# Table of Contents

## Overview 3
- Features:
- Materials & Tools:
- Materials:

## Mod the Pedal 6
- How it works

## Installing CircuitPython 13
- CircuitPython Quickstart
- Safe Mode
- Flash Resetting UF2

## Code the Pedal 16
- Text Editor
- Download the Project Bundle
- How It Works

## Use the Pedal 21

## Use as Controller (optional) 22
- Custom GP2040-CE Software
- Button Layout
- Input Modes
- Web Configurator
- Game Time
Overview

If your little one has the Fisher-Price Deluxe Kick & Play Piano Gym, then you will know all about the "purple monkey in a bubble-gum tree" ♫ song and the endless hours of entertainment. Its a great sensory toy but when it reaches its end of life, it can be easily modded into a pretty durable 5-button USB foot pedal.

This guide will show you how to turn this Fisher-Price Kick Piano into a real USB controller with an easily configurable keyboard mode. Map extra buttons to your feet when gaming or create shortcuts for repetitive work tasks. With just a KB2040, a 3D printed mount, and some basic soldering skills, you can give this toy new life as a productivity tool.

Features:

- **Keyboard + Macros**: CircuitPython code simulating keyboard output.
  - Five programmable buttons
  - Single button press
  - Combo button press
  - Macro sequence of single/combo button presses
- **Controller with GP2040**: Optional open source gamepad firmware.
  - Multiple controller modes (X-Input, DirectInput, Nintendo Switch)
  - Custom button mapping (default: 1=Left, 2=Right, 3=Start, 4=B, 5=A)
  - Web Configurator
Materials & Tools:

- Deluxe Kick & Play Piano Gym
- An Adafruit KB2040 microcontroller
- About 4-5ft of jumper wire (Thin 30 AWG wire wrap recommended)
- A soldering iron and solder
- Flush angled cutter pliers
- A drill and 1/8 in drill bit (optional)
- 3D printed cutting jig (optional)
- A multimeter (optional)
- Phillips screwdriver
- USB-C cable

Materials:

Fisher-Price Deluxe Kick & Play Piano Gym
Made by Mattel (), available at Amazon and other resellers online.

Adafruit KB2040 - RP2040 Kee Boar Driver
A wild Kee Boar appears! It’s a shiny KB2040! An Arduino Pro Micro-shaped board for Keebs with RP2040. (#keeblife 4 evah) A lot of folks like using Adafruit... https://www.adafruit.com/product/5302
Rainbow "Wire Wrap" Thin 30 AWG Prototyping & Repair Wire
This stuff is called "wire-wrap wire" because it used to be used for wire-wrapping high-speed digital circuits on a special kind of contact board. It's pretty rare to see...
https://www.adafruit.com/product/4730

USB Type A to Type C Cable - approx 1 meter / 3 ft long
As technology changes and adapts, so does Adafruit. This USB Type A to Type C cable will help you with the transition to USB C, even if you're still...
https://www.adafruit.com/product/4474

If needed/desired:

Flush diagonal cutters
These are the best diagonal cutters, large super-comfortable grip to use and have strong nippers for perfect trimming of wires and leads. I've used my pair every day for years.
https://www.adafruit.com/product/152
Digital Multimeter - Model 9205B+
This massive multimeter has everything but the kitchen sink included. It's a great addition to any workbench or toolbox. It's low cost, simple to use, and has a big clear...
https://www.adafruit.com/product/2034

Mod the Pedal

How it works

The piano has five piano keys with a button underneath each. These can be easily tapped into by soldering a wire directly to one of the pins on each button board. When a key is pressed the voltage signal of the corresponding button wire will go high.

By connecting each of these buttons to GPIO pins of a microcontroller like the Adafruit KB2040, you can programmatically detect a button press when the voltage goes from low to high. (ex: 0 → 3.3v)
Disassemble

Start off by removing the 16 Phillips-head screws along the bottom side of the piano.

Gently pull apart the two halves with the top half remaining facedown. This will help prevent the plastic piano keys from falling out of place.

Mount the Microcontroller

There is a lot of free space within this toy to mount your microcontroller. So you have a lot of options. For this guide we'll use a 3D printed mount for holding the KB2040 and cutting a small opening for exposing the USB-C port within the barrel on the far right end.

Download STL Files

The 3D printed mount fits the KB2040 stealthily within either the left or right ends (used to screw the piano to the rest of gym). This way the piano can be easily restored to normal function later down the road and the cuts will remain hidden.

Mounting to the right-side is ideal as it is close enough to reach the existing power wires.
Cut the Case

Using the 3D printed jig you can drill some initials holes on either end and then use flush cutter pliers to trim out the remaining.

Roughly a 9mm x 2.8mm opening for the KB2040's USB-C port.
Fit the Mount

Slide the button end of the KB2040 into the notches of the mount.

Then slide the mount and KB2040 into place to verify the USB-C connector fits within the hole cut.

Connect the Power

To make sure the piano is within a safe operating range for the KB2040 GPIO to read input from, you can power the piano by USB through the KB2040’s built-in voltage regulated 3.3v pin.

Start by cutting the red and black wires coming out of the battery compartment.
Cut as close to the plastic as possible to ensure there is enough length to reach the microcontroller.

Then solder the red wire to the 3V pin and the black wire to a GND pin of the KB2040.

Wire the Boot Button (optional)

Optionally the "Try Me" button, which is normally triggered by the packaging, can be repurposed into an externally accessible boot button for the KB2040.

Start by cutting the two wires for the "Try Me" button close to the main PCB of the piano.
Then solder either one of leads to the small circle solder pad on the bottom of the KB2040 and other to one of the ground pins.

Make sure to route the wire going to the underside of the KB2040, through the small slot under the mount.

Having this boot button wired for external access will come in handy later if you would ever like to flash a different firmware without having to completely disassemble the whole piano again.
Wire the Keys

Each piano key button board has three wires connected to it. The lead with the white wire, closest to the button membrane, carries the button's state signal. It is 3.3v when pressed and 0v when released.

For each button, cut a wire long enough to run between it and the mounted KB2040 board. Then solder the wire to the white wire's pin hole.

<table>
<thead>
<tr>
<th>PIANO KEY</th>
<th>KB2040 PIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red - 1</td>
<td>A0 - Left</td>
</tr>
<tr>
<td>Orange ♥ - 2</td>
<td>A2 - Right</td>
</tr>
<tr>
<td>Yellow ▲ - 3</td>
<td>CLK - Start</td>
</tr>
<tr>
<td>Green ■ - 4</td>
<td>D2 - Button 1</td>
</tr>
<tr>
<td>Blue ★ - 5</td>
<td>D3 - Button 2</td>
</tr>
</tbody>
</table>

Solder the other end of the wires to the corresponding KB2040 pin labels in the chart above.

Button 1 (red) to pin A0.
Button 2 (orange) to pin A2.
Button 3 (yellow) to pin CLK.
Button 4 (green) to pin 2.
Button 5 (blue) to pin 3.
Installing CircuitPython

CircuitPython is a derivative of MicroPython 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 running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org

Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset
button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

A lot of people end up using charge-only USB cables and it is very frustrating! 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! :)

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.

Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

Entering Safe Mode

To enter safe mode when using CircuitPython, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)
In Safe Mode

If you successfully enter safe mode on CircuitPython, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.
CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.
Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading 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 loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

Code the Pedal

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in [this guide](#).

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 KB2040 into the computer via USB.

```python
# SPDX-FileCopyrightText: 2023 Robert Dale Smith for Adafruit Industries
# SPDX-License-Identifier: MIT
# The Fisher-Price Kick and Play Piano Gym has five buttons that are
# active high. Pressed = 1, Released = 0. This code turns that into
# keyboard key press, key combos, and/or key press/combo macros.

import time
import board
import usb_hid

from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from digitalio import DigitalInOut, Direction, Pull

# Set up a keyboard device.
kbd = Keyboard(usb_hid.devices)
layout = KeyboardLayoutUS(kbd)

# Setup the buttons with internal pull-down resistors
buttons = []
for pin in [board.A0, board.A2, board.CLK, board.D2, board.D3]:  # kb2040 pins
    button = DigitalInOut(pin)
    button.direction = Direction.INPUT
    button.pull = Pull.DOWN
    buttons.append(button)

# Each button corresponds to a key or key combination or a sequence of keys
keys = [
    Keycode.A,
    (Keycode.COMMAND, Keycode.TAB),
    [Keycode.UP_ARROW, Keycode.ENTER],
    [Keycode.END, (Keycode.SHIFT, Keycode.HOME), (Keycode.COMMAND, Keycode.C)],
    [(Keycode.CONTROL, Keycode.A), 'Hello World', Keycode.PERIOD]
]

while True:
    # check each button
    for button, key in zip(buttons, keys):
        button.key_press(key)
```

