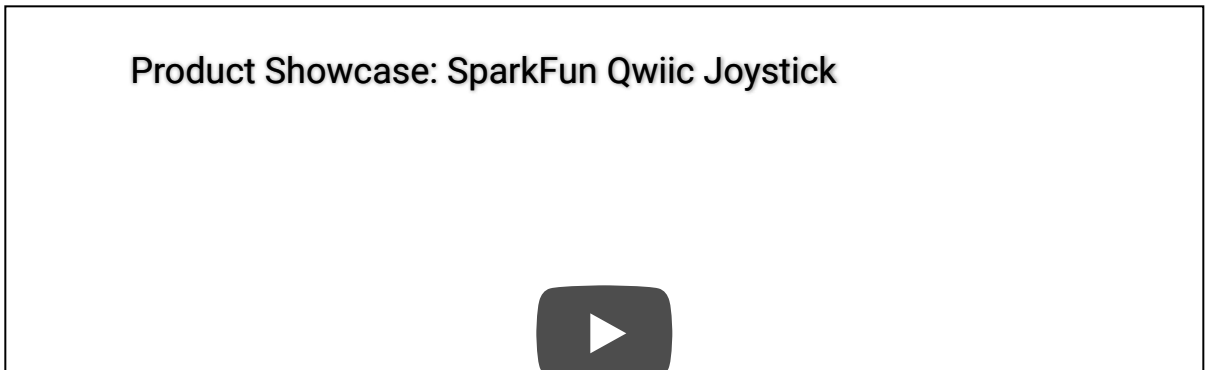
Qwiic Joystick Hookup Guide

Introduction

Now, you can easily add an HID/controller to your project! The SparkFun Qwiic Joystick combines the convenience of the Qwiic connection system and an analog joystick that feels reminiscent of the *thumbstick* from a PlayStation 2 controller.

SparkFun Qwiic Joystick
COM-15168

Thanks to an ATtiny85 on the Qwiic Joystick, all the necessary bits are taken care of and your microcontroller only needs to look for your inputs in the registers of the I^2C device.

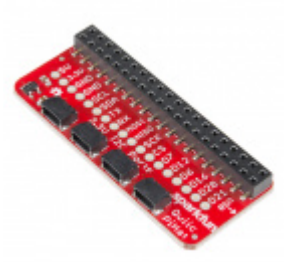


Required Materials

The Qwiic Joystick does need a few additional items for you to get started. The RedBoard Qwiic is for the Arduino examples and the Qwiic Hat is for the Raspberry Pi example (see note below). You may already have a few of these items, so feel free to modify your cart based on your needs.

SparkFun RedBoard Qwiic
DEV-15123

SparkFun Qwiic Cable Kit
KIT-15081



SparkFun Qwiic HAT for Raspberry Pi
DEV-14459

Raspberry Pi Example: If you don't already have them, you will need a Raspberry Pi and standard peripherals. An example setup is listed below.

Raspberry Pi 3
DEV-13825

pi-topCEED (Green)
KIT-14035

Multimedia Wireless Keyboard
WIG-14271

Raspberry Pi™ - 16GB MicroSD NOOBS Card
COM-13945

Suggested Reading

If you're unfamiliar with jumper pads, I^{2}C, or Python be sure to checkout some of these foundational tutorials.

Logic Levels

Learn the difference between 3.3V and 5V devices and logic levels.

I2C

An introduction to I2C, one of the main embedded communications protocols in use today.

How to Work with Jumper Pads and PCB Traces

Handling PCB jumper pads and traces is an essential skill. Learn how to cut a PCB trace, add a solder jumper between pads to reroute connections, and repair a trace with the green wire method if a trace is damaged.

RedBoard Qwiic Hookup Guide

This tutorial covers the basic functionality of the RedBoard Qwiic. This tutorial also covers how to get started blinking an LED and using the Qwiic system.

Raspberry Pi SPI and I2C Tutorial

Learn how to use serial I2C and SPI buses on your Raspberry Pi using the wiringPi I/O library for C/C++ and spidev/smbus for Python.

