LITTLE BANG VIRTUAL PINBALL CONTROLLER

Version 1.00, first hack

Author: Rickey M. Horwitz

INTRODUCTION

This document details the design and assembly of a game controller specifically used to interface with a virtual pinball machine. The document provides detailed instructions on theory of operation, logistics, assembly, test, and implementation.

DESCRIPTION

This project revolves around a Raspberry Pi Pico module. The Raspberry Pi Pico is a system on a chip that provides a peripheral interface between the human and host PC using USB, like a keyboard, mouse, or joystick. These are commonly referred to as USB HID devices. Devices such as mechanical buttons, accelerometer, and a ball plunger are connected to the Pico and the signals from these devices are processed in the Pico and sent to the host PC using a Joystick and Keyboard HID interface to ultimately control a virtual pinball application.

SPECIFICATIONS

- Interface: USB 1.1
- USB HID Types: Joystick/Gamepad and Keyboard on a single USB
- Button inputs: 16 each, debounced at .03 seconds (30 milliseconds).
- Button latency: .005 seconds (5 milliseconds)
- Ball Plunger: 16-bit, sampled at 200 Hz (.005 seconds)
- Accelerometer: LIS3DH, 10-bit, X/Y axis sampled at 100 Hz (.01 seconds)

PROJECT SUMMARY

The project requires only a few components and shouldn't take long to assemble. The Raspberry Pi Pico is the most powerful and versatile hobby processor on the market today. The extremely low cost makes it an ideal choice for almost any project. The LIS3DH accelerometer module was chosen as it is the most popular model available and should be available for at least 10 years.

The RP Pico module is interfaced to a LIS3DH module by four wires: 3.3V, GND, SDL-I2C, SDA-I2C. Up to 16 buttons are soldered to I/O pins to the Pico. The other ends to the buttons are all tied to 3.3V supplied by pin 36 on the Pico module. A linear potentiometer (for ball plunger) is wired to the Pico module using the 3.3V, GND, and ADC2 (pin 34).

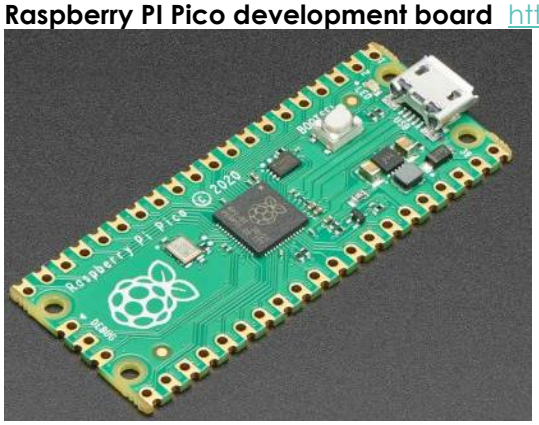
INSTRUCTIONS

Follow the instructions in the following sections.

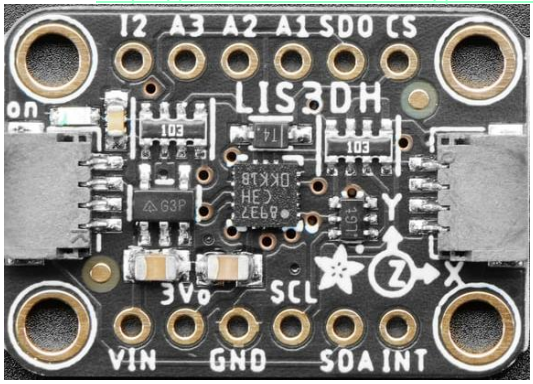
BILL OF MATERIALS

HARDWARE

- **Raspberry PI Pico development board** https://www.adafruit.com/pico?src=raspberrypi



