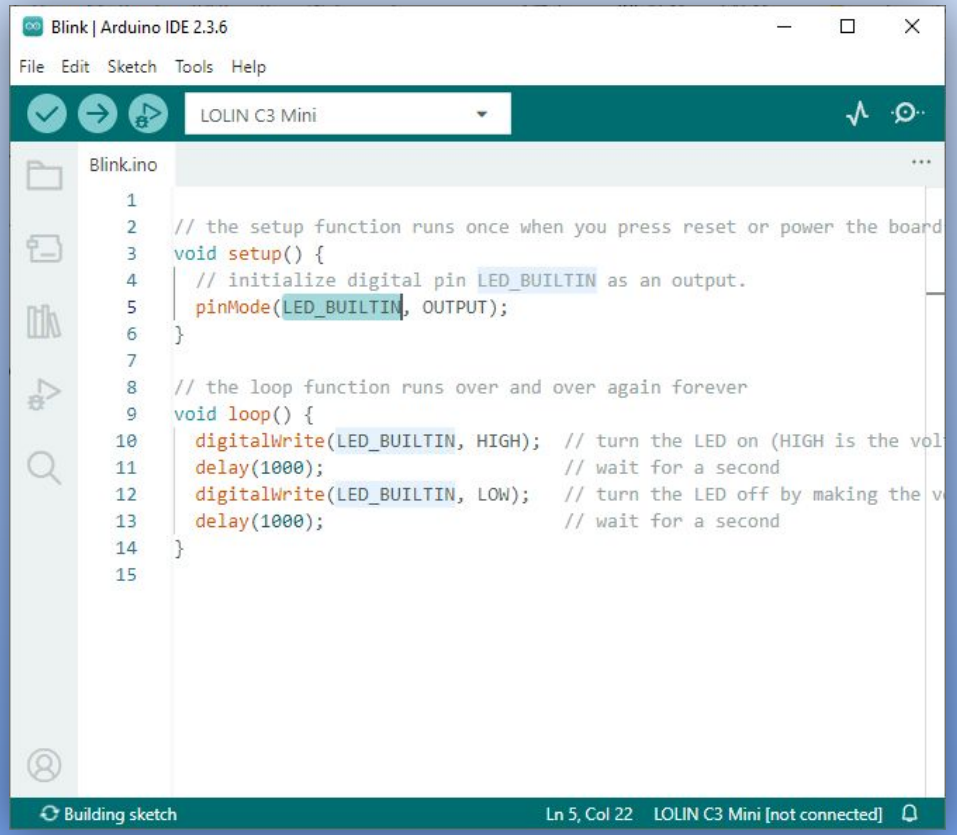


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Program 1: LED blink program:

1. Open the example Blink Sketch: File->Examples->01.Basics->Blink
2. Locate the `LED_BUILTIN` variable
3. Edit to use to the proper pin number (from your pinout)
4. Press the upload (->) button and wait for program to compile and upload.
5. The blue light on your board should start to blink
6. Mess around with the delay time



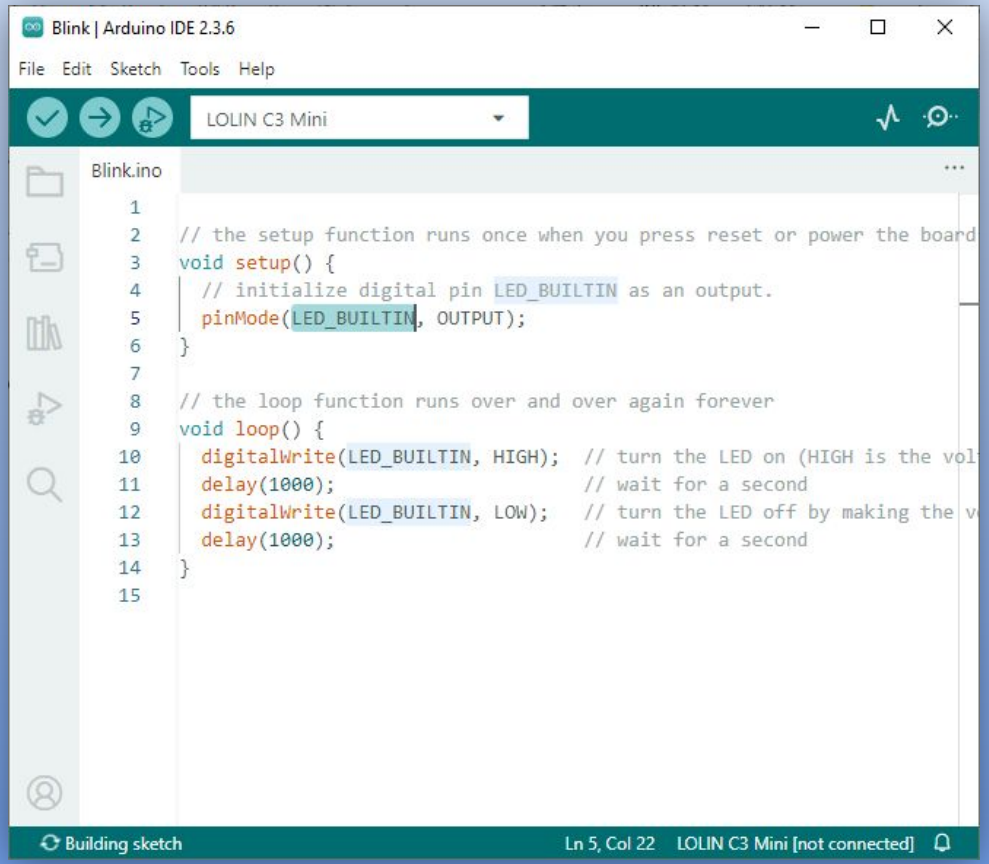
Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Program 2: Laser blink program