Qwiic HAT for Raspberry Pi Hookup Guide

Get started interfacing your Qwiic enabled boards with your Raspberry Pi. This Qwiic connects the I2C bus (GND, 3.3V, SDA, and SCL) on your Raspberry Pi to an array of Qwiic connectors.

Python Programming Tutorial: Getting Started with the Raspberry Pi

This guide will show you how to write programs on your Raspberry Pi using Python to control hardware.

If you aren't familiar with the Qwiic system, we recommend reading here for an overview.

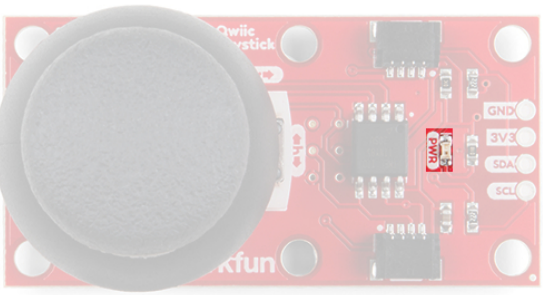
Qwiic Connect System

Hardware Overview

The Qwiic Joystick will report the *thumbstick* position over I^2C and is designed to be compatible with the Qwiic system so you can add it to your project solder-free!

Power LED

There is a power status LED to help make sure that your Qwiic Joystick is getting power. You can power the board either through the *polarized* **Qwiic connector** system or the **breakout pins** (**PWR** and **GND**) provided. The Qwiic system is meant to run on **3.3V**, be sure that you are **NOT** using another voltage when using the Qwiic system.



Joystick

The joystick is similar to the *analog* joysticks on PS2 (PlayStation 2) controllers. Directional movements are simply measured with two **10 kΩ** potentiometers, one for each axis. This joystick also has a select button that is actuated when the joystick is pressed down.

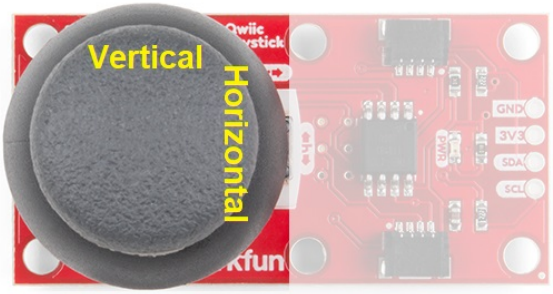
Thumb Joystick
COM-09032

The two potentiometers are connected to the analog to digital converter (ADC) of the ATtiny85. The select button operates as a digital input on the ATtiny85. This allows the firmware to read the knob position and any button presses.

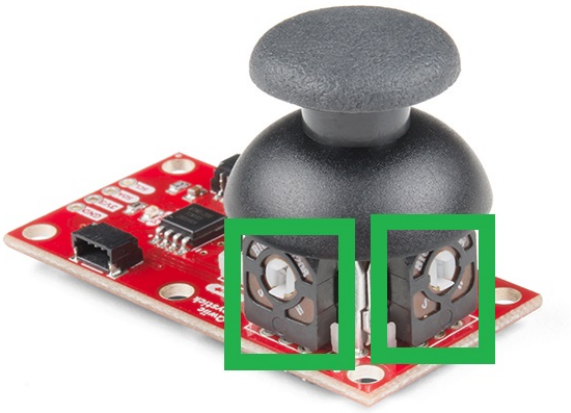
Troubleshooting Tip: Due to resistor manufacturing tolerances, the joystick may not read exactly half of VCC when at rest. You may want to make a note of the "zero" reading, and store that value into your program if necessary.

Joystick Orientation

The joystick contains two potentiometers, connected with a gimbal mechanism that separates the *horizontal* and *vertical* movements (orientation shown below).

