USB HID Crank Controller

Created by Ruiz Brothers

https://learn.adafruit.com/usb-hid-crank-controller

Last updated on 2021-11-15 07:41:29 PM EST

Table of Contents

Overview

A USB Crank Device

Ever wonder what it's like to control things with a crank? This project explores the use of a crank mechanism that attaches to a rotary encoder. With the Adafruit ItsyBitsy M0 and CircuitPython, we can simulate a USB Human Interface Device (HID) device (like computer keyboards, mice, and gamepads) to trigger commands, macros and key presses. This idea was inspired by the Play Date (https://adafru.it/F0d) device by Panic.

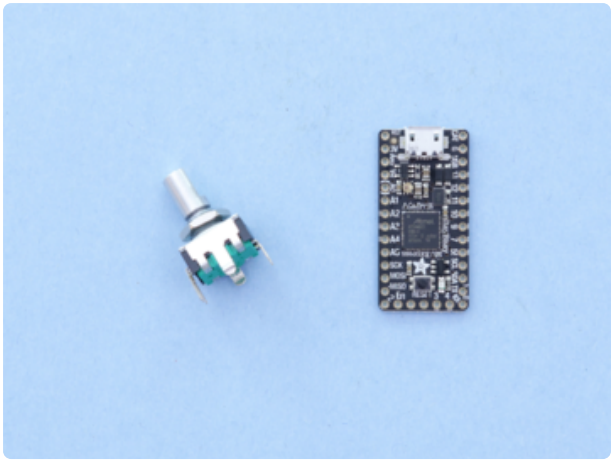
Hinged Crank Mechanism

Have you ever cranked a rotary encoder? With the right application, this actually feels intuitive and fun. This design features a hinged arm and free-rotating handle. It pops out of the case and hinges out making a crank. The mechanism is a 3D print-in-place design. It press fits over any flat nose rotary encoder.

Turbo Mode

The rotary encode simulates key presses each time a pulse is produced. By turning the encoder, rapid key presses are generated – This is similar to the "Turbo" button prominently featured on USB gamepads. We've come up with some use cases for this.

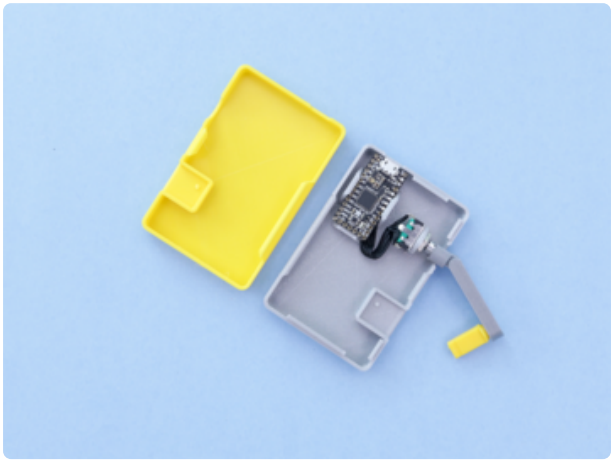
- Turn up/down volume or screen brightness
- Play MakeCode Arcade games
- Scroll webpages
- Rotate 3D models in CURA Slicer



DIY USB HID

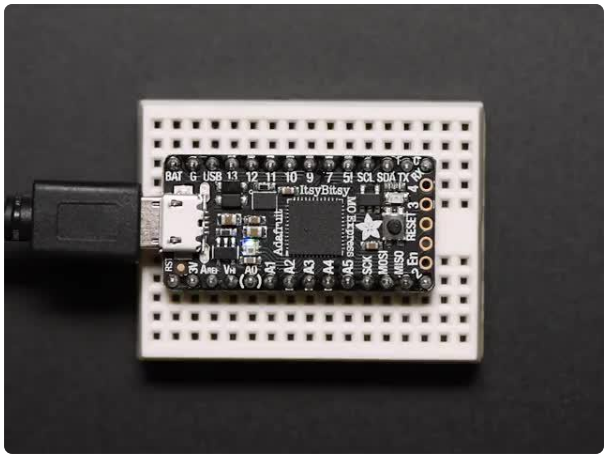
This project uses the USB HID CircuitPython library. It's designed for creating custom USB input devices for creative and assistive applications. The library supports most US keypresses, multimedia and gamepad controls.

Parts List



A copy and paste friendly list of parts linked to products page.

- Adafruit ItsyBitsy M0 ()
- Rotary Encoder ()
- USB microB Cable – Fully reversible ()
- 10-wire silicone cover ribbon cable (https://adafru.it/CJj)
- 3D Printer – Inventor II ()
- Filament for 3D Printers ()

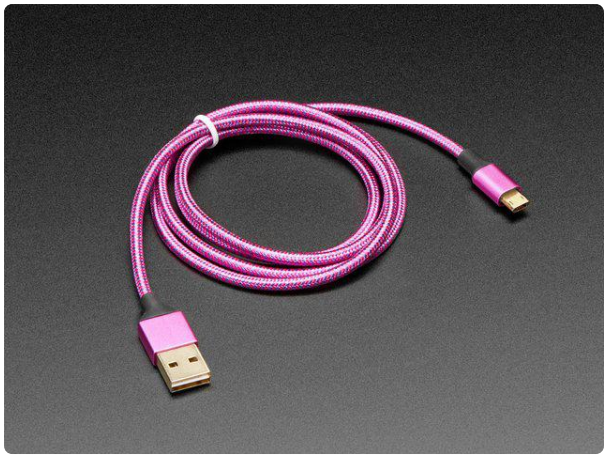


Adafruit ItsyBitsy M0 Express - for CircuitPython & Arduino IDE

What's smaller than a Feather but larger than a Trinket? It's an Adafruit ItsyBitsy M0 Express! Small, powerful, with a rockin' ATSAMD21 Cortex M0...

