

The following document is intended to document and outline the path taken to achieve what up to this point could be called the VRI prototype. It will outline the pinouts, best practices, and code challenges that were encountered to better get one up to speed if they wish to emulate the procedure to create a functional prototype. The structure of the document will follow the below outline

1. Hardware
 - a. Links to each component used
 - b. Pinout according to 3rd party ESP32 device to screw termination
 - c. Pins used
 - d. Cable and color code
 - e. Fan connection notes
2. Software
 - a. Links to each necessary library
 - b. Links to the code used
 - c. Overall Unity setup and configuration
 - i. Unity
 - ii. Visual Studio Code
 - d. Final script
3. Putting it all together
 - a. VTAs on glove
 - b. Assembly tips

Hardware

- a. LINKS TO HARDWARE
 - i. [ESP32](#)
 - ii. [12VDC FAN](#)
 - iii. [VTA](#)
 - iv. [RUBBER WORK GLOVE](#)
 - v. [10 CONN CABLE](#)
 - vi. [SCREW TERMINATION](#)
- b. ASSOCIATED PINOUTS
 - i. Description
 1. The ESP32 that was available for purchase does not obtain the same pinout as a typical ESP32 board found [here](#). As a result, the pinout on the screw termination kit does not match up verbatim and a conversion for the pinout must be created to ensure that the cables are wired into the correct termination. The table below will outline the pinouts with the assumption that the EN pin of the ESP32 is lined up with the EN of the screw termination kit. With the ESP32 USB-C facing towards you, the pinout will start on the top left side and move down. It will then proceed with the right side starting with pin 23 and likewise move downwards. Pins that are different than their associated ESP32 will be bolded to ensure that it is

known the screw termination will be a different name than what is on the ESP32. The '-' indicates no pin present.

ii.

ESP32	Screw Terminal
-	3v3
EN	EN
VP	VP
VN	VN
D34	34
D35	35
D32	32
D33	33
D25	25
D26	26
D27	27
D14	14
D12	12
D13	GND
GND	13
VIN	D2
-	D3
-	CMD
-	5v
-	GND
D23	23
D22	22
TX0	TX
RX0	RX

D21	21
D19	GND
D18	19
D5	18
TX2	5
RX2	17
D4	16
D2	4
D15	0
GND	2
3v3	15
-	D1
-	D0
-	CLK

- iii. Ensure this table is used to convert whatever the hardware code calls for in terms of wiring. (e.g. The code calls for pin D19 to a VTA. Connect it to GND on the right side of the ESP32)

c. PINS USED

i. VTA PIN ASSIGNMENTS

FINGER	GPIO
Thumb	32
Index	33
Middle	25
Ring	26
Little	27

ii. FAN PIN ASSIGNMENT

PWM	14
-----	----

d. CABLE PINOUT COLOR CODE

i.

Color	Component
BLACK	THUMB_VCC
BLUE	INDEX_VCC
GREEN	MIDDLE_VCC
RED	RING_VCC
ORANGE	LITTLE_VCC
YELLOW	THUMB_GND
GREY	INDEX_GND
BROWN	MIDDLE_GND
VIOLET	RING_GND
WHITE	LITTLE_GND

1. The GND return legs will all be joined together and put into PIN_13 of the screw termination expansion board. If this is the case, a simple ethernet cable can be used and wired in the following form

ii.

Color	Component
BLUE	THUMB_VCC
BLUE WHITE	ALL_FINGER_GND
GREEN	INDEX_VCC
GREEN WHITE	MIDDLE_VCC
ORANGE	RING_VCC
ORANGE WHITE	LITTLE_VCC
BROWN	FAN_PWM
BROWN WHITE	-

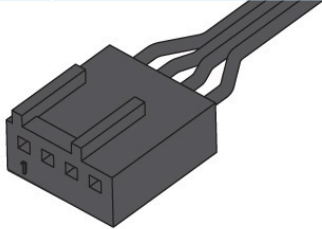
1. The VTAs are non-polarized so it does not matter which side is used for GND or VCC.

e. FAN CONNECTION NOTES

i. 120mm PC Fans

1. Fan power to the outputs of 12v PSU

Pin No.	Pin Assign
1	Ground
2	VCC (5-12V)
3	Signal
4	PWM

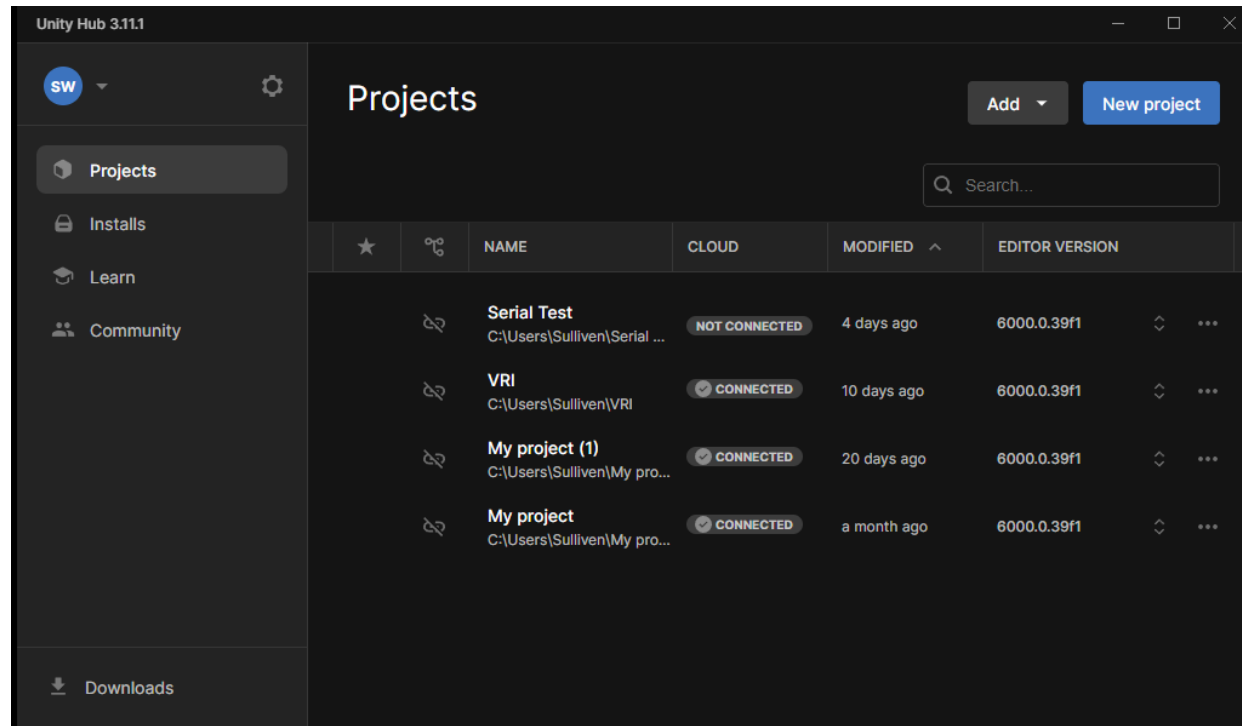


- 2.
3. 1: Ground to GND(-) of PSU
4. 2: VCC to 12V(+) of PSU
5. 4: PWM to pin9 of MCU

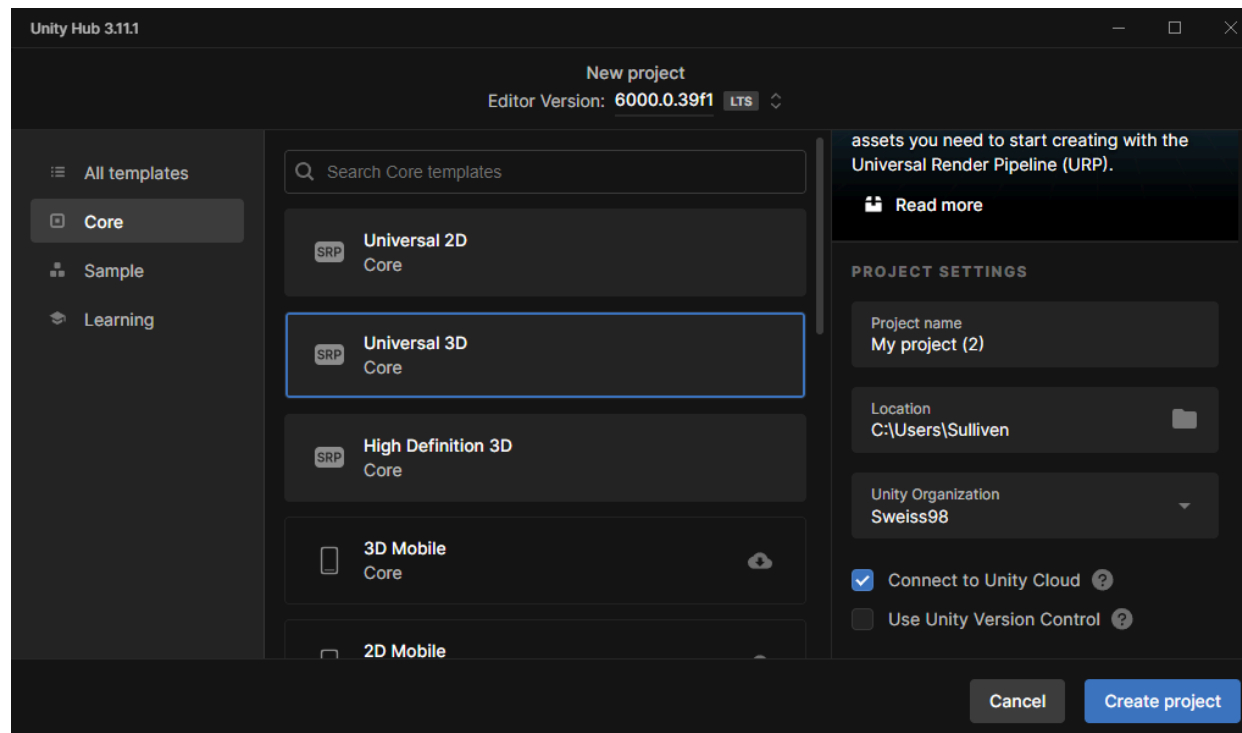
Software (serialTest)