Modify your LED Blink program to instead blink the Laser:

1. Edit the LED Digital IO pin to the proper pin for the laser (from your pinout)
2. The laser should start to blink
3. Mess around with the delay time
4. OPTIONAL: Try to add the LED so it blinks in time with the laser
    a. HINT: Just duplicate all the digitalWrite() calls with a different pin

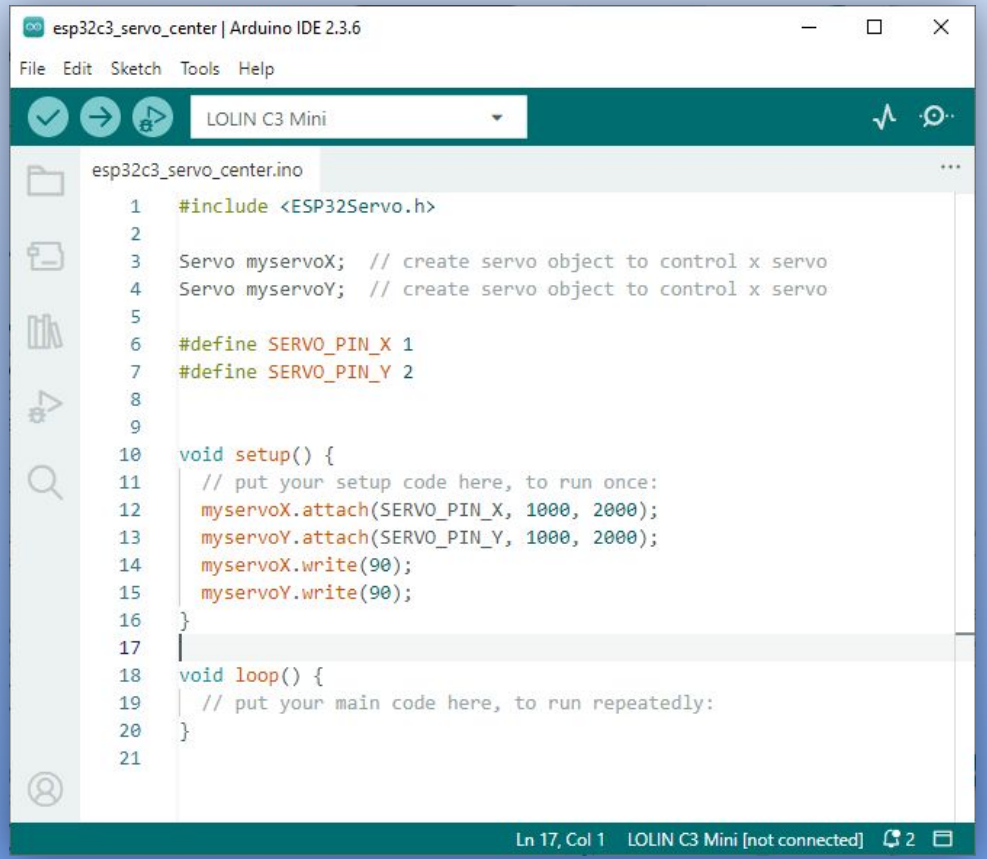


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Program 3: Servos Part 1: Center Servos

1. Install the **ESP32Servo** library from Sketch->Include Library->Manage Libraries and searching for **ESP32Servo** (One Word)
2. Click Install
3. Make a new Sketch with File->New
4. Zero the servos out by running the program on the right. This will center your servos to 90 degrees
5. physically move your servos so that the horizontal one points straight forward, and the vertical one is level (flat)



