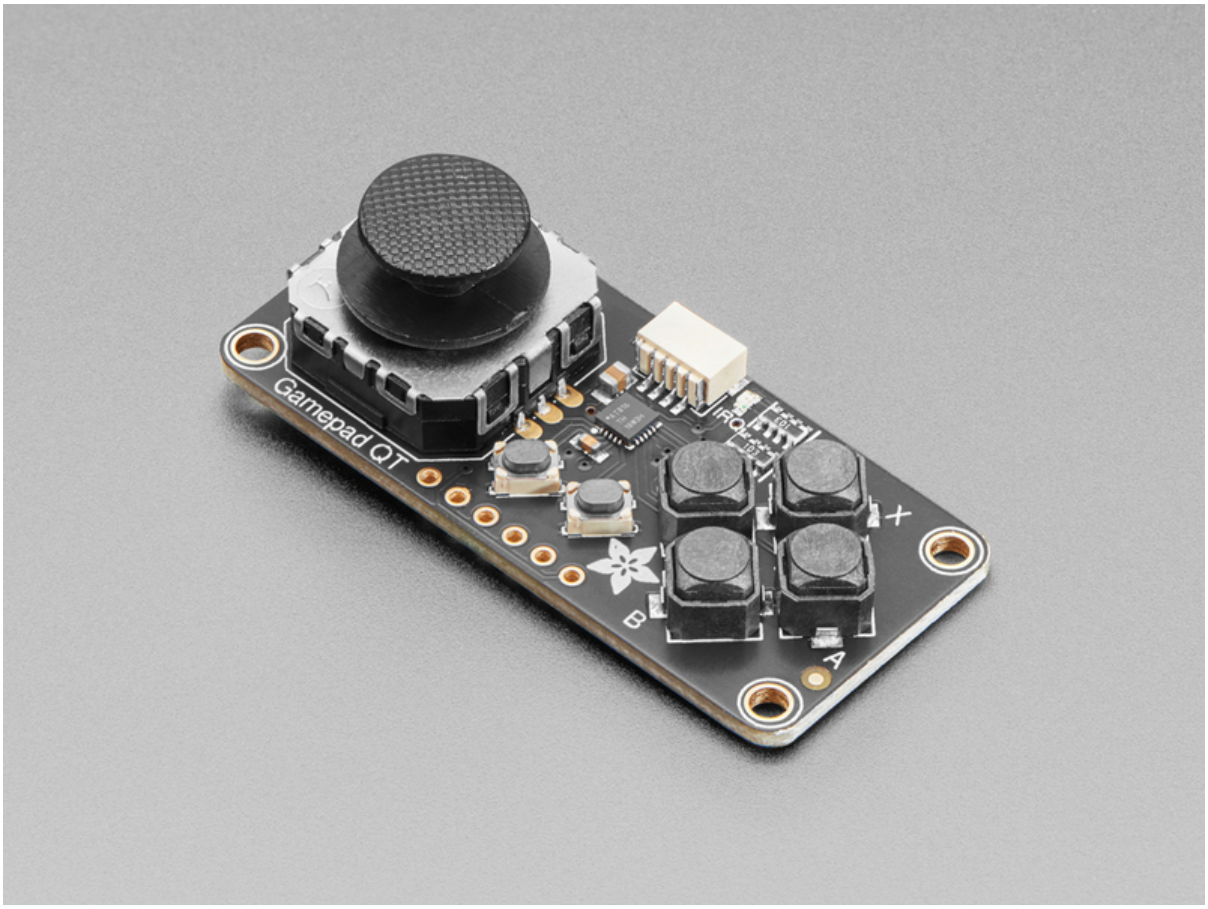
Adafruit Mini I2C STEMMA QT Gamepad with seesaw

Created by Kattni Rembor

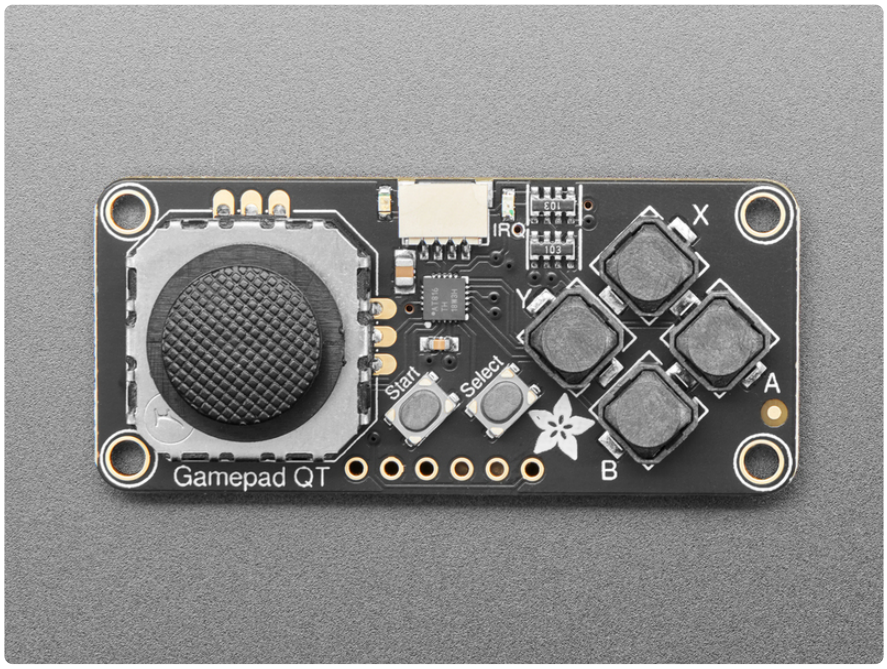


https://learn.adafruit.com/gamepad-qt

Last updated on 2025-08-06 05:34:31 PM EDT

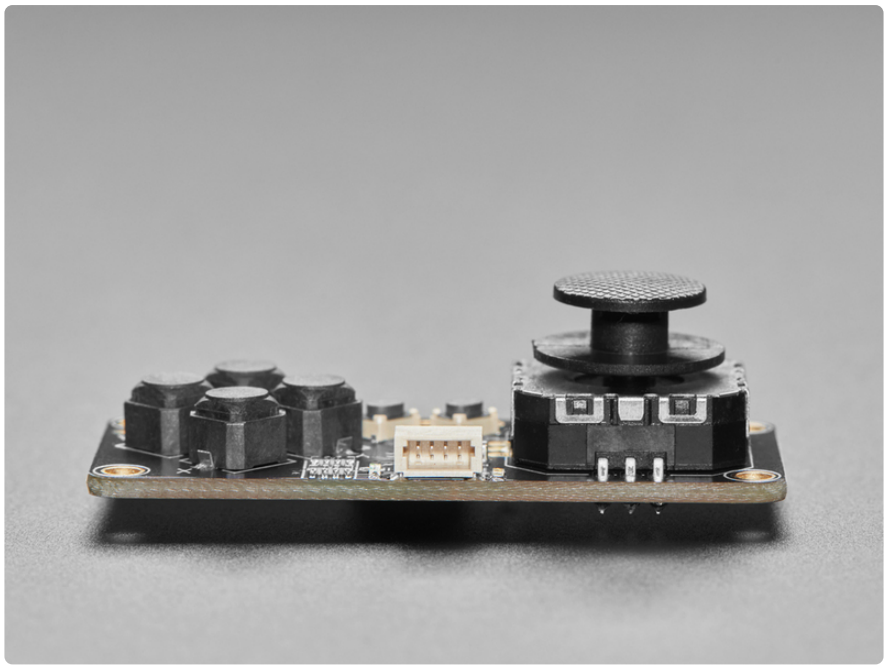
Table of Contents

Overview

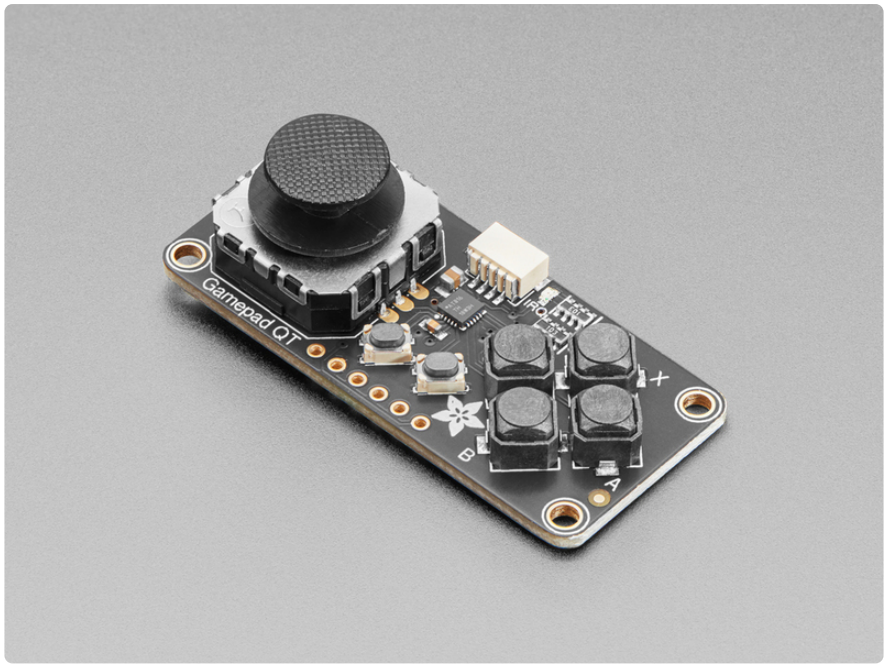


Make a game or robotic controller for any I2C microcontroller or microcomputer with this tiny gamepad breakout board. This design has a 2-axis thumb joystick and 6 momentary buttons (4 large and 2 small). The board communicates with your host microcontroller over I2C so it's easy to use and doesn't take up any of your precious analog or digital pins. There is also an optional **interrupt pin** that can alert your feather when a button has been pressed or released to free up processor time for other tasks.

You can use our Arduino library to control and read data (https://adafru.it/BrV) with any compatible microcontroller. We also have CircuitPython/Python code (https://adafru.it/BrW) for use with computers or single-board Linux boards.

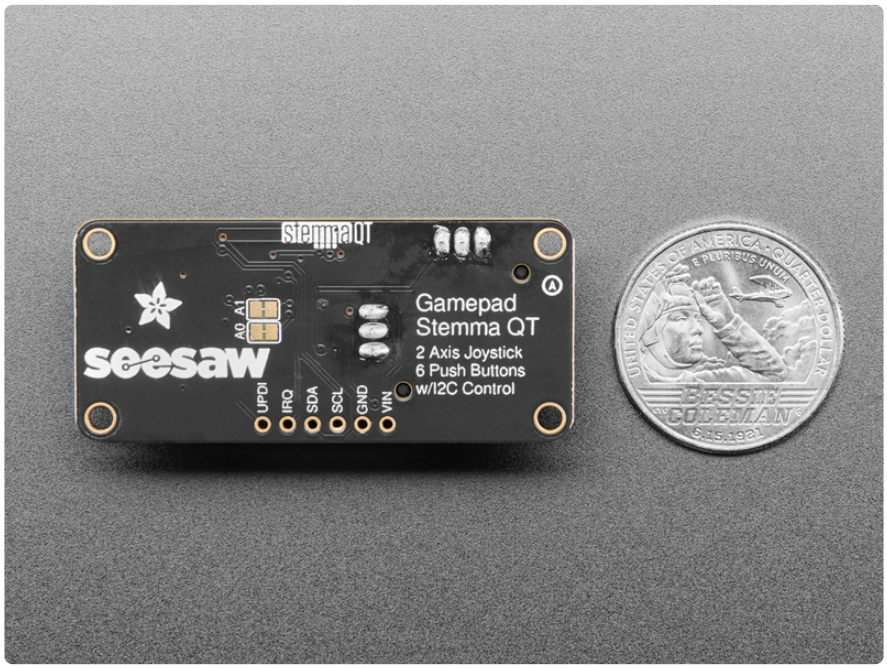


To get you going fast, this board comes with a **STEMMA QT** (https://adafru.it/LBQ) connector, making them easy to interface with. The STEMMA QT connector (https://adafru.it/JqB) is compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the gamepad or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB) and a 'hub' board (http://adafru.it/5625). QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE).