©Adafruit Industries
if button.value:  # button is pressed
    if isinstance(key, tuple):
        kbd.press(*key)
        kbd.release_all()
    elif isinstance(key, list):
        for macro_key in key:
            if isinstance(macro_key, str):  # print a string
                layout.write(macro_key)
            elif isinstance(macro_key, tuple):  # press combo keys
                kbd.press(*macro_key)
                kbd.release_all()
            else:  # press a single key
                kbd.press(macro_key)
                kbd.release_all()
        time.sleep(0.1)  # delay between keys
    else:  # press a single key
        kbd.press(key)
        kbd.release_all()
        time.sleep(0.1)  # debounce delay

time.sleep(0.1)

How It Works

The code uses the `digitalio` library to read the piano key button presses, then the `adafruit_hid` library is used to send USB keyboard commands.

Libraries

First, the necessary libraries and modules are imported.

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

Keyboard Setup

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

©Adafruit Industries

Page 18 of 25
# Set up a keyboard device.
kbd = Keyboard(usb_hid.devices)
layout = KeyboardLayoutUS(kbd)

Digital In Out

We'll create an array named `buttons`. This array will store instances of the `DigitalInOut` class, each configured to correspond with one of the five KB2040 pins connected to the piano keys.

Each pin direction will be set to `INPUT`. The piano keys are active high. Meaning the voltage goes from 0 to 3.3v when a button is pressed and returns to 0 when released. So the default state of each pin is set to `Pull.DOWN`.

```python
# Setup the buttons with internal pull-down resistors
buttons = []
for pin in [board.A0, board.A2, board.CLK, board.D2, board.D3]:
    button = DigitalInOut(pin)
    button.direction = Direction.INPUT
    button.pull = Pull.DOWN
    buttons.append(button)
```

Key Commands

The `keys` array contains five elements, one for each piano key button.

Each item can be either a single keyboard key press, a combination of simultaneously pressed keys, or a sequenced macro of individual or combo key presses.

This is where you can customize the individual piano keys' commands to fit your own needs.

For example:

- Single - press a single key
  - example: `Keycode.A`
- Combo - press a combination of keys simultaneously
  - example: `(Keycode.COMMAND, Keycode.TAB)`
- Macro - press a sequence of single keys or combos one after the other
  - example: `[ Keycode.UP_ARROW, Keycode.ENTER ]`
- Macro with a combo, a string, and a single key
  - example: `[ (Keycode.CONTROL, Keycode.A), 'Hello World', Keycode.D ]`

```python
# Each button corresponds to a key or key combination or a sequence of keys
keys = [
    Keycode.A,  # single key
    (Keycode.COMMAND, Keycode.TAB),  # key combo
    [Keycode.UP_ARROW,
     Keycode.ENTER],  # macro of single key presses
    Keycode.END,
    (Keycode.SHIFT, Keycode.HOME),
    (Keycode.COMMAND, Keycode.C),
],  # macro with single key and combo keys
[  
    (Keycode.CONTROL, Keycode.A),
    'Hello World',
    Keycode.PERIOD
]  # macro with combo, string, and single
```

**Main Loop**

The main loop of the program first checks the `button.value` of each of the five buttons wired up to the GPIO pins of the KB2040.

Each button's corresponding keys array key is checked to see if it is either a tuple `()`, a list `[]`, a single key `Keycode.`, or a string of key characters `'Hello World'`.

Tuples, values wrapped in parentheses `()` will send simultaneous button combo presses. Lists, values wrapped in square brackets `[]` will send a macro sequence of individual or combo keys one after the other.

`Keycode.` values or strings wrapped in single quotes `'` can be used for pressing one or more keys in sequence.

```python
while True:
    # check each button
    for button, key in zip(buttons, keys):
        if button.value:  # button is pressed
            if isinstance(key, tuple):
                kbd.press(*key)
                kbd.release_all()
            elif isinstance(key, list):
                for macro_key in key:
                    if isinstance(macro_key, str):  # print a string
                        layout.write(macro_key)
                    elif isinstance(macro_key, tuple):  # press combo keys
                        kbd.press(*macro_key)
                        kbd.release_all()
```

© Adafruit Industries
else:  # press a single key
    kbd.press(macro_key)
    kbd.release_all()
    time.sleep(0.1)  # delay between keys
else:  # press a single key
    kbd.press(key)
    kbd.release_all()
    time.sleep(0.1)  # debounce delay

time.sleep(0.1)

Use the Pedal

Finally, let's test and make sure everything is working before fully re-assembling.

In order to properly read the piano key button state, the piano's audio switch needs to be turned on to either of the two volume settings.

Then connect the piano via USB to a computer and open your favorite text editor.

Using the CircuitPython code in this guide as-is:

Button 1 will type a lowercase 'a' character.
Button 2 will tab between active windows.
Button 3 will fire up and enter keys to easily to repeat the last terminal command.
Button 4 will highlight and copy the current line of code.
Button 5 will type "Hello World." at the start of the line.
Congratulations! You have successfully created a keyboard foot pedal with programmable macros.

Customize the code in this example to match your own workflow. Whether its gaming, video editing, or programming, there are so many new uses for this old toy.

**Use as Controller (optional)**

The open source gamepad firmware [GP2040-CE](#) can be used to turn this piano into a controller compatible with just about everything. PC, Mister, PS4, and even Nintendo Switch consoles.

In fact, the same exact firmware used for the [Fisher-Price USB Controller](#) mod will just work with the wiring in this guide.

**Custom GP2040-CE Software**

GP2040-CE out of the box reads button presses in the opposite direction of how the Fisher-Price controller behaves. (pressed = LOW; released = HIGH)

Luckily with it being open source, it was easily modified to support this. And a precompiled UF2 firmware file of this custom build is available by clicking the download button below.

[Download UF2 Firmware File](#)
1. Download UF2 firmware file
2. Hold the boot button while connecting the USB cable to your computer
3. A virtual drive will in your computer File Explorer or Finder (depending on your operating system) named RPI-RP2
4. Release the boot button once the drive shows up
5. Drag-and-drop the UF2 file onto the RPI-RP2 drive
6. The drive will auto disconnect when update is complete

Button Layout

By default, the wiring within this guide will map buttons to:

<table>
<thead>
<tr>
<th>PIANO KEY</th>
<th>X-INPUT MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red  - 1</td>
<td>Left</td>
</tr>
<tr>
<td>Orange ♥ - 2</td>
<td>Right</td>
</tr>
<tr>
<td>Yellow ▲ - 3</td>
<td>Start</td>
</tr>
<tr>
<td>Green ■ - 4</td>
<td>B Button</td>
</tr>
<tr>
<td>Blue ★ - 5</td>
<td>A Button</td>
</tr>
</tbody>
</table>
Input Modes

The GP2040-CE software supports multiple input modes for use on a wide array of devices. In order to change input modes, simply hold either button 3, 4, or 5 while connecting the USB cable.

3 - Web Configurator (web UI to select other modes)
4 - X-Input Mode
5 - Nintendo Switch Mode

Web Configurator

The GP2040-CE software also has a built-in web server that can be enabled by holding the start button (button 3) while connecting the USB cable to a computer.

Then access http://192.168.7.1 in a web browser on your computer to begin configuration. This mode is compatible with Windows, Mac, Linux and SteamOS.

The web configurator can be used to remap the buttons to anything. But be aware that if no key is mapped to the Start (S1) button, then you will no longer be able to access the web configurator. Access can be restore by nuking the KB2040 flash.
Game Time

Now its time to play some games. Although the button mapping is limited to only five keys, there are actually plenty of retro games this plays well with. Pair it with a mouse and keyboard on PC and it can double as secondary foot pedal input.