- **LIS3DH** https://www.adafruit.com/product/2809



- **24 AWG Zip wire** https://www.amazon.com/American-Stranded-Power-Speaker-Available/dp/B0791C68HJ/ref=sr_1_1?crid=3HALH81KQ5DBN&dib=eyJ2IjoiMSJ9.TMAyo69AeFaKcGeklbvzunnyC4ltIkiu-QYxZ2y1aAK2E4e_qNzrlE2vTllNz31SSXYEYiRn-KdjiHqyKsKESTx8jjEIcgUqbLIFbe-kQwBVPCJJzsVMetWYaHSa06bEyXrYm3B2X3QdNwokGlcl3eOd-ugpjQNU8rgMZk-fqUaJP6fguTylJfDo8jI2xXH0SgR8H45Lhs3qc7zLlr3GAJUlcBtCnOqBe_G3NXKnjNzmVPkViZJLlPeaDikVBGFEDriwzjnzHX4_aHcZd-bF5yzhOngNmp3rqQQBmCJAGkc.AjGd9jBQ-r1Yd_IDxv68nj3t93yKsz_N-WcXdJ_uX8U&dib_tag=se&keywords=24%2BAWG%2Bzip%2Bwire&qid=1714151230&sprefix=24%2Bawg%2Bzip%2Bwire%2Caps%2C78&sr=8-1&th=1

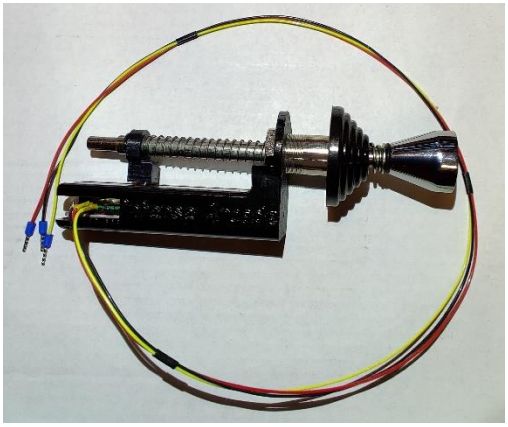
- **USB Micro to USB A**

OPTIONAL HARDWARE

This hardware is beyond the scope of this document but a useful reference for needed items to complete a pinball cabinet.

Arcade Buttons - https://www.diyretroarcade.com/products/american-style-28mm-standard-arcade-push-button-for-arcade1up?variant=32334177042567

Ball Plunger - Complete system includes linear potentiometer, plunger mechanism, Beehive bezel, feruled wiring, and binding post. This assembly is provided by Intense Arcade which authored this project.



LIS3DH module generic https://www.amazon.com/HiLetgo-Tri-Axis-Acceleration-Development-TemperatureSensor/dp/B082W63MWL/ref=dp_prsubs_sccl_1/145-7467177-1190224?pd_rd_w=HTjzt&content-id=amzn1.sym.53e0c629-1936-47cb-93a2-c361b12e7d3c&pf_rd_p=53e0c629-1936-47cb-93a2-c361b12e7d3c&pf_rd_r=G0VSMSGHQVPGA6CTJXNF&pd_rd_wg=rgomO&pd_rd_r=3c0d6f94-ce74-4eb3-976f-ec1fa4acbd96&pd_rd_i=B082W63MWL&psc=1

This is an alternative to the Adafruit module. I used these on my prototype, and they worked fabulously. This generic version does not have a built-in 3.3V LDO and as such relies on the Pico's built in 3.3V regulator.

Project Bread Board
https://www.amazon.com/gp/product/B00FXHXT80/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1