- a. With the assumption that Unity, Arduino IDE, and Visual Studio code are downloaded, proceed with step b. If you do not have them downloaded, you can find the software here:
 - i. Unity
 1. <https://unity.com/download>
 - ii. Visual Studio

1. <https://code.visualstudio.com>
- iii. Arduino IDE latest
 1. <https://www.arduino.cc/en/software/>
- b. Open Unity Hub and create a new project

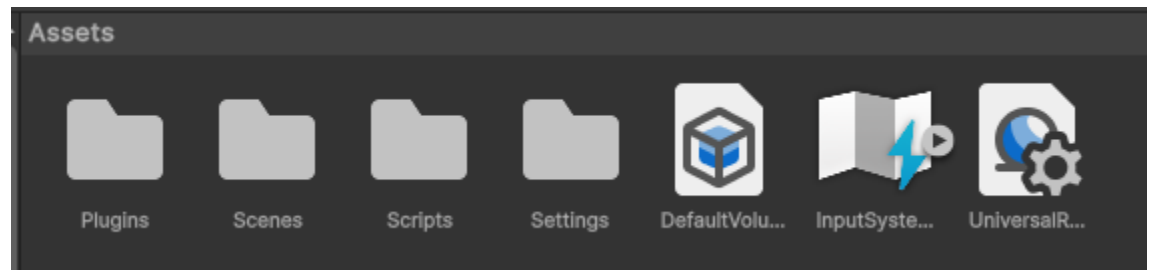


- i.
- c. Select Universal 3D as your Core template



i.

- d. Verify there is a file folder structure created to support additional libraries. It should resemble the following (ignore the root files present within this Assets directory). If any folders are missing, simply right click -> create -> folder



- i.
- e. Download the following libraries and place the .dlls into the Plugins folder
- [System.ComponentModel](#)
 - [System.IO.Ports](#)
 - [System.ComponentModel.Primitives](#)
 - [System.ComponentModel.Composition](#)
- f. Plug the ESP32 into your PC and record the COM interface it's plugged into
- Hit "Win+R"
 - Type in "devmgmt.msc" and hit Enter
 - Locate Ports (COM & LPT) and click the dropdown
 - Find the COM interface which the ESP32 is connected to and record this for future use
- ▼ Ports (COM & LPT)