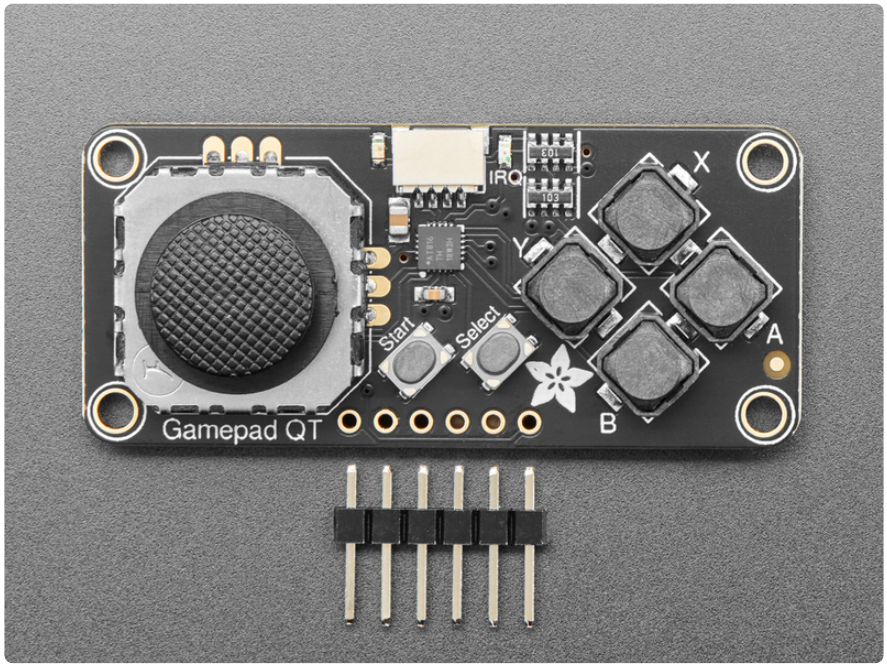
This board features Adafruit **seesaw** technology (https://adafru.it/VdL) - a custom programmed little helper microcontroller that takes the two analog inputs from the joystick, and 6 button inputs, and converts it into a pretty I2C interface. This I2C

interface means you don't 'lose' any GPIO or analog inputs, and it works with any and all microcontrollers or microcomputers - even if they don't have an analog input for the thumbstick!



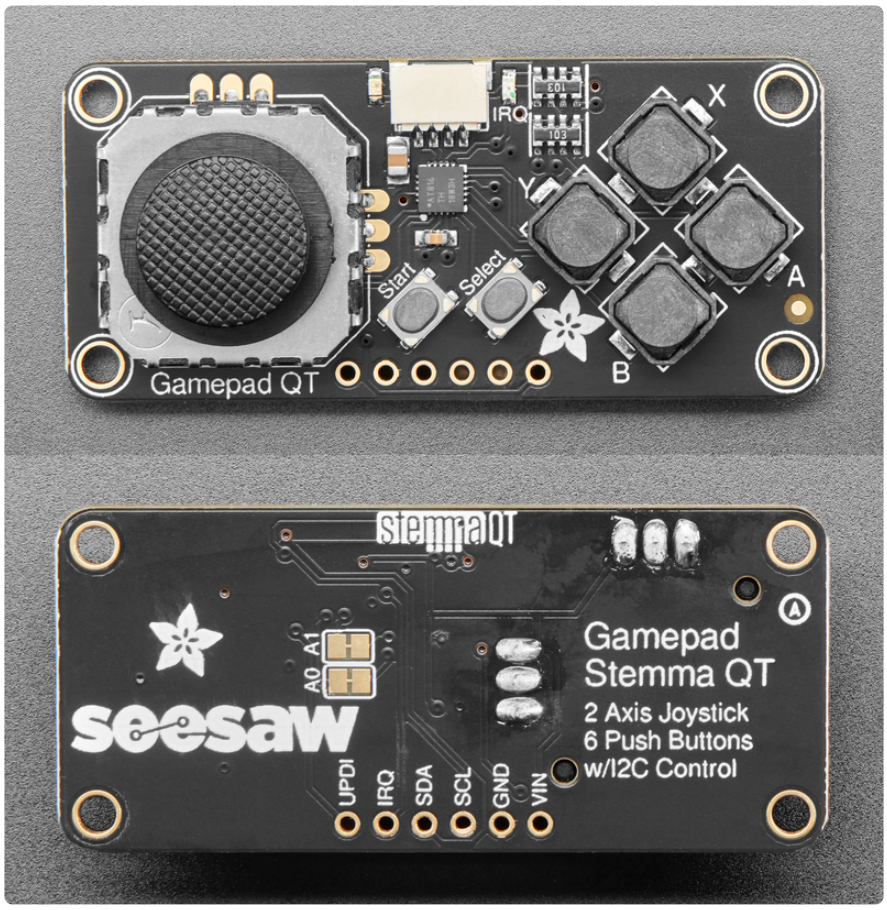
If you have an I2C address conflict, or you want to connect more than one of these to a board, there are two address-select jumpers so you have 4 options of I2C addresses.

There's an optional IRQ (interrupt) line that you can use if you'd like the gamepad to let you know when a button has been pressed. Since it's optional, you will need to connect a separate wire for the IRQ line, or just leave it disconnected.



Comes with one assembled and programmed Gamepad, and some 0.1" header. If you aren't using Stemma QT cables, some soldering is required to attach the header for breadboarding.

Pinouts



Power Pins

- **VIN** - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 3V microcontroller like a Feather RP2040, use 3V, or for a 5V microcontroller like Arduino, use 5V.
- **GND** - This is common ground for power and logic.