SOFTWARE

CircuitPython 9.X - https://circuitpython.org/board/raspberry_pi_pico/

CircuitPython Library bundle 9.X https://circuitpython.org/libraries

Joystick_XL https://github.com/fasteddy516/CircuitPython_JoystickXL

Little Bang Pico boot file -in this github

Little Bang Pico code file -in this github

REFERENCE DOCUMENTATION

RP Pico description https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

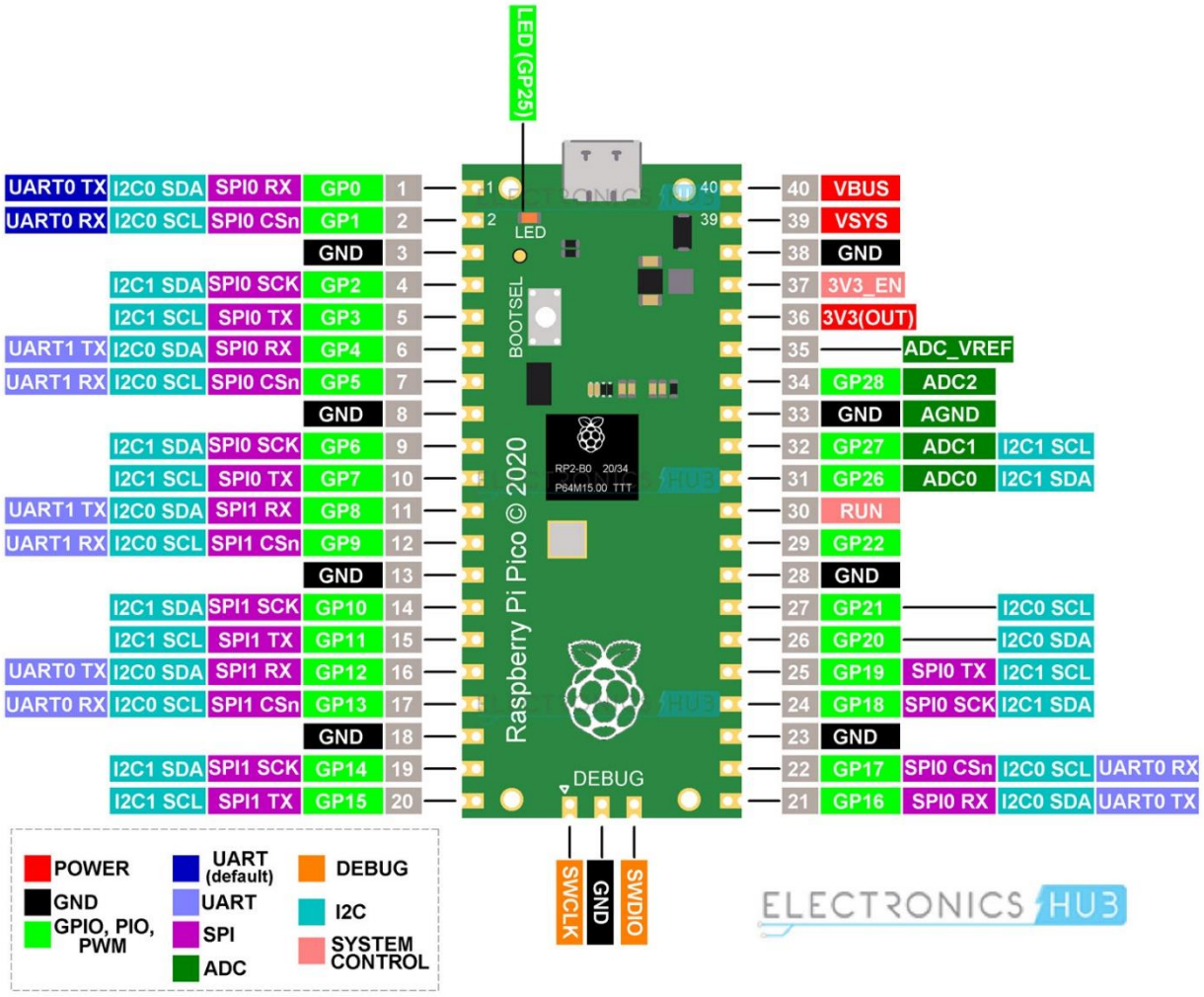
LIS3DH data sheet https://www.st.com/resource/en/datasheet/lis3dhh.pdf

Joystick_XL documents https://circuitpython-joystickxl.readthedocs.io/en/latest/

HARDWARE ASSEMBLY

The hardware assembly is presented in three illustrations: accelerometer, buttons, and ball plunger potentiometer. All three of these devices need to be wired and installed as shown. Common to all three device interfaces is Pico pin 36, 3.3V (If using the generic LIS3DH). Ensure that accommodation is made to accept three connections to this single pin. As shown in the RP Pico figure below, there are several ground pins (GND). The ground pin for the Ball Plunger is exclusive to the ground on pin 33. NO OTHER DEVICES SHOULD USE THIS GROUND!!

The Illustration below shows an overview of the RP Pico.