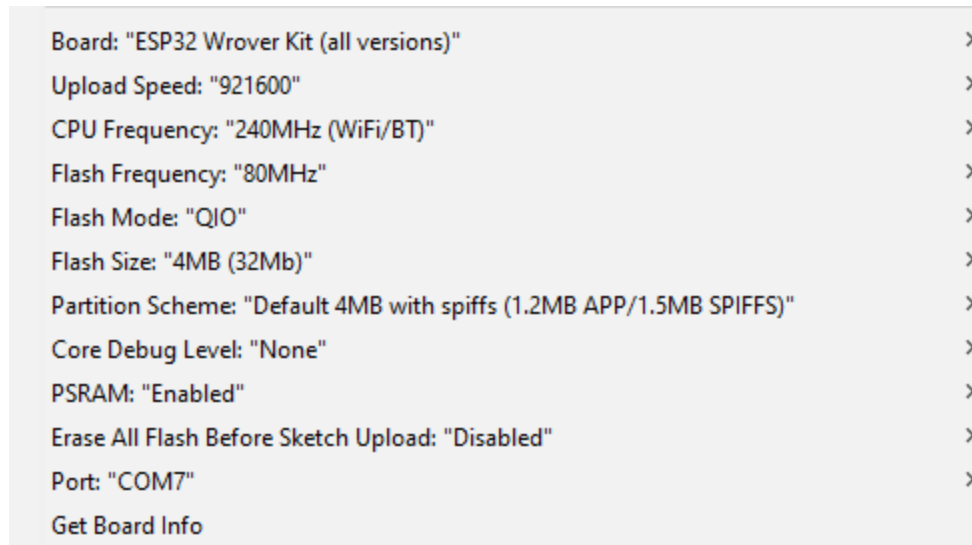
 - Communications Port (COM1)
 - Silicon Labs CP210x USB to UART Bridge (COM7)
 - Standard Serial over Bluetooth link (COM3)
 - Standard Serial over Bluetooth link (COM4)
- v.
- g. Open the Arduino IDE and paste in the following code
- [serialTest.ino](#)
 - It should resemble the following

serialTest

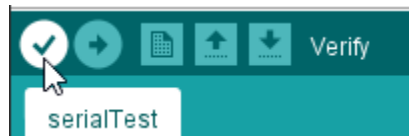
```
1 // ESP32 Serial Communication for LED Control
2 // Listens for commands from Unity and toggles LED accordingly
3
4 const int ledPin = 2; // Built-in LED on ESP32
5
6 void setup() {
7     Serial.begin(115200); // Start Serial communication
8     pinMode(ledPin, OUTPUT);
9     Serial.println("ESP32 Ready. Send 'LED_ON' or 'LED_OFF' via Serial.");
10 }
11
12 void loop() {
13     if (Serial.available()) {
14         String command = Serial.readStringUntil('\n'); // Read input command
15         command.trim(); // Remove any whitespace/newline
16
17         Serial.print("Received: ");
18         Serial.println(command);
19
20         if (command == "LED_ON") {
21             digitalWrite(ledPin, HIGH); // Turn LED ON
22             Serial.println("LED turned ON");
23         } else if (command == "LED_OFF") {
24             digitalWrite(ledPin, LOW); // Turn LED OFF
25             Serial.println("LED turned OFF");
26         } else {
27             Serial.println("Unknown command.");
28         }
29     }
30 }
```

iii.

- h. The objective of this code is to allow the ESP32 to read it's serial interface and active an LED based on an action that takes place via serial. All commands that are not sent via Wi-Fi can be sent over USB serial.
- i. Within the Arduino IDE, locate the tools tab and ensure the following is changed.



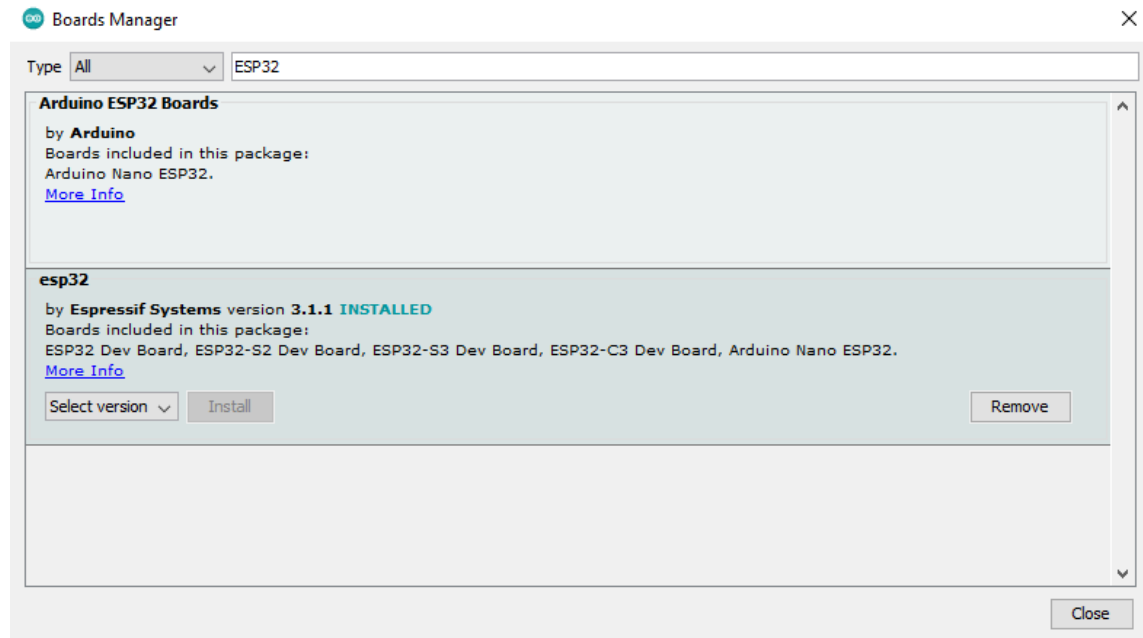
- i.
- ii. Ensure your COMx interface reflects what is shown within device manager.
- j. Click the check icon on the upper left hand side to verify the code