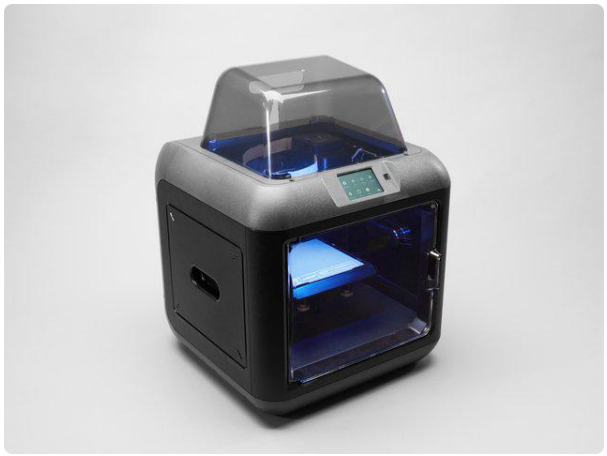
https://www.adafruit.com/product/3727

[Rotary Encoder + Extras]
This rotary encoder is the best of the best, it's a high-quality 24-pulse encoder, with detents and a nice feel. It is panel mountable for placement in a box, or you can plug it...
https://www.adafruit.com/product/377



[Fully Reversible Pink/Purple USB A to micro B Cable - 1m long]
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also fully reversible! That's right, you will save seconds a day by...
https://www.adafruit.com/product/4111

[Silicone Cover Stranded-Core Ribbon Cable - 10 Wire 1 Meter Long]
For those who are fans of our silicone-covered wires, but are always looking to up their wiring game. We now have Silicone Cover Ribbon cables! These may look...
https://www.adafruit.com/product/3890



[Monoprice Inventor II 3D Printer with Touchscreen and WiFi]
The Monoprice Inventor II 3D Printer Touchscreen with WiFi is a perfect entry-level 3D printer with small footprint and reliable performance. It comes equipped with...
https://www.adafruit.com/product/3897

Filament for 3D Printers in Various Colors and Types

Having a 3D printer without filament is sort of like having a regular printer without paper or ink. And while a lot of printers come with some filament there's a good chance...

https://www.adafruit.com/product/2080

Prerequisite Guides

If you're new to soldering and CircuitPython, we suggest you walk through the following guides to get the basics.

- Collin's Lab – Soldering (https://adafru.it/wsa)
- Welcome to CircuitPython! (https://adafru.it/Bfx)
- Adafruit ItsyBitsy M0 Guide (https://adafru.it/F0e)

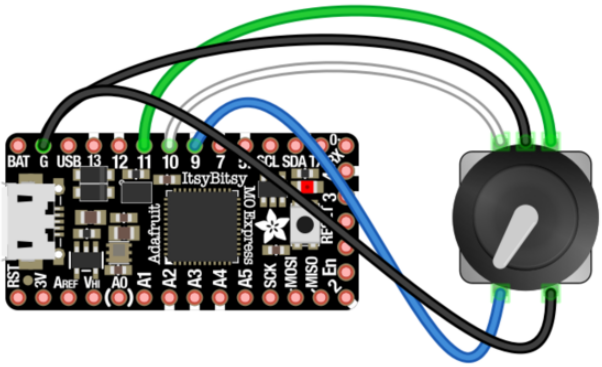
Circuit Diagram

Circuit Diagram

This provides a visual reference for wiring of the components. They aren't true to scale, just meant to be used as reference. This diagrams was created using Fritzing software (https://adafru.it/oEP).

Adafruit Library for Fritzing

Use our Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab the individual parts. Get library and parts from GitHub Adafruit Fritzing Parts (https://adafru.it/AYZ).



Wired Connections

The rotary encoder connects to the Adafruit ItsyBitsy M0 with five wired connections. Ground from the switch and rotary can be shared on the ground pin or ground pad (on the back of the PCB).

- Ground from switch on rotary encoder to Ground on ItsyBitsy M0
- Pin from switch on rotary encoder to pin 9 on ItsyBitsy M0
- Right pin on rotary encoder to pin 10 on ItsyBitsy M0
- Left pin on rotary encoder to pin 11 on ItsyBitsy M0
- Middle pin on rotary encoder to ground on ItsyBitsy M0

Powering

The Adafruit ItsyBitsy M0 can be powered via 5V via a USB connection to another device such as a computer.