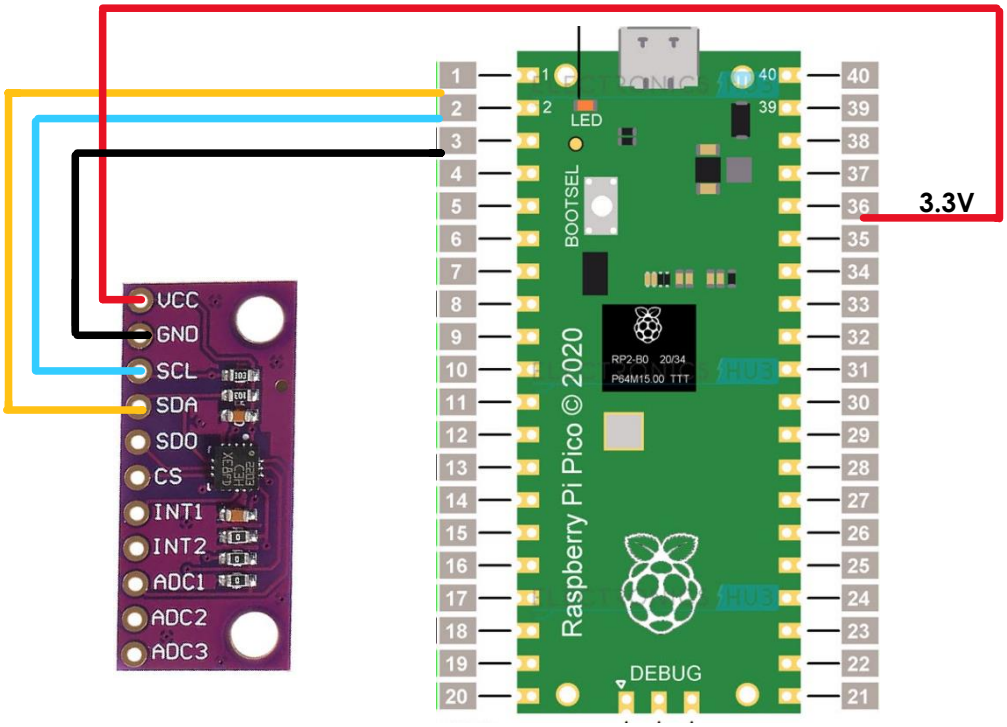
LIS3DH INTERFACE TO THE PICO

This illustration highlights the LIS3DH to Pico Interface. The ball plunger and buttons are omitted for clarity. The accelerometer must be mounted close to the SCL/SDA pins to ensure data integrity. The wire length should not exceed 75 mm for these two data lines. The LIS3DH module shown in this example is a generic module and is physically different than an Adafruit. This module needs to be securely mounted flat and level inside the pinball cabinet. The Y axis should run forward and aft of the cabinet, while the Y axis runs left and right. Again, this module must be level in both axis to work properly.

The example shown here uses a generic LIS3DH module that does not have a built-in LDO 3.3V supply. As such, it gets its power from the Pico's built-in 3.3V. If using the Adafruit module, use the Vin for supplying the needed 5V from the Pico's VBUS on pin 40.

CAUTION

DO NOT FEED VBUS (PIN 40 FROM PICO) TO THE 3Vo PIN. IT WILL DESTROY THE MODULE



If the axis appears to operate backwards, a python code modification will be needed. Removing or adding a (-) to the 3500 gain factor will change the axis direction.

```
x_acc.value = int((lis3dh.acceleration.x) * -3500)

y_acc.value = int((lis3dh.acceleration.y) * 3500)
```

BUTTON INTERFACE TO THE PICO MODULE

This illustration highlights the connections for the buttons on the Pico. LIS3DH and ball plunger have been omitted for clarity. The GP I/O outputs are pulled low and use the Pico's onboard 3.3V supply to switch the I/Os high. Refer to table 1 that shows the GP I/O to the designated keys.