I2C Logic Pins

The default I2C address is **0x50**.

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. There's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. There's a **10K pullup** on this pin.

STEMMA QT Connector

The default I2C address is **0x50**.

- **STEMMA QT connector** (https://adafru.it/Ft4) **-** This connector is located on the front of the board, at the top in the center. It allows you to connect to development boards with **STEMMA QT** / Qwiic connectors or to other things with various associated accessories (https://adafru.it/JRA).

Buttons

There are two small buttons towards the center of the board, and four large buttons arranged in a diamond on the right side of the board.

- **Start** - This is a small button, located to the left of center. It is pin **16** in the seesaw firmware.
- **Select** - This is a small button, located to the right of center. It is pin **0** in the seesaw firmware.
- **A** - This is a large button, located to the right in the diamond. It is pin **5** in the seesaw firmware.
- **B** - This is a large button, located at the bottom of the diamond. It is pin **1** in the seesaw firmware.
- **X** - This is a large button, located at the top of the diamond. It is pin **6** in the seesaw firmware.
- **Y** - This is a large button, located to the left in the diamond. It is pin **2** in the seesaw firmware.

Joystick

This is the 2-axis joystick located on the left side of the board.

- **X-axis** - The joystick x-axis (horizontal) is on pin **14** in the seesaw firmware.
- **Y-axis** - The joystick y-axis (vertical) is on pin **15** in the seesaw firmware.

Power LED

- **Power LED -** On the front of the board, to the right of the STEMMA connector, is the power LED. It is the green LED.

Interrupt Pin and LED

- **IRQ** - This is the optional interrupt pin. It is pin **5** in the seesaw firmware. This pin will let you know when a button on the gamepad has been pressed. You can use

it to alert the Feather to the button press or release, to free up processor time for other tasks. Since its optional you will need to connect a separate wire for the IRQ line, or simply leave it disconnected.

- **IRQ LED** - On the front of the board, to the left of the STEMMA connector, is the IRQ LED. It is the red LED. It turns on whenever the interrupt is detected.

Address Jumpers

On the back of the board are **two address jumpers**, labeled **A0** and **A1**, on the back of the board above the "w" in the "seesaw" label. These jumpers allow you to chain up to 4 of these boards on the same pair of I2C clock and data pins. To do so, you cut the jumpers "open" by separating the two pads.

If you happen to need more than 4 addresses, it's possible to set the I2C address with a special address-change command that is saved to the onboard non-volatile EEPROM memory.

The default I2C address is **0x50**. The other address options can be calculated by "adding" the **A0/A1** to the base of **0x50**.

A0 sets the lowest bit with a value of **1** and **A1** sets the next bit with a value of **2**. The final address is **0x50** + **A0** + **A1** which would be **0x53**.

If only **A0** is cut, the address is **0x50** + **1** = **0x51**.

If only **A1** is cut, the address is **0x50** + **2** = **0x52**.

If both **A0 and A1** are cut, the address is **0x50** + **1** + **2** = **0x53**.

ADDR	A0	A1
0x50	L	L
0x51	H	L
0x52	L	H
0x53	H	H

UPDI Pin

- **UPDI** - This is the single-pin **U**nified **P**rogram and **D**ebug **I**nterface. This pin is for external programming or on-chip-debugging for the ATtiny816 running the seesaw firmware (https://adafru.it/VdL). We have a page in the ATtiny Breakouts with seesaw Learn Guide (https://adafru.it/18ED) detailing how to reprogram these chips with your own firmware (at your own risk). We don't provide any support for custom builds of seesaw - we think this is cool and useful for the Maker community.

CircuitPython and Python

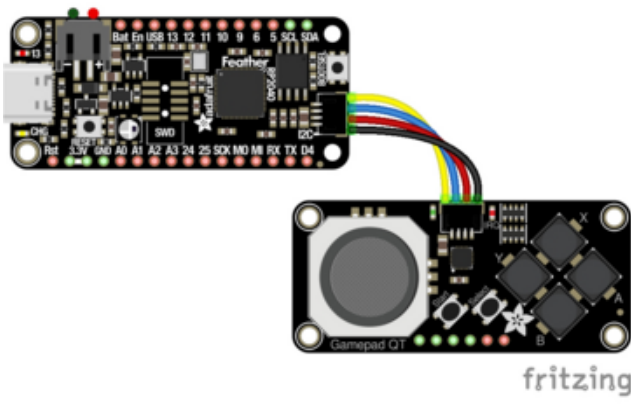
It's easy to use the **Mini I2C STEMMA QT Gamepad** with Python or CircuitPython, and the Adafruit_CircuitPython_seesaw (https://adafru.it/BrW) module. This module allows you to easily write Python code that reads the joystick analog values and the six button presses (X, Y, A, B, Start, and Select).

You can use the Gamepad with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

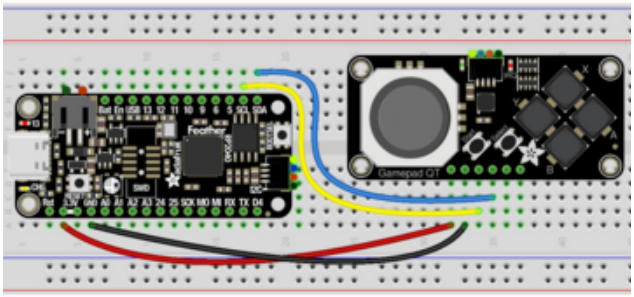
First wire up the Gamepad to your board, exactly as shown below.

The following is the Gamepad connected to a Feather RP2040 using STEMMA QT:



Simply connect a STEMMA QT cable (https://adafru.it/JRA) from the **STEMMA QT connector on the Feather** to the **STEMMA QT connector on the Gamepad**.