- i.
- k. Once verified, click on the upload icon

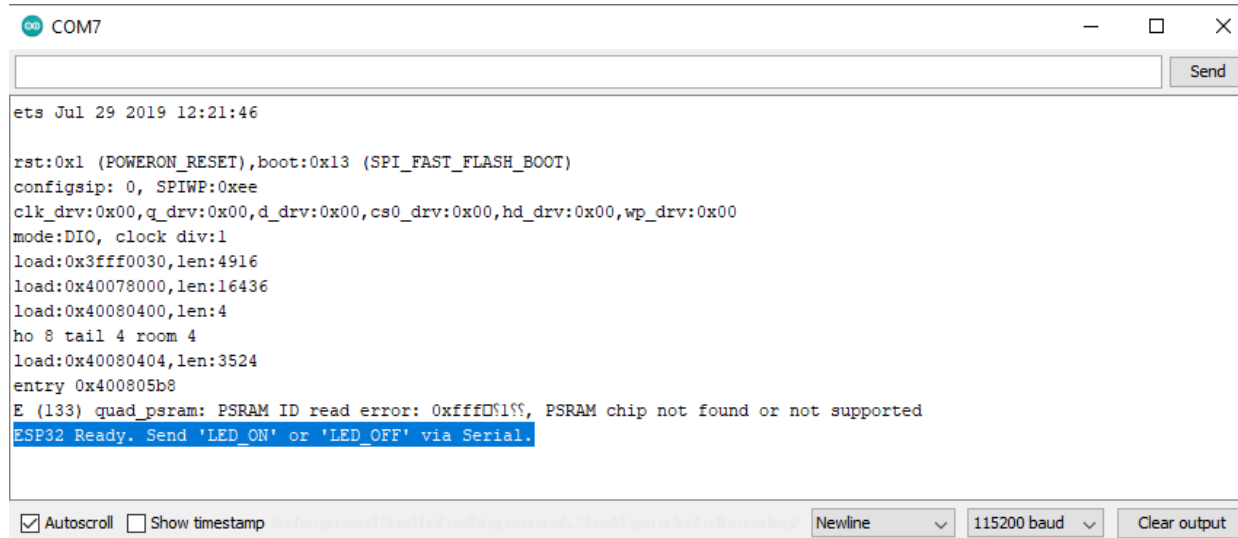


- i.
- l. The output will be shown below in the console as to whether the upload worked or not. If it fails out, ensure all the settings are correct for the board type and COM interface.
 - i. If the board is not present within the board section, do the following.
 1. Click on boards and board manager, this will open another window.
 2. Type in ESP32 and install the esp32 board libraries



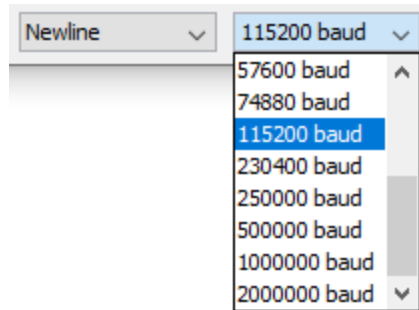
3.