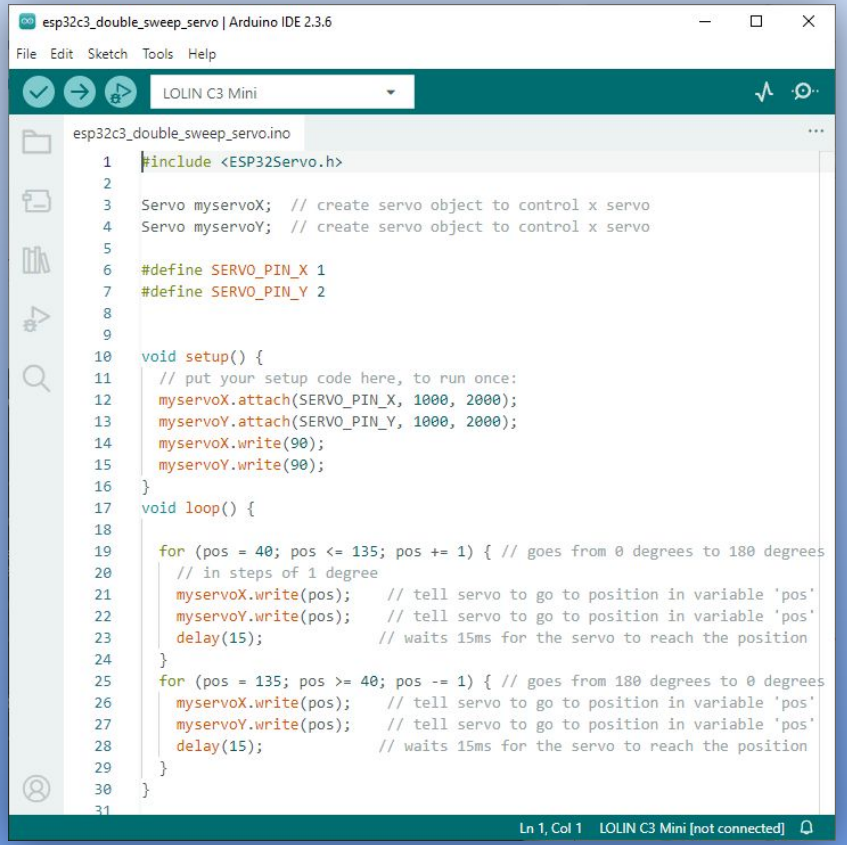
Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Program 3: Servos Part 2: Sweep Servos

Inside the loop() function:

1. Add a `for` loop to loop to move the servos from 45 degrees to 135 degrees.
2. Add another `for` loop to move the servos back from 135 degrees to 45 degrees
3. Make sure you add some `delays`, like delay(15)
4. Try swapping your servo pins. What happens?
5. try doing only one? the other?
6. set your servo pins to reflect reality, for example, SERVO_PIN_X should be the servo that makes it go right and left.

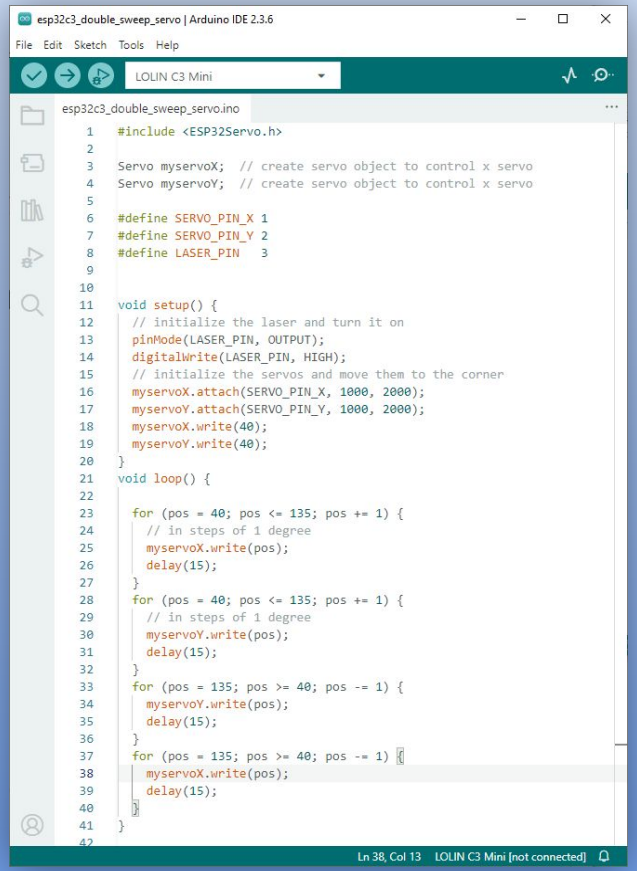


Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Put it all together!

1. Add in the lines of code in `setup()` that turn on the laser (make sure you use the right pin)
2. Add in the lines of code in `loop()` that turn on and off the laser (if you want)
3. Try to draw a square.
   a. `setup()`: move servos to 45,45
   b. `for` loop on servo1 from 45 to 135
   c. `for` loop on servo2 from 45 to 135
   d. `for` loop on servo2 from 135 to 45
   e. `for` loop on servo1 from 134 to 45



Mad Science 101: Mechatronics for World Domination!

Part 3: Programming

Step 9: Mess with stuff!

Experiment! Can you make an x? Or another letter? Write your name? Make a circle? Make a wave pattern? Move around randomly?

Hint on the last one - for a random number from 45 to 135:

```
long randNumber =
random(45,136);
```

Mad Science 101: Mechatronics for World Domination!

Links

ESP32-C3 Datasheet

Arduino IDE

Adafruit

Sparkfun

Link to ESP32-C3 Super Mini Board

Astable Multivibrator Simulator

EasyEDA PCB Design Software

Unicornmafia's Mad Science github