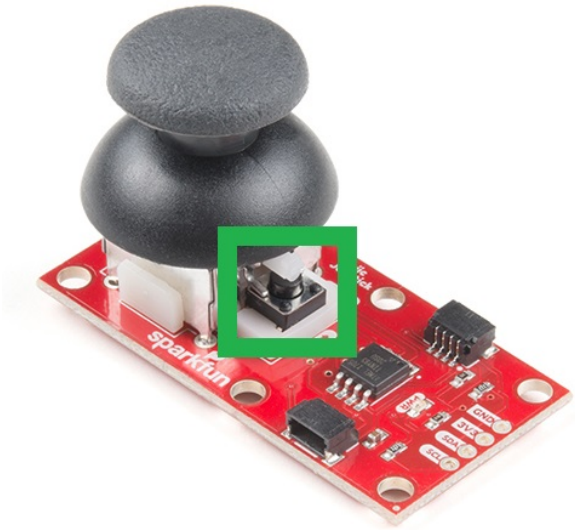


The potentiometers are the two blue or black boxes on the sides of the joystick. If you move the joystick while watching the center shaft of each potentiometer, you'll see that each of the potentiometers will pick up movement on only one axis. Clever, isn't it?



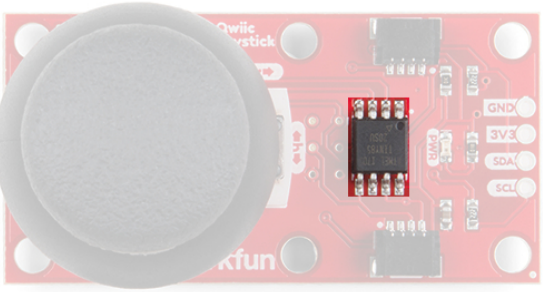
Select Button

The joystick also contains a momentary button which activates when you push down on the cap. The button is the small black box on the side of the joystick. If you push down on the cap, you can see a lever pushing down on the head of the button. The lever works no matter what position the joystick is in. That is pretty cool!



ATtiny85

With the pre-installed firmware, the ATtiny85 is acts as an intermediary (microcontroller) for the analog and digital inputs from the joystick. This allows the Qwiic Joystick to report its position over I^{2}C.



> **Note:** To flash your own firmware, at minimum, you will need an programmer for the ATtiny and a pogo pin adapter. Writing and uploading firmware is not easily done, unless you know what you are doing; this is **inadvisable** for beginners.

Qwiic & I^{2}C

I^{2}C Address

In the firmware, the Qwiic Joystick's I^{2}C address is configurable so you can add a bunch of them to the same bus without collision issues.

Factory Default I^{2}C Slave Address: 0x20

I^{2}C Registers

Address	Description
0x00	Default I^{2}C Slave Address from EEPROM
0x01 - 0x02	Firmware Version (MSB First)

0x03 - 0x04	Current Horizontal Position (MSB First)
0x05 - 0x06	Current Vertical Position (MSB First)
0x07	Current Button Position
0x08	Button Status: Indicates if button was pressed since last read of button state. Clears after read.
0x09	Configuration or "*Locking*" Register - Prevents random I^{2}C address changes during I^{2}C scans and register dumps. Must be set to 0x13 before an address change gets saved to the EEPROM.
0x0A	Current/Set I^{2}C Slave Address (Write). *Stored in EEPROM.*

Note: In the registers for the joystick position, the MSB contains the first 8 bits of the 10-bit ADC value and the LSB contains the last two bits. As an example, this is how the library converts the two registers back to a 10-bit value.

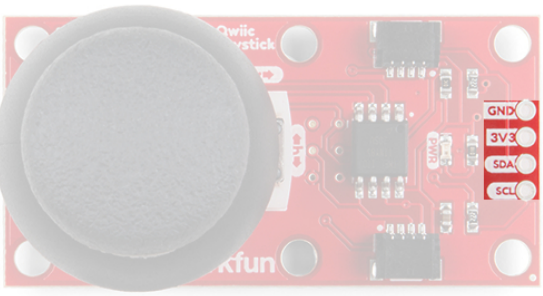
```
uint_16t full10bitvalue = ((MSB << 8) | LSB) >> 6;
```

You could potentially only look at the MSB and get an 8-bit reading (for 256 positions). The firmware was intentionally written this way in the hopes that it would be useful for customers who didn't need the full resolution of the joystick position.