The following is the Gamepad connected to a Feather RP2040 using a solderless breadboard:

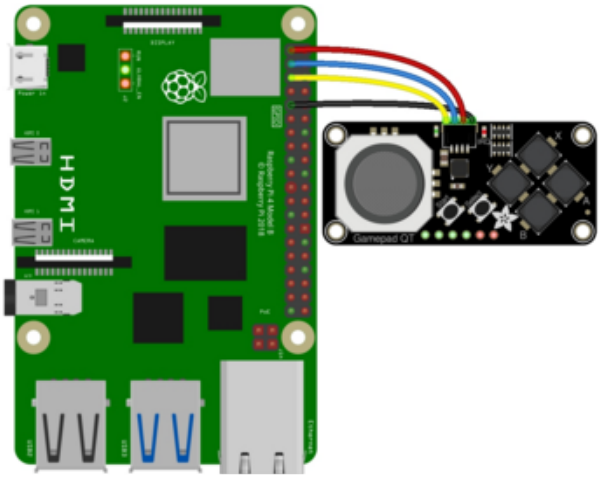


Board STEMMA 3V to **Gamepad VIN (red wire)**
Board STEMMA GND to **Gamepad GND (black wire)**
Board STEMMA SCL to **Gamepad SCL (yellow wire)**
Board STEMMA SDA to **Gamepad SDA (blue wire)**

Python Computer Wiring

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (https://adafru.it/BSN).

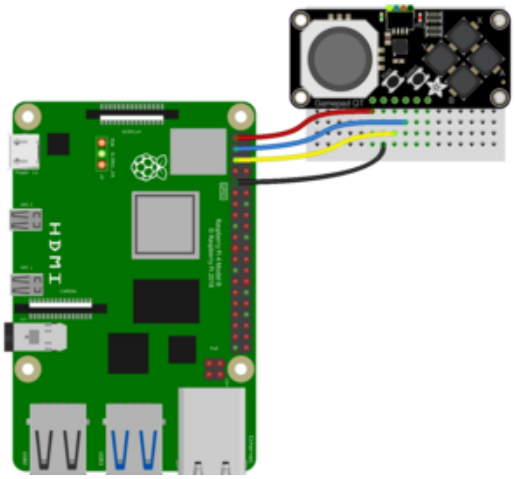
The following is a Raspberry Pi wired with I2C using the Gamepad STEMMA connector:



Using a [STEMMA QT to socket cable](http://adafru.it/4397) (http://adafru.it/4397) plugged into the **STEMMA QT connector on the Gamepad**, connect the wires to the Pi as follows:

Red wire to **Pi 3V**
Black wire to **Pi GND**
Yellow wire to **Pi SCL**
Blue wire to **Pi SDA**

The following is a Raspberry Pi wired with I2C using a solderless breadboard:



Pi 3V to **Gamepad VIN (red wire)**
Pi GND to **Gamepad GND (black wire)**
Pi SCL to **Gamepad SCL (yellow wire)**
Pi SDA to **Gamepad SDA (blue wire)**

Increase the I2C Bus Speed

The Pi OS default I2C baudrate is 100 kHz, but the Gamepad works best at 400 kHz.

Modify the **i2c_arm_baudrate** variable in the **/boot/firmware/config.txt** file to look like this:

```
i2c_arm_baudrate=400000
```

Python Installation of seesaw Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `pip3 install adafruit-circuitpython-seesaw`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython Usage

To use with CircuitPython, you need to first install the **Adafruit_CircuitPython_seesaw** library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file

in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders and file:

- **adafruit_bus_device/**
- **adafruit_seesaw/**
- **adafruit_pixelbuf.mpy**

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

```
python3 code.py
```

Example Code

If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-FileCopyrightText: 2023 Kattni Rembor for Adafruit Industries

# SPDX-License-Identifier: MIT

import time

import board
from micropython import const

from adafruit_seesaw.seesaw import Seesaw

BUTTON_X = const(6)
BUTTON_Y = const(2)
BUTTON_A = const(5)
```

```
BUTTON_B = const(1)
BUTTON_SELECT = const(0)
BUTTON_START = const(16)
button_mask = const(
    (1 << BUTTON_X)
    | (1 << BUTTON_Y)
    | (1 << BUTTON_A)
    | (1 << BUTTON_B)
    | (1 << BUTTON_SELECT)
    | (1 << BUTTON_START)
)

i2c_bus = board.STEMMA_I2C()  # The built-in STEMMA QT connector on the
microcontroller
# i2c_bus = board.I2C()  # Uses board.SCL and board.SDA. Use with breadboard.

seesaw = Seesaw(i2c_bus, addr=0x50)

seesaw.pin_mode_bulk(button_mask, seesaw.INPUT_PULLUP)

last_x = 0
last_y = 0

while True:
    x = 1023 - seesaw.analog_read(14)
    y = 1023 - seesaw.analog_read(15)

    if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
        print(x, y)
        last_x = x
        last_y = y

    buttons = seesaw.digital_read_bulk(button_mask)

    if not buttons & (1 << BUTTON_X):
        print("Button x pressed")

    if not buttons & (1 << BUTTON_Y):
        print("Button Y pressed")

    if not buttons & (1 << BUTTON_A):
        print("Button A pressed")

    if not buttons & (1 << BUTTON_B):
        print("Button B pressed")

    if not buttons & (1 << BUTTON_SELECT):
        print("Button Select pressed")

    if not buttons & (1 << BUTTON_START):
        print("Button Start pressed")

    time.sleep(0.01)
```

Reading Joystick Values

The joystick is made up of two potentiometers at a 90 degree angle to each other, creating an x and y axis across it, each of which produce analog values ranging from 0 to 1023. To read the values, you use the seesaw firmware `analog_read` and provide it the seesaw firmware pin to which each axis is connected. The x axis is on pin 14, and the y axis is on pin 15.

Due to the physical orientation of the joystick on the gamepad, it is necessary to reverse the analog values so that when x is to the left or y is down, both report `0`, and when x is to the right or y is up, both report `1023`. This is done by subtracting both axis' values from 1023 when reading the value.

```
x = 1023 - seesaw.analog_read(14)
y = 1023 - seesaw.analog_read(15)
```

The values printed to the serial console are printed within an `if` block. This code is used to only print the values when they have changed more than 3. This avoids constant printing in the event of noise or vibration of the gamepad itself. That way, it only prints when you are actively and deliberately moving the joystick around.

