PlayStation Spinner Controller
Created by John Park

https://learn.adafruit.com/playstation-spinner-controller

Last updated on 2024-03-08 03:55:42 PM EST
# Table of Contents

**Overview**
- Parts

**Wire the PlayStation Controller**
- Harvest the Connector Cable
- Wire the QT Py
- Optional Heat Shrink Tubing
- Hook Up

**Installing CircuitPython**
- CircuitPython Quickstart
- Flash Resetting UF2

**Code the PlayStation Spinner**
- Text Editor
- Download the Project Bundle
- How It Works

**Use the Controller**

©Adafruit Industries
Harness the power of a very specific and weird PlayStation controller! The Takara Game of Life controller is just like a normal PlayStation (PSX) controller, with a twist -- literally -- it has a colorful spinner on it that you can twirl like a roulette wheel to spam the circle button!

While this was originally designed for use with a specific Game of Life game on PSX, you can bring it into the modern era and turn it into a USB controller that can be used for emulation, game control, and perhaps most importantly -- as a very fast mouse clicker!

Using a QT Py RP2040 running CircuitPython and the ps2controller library by Tod Kurt, this novelty controller can be used on your computer or mobile device, sending USB HID keyboard and mouse commands, or, alternatively as a USB MIDI controller.
Parts

Adafruit QT Py RP2040
What a cutie pie! Or is it... a QT Py? This diminutive dev board comes with one of our new favorite chip, the RP2040. It's been made famous in the new https://www.adafruit.com/product/4900

Takara TAKC-00001 PlayStation Controller
Phil Torrone has a great talent for discovering unusual video game controllers. He found this one and said, "JP, get this and see what you can do with it!" -- it's not very easy to find, partly as it was only released for the Japanese market it seems, but you can find them on eBay (https://adafruit.it/18Kc) and other online auction sites.

PlayStation Extension Cable - Compatible with PS1 and 2
Ostensibly this extension cord was originally designed to allow PlayStation game players to be able to sit across the room from their game system, but what we are using it for is to...
https://www.adafruit.com/product/5773
PlayStation Controller Extension Cable
These inexpensive connector cables are great for building DIY PSX and PS2 breakouts (the PlayStation and PlayStation 2 use the same 9-pin connector and pinout).

Pink and Purple Woven USB A to USB C Cable - 2 meters long
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also made for USB C for our modernized breakout boards, Feathers and more.
https://www.adafruit.com/product/5044

Pre-Cut Multi-Colored Heat Shrink Pack Kit - 280 pcs
Heat shrink is the duct tape of electronics which we guess makes this heat shrink the colorful and exciting duct tape they sell at craft stores. This pack contains heat...
https://www.adafruit.com/product/4559
Wire the PlayStation Controller

PlayStation controllers use a nine-conductor, wired connector plug. Only six pins are necessary for our needs (we'll ignore the white VCC2 wire that's used for rumble motors, and the green ACK wire).

This is the wiring we'll use:

- **QT Py A0 - DAT** (data) in from PlayStation controller **brown wire**
- **QT Py A1 - CMD** (command) data **out** to PlayStation controller **orange wire**
• QT Py A2 - ATT (attention / chip select) out to PlayStation controller yellow wire
• QT Py A3 - CLK (clock) out to PlayStation controller blue wire
• QT Py GND (ground) to PlayStation controller black wire
• QT Py 3V - VCC (power) out to PlayStation controller power red wire
Harvest the Connector Cable

Plug your controller into the extension cable to be sure you have the proper end selected. Then, cut the wire about 6 inches from that end.

Carefully remove about three inches of the outer insulation, making sure not to cut any of the wires.
Wire the QT Py

Feed the wires through the QT Py holes as shown, according to the color coding shown in the Fritzing diagram (you can ignore white and green).

Trim the wires to a consistent length, leaving about 3/4” extra. Then strip insulation from the tips and back the wires out of the holes until the bare wire is centered in each hole, then solder them in place.

PlayStation connector wire color coding tends to be consistent, but it's still a good idea to use a multimeter in continuity mode to confirm the connector pin ordering is correct.
Optional Heat Shrink Tubing
You can optionally tidy up the build with a bit of heat shrink tubing. Feeling really extra? Poke a small hole in the tubing to see the NeoPixel light up!
Hook Up
The adapter should work with any PSX or PS2 controller -- a PS2 DualShock controller is shown here connected to the QT Py adapter and USB.
Installing CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython working on your board.

Download the latest version of CircuitPython for the Raspberry Pi Pico from circuitpython.org

https://adafru.it/QaP

Click the link above and download the latest UF2 file.

Download and save it to your desktop (or wherever is handy).
Start with your Pico unplugged from USB. Hold down the **BOOTSEL** button, and while continuing to hold it (don't let go!), plug the Pico into USB. **Continue to hold the** **BOOTSEL** **button** **until** **the** **RPI-RP2** **drive appears!**

If the drive does not appear, unplug your Pico and go through the above process again.

A lot of people end up using charge-only USB cables and it is very frustrating! **So** make sure you have a USB cable you know is good for data sync.

You will see a new disk drive appear called **RPI-RP2**.

