Inhaltsverzeichnis

USB Keyboard and Gamepad Emulation for an old analog Joystick

The Internet Archive provides some great browser emulations for tons of old arcade games.

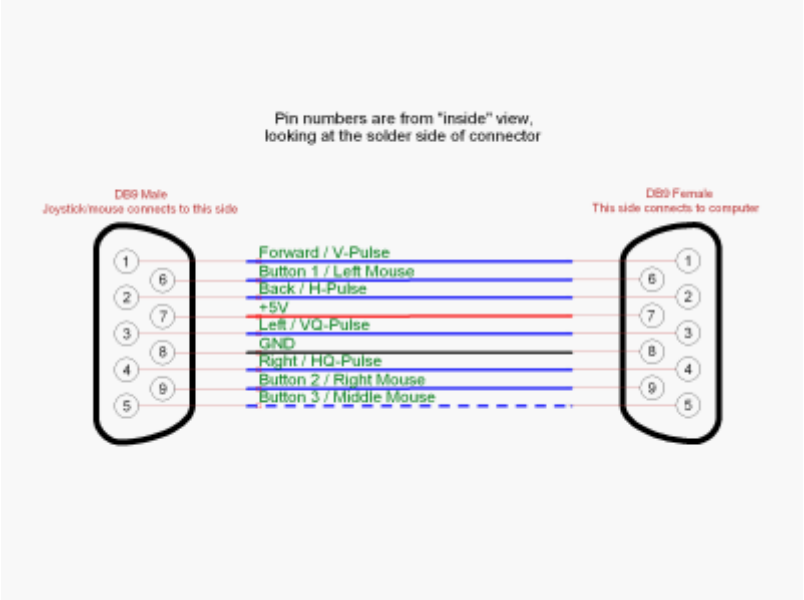
But for myself it came out as very disappointing, that the emulators can only be played with the keyboard, and any trials to connect a usb gamepad finally failed because of weak or no gamepad support of the emulator (JSMAME).

But inspired by the video from Lady Ada and the Adafruit DIY Pico Keypad tutorial I made the plan to use the great Raspi Pico as keyboard emulator for an old analog joystick. By that it came out that it's cheaper to internally modify a newer USB joystick as to buy an old one and make an external adapter.

But for the pico and for the software that does not make a difference. I connect the wires directly inside the joystick housing to the switches, but in the same way the wiring could be connected to a 9-way DSUB male connector to plug an old unmodified joystick into that.

That then gives the following connection layout

Pico Signal	Pico Pin	Function	Joystick dSub Pin	Keyboard Mapping	Gamepad Mapping
GP6	9	Forward / V-Pulse	1	Keycode.UP_ARROW	
GP7	10	Button 1 / Left Mouse	6	Keycode.LEFT_CONTROL	
GP8	11	Back / H-Pulse	2	Keycode.DOWN_ARROW	
do **not connect** this!		+5 Volt	7	-	-
GP9	12	Left / VQ-Pulse	3	Keycode.LEFT_ARROW	
GND	13	Ground	8	-	-
GP10	14	Right / HQ-Pulse	4	Keycode.RIGHT_ARROW	
GP11	15	Button 2 / Right Mouse	9	Keycode.LEFT_ALT	
GP12	16	Button 3 / Middle Mouse	5	Keycode.SPACEBAR	
GP13	17	reserved for permanent fire switch	do **not connect** this!	-	-



picture © https://retrocomputing.stackexchange.com/a/2081

Install Circuit Python and the Adafruit HID library

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.

Download the latest UF2 file from the circuitpython homepage and install it by drag & drop onto your Pico.

A new drive is shown as `CIRCUITPI`. Let this drive be connected, as Mu will try to find it during its installation

Installing the Adafruit HID Library

Download the library bundle.

Copy the adafruit_hid folder from the bundle to the lib folder on your CIRCUITPY drive.

raspberry_pi_picokeeblib.jpg Before continuing make sure your board's lib folder has the adafruit_hid library folder copied over.

Install the Mu python IDE

sudo apt install mu-editor

As `mode` choose `CircuitPython`

The Software

```
# stko: copied from
# SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
# SPDX-License-Identifier: MIT
# RaspberryPi Pico RP2040 Mechanical Keyboard

import time
import board
from digitalio import DigitalInOut, Direction, Pull
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode

print("---Pico Joystick emulator Keyboard---")

led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT
led.value = True

kbd = Keyboard(usb_hid.devices)


# list of pins to use (skipping GP15 on Pico because it's funky)
# and as I soldered my sample exactly the wrong way round :-( :-( :-(
# I need to adjust this here in the software by mirroring the pins :-(

# so if you want to use the software, rename this pins_original array back
to pins
# and delete the other, bugfix pin array
pins_original = [
    board.GP6,
    board.GP7,
    board.GP8,
    board.GP9,
    board.GP10,
    board.GP11,
    board.GP12,
    board.GP13,
]
# this is a bugfix workaround pins table to correct the wrong soldering :-(
pins = [
    board.GP13, #up
    board.GP12, #Lbutton 1
    board.GP8, # down
    board.GP9, # left
    board.GP10, # right
    board.GP11,  # Rbutton 2
    board.GP7,  # MButton 3
    board.GP6, # no key
```

```
]

keymap = [
    Keycode.UP_ARROW,
    Keycode.LEFT_CONTROL,
    Keycode.DOWN_ARROW,
    Keycode.LEFT_ARROW,
    Keycode.RIGHT_ARROW,
    Keycode.LEFT_ALT,
    Keycode.SPACEBAR,
]

nr_of_keys = len(keymap)
switches = []
switch_state = []

for i in range(nr_of_keys):
    switch_state.append(0)
    switches.append(DigitalInOut(pins[i]))
    switches[i].direction = Direction.INPUT
    switches[i].pull = Pull.UP

while True:
    for button in range(nr_of_keys):
        if switch_state[button] == 0:
            if not switches[button].value:
                print(button, 'on')
                try:
                    kbd.press(keymap[button])
                except ValueError:  # deals w six key limit
                    print('e1')
                    pass
                switch_state[button] = 1

        if switch_state[button] == 1:
            if switches[button].value:
                print(button, 'off')
                try:
                    kbd.release(keymap[button])

                except ValueError:
                    print('e1')
                    pass
                switch_state[button] = 0

    time.sleep(0.01)  # debounce
```

The Hardware

Build into the Competition Pro: