Crankfruit

Created by John Park

https://learn.adafruit.com/crankfruit

Last updated on 2026-10-19 10:09:52 PM UTC

Table of Contents

Overview 3

- Credits
- Parts

Circuit 7

- Wire Prep
- Heat Shrink Tubing
- Connect to Encoder
- Join Cables

3D Printed Enclosure 12

Assembly 14

- Fasten Encoder to Lid
- Snap Parts Together
- Fruit Jam Case
- Plug in Crank

Code 20

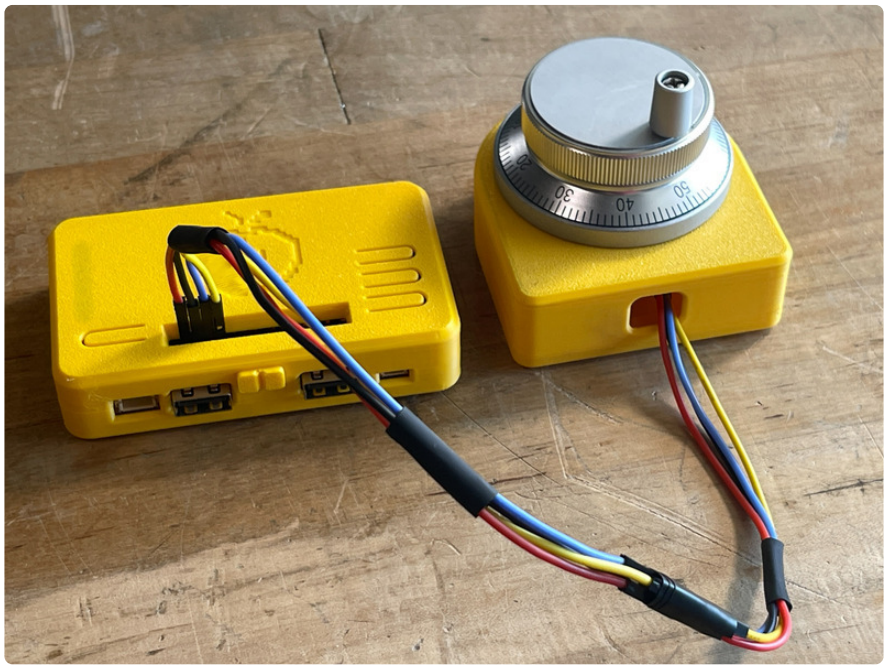
- Installation
- How it Works
- Limitations

Play 23

- Connections

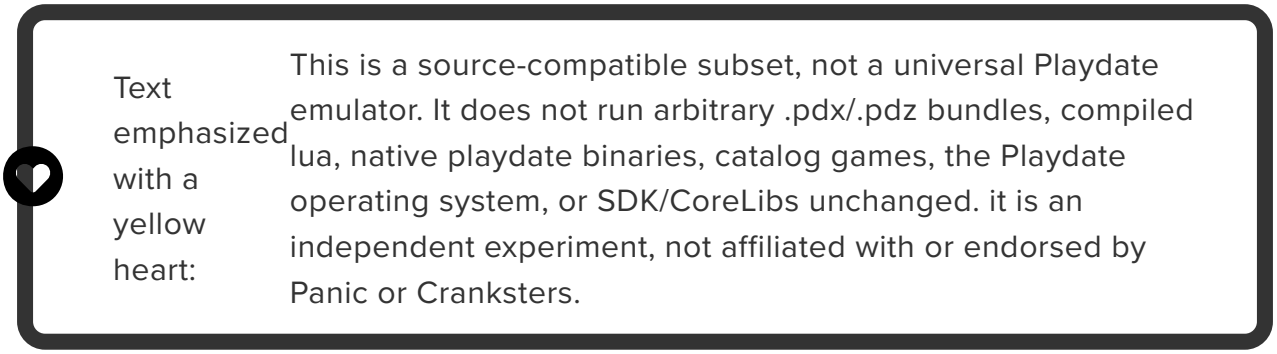
Overview

Note: the example video shows an old version with STEMMA QT cable connection, the current uses GPIO pins to connect the crank as shown in the photos, steps, and diagrams.



Play with this unofficial, experimental Playdate-inspired CircuitPython game console on your TV using a controller and custom crank!

Crankfruit is a standalone game player for the Adafruit Fruit Jam that runs Playdate-style Lua games from source, with crisp one-bit graphics and synth sound over HDMI. Play with a USB PlayStation controller, and add a crank with a 100-pulse CNC handwheel. Then scale the picture up to 1.6× with optional antialiasing at the press of a button.

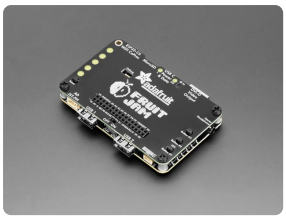


Credits

- Almost Tennis: adapted from https://github.com/SquidGodDev/almost-tennis () (`69172662a659849c8b452e257c1f41f8cd240ad8`), MIT, SquidGod.
- Simple Snake: adapted from https://gitlab.com/mocchimochi/playdate/playdate-snake-game () (`c9cc2a924dcbe31156529ae37326940360865530`), MIT, Mocchi Mochi.
- Playdate file-format research: https://github.com/cranksters/playdate-reverse-engineering () (`e7be3ff0141b7cd3b9f008737eda1f0539aca45c`), CC0. Format notes are community documentation, not an SDK compatibility guarantee.
- CircuitPython, Lua 5.4.9, TinyUSB, Pico-PIO-USB, `fast_float` , `stb_sprintf` and other components: see **engine/licenses/** and the notices in their source folders.

Playdate is a trademark of Panic Inc.

Parts



Featured

Adafruit Fruit Jam - Mini RP2350 Computer

We were catching up on a recent hackaday hackchat with eben upton and learned some fun facts:...

https://www.adafruit.com/product/6200

HDMI Cable - 1 meter

Connect two HDMI devices together with this basic HDMI cable. It has nice molded grips for easy installation, and is 1 meter long (about 3 feet). This is a HDMI 1.3...

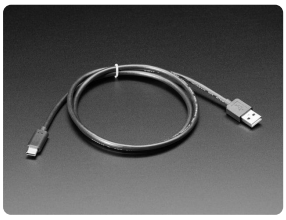
https://www.adafruit.com/product/608

Featured

7" Display 1280x800 (720p) IPS + Speakers - HDMI/VGA/NTSC/PAL

Yes, this is an adorable small HDMI television with incredibly high resolution and built in 3W stereo speakers! We tried to get the smallest possible HDMI/VGA display with...

https://www.adafruit.com/product/1667



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...

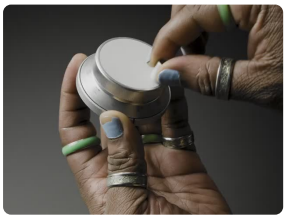
https://www.adafruit.com/product/4474

Featured

USB Game Controller with SNES-like Layout

This is a generic USB game controller, which plugs into to provide a two-handed gaming experience for retro gaming, or really any game you want to use a handheld rather than...

https://www.adafruit.com/product/6285

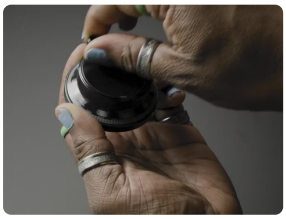


Featured

CNC Rotary Encoder - 100 Pulses per Rotation - 60mm Silver

We have various PCB-mount rotary encoders in the shop, great for making interfaces where you may want to spin a knob...

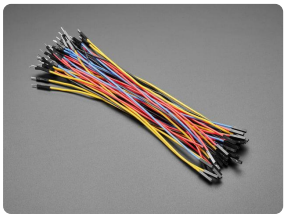
https://www.adafruit.com/product/5735



CNC Rotary Encoder - 100 Pulses per Rotation - 60mm Black

We have various PCB-mount rotary encoders in the shop, great for making interfaces where you may want to spin a knob...

https://www.adafruit.com/product/5734



Featured

Premium Silicone Covered Extension Jumper Wires - 200mm x 40

These premium extension jumper wires are handy for making wire harnesses or jumpering between headers on PCBs. They're 200mm (~7.8") long and come loose as a pack of...

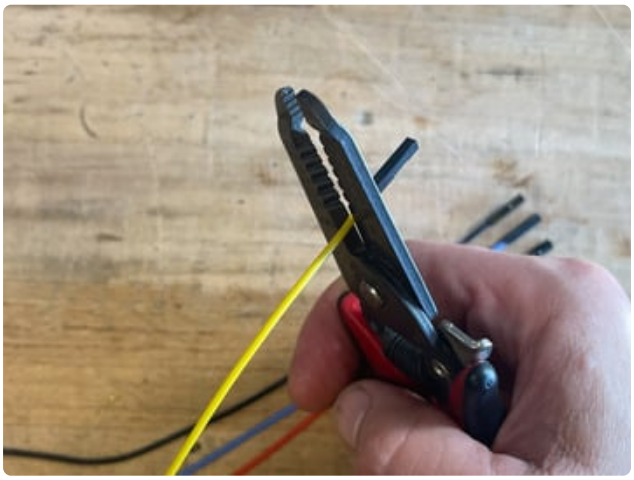
https://www.adafruit.com/product/4635

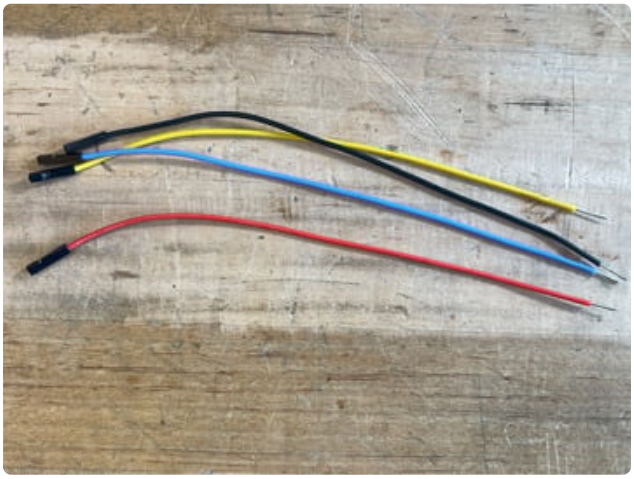
Pre-Cut Multi-Colored Heat Shrink Pack Kit - 280 pcs

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

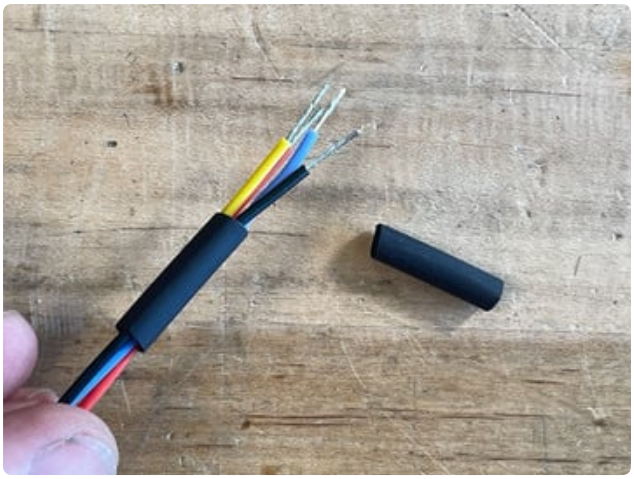
Circuit





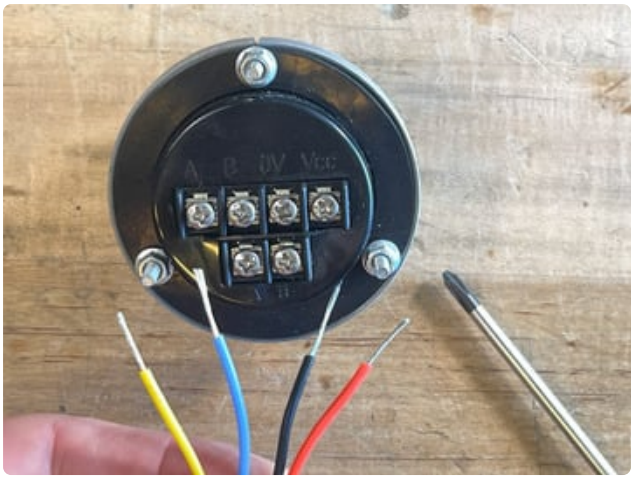
Wire Prep

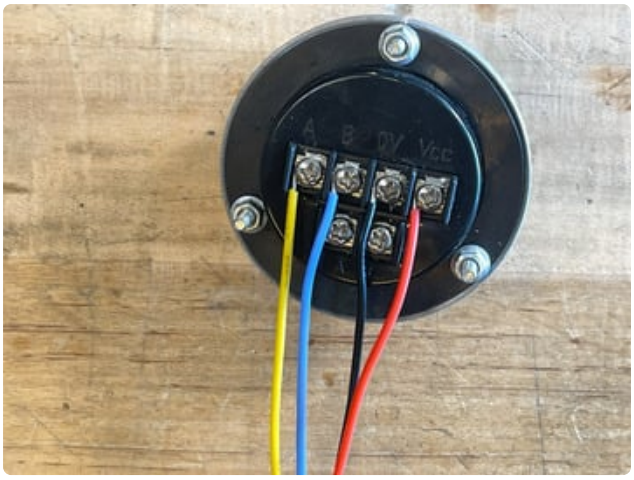
Prep four jumper wires with Dupont socket connectors (one each red, black, blue, yellow) by snipping off one end and then stripping about 1/2" of insulation.

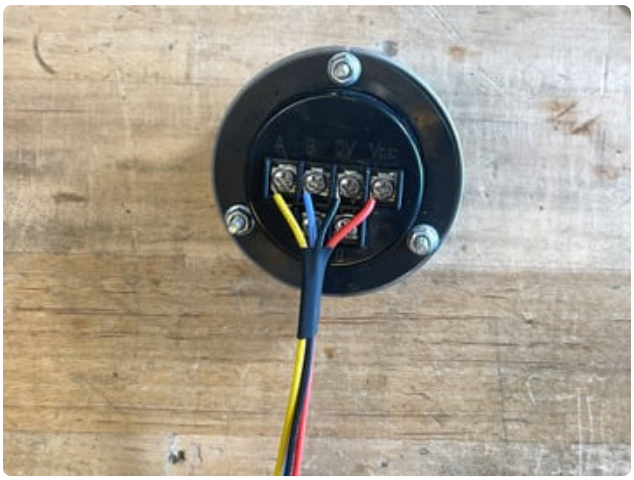


Heat Shrink Tubing

Slide three lengths of heat shrink tubing over the four-wire jumper cable bundle. The tubing near the DuPont connectors should be just wide enough to fit over all four ends. (See photos below)







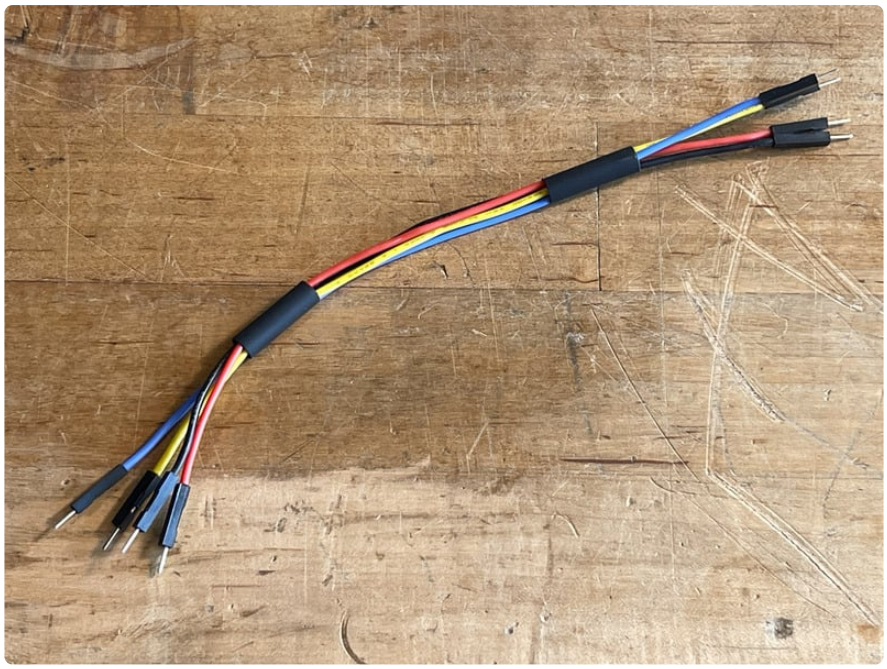
Connect to Encoder

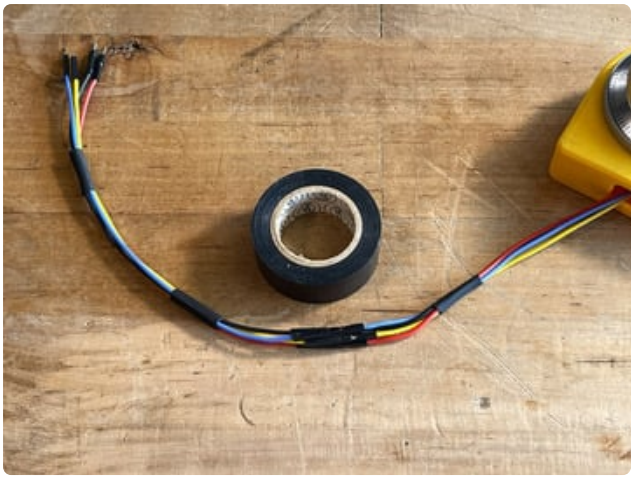
Loosen the four screws in the main row of the encoder, then hook each wire over it's screw. Use a Philips screwdriver to tighten them, making sure stray strands are not shorting any positions.

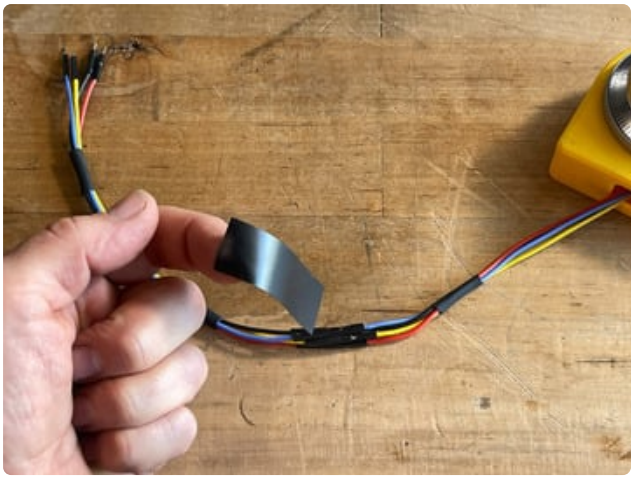
A = **Yellow**
B = **Blue**
0V = **Black**
Vcc = **Red**

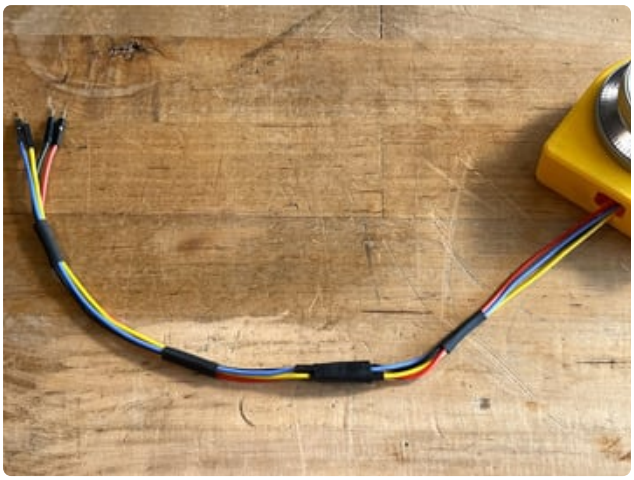
Slide one piece of the heat shrink tubing up and heat it to, you know, shrink it.

Slide two sections of heat shrink tubing over the four Dupont pin connector cables.









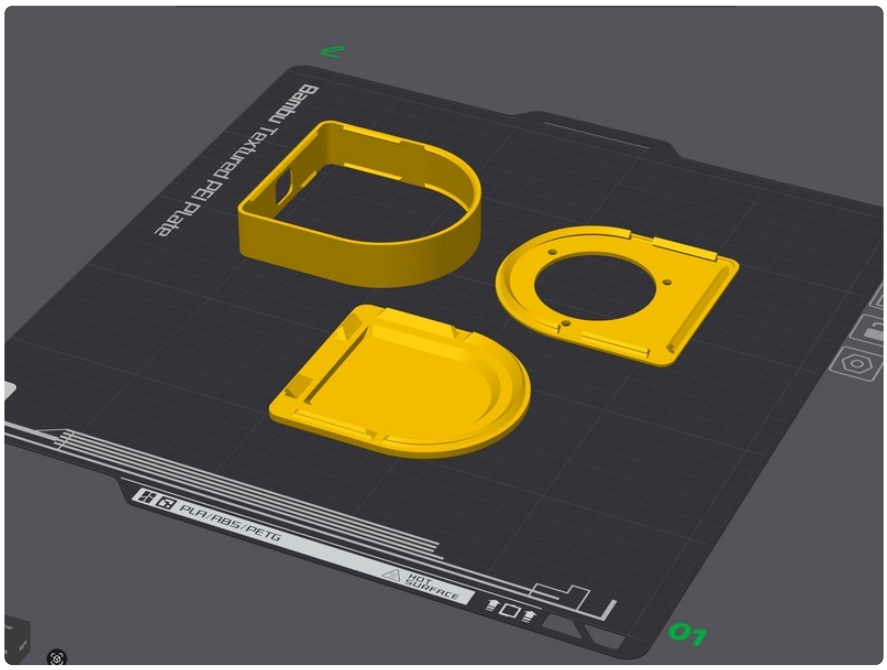
Join Cables

Connect each Dupont pin connector to it's associated, same-color socket connector on the encoder cable (these photos show the 3D printed case we'll create on the next page).

Secure the connection with heat shrink tubing or electrical tape.

The encoder cable is now ready to be fit into a 3D printed enclosure and used on the Fruit Jam.

3D Printed Enclosure



3MF files for 3D printing are oriented and ready to print on FDM machines using PLA filament.

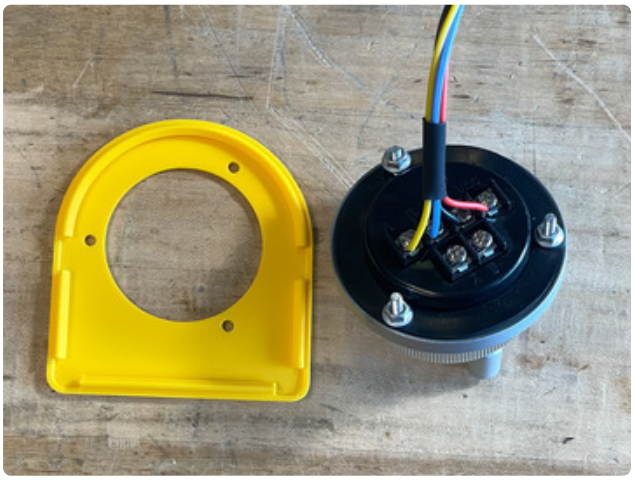
This case is a remix of the Ruiz Bros. CNC Rotary Macropad () enclosure design.

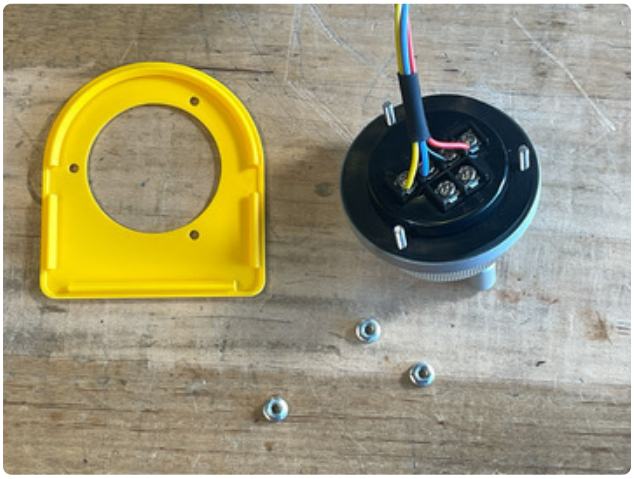
Slice with settings for PLA material

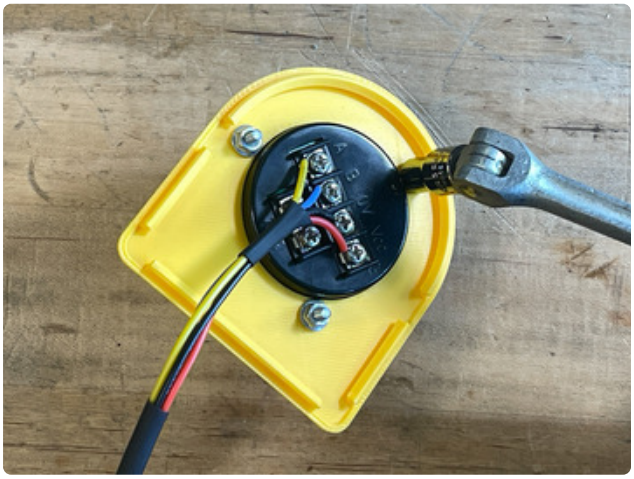
The parts were sliced using BambuStudio using the slice settings below.

- PLA filament 220c extruder
- 0.2 layer height
- 15% grid infill
- 200mm/s print speed

Assembly



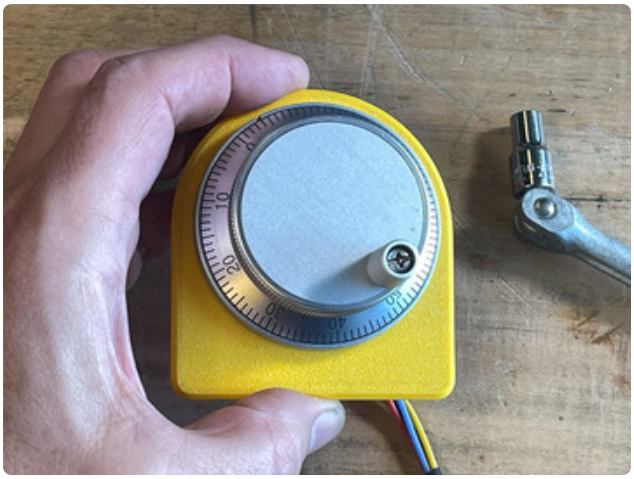


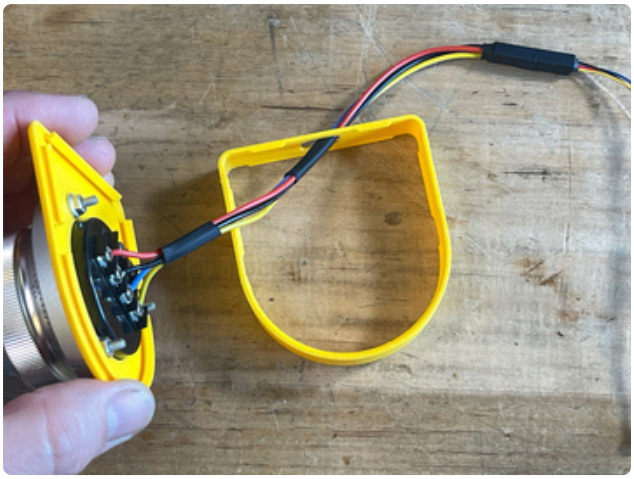


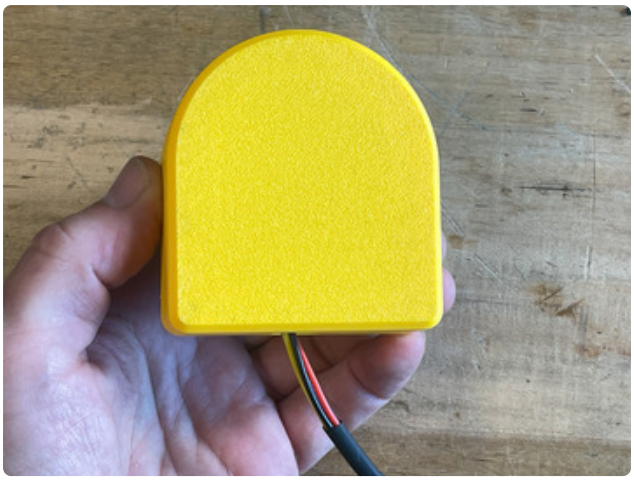
Fasten Encoder to Lid

Unscrew the three nuts from the encoder's screws.

Fit the encoder onto the lid as shown, then screw down the nuts to fasten.


10
20
30
40



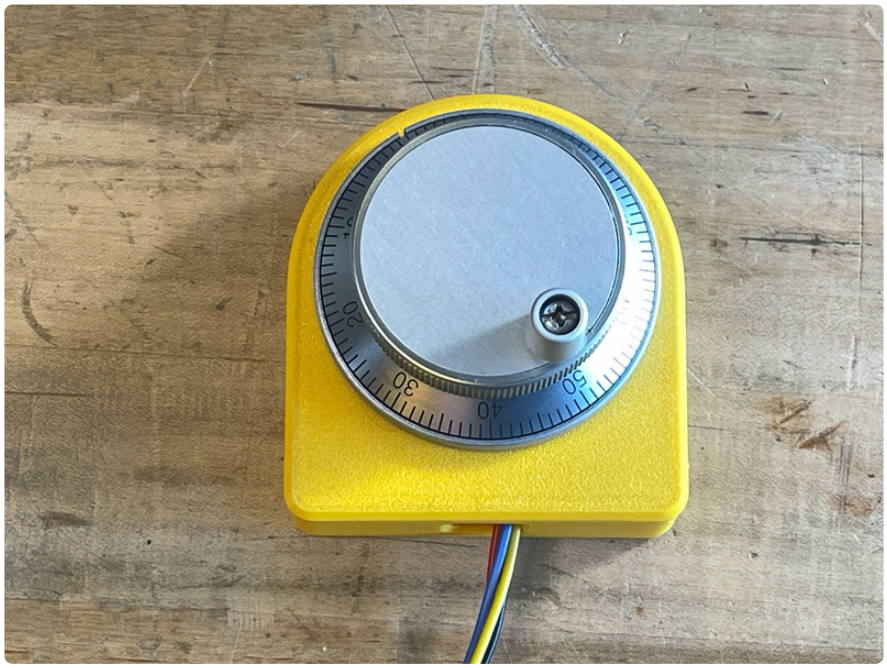


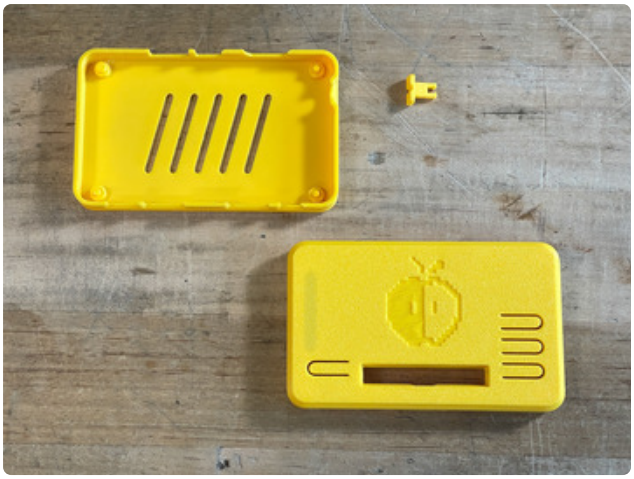
Snap Parts Together

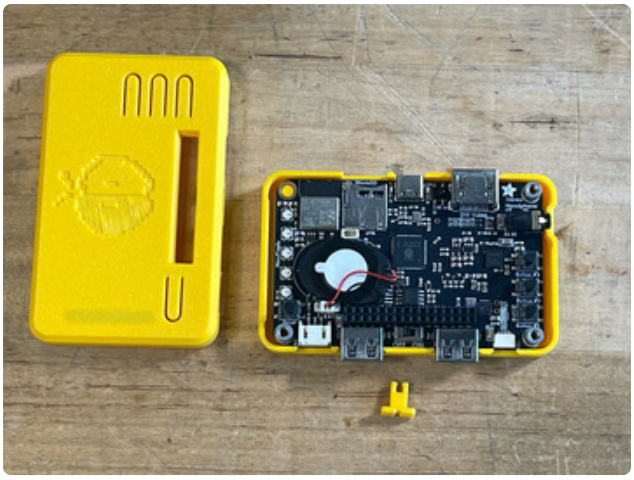
Feed the cable through the case hole, then snap the lid onto the case.

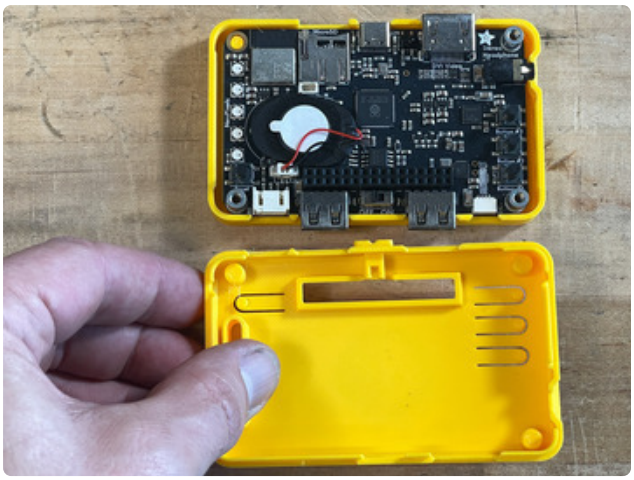
Snap the bottom onto the case.

60
50
40
30


10
20
30
40
50

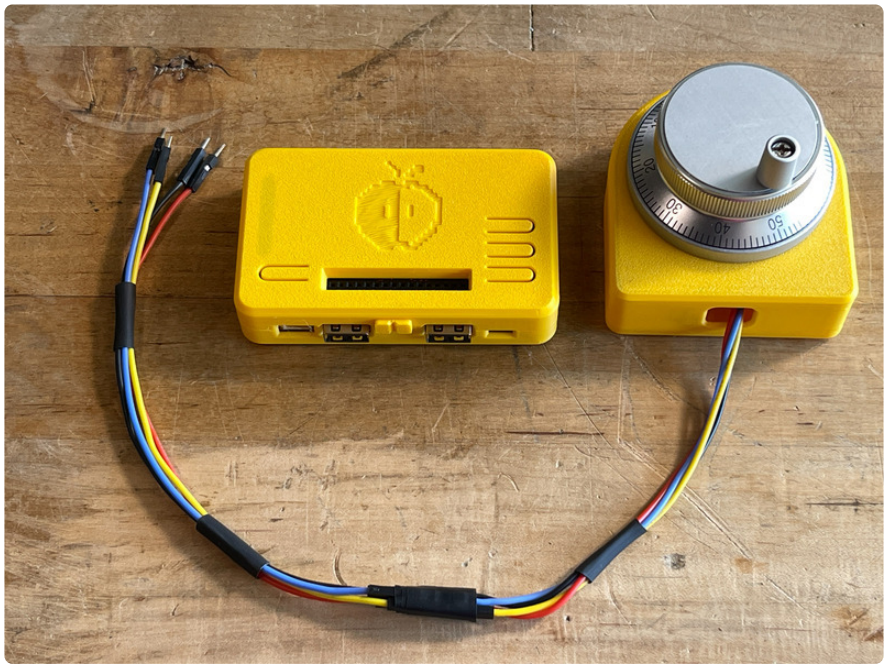


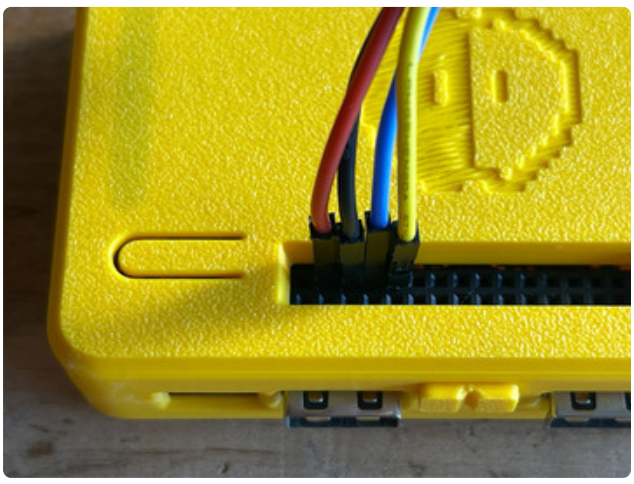




Fruit Jam Case

Print and assemble the optional Fruit Jam case as shown in this guide () by the Ruiz Brothers.

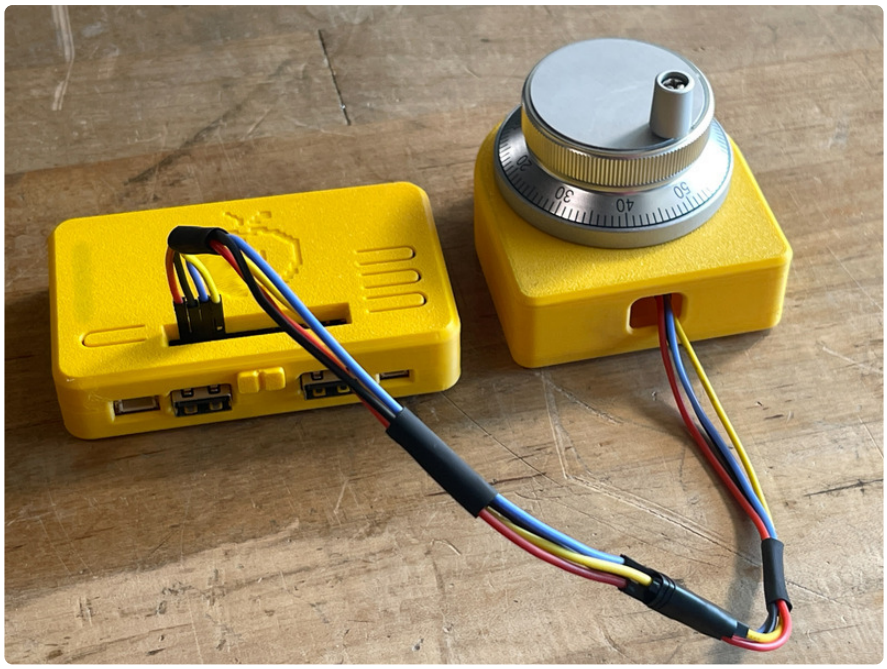




Plug in Crank

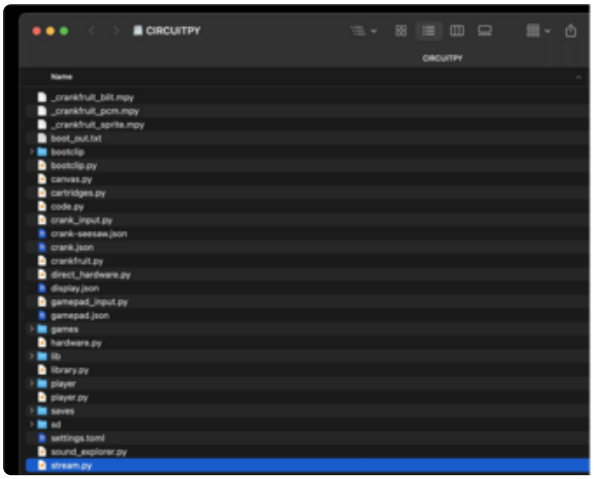
Plug the crank's cable into the Fruit Jam's GPIO header:

Red to **3V**
Black to **GND**
Blue to **A1**
Yellow to **A2**



Code

The source code for Crankfruit lives here ().



Installation

Download the .uf2 file from the latest Releases ().

To enter the bootloader, hold the **BOOT** button on the Fruit Jam then press/release the **RESET** button and finally release the **BOOT** button.

The **RPI-RP2** drive will appear on your desktop.

Drag the **crankfruit-fruit-jam.uf2** file onto the **RPI-RP2** drive. I'll dissapear after the file copies over and a new drive named **CIRCUITPY** will appear.

How it Works

Crankfruit is not stock CircuitPython plus a few libraries. It has three layers:

1. **Custom CircuitPython firmware** (**firmware/crankfruit-fruit-jam.uf2**, built from CircuitPython 10.3 with the patches in **engine/firmware/**):

    - the `crankjam` **Lua engine** is compiled into the firmware as a C module (`USER_C_MODULES`): Lua 5.4.9 plus Crankfruit's Playdate-style API for drawing, text, buttons, crank, time and synth sound (**engine/src/**, **engine/backends/circuitpython/**).
    - **HDMI audio:** the `picodvi` display driver gains `audio_write()`, `audio_pending` and `audio_stop()`, sending 32kHz PCM to the TV inside the HDMI signal (based on pico-hdmi's packet code).
    - **native .mpy loading** is enabled so the app can import compiled C helpers.
    - small fixes in CircuitPython's USB host (`usb.core`), TinyUSB and Pico-PIO-USB, and a flash layout that reserves 2MiB for the firmware.

    The firmware is reproducible: **engine/firmware/prepare.py** and **build.py** rebuild it from pinned public sources (see **engine/firmware/README.md**).

2. **Native helpers** (**app/_crankfruit_*.mpy**): small C functions compiled for the RP2350 as CircuitPython native modules (MPY ABI 0x1f06, ARMv7EMSP) from **native/**, for work too slow in Python:

    - `_crankfruit_blit`: copies and scales each frame into the HDMI framebuffer (1x, 1.6x sharp, 1.6x antialiased) and skips unchanged frames.
    - `_crankfruit_sprite`: draws the menu artwork and sprites.
    - `_crankfruit_pcm`: moves audio buffers without Python per-sample loops.

3. **The CircuitPython app** (**app/**, installed to CIRCUITPY): the player itself, in Python.

    - **crankfruit.py** runs the scenes: startup animation, library, games, Sound Explorer.
    - **cartridges.py** finds games in **/games/** and checks their manifests and source without running them.
    - **player.py** keeps everything at 30 frames per second and mixes audio a few frames ahead.
    - **stream.py** feeds that audio to HDMI.
    - **hardware.py** / **direct_hardware.py** set up HDMI and present frames.
    - **gamepad_input.py and crank_input.py read the USB controller and crank.**

Each frame: the player reads the buttons, gamepad and crank, then calls the game's `playdate.update()` inside `crankjam`. The Lua game draws into a 400x240 one-bit framebuffer and plays notes on its synth voices. The engine mixes the audio, which **stream.py** queues for HDMI; `_crankfruit_blit` copies and scales the frame onto the TV.

Games are plain Lua source. When a game starts, the engine compiles it with Lua's own parser in a fresh, sandboxed interpreter with a memory cap and an instruction budget per frame. Compiled Lua and native code are refused, and unknown Playdate APIs raise an error.

The C/Lua engine also builds on a computer, which provides the desktop player and the engine's test suite.

Limitations

- **Not an emulator.** Only compatible Lua source games run; compiled Playdate games, CoreLibs, sprites, images, timers and sample playback are not supported.
- **Brief freezes when the computer writes to CIRCUITPY.** Writing to the board's flash (for example an editor closing a file on the drive) pauses the board for up to a few hundred ms, which can cause an audio dropout. The library recovers by itself. For play, eject the drive or use standalone power.
- **Heavy games are close to the frame budget in 1.6x smooth.** Almost Tennis's busiest frames can run late; the audio buffer has absorbed it in testing. Use 1.6x sharp for the most headroom.
- **Fixed 30 fps, 64KB Lua heap by default (256KB maximum), no save data.**
- **Wired DualShock 4 only.** Other USB controllers, Bluetooth and touch are not supported.
- **The 1.6x and 2-bit modes and HDMI audio** are tested on one TV; other displays may differ. The firmware is a patched CircuitPython 10.3 build; a clean rebuild from the pinned public sources reproduces the shipped UF2 except for its build date (see **engine/firmware/REBUILD-RESULTS.json**).

Play

Connections

Plug the Fruit Jam into the TV or monitor using an HDMI cable.

Plug the Fruit Jam into power with a USB C cable and power supply.

Connect the optional gamepad to the Fruit Jam with a USB data/power cable (either port works).

Turn on the TV and the Fruit Jam and play!

Select your game/demo with the crank or gamepad L/R or B1/B2 buttons on the Fruit Jam.

Start the game/demo with X or Fruit Jam B3.

Note: the example video shows an old version with STEMMA QT cable connection, the current uses GPIO pins to connect the crank as shown in the photos, steps, and diagrams.