```
if (abs(x - last_x) &gt; 3) or (abs(y - last_y) &gt; 3):
    print(x, y)
    last_x = x
    last_y = y
```

Reading Button Presses

The button variables are created and set to be equal to the pin to which they are connected in the seesaw firmware (as shown on the Pinouts page (https://adafru.it/18KA)). For example, the X button is connected to pin 6 in the seesaw firmware, and therefore `BUTTON_X = const(6)`.

```
BUTTON_X = const(6)
BUTTON_Y = const(2)
BUTTON_A = const(5)
BUTTON_B = const(1)
BUTTON_SELECT = const(0)
BUTTON_START = const(16)
```

Then the button mask is created. This is what tells the seesaw on the Gamepad which pins are of interest, e.g. only the pins connected to each of the six buttons.

```
button_mask = const(
    (1 &lt;&lt; BUTTON_X)
    | (1 &lt;&lt; BUTTON_Y)
    | (1 &lt;&lt; BUTTON_A)
    | (1 &lt;&lt; BUTTON_B)
    | (1 &lt;&lt; BUTTON_SELECT)
    | (1 &lt;&lt; BUTTON_START)
)
```

The button mask is then used in two places. It is first used to set all the buttons up as inputs with pullups.

```
seesaw.pin_mode_bulk(button_mask, seesaw.INPUT_PULLUP)
```

It is then used inside the loop to begin looking for button presses.

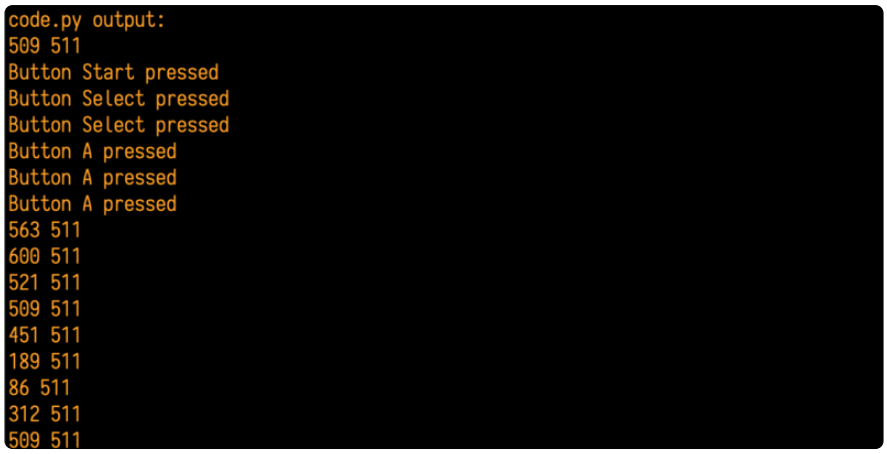
```
buttons = seesaw.digital_read_bulk(button_mask)
```

The remainder of the loop checks for button presses, one at a time, with an `if` block for each. For any button that is pressed (can be more than one) a print statement is output.

```
if not buttons &amp; (1 &lt;&lt; BUTTON_X):
    print("Button x pressed")

if not buttons &amp; (1 &lt;&lt; BUTTON_Y):
    print("Button Y pressed")

if not buttons &amp; (1 &lt;&lt; BUTTON_A):
    print("Button A pressed")

if not buttons &amp; (1 &lt;&lt; BUTTON_B):
    print("Button B pressed")

if not buttons &amp; (1 &lt;&lt; BUTTON_SELECT):
    print("Button Select pressed")