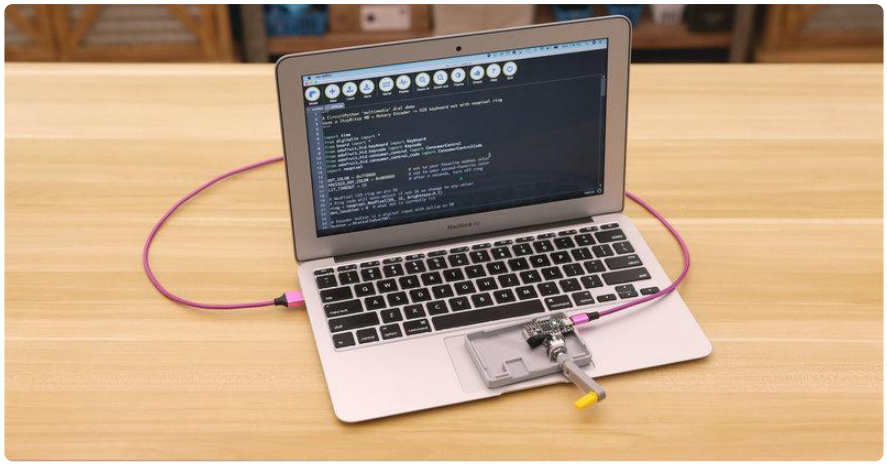
Software

Install Circuit Python for ItsyBitsy M0

Download the latest version of CircuitPython for the Adafruit ItsyBitsy M0. Click the green button to launch the page and click on the purple download button for the latest stable release.

Download Circuit Python for ItsyBitsy M0

https://adafru.it/Emi



Quick Start

- Connect the ItsyBitsy board to a computer (PC, mac, Linux) via a known good USB and double press the reset button.
- Download circuitpython UF2 and upload to the ITSYBOOT flash drive.
- Open the CIRCUITPY drive and upload the required libraries (listed below) and the code for this project in a file named code.py in the root directory of CIRCUIT PY.

Adafruit Circuit Python Libraries

Download the CircuitPython library bundle and unzip the folder. Create a new folder in the CIRCUITPY drive and name it "lib". The following libraries are required to run the code properly. Double check to ensure all of the files and folders are inside the lib folder on the CIRCUITPY drive.

- adafruit_hid (directory)
- neopixel.mpy

Download Circuit Python Library Bundle

https://adafru.it/ENC

Upload code.py

You can click Download and download the file. Insure it saves to a file named code.py (if you select Download Zip you'll need to unzip the file to see code.py. Upload the co de.py file to the CIRCUITPY drive.

```
"""
A CircuitPython 'multimedia' dial demo
Uses a ItsyBitsy M0 + Rotary Encoder -> HID keyboard out with neopixel ring
"""

import time
import board
from digitalio import DigitalInOut, Direction, Pull
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode
import neopixel
import usb_hid

DOT_COLOR = 0xFF0000              # set to your favorite webhex color
PRESSED_DOT_COLOR = 0x008080      # set to your second-favorite color
LIT_TIMEOUT = 15                  # after n seconds, turn off ring

# NeoPixel LED ring on pin D1
# Ring code will auto-adjust if not 16 so change to any value!
ring = neopixel.NeoPixel(board.D5, 16, brightness=0.2)
dot_location = 0  # what dot is currently lit

# Encoder button is a digital input with pullup on D9
button = DigitalInOut(board.D9)
button.direction = Direction.INPUT
button.pull = Pull.UP

# Rotary encoder inputs with pullup on D10 & D11
rot_a = DigitalInOut(board.D10)
rot_a.direction = Direction.INPUT
```

```
rot_a.pull = Pull.UP
rot_b = DigitalInOut(board.D11)
rot_b.direction = Direction.INPUT
rot_b.pull = Pull.UP

# Used to do HID output, see below
kbd = Keyboard(usb_hid.devices)
consumer_control = ConsumerControl(usb_hid.devices)

# time keeper, so we know when to turn off the LED
timestamp = time.monotonic()

########################## MAIN LOOP ##############################

# the counter counts up and down, it can roll over! 16-bit value
encoder_counter = 0
# direction tells you the last tick which way it went
encoder_direction = 0

# constants to help us track what edge is what
A_POSITION = 0
B_POSITION = 1
UNKNOWN_POSITION = -1  # initial state so we know if something went wrong

rising_edge = falling_edge = UNKNOWN_POSITION

# get initial/prev state and store at beginning
last_button = button.value
rotary_prev_state = [rot_a.value, rot_b.value]

while True:
    # reset encoder and wait for the next turn
    encoder_direction = 0

    # take a 'snapshot' of the rotary encoder state at this time
    rotary_curr_state = [rot_a.value, rot_b.value]

    if rotary_curr_state != rotary_prev_state:
        #print("Changed")
        if rotary_prev_state == [True, True]:
            # we caught the first falling edge!
            if not rotary_curr_state[A_POSITION]:
                #print("Falling A")
                falling_edge = A_POSITION
            elif not rotary_curr_state[B_POSITION]:
                #print("Falling B")
                falling_edge = B_POSITION
            else:
                # uhh something went deeply wrong, lets start over
                continue

        if rotary_curr_state == [True, True]:
            # Ok we hit the final rising edge
            if not rotary_prev_state[B_POSITION]:
                rising_edge = B_POSITION
                # print("Rising B")
            elif not rotary_prev_state[A_POSITION]:
                rising_edge = A_POSITION
                # print("Rising A")
            else:
                # uhh something went deeply wrong, lets start over
                continue

            # check first and last edge
            if (rising_edge == A_POSITION) and (falling_edge == B_POSITION):
                encoder_counter -= 1
                encoder_direction = -1
                print("%d dec" % encoder_counter)
            elif (rising_edge == B_POSITION) and (falling_edge == A_POSITION):
```

```
                encoder_counter += 1
                encoder_direction = 1
                print("%d inc" % encoder_counter)
            else:
                # (shrug) something didn't work out, oh well!
                encoder_direction = 0

            # reset our edge tracking
            rising_edge = falling_edge = UNKNOWN_POSITION

    rotary_prev_state = rotary_curr_state

    # Check if rotary encoder went up
    if encoder_direction == 1:
        consumer_control.send(ConsumerControlCode.VOLUME_DECREMENT) #Turn Down
Volume
    #    kbd.press(Keycode.LEFT_ARROW)
    #    kbd.release_all()
    # Check if rotary encoder went down
    if encoder_direction == -1:
        consumer_control.send(ConsumerControlCode.VOLUME_INCREMENT) #Turn Up Volume
    #    kbd.press(Keycode.RIGHT_ARROW)
    #    kbd.release_all()
    # Button was 'just pressed'
    if (not button.value) and last_button:
        print("Button pressed!")
        kbd.press(Keycode.SPACE) #Keycode for spacebar
        kbd.release_all()
        ring[dot_location] = PRESSED_DOT_COLOR # show it was pressed on ring
        timestamp = time.monotonic()        # something happened!
    elif button.value and (not last_button):
        print("Button Released!")
        # kbd.press(Keycode.SHIFT, Keycode.SIX)
        # kbd.release_all()
        ring[dot_location] = DOT_COLOR      # show it was released on ring
        timestamp = time.monotonic()        # something happened!
    last_button = button.value

    if encoder_direction != 0:
        timestamp = time.monotonic()        # something happened!
        # spin neopixel LED around!
        previous_location = dot_location
        dot_location += encoder_direction   # move dot in the direction
        dot_location += len(ring)           # in case we moved negative, wrap around
        dot_location %= len(ring)
        if button.value:
            ring[dot_location] = DOT_COLOR  # turn on new dot
        else:
            ring[dot_location] = PRESSED_DOT_COLOR # turn on new dot
        ring[previous_location] = 0         # turn off previous dot

    if time.monotonic() > timestamp + LIT_TIMEOUT:
        ring[dot_location] = 0   # turn off ring light temporarily
```