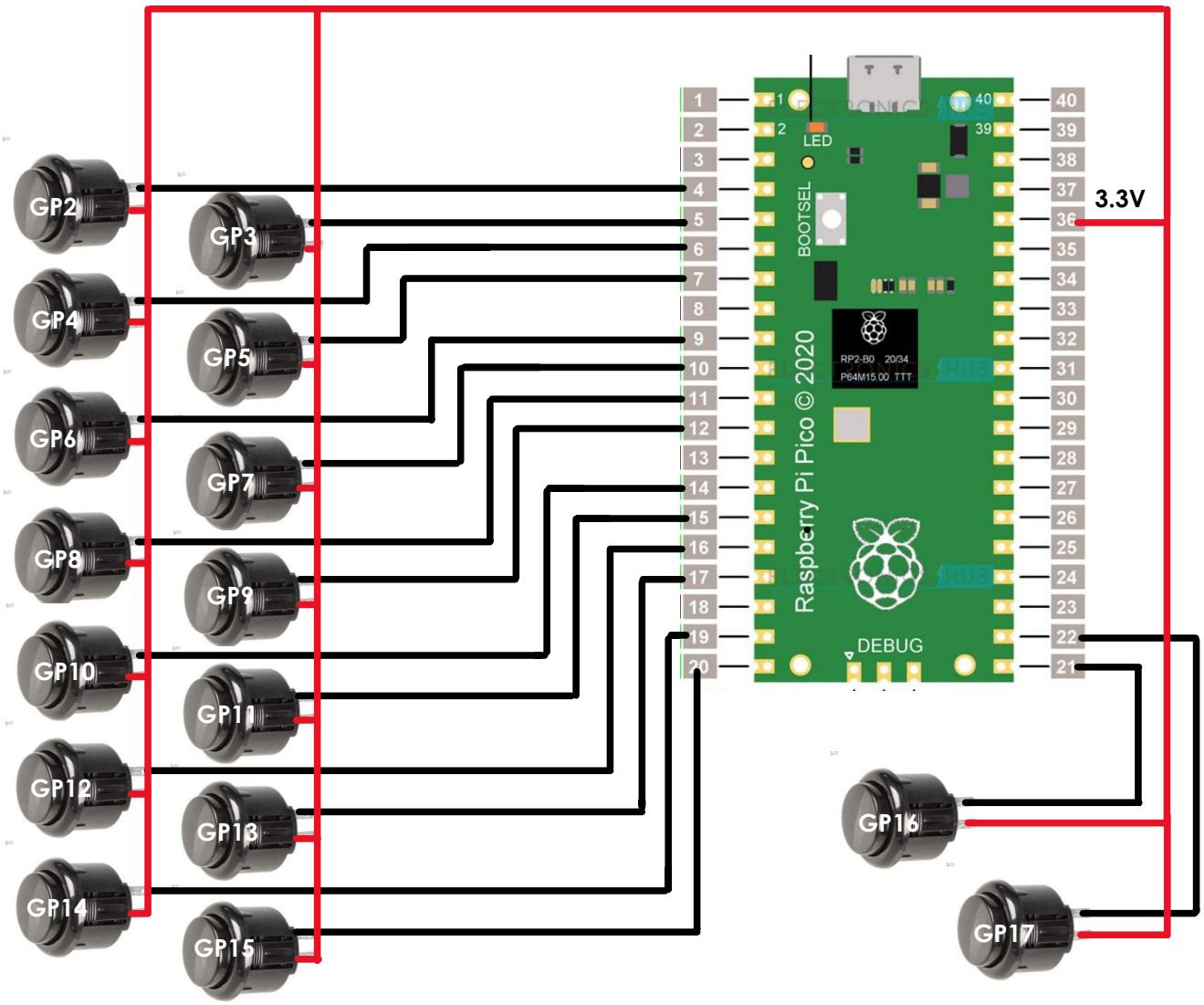


TABLE 1

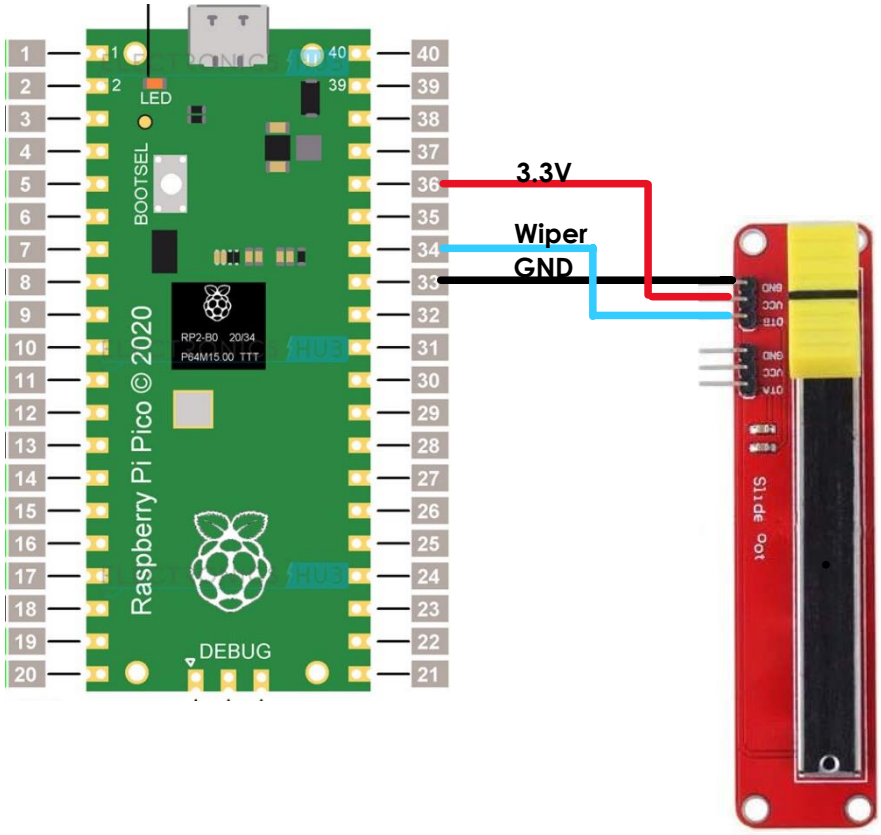
GP I/O Pin	BTN #	Keystroke
GP2	Button_1	Left Shift
GP3	Button_2	Right Shift
GP4	Button_3	Left Control
GP5	Button_4	Right Control
GP6	Button_5	1
GP7	Button_6	5
GP8	Button_7	4
GP9	Button_8	Enter
GP10	Button_9	q
GP11	Button_10	=
GP12	Button_11	-
GP13	Button_12	F11
GP14	Button_13	t
GP15	Button_14	5
GP16	Button_15	4
GP17	Button_16	Enter

The designated keys above can be changed in the python code.

BALL PLUNGER INTERFACE TO THE PICO

This illustration highlights the connections for the ball plunger to the Pico. LIS3SH and buttons have been omitted for clarity. The ball plunger uses a linear potentiometer (pot) to measure the movement of the ball plunger. As the plunger moves, the pot's