if not buttons &amp; (1 &lt;&lt; BUTTON_START):
    print("Button Start pressed")
```

Using the Gamepad Demo

This example prints the x and y analog joystick values to the serial console, as well as the button presses. Try moving the joystick to see the x and y values change, and press the buttons to see the button name.



Python Docs

Python Docs (https://adafru.it/18EG)

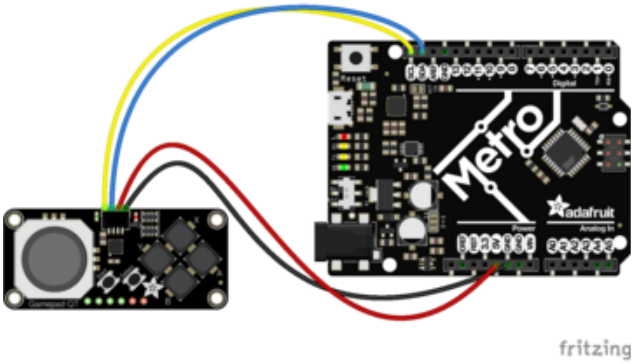
Arduino

Using the Mini I2C STEMMA QT Gamepad with Arduino involves wiring up the Gamepad to your Arduino-compatible microcontroller, installing the Adafruit_Seesaw (https://adafru.it/BrV) library and running the provided example code.

Wiring

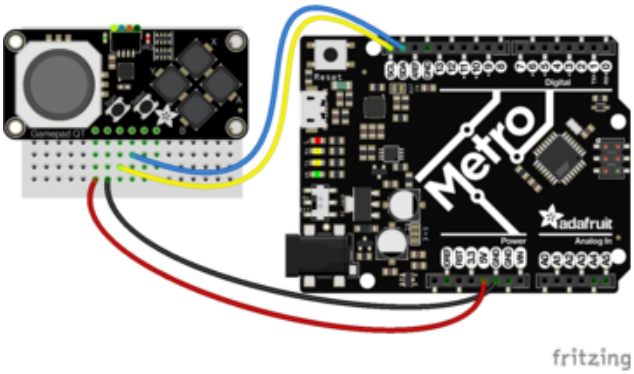
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the Gamepad VIN.

Here is an Adafruit Metro wired up to the Gamepad using the STEMMA QT connector:



Board 5V to **Gamepad VIN (red wire)**
Board GND to **Gamepad GND (black wire)**
Board SCL to **Gamepad SCL (yellow wire)**
Board SDA to **Gamepad SDA (blue wire)**

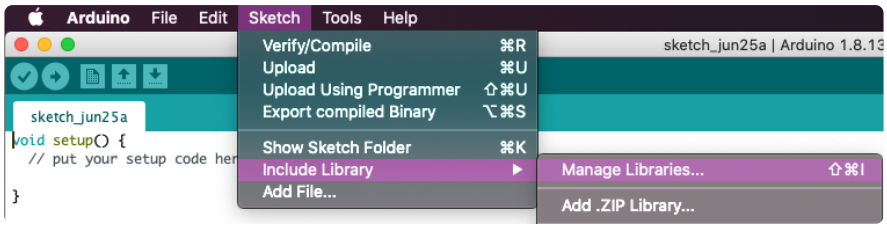
Here is an Adafruit Metro wired up using a solderless breadboard:



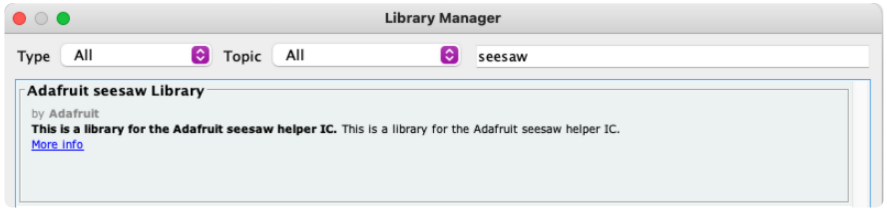
Board 5V to **Gamepad VIN (red wire)**
Board GND to **Gamepad GND (black wire)**
Board SCL to **Gamepad SCL (yellow wire)**
Board SDA to **Gamepad SDA (blue wire)**

Library Installation

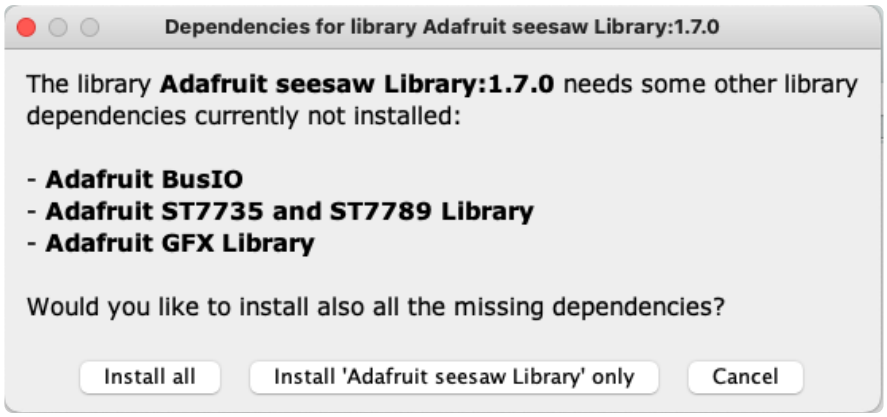
You can install the **Adafruit_Seesaw** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **seesaw**, and select the **Adafruit seesaw Library** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

```
#include "Adafruit_seesaw.h"

Adafruit_seesaw ss;

#define BUTTON_X         6
#define BUTTON_Y         2
#define BUTTON_A         5
#define BUTTON_B         1
#define BUTTON_SELECT    0
#define BUTTON_START    16
```

```
uint32_t button_mask = (1UL << BUTTON_X) | (1UL << BUTTON_Y) | (1UL << BUTTON_START)
|
                       (1UL << BUTTON_A) | (1UL << BUTTON_B) | (1UL <<
BUTTON_SELECT);

//#define IRQ_PIN   5


void setup() {
  Serial.begin(115200);

  while(!Serial) {
    delay(10);
  }

  Serial.println("Gamepad QT example!");

  if(!ss.begin(0x50)){
    Serial.println("ERROR! seesaw not found");
    while(1) delay(1);
  }
  Serial.println("seesaw started");
  uint32_t version = ((ss.getVersion() >> 16) & 0xFFFF);
  if (version != 5743) {
    Serial.print("Wrong firmware loaded? ");
    Serial.println(version);
    while(1) delay(10);
  }
  Serial.println("Found Product 5743");

  ss.pinModeBulk(button_mask, INPUT_PULLUP);
  ss.setGPIOInterrupts(button_mask, 1);

#if defined(IRQ_PIN)
  pinMode(IRQ_PIN, INPUT);
#endif
}


int last_x = 0, last_y = 0;

void loop() {
  delay(10); // delay in loop to slow serial output

  // Reverse x/y values to match joystick orientation
  int x = 1023 - ss.analogRead(14);
  int y = 1023 - ss.analogRead(15);

  if ( (abs(x - last_x) > 3)  ||  (abs(y - last_y) > 3)) {
    Serial.print("x: "); Serial.print(x); Serial.print(", "); Serial.print("y: ");
Serial.println(y);
    last_x = x;
    last_y = y;
  }

#if defined(IRQ_PIN)
  if(!digitalRead(IRQ_PIN)) {
    return;
  }
#endif

    uint32_t buttons = ss.digitalReadBulk(button_mask);

    //Serial.println(buttons, BIN);

    if (! (buttons & (1UL << BUTTON_A))) {
      Serial.println("Button A pressed");
    }
    if (! (buttons & (1UL << BUTTON_B))) {
```

```
      Serial.println("Button B pressed");
    }
    if (! (buttons & (1UL << BUTTON_Y))) {
      Serial.println("Button Y pressed");
    }
    if (! (buttons & (1UL << BUTTON_X))) {
      Serial.println("Button X pressed");
    }
    if (! (buttons & (1UL << BUTTON_SELECT))) {
      Serial.println("Button SELECT pressed");
    }
    if (! (buttons & (1UL << BUTTON_START))) {
      Serial.println("Button START pressed");
    }
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. You'll see the seesaw firmware recognized by the code. Then, when you press any of the buttons or move the joystick, it will print to the Serial Monitor. You'll also see the interrupt LED light up with each button press.