Modify Key Codes

You can customize the key codes to form custom commands, which can be multiple keys, or have it execute just single keyboard characters.

The rotary encoder can execute up to 3 different commands based on features. Pressing the knob and turning the knob left or right. These are commented in the code and can be changed by adjusting the key code value.

List of USB HID Keycodes

The long list of available keyboard characters are listed in the webpage linked below. Most of the characters are for US keyboard only. Function keys and modifiers can be used but some special characters are not currently supported.

Adafruit HID Keycodes

https://adafru.it/AOi

Starting with the first command, turning the knob to the right will execute the ctrl+up arrow keys. These are two different keyboard characters that are separated with commas. This will essentially press the two keys simultaneously. The values inside the parentheses kbd.press(keycode.THISKEY) are the ones you want to change. For example, the block of code below is executed when turning the knob to the right.

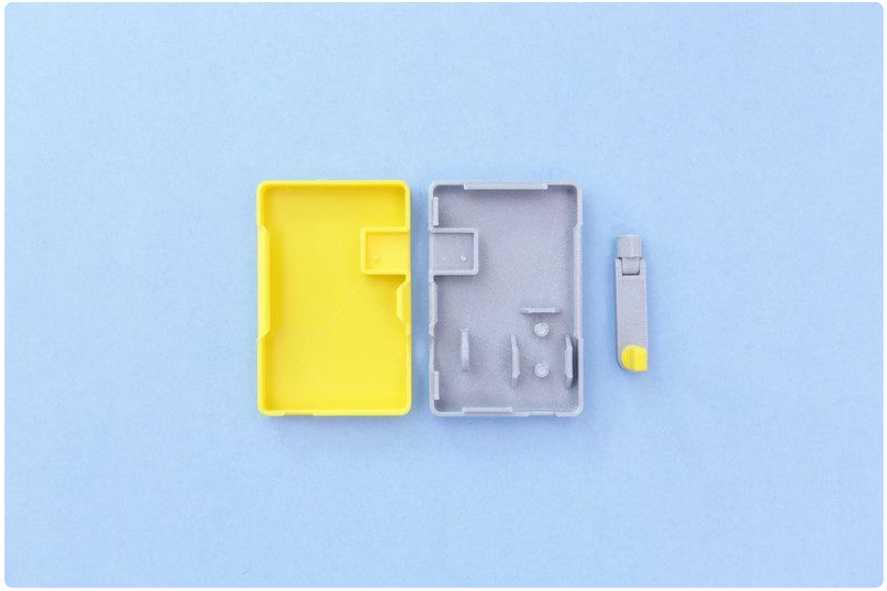
```
# Check if rotary encoder went up
if encoder_direction == 1:
kbd.press(Keycode.CONTROL, Keycode.UP_ARROW)
kbd.release_all()
```

For more information and troubleshooting, please check out the CircuitPython library guide, linked below.

Welcome to CircuitPython Libraries

https://adafru.it/ABU

3D Printing

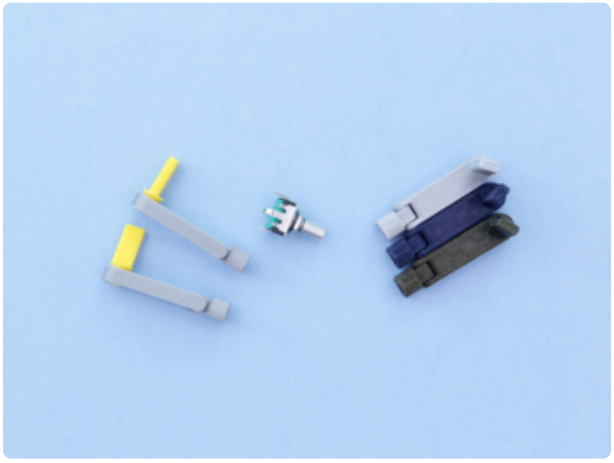


3D Printed Parts

The parts in this kit are designed to be 3D printed with FDM based machines. STL files are oriented to print "as is". Parts require tight tolerances that might need adjustment of slice settings. Reference the suggested settings below.

Download on Thingiverse

https://adafru.it/F0G



DIY 3D Printed Crank for Rotary Encoders

Need a crank for a rotary encoder? Here's a print-in-place model that features a hinged arm and free-rotating handle. The pieces can not be disassembled. It's just one part that prints in about 20 minutes. Press fits over any flat nose rotary encoder, for those times where you need a cranky crank!

Fusion 360 model included with accompanying STEP file. User parameters are setup for tolerances, length, shaft diameter etc. Gap is set to 0.4mm in STL file. Check out the source for your own tweaks!

CURA Slicing

Parts were sliced using Ultimaker's CURA software and tested with an Ultimaker 3 and Flashforge Inventor II. The kit requires a minimum build volume of 150mm cubed. No support material is necessary for any of the parts. Double check parts are positioned in the center of the build plate before printing.

Settings

Use these settings as reference. Values listed were used in Ultimaker's CURA (https://adafru.it/C26) slicing software.

- 0.2mm Layer Height / 0.4mm nozzle
- 0.38mm Line Width (inner & outer widths)
- 40mm/s printing speed
- 20% infill
- Supports: No

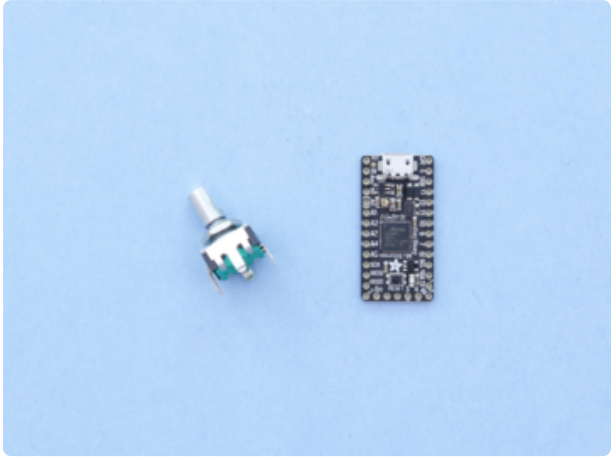
Designing Things

The fusion 360 source file is included and features original sketches and feature timeline along with easily editable user parameters. The parts can further be separated into small pieces for fitting on printers with smaller build volumes. Note: STEP file is included for other 3D surface modeling programs such as Onshape, Solidworks and Rhino.

Layer by Layer

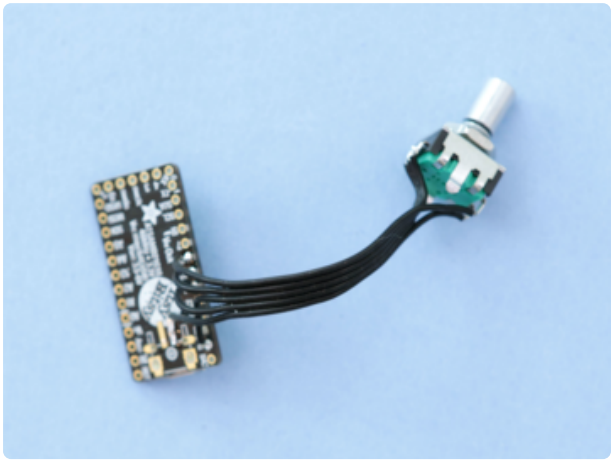
Interested in CAD tutorials? Check out my playlist on YouTube (https://adafru.it/Ddm) – There's over 100 of them! My personal favorite is the snap fit tutorial for cases and enclosures.

Wiring



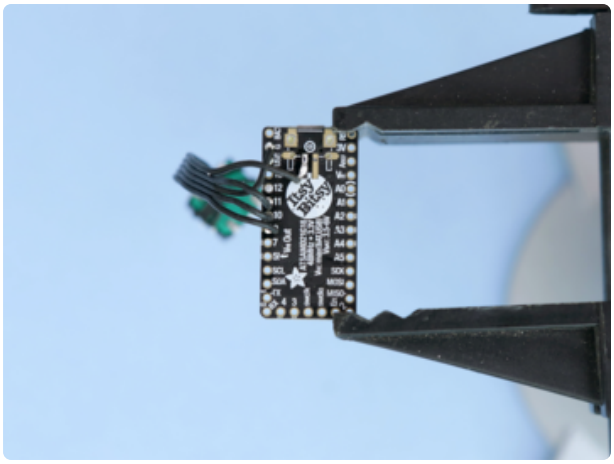
Connect Rotary Encode to ItsyBisty

We'll need to solder wires to connect the rotary encoder to the Adafruit ItsyBitsy M0.