wiper changes resistance which is measured as volts to ADC 02 on the Pico. The Pico's 3.3V onboard supply (pin 36) is used to power the pot, while the pot return is attached to AGND (pin 33). The value of this pot must be in the range of 5k to 20k ohms and cannot use an audio or logarithmic taper.

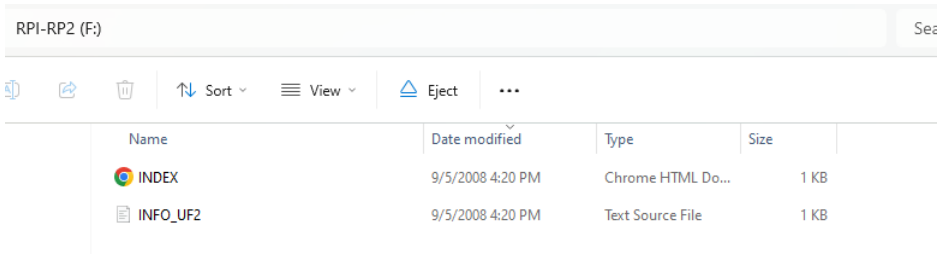
The length of the wires for the ball plunger depends on the value of the pot, how the wires are routed, and how much electrical noise is in the cabinet. Noise will cause the plunger to jitter. 500mm is the nominal length using a 20k ohm pot. 1 meter length is okay for a 5 to10k pot. If noise is persistent in your cabinet, try using shielded wire. Ensure only to ground one end of the shield to any ground on the Pico provided it's not pin 33.