- m. Once the code is uploaded, locate Tools -> serial monitor and open up the monitor.
- n. Click the reset button on the ESP32 and verify the code functions (button to the left of the USB-C interface while the USB-C is facing you)
 - i. You should see an output

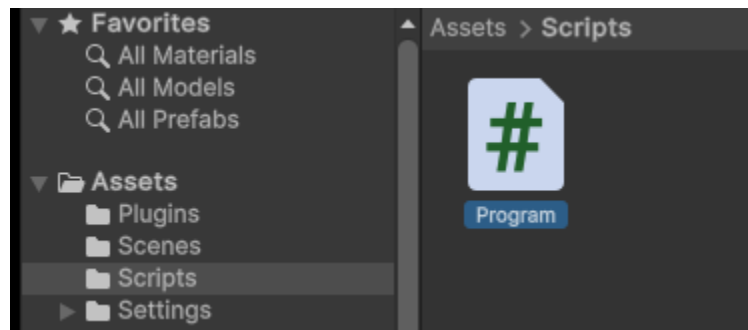


ii.

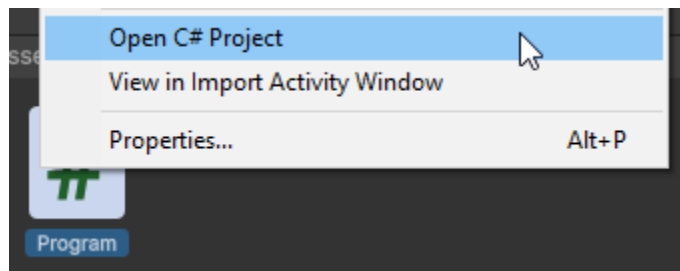
- iii. If there is no output, verify that the baud in the lower right dropdown is set to 115200.



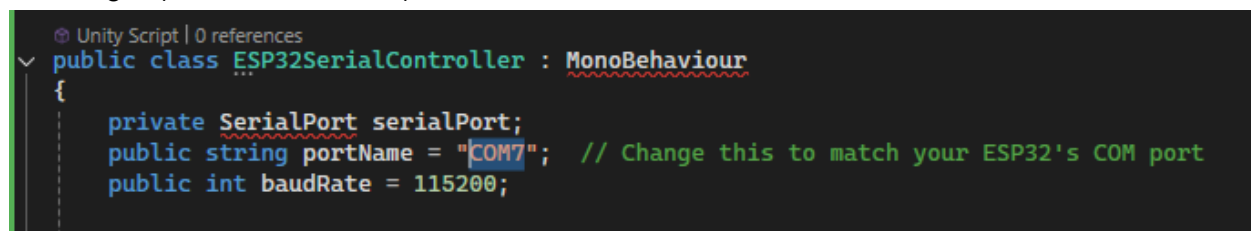
- iv.
- o. Close the monitor and the IDE, we do not want anything else taking up the serial communication. Open Unity back up.
- p. Within your scripts folder, place the following .cs file
 - i. [serialTest.cs](#)
 - ii. It should resemble the following



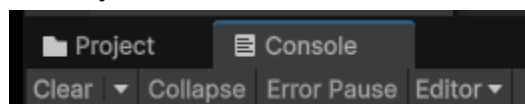
- 1.
- q. Right click on the file and click Open C# project



- i.
- r. When the code opens up in Visual Studio, change the COM interface to that which is set in device manager (COM7 in our case)



- i.
- s. Go back to the Unity project and ensure there are no errors within the Console tab to the right of the Project tab



i.

- t. Click the play button at the very top and ensure the project runs.
 - i. Press E → Sends "LED_ON" to the ESP32.
 - ii. Press D → Sends "LED_OFF" to the ESP32.

Once verifying that this works, we can expand our code to ensure that different key presses can interact with the pins going to the VTAs and fans.

- a. Ensuring that we have tested everything from a Unity standpoint, let's now ensure that all the individual facets of the design cohere adequately.
 - i. BUTTON SETUP
 - ii. GAME OBJECT INTEGRATION
 - iii. CODE UNITY
 - iv. CODE ESP32

Putting it all together

- a. Vibrotactile Actuators (VTAs)
 - i. The aforementioned VTAs within the BoM have a sticky backing that can easily adhere to rubber gloves. Each VTA is about 3v and has ample power from the GPIO interfaces.
 - 1. Each VTA has two leads, red and blue, that can be wired in any direction.
 - 2. Ensure one lead goes to the GPIO and the other lead to GND
 - 3. Each ground lead can connect to another ground. Whether this is twisted together into the screw terminal or daisy chained on the glove.
 - ii. The fan has three connections that will be needed to function. Ensure you verify from the diagram (*e.i.1*) which one goes to GND or (-) of the 12 VDC PSU and VCC (+). The last PWM lead will connect directly to the screw termination (14).
 - iii. 3d fabrication
 - iv. Assembly uses M6 x 32mm