Drag the `adafruit_circuitpython_etc.uf2` file to **RPI-RP2**.
The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

Flash Resetting UF2

If your Pico ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try installing this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After nuking, re-install CircuitPython

flash_nuke.uf2
https://adafru.it/QAJ

Code the PlayStation Spinner

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in this guide (https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries, and the code.py file. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your board's CIRCUITPY drive, replacing any existing files or directories with the same
names, and adding any new ones that are necessary. The CIRCUITPY drive appears when you plug the QT Py into the computer via USB.

```python
# SPDX-FileCopyrightText: 2023 John Park w/ Tod Kurt ps2controller library
#
# SPDX-License-Identifier: MIT
# The Takara Game of Life PlayStation roulette wheel controller spinner (TAKC-00001)
# sends two sets of held CIRCLE buttons with randomized hold time periods
# this code turns that into mouse click spamming (the CIRCLE button also spams)
# Tested on QT Py RP2040

import time
import board
import usb_hid
import neopixel
from adafruit_hid.keycode import Keycode
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.mouse import Mouse
from ps2controller import PS2Controller

# turn on neopixel
led = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.1)
led.fill(0x331000) # amber while we wait for controller to connect

mouse = Mouse(usb_hid.devices)
keyboard = Keyboard(usb_hid.devices)
layout = KeyboardLayoutUS(keyboard)

# create controller object with QT Py wiring
psx = PS2Controller(
    dat=board.A0,
    cmd=board.A1,
    att=board.A2,
    clk=board.A3
)
led.fill(0x0010ee) # a nice PlayStation blue

circle_held = False
spam_speed = 0.001

buttonmap = {
    "SELECT": (0, Keycode.SPACEBAR),
    "START": (0, Keycode.X),
    "UP": (0, Keycode.W),
    "DOWN": (0, Keycode.S),
    "RIGHT": (0, Keycode.D),
    "LEFT": (0, Keycode.A),
    "L3": (0, Keycode.V),
    "R3": (0, Keycode.B),
    "L2": (0, Keycode.R),
    "R2": (0, Keycode.T),
    "L1": (0, Keycode.F),
    "R1": (0, Keycode.G),
    "TRIANGLE": (0, Keycode.I),
    "CIRCLE": (1, Mouse.LEFT_BUTTON), # for mouse clicks
    "CROSS": (0, Keycode.K),
    "SQUARE": (0, Keycode.L),
}

print("PlayStation Roulette Wheel controller")

while True:
    events = psx.update()
    if events:
        print(events)
```

©Adafruit Industries
for event in events:
    if buttonmap[event.name][0] == 0:  # regular button
        if event.pressed:
            keyboard.press(buttonmap[event.name][1])
        if event.released:
            keyboard.release(buttonmap[event.name][1])
    if buttonmap[event.name][0] == 1:  # mouse button
        if event.pressed:
            circle_held = True
        if event.released:
            circle_held = False
    if circle_held:  # spam the mouse click
        mouse.press(buttonmap["CIRCLE"])[1])
        mouse.release_all()
        time.sleep(spam_speed)

How It Works

The code uses the `ps2controller` library to interpret the PlayStation controller messages, then the `adafruit_hid` library is used to send USB keyboard commands and mouse clicks.

Libraries

First, the necessary libraries and modules are imported.

```python
import time
import board
import usb_hid
import neopixel
from adafruit_hid.keycode import Keycode
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.mouse import Mouse
from ps2controller import PS2Controller
```

NeoPixel Setup

The on-board NeoPixel is set up next to be used as a status indicator -- amber during boot up and blue when a PSX/PS2 controller has been successfully connected.
led = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.1)
led.fill(0x331000)  # amber while we wait for controller to connect

Mouse and Keyboard Setup

Next, we create instances of Mouse, Keyboard, and KeyboardLayoutUS.

mouse = Mouse(usb_hid.devices)
keyboard = Keyboard(usb_hid.devices)
layout = KeyboardLayoutUS(keyboard)

PS2Controller

We'll created an object called psx that is an instance of the PS2Controller with the necessary pin configuration to use it on the QT Py. After the controller is set up the NeoPixel is set to blue.

psx = PS2Controller(
    dat=board.A0,
    cmd=board.A1,
    att=board.A2,
    clk=board.A3
)
led.fill(0x0010ee)

Variables

A boolean variable called circle_held is used to hold the state of the spinner -- while the O (circle) button is held (meaning the spinner is spinning) this variable is True and the rapid mouse button spamming will occur. When the spinner stops the mouse button stops spamming.

The spam_speed variable is used to adjust the mouse-click rate.

circle_held = False
spam_speed = 0.001

Buttonmap

The buttonmap dictionary is defined to map the controller button names to USB HID keycodes or mouse-clicks. The first item in each key is a 0 or 1 to indicate if keycodes or mouse events are sent, while the second item is the specific keycode or mouse event.
Main Loop

The main loop of the program first checks for any PSX controller events with `psx.update()`.

Each controller event is checked to see if it should trigger a keycode or mouse event.

The events are then sent as per the `buttonmap` mappings.

Keycodes are pressed and then released along with their corresponding buttons, while the O key that the spinner sends is treated specially -- it rapidly clicks the left mouse button, making it perfect for clicker games!

```python
while True:
    events = psx.update()
    if events:
        print(events)
        for event in events:
            if buttonmap[event.name][0] == 0:  # regular button
                if event.pressed:
                    keyboard.press(buttonmap[event.name][1])
                if event.released:
                    keyboard.release(buttonmap[event.name][1])
            if buttonmap[event.name][0] == 1:  # mouse button
                if event.pressed:
                    circle_held = True
                if event.released:
                    circle_held = False
                if circle_held:  # spam the mouse click
                    mouse.press(buttonmap["CIRCLE"][1])
                    mouse.release_all()
                    time.sleep(spam_speed)
```
Use the Controller

The above video shows how to use the controller -- simply plug the controller into the adapter, then plug the QT Py USB into the computer or mobile device USB host.

Spinning the spinner sends two bursts of variable mouse clicks, as shown in the video. Pressing any other buttons sends USB HID keycodes, which can be customized for your needs in the code.