Silicone Cover Ribbon Cable

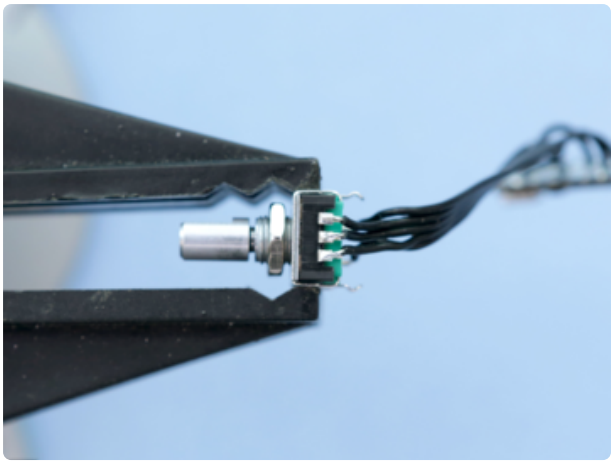
To keep the wiring neat and tidy, we'll use a 10-wire silicon cover ribbon cable – We only need five wires, so peel off a set. Measure out a piece to about 8 cm in length. Peel apart individual wires, on both ends, about 1 cm in length. Use a pair of wire strippers to remove a bit of insulation from the tips. Tin the exposed wire by adding a bit of solder – This will help prevent the wires from fraying.



Wire ItsyBitsy

Solder the wires to the following pins on the bottom of the Adafruit ItsyBitsy M0.

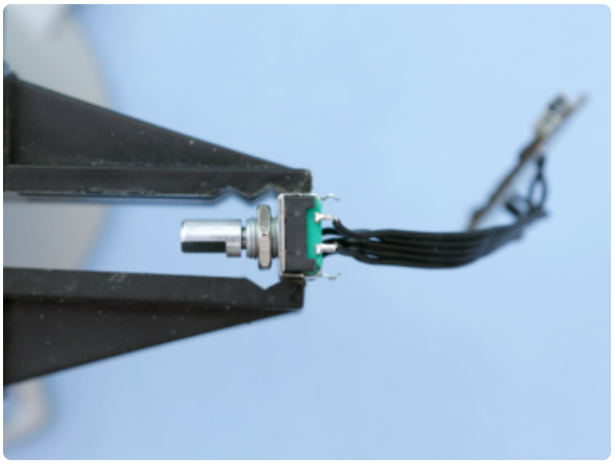
- Ground pin
- Ground pad
- Pin 10
- Pin 11
- Pin 9



Wire Rotary Encoder

Solder the wires to the following pins on the rotary encoder. Polarity on the left and right pins don't matter as long as they match pins on the ItsyBitsy. Double check and make sure the connections match.

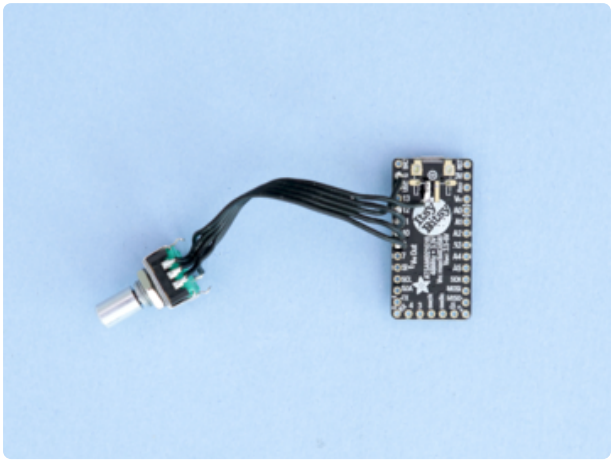
- ItsyBitsy ground pin to center pin on rotary encoder
- ItsyBitsy pin 10 to left pin on rotary encoder
- ItsyBitsy pin 11 to right pin on rotary encoder



Wire Switch on Rotary Encoder

Solder wires to the switch on the rotary encoder.

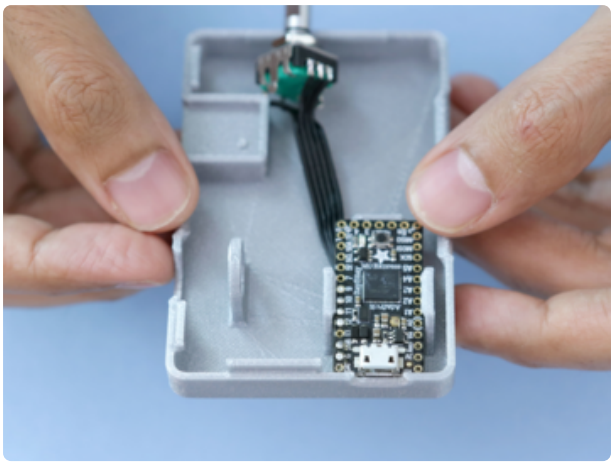
- ItsyBitsy ground pin to left pin on rotary encoder switch
- ItsyBitsy pin 9 to right pin on rotary encoder switch



Wired Rotary Encoder

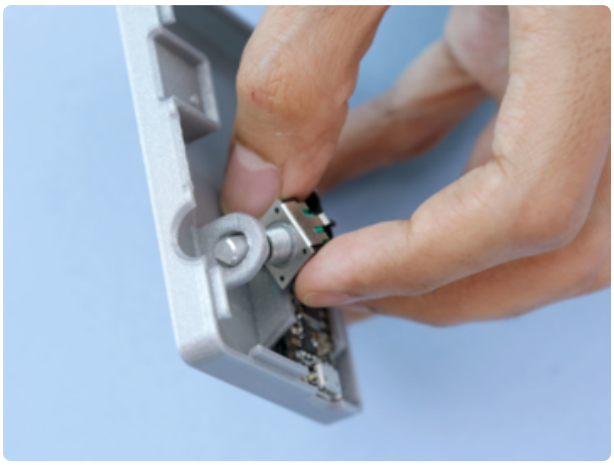
And now we have a nice and neatly wired rotary encoder! Double check the wiring and make sure the connections match.

Assembly



Install ItsyBitsy M0

Place the ItsyBitsy PCB over the two standoffs on the bottom part of the enclosure. Line up the microUSB so with the notch on the side of the case. Insert the PCB at an angle so it's under the clip. Press the edges of the PCB down to snap fit the PCB into place.

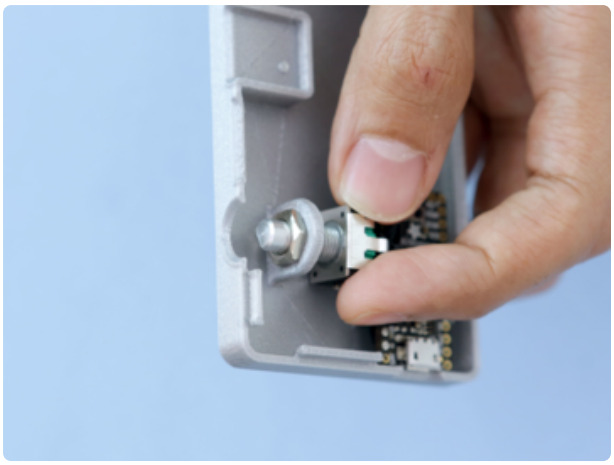


Insert Rotary Encoder

Insert the shaft of the rotary encoder into the circular mounting tab on the inside of the case.

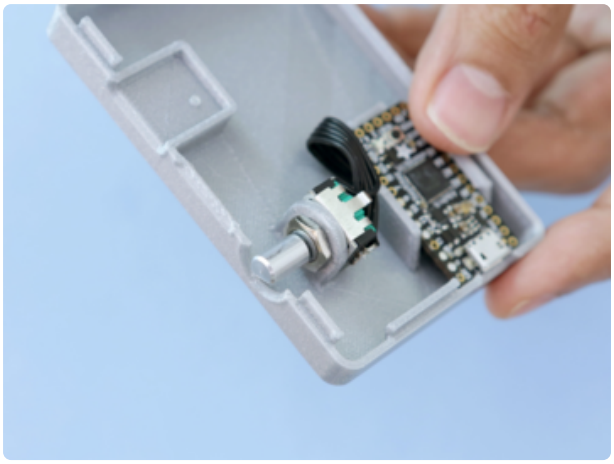
Install Rotary Encoder

Push the body of the rotary encoder up against the mounting tab.



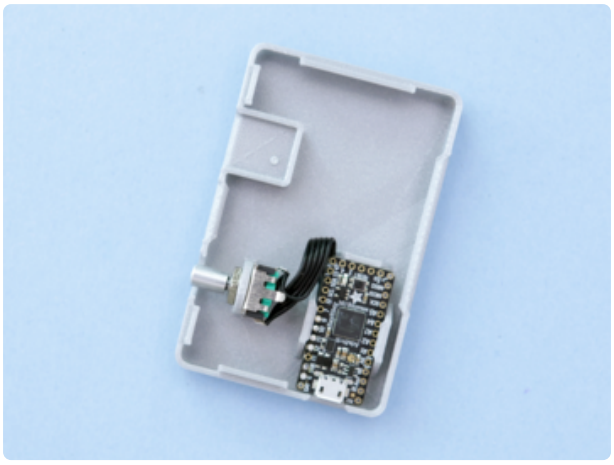
Install Hex Nut

Pull the rotary encoder back and place the hex nut over the top of the shaft. There should be enough clearance to get the nut over the shaft.



Fasten Nut

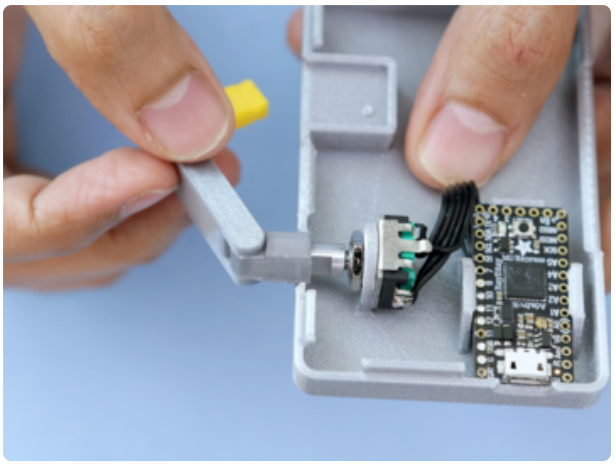
Use fingers to fasten the hex nut onto the threading. Rotate the rotary encoder to tighten or loosen the nut. Optionally use a pair of needle nose pliers to grab hold of the nut.



Installed Rotary Encoder + ItsyBitsy

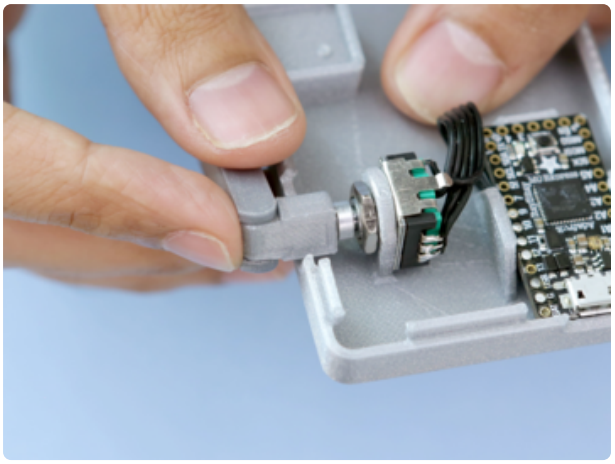
Take note of the rotary encoders orientation – The two metal prongs are right side up. This allows the space for the pins and wires.

Looks like there's a good amount of free space, eh? Maybe try adding some extra buttons!

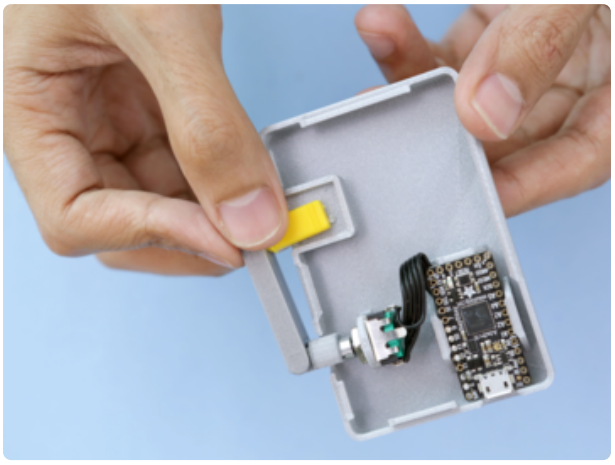


Install Crank

Place the cap end of the crank onto the tip of the rotary encoder.

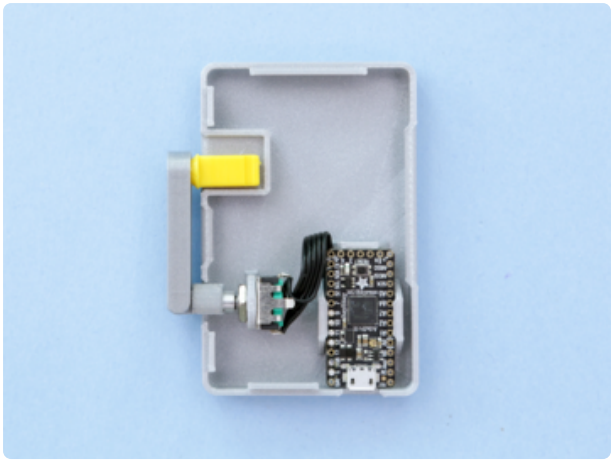


Firmly press the crank onto shaft of the rotary encoder until it's fully seated.



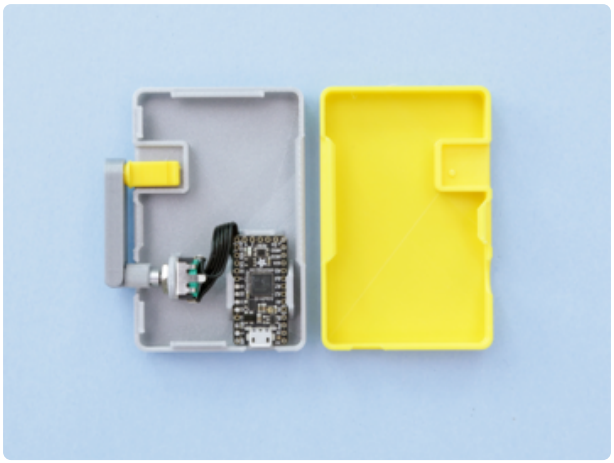
Handle Holster

The case features a spot for holding the handle of the crank. There's a little dimple that clips into a small grove near the end of the handle. When the two halves of the enclosure are closed, the handle can "click" and hide inside the case.



Set Handle

Place the handle of the crank over the dimple.



Closing Case

Grab the top half of the enclosure and orient the halves so the features are matching.

Close Case

Place the top half over the bottom half. Make sure the cutouts are lined up and matching.

Snap Fit Shut

The case has snap fit features that keep the halves secured and closed shut.

To open the case back up, place fingernail in between the indentations on the side. Pull apart to separate the two halves.

Case Closed

Plug it into a computer and try it out!