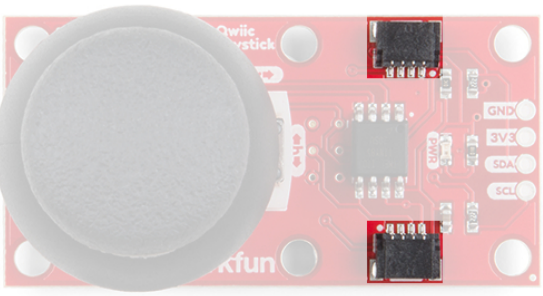
Troubleshooting Tip:
Please be aware, the first time the joystick position registers are read, it will have a maximum value (all 1's in binary). I am not sure if it is something in the firmware or if has something to do with the ATtiny85, but the initial read will be 1023. Once that is read, everything reading after should be accurate (excluding the manufacturing tolerances mentioned above). If you have a fix for this issue, please feel free to comment on this tutorial or create a pull request on the GitHub repository.

Connections

The board provides four labeled breakout pins. You can connect these lines to the I^{2}C bus of your microcontroller and power pins (**3.3V** and **GND**), if it doesn't have a Qwiic connector.

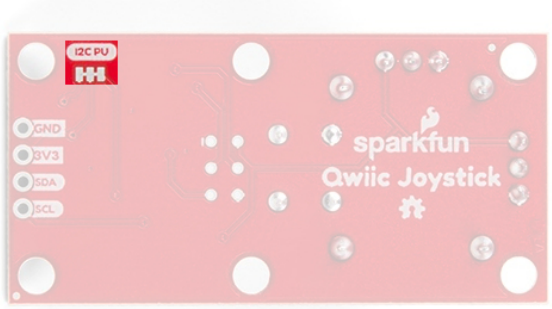


However, the simplest way to use the Qwiic Joystick is through the Qwiic connect system. The connectors are polarized for the I^{2}C connection and power. (**They are tied to the breakout pins.*)



I²C Pull-up Jumpers

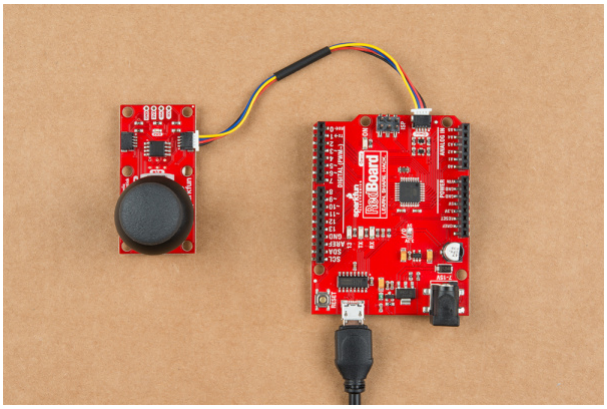
Cutting the **I²C** jumper will remove the 2.2k Ohm resistors from the I²C bus. If you have many devices on your I²C bus you may want to remove these jumpers. Not sure how to cut a jumper? Read here!



Hardware Assembly

Arduino Examples

With the Qwiic connector system, assembling the hardware is simple. All you need to do is connect your Qwiic Joystick to the RedBoard Qwiic with a Qwiic cable. Otherwise, you can use the I²C pins of your microcontroller; just be aware of logic levels, that the operating voltage of the ATtiny85 ranges from 1.8 to 5.5V, and the EEPROM needs at least 2V or it will get corrupted .



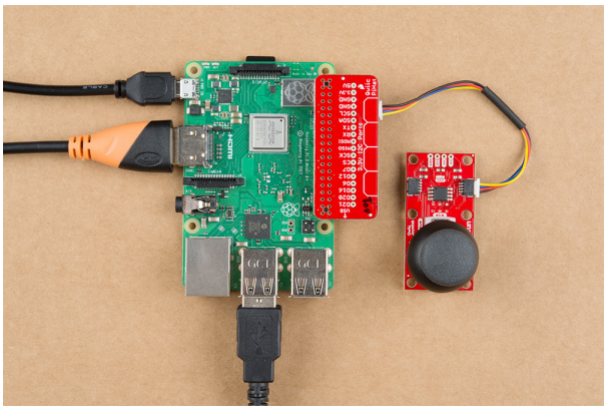
Note: This tutorial assumes you are familiar with Arduino products and you are using the latest stable version of the Arduino IDE on your desktop. If this is your first time using the Arduino IDE, please review our tutorial on installing the Arduino IDE.