For details on the concepts used in this example, check out Reading Joystick Values (https://adafru.it/18KC) and Reading Button Presses (https://adafru.it/18KC) on the CircuitPython and Python page in this guide.

Arduino Docs

Arduino Docs (https://adafru.it/18KD)

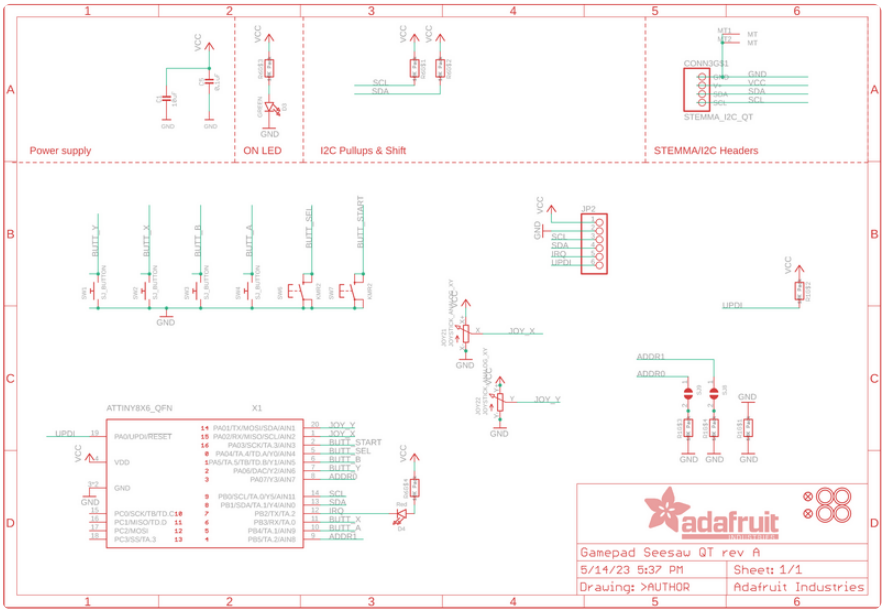
Downloads

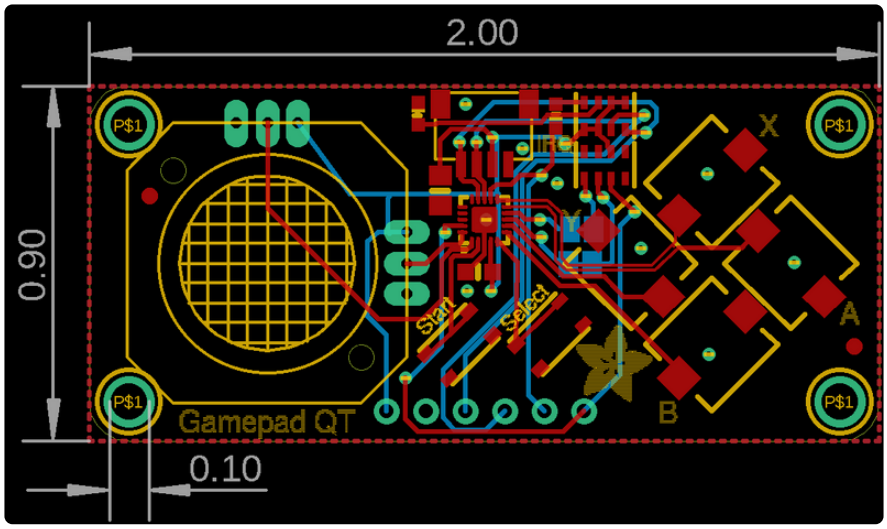
Files

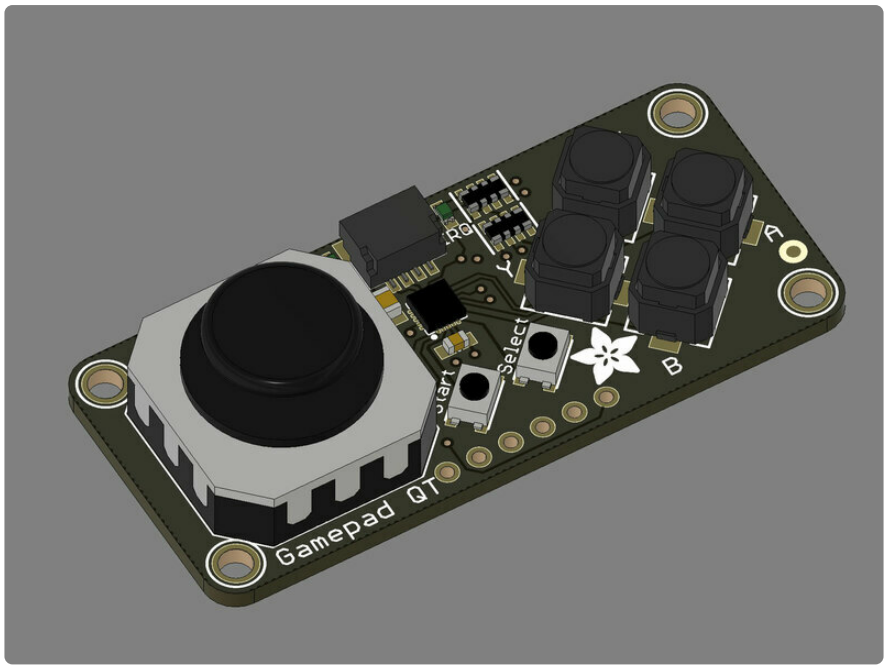
- ATtiny816 Datasheet (https://adafru.it/18EI)
- EagleCAD PCB Files on GitHub (https://adafru.it/18KE)
- 3D models on GitHub (https://adafru.it/19FG)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/18Lb)

Schematic and Fab Print

The x-axis (`JOY_X`) and y-axis (`JOY_Y`) are swapped on the schematic. In the seesaw firmware, X is pin 14, and Y is pin 15.






Y
Select
B
A
Gamepad QT