SOFTWARE CONFIGURATION

The Little Bang Board uses CircuitPython programming language to tell it what to do. Assuming that all the software outlined in this instruction has been downloaded and all devices have been wired as detailed, follow the steps below:

- Attach a micro USB cable between a host computer and the RP Pico.
- The RP Pico should show up as a drive with a volume named RPI RP2 on the PC. See below:

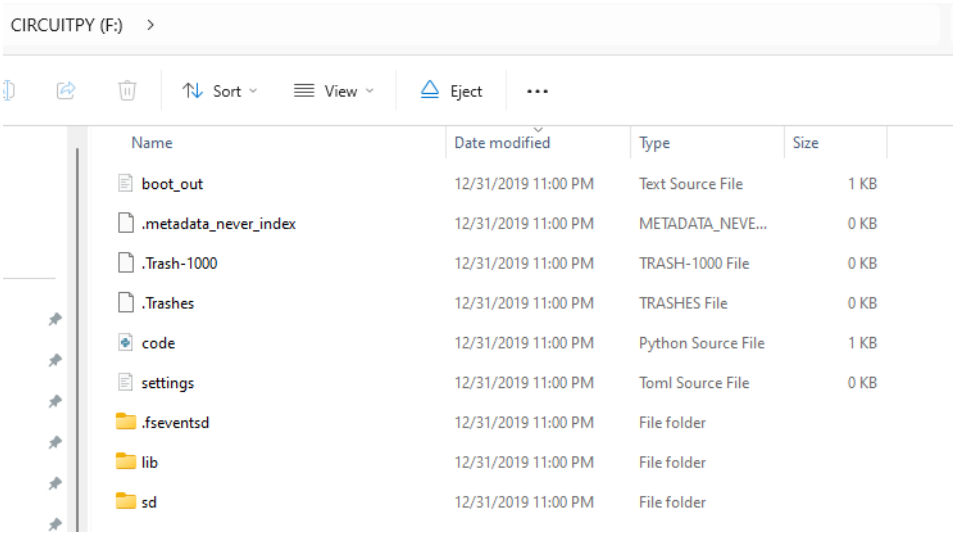


- Locate the latest release version of CircuitPython and copy it to the root drive of the RP Pico.

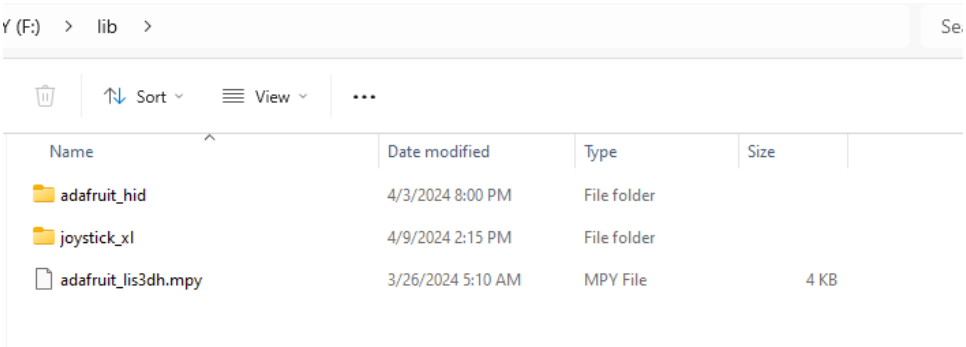
Note

This project was tested using CircuitPython version 9.x with 9.x libraries

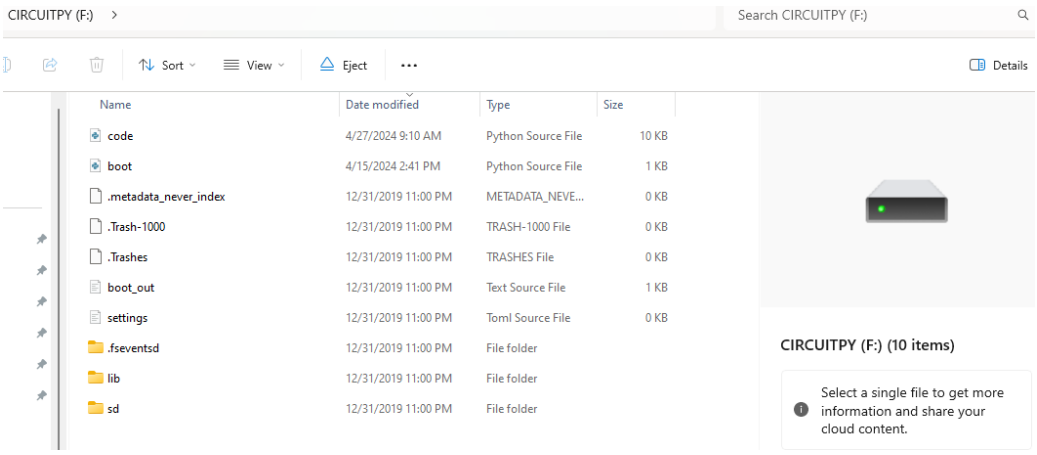
- Once the file is copied, the RP Pico will reboot and when it returns, the volume name will change to CIRCUITPY, and the drive should look like this:



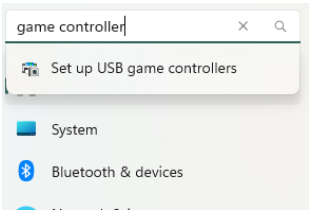
- Assuming that the Adafruit libraries and Joystick_XL library are both present, copy the following folders/files and place them directly into the \lib folder on the RP Pico. It should look like this:



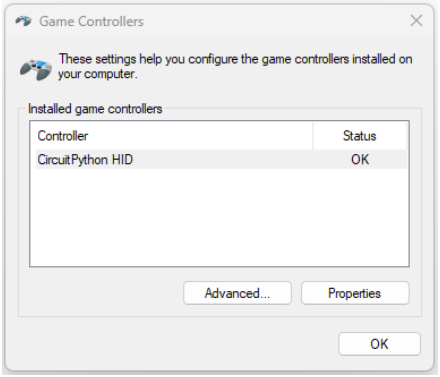
- Take both the boot.py and code.py files found on Github and copy them to the root of the RP Pico. Your root folder on the RP Pico should look like this.



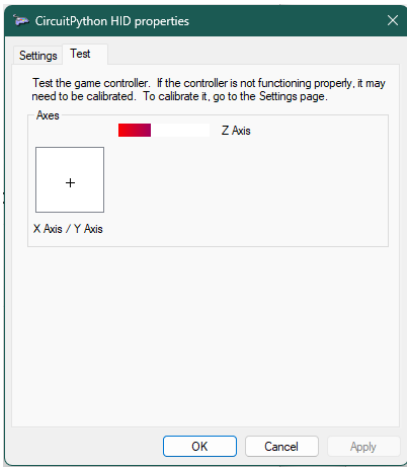
- On your PC, bring up Device Manager and look for an extra HID Compliant Consumer Control Device, HID Compliant Game Controller, and a USB Input Device.
- In Windows settings, select '*Set up USB game controllers*':



- A list should appear in a window that looks like:



- Select the CircuitPython HID and click on *'Properties'*
- The following window should appear:



- Shake the Little Bang Controller and ensure that the X/Y axis moves accordingly.
- Pull the ball Plunger back and forth and ensure that the Z axis reacts accordingly.
- Click on OK and close the CircuitPython HID properties.
- Open the notepad app on the PC, in the session press the buttons on the Little Bang Controller that correspond to the keystrokes on the keyboard. Ensure that they all work accordingly.

INSTALLING THE LITTLE BANG BOARD ASSEMBLY

Mounting the assembly should follow these requirements:

- The assembly should be mounted within reach of the button wires, plunger wires, and USB cable
- The USB plug requires at least 40 mm length allowance.
- The mounting area needs to be flat and level.
- The Accelerometer Y-axis should run forward and aft while the X-axis runs left and right.
- Route plunger wires away from other wires. This will reduce electrical noise and reduce plunger jitter.
- After mounting, check the level by running the windows game controller utility.
- If the X/Y accelerometer is picking up too much vibration from the cabinet during no activity, the dead band may need to be increased. This can be accomplished using a text editor or Mu by modifying the following lines of code shown below:

```
js.add_input(
    Axis(x_acc, 1000, -32000, 32000), # the first arg is the dead band, this keeps the table from shaking.
    Axis(y_acc, 1000, -32000, 32000), #2nd is the lowest int, and the 3rd is the highest int
    Axis(Plunger),
```

Increase the yellow highlighted numbers. Be careful not to raise these numbers too high, as this may cause the program to crash.