Troubleshooting Tip:
It is best to power the board at the recommended 3.3V of the Qwiic system. However, if you have a board with firmware version 2.3 and happen to accidentally corrupt the EEPROM and inject a new, random I^2C address. There is a way to replace the I^2C address and store it in the EEPROM again. Follow the following steps exactly, you will need to use the Arduino Library from below:

1. Connect the Qwiic Joystick to a RedBoard Qwiic with a Qwiic cable.
2. Run Example 3 to find the current I^2C address.
3. Modify Example 2 to use the I^2C address you just found.
4. Run Example 2 and change the I^2C address back to the default address.
5. Unplug and the Qwiic Joystick and plug it back in. The EEPROM should be reset to the default I^2C address and be ready to use again.

Raspberry Pi Example

With the Qwiic connector system, assembling the hardware is simple. You will need a Qwiic cable, a SparkFun Qwiic HAT for Raspberry Pi, and a Raspberry Pi setup with the Raspbian OS, monitor, and standard peripherals. (**If you are unfamiliar with the Qwiic Hat, you can find the Hookup Guide here*.)



Note: This tutorial assumes you are familiar with using a Raspberry Pi and you have the latest (full... with recommended software) version of Raspbian OS your Raspberry Pi. You can download the latest version of the Raspbian OS from the Raspberry Pi Foundation website. As of Feb. 13th 2019, we recommend the **Raspbian Stretch with desktop and recommended software** option.

If this is your first time using a Raspberry Pi, please head over to the Raspberry Pi Foundation website to use their quickstart guides. We have listed a few of them here:

1. Setting up your Raspberry Pi
2. Using your Raspberry Pi
3. Documentation:
   Setup Documentation
   Installation Documentation
   Raspbian Documentation
   SD card Documentation

Arduino Library

The easiest way to install the Arduino library is by searching **SparkFun Qwiic Joystick** inside the Arduino library manager. To manually install, head on over to the GitHub Page or feel free to download the library here!

DOWNLOAD: SPARKFUN QWIIC JOYSTICK LIBRARY (ZIP)

Library Functions

The Arduino library is commented and the functions should be self-explanatory. However, below is a detailed list of the available library functions.

`.begin()` or `.begin(Wire, deviceAddress)`
Creates a connection to an I^{2}C device over the I^{2}C bus using the default or specified I^{2}C address.

- Input: uint8_t *deviceAddress*
  - Unasssigned 8-bit integer for device address. If not defined, the library will use the default I^{2}C address stored in the I^{2}C library (0x20).
- Output: Boolean
  - True- Connected to I^{2}C device.
  - False- No device found or connected.

`isConnected()`
Tries to connect to the device at the I^{2}C address stored in the library.

- Output: Boolean
  - True- Connected to I^{2}C device.
  - False- No device found or connected.

`setI2CAddress(newAddress)`
Changes the I^{2}C address to *newAddress*. Once changed, the new I^{2}C address is saved in the EEPROM of the Qwiic Joystick. Reconnects to device using new I^{2}C address. (*The new address is printed out in the Serial Monitor.*)

- Input: uint8_t *newAddress*
  - Unasssigned 8-bit integer for device address.

`getHorizontal()`
Returns the ADC value of the joystick horizontal position.

- Output: uint16_t
  - Unasssigned 10-bit integer [0 - 1023] for ADC value of the joystick position in the horizontal axis.

`getVertical()`
Returns the ADC value of the joystick vertical position.

- Output: uint16_t
  - Unasssigned 10-bit integer [0 - 1023] for ADC value of the joystick position in the vertical axis.

`getButton()`
Returns the current button state (clears check button indicator - see below).

- Output: byte
  - 0 - Button is currently being pressed.
  - 1 - Button is currently unpressed.

`checkButton()`
Indicates if the button was pressed between reads of the button state or calls of this function. The register for this indicator is then cleared.

Output: byte
0 - Button is was NOT pressed.
1 - Button is was pressed.

`getVersion()`
Returns a string of the firmware version number.

Output: String *AAA.BBB*
AAA - Firmware Version (Major)
BBB - Firmware Version (Minor)

Examples

There are four examples in the Qwiic Joystick Arduino Library to get you started with using Qwiic Joystick. These example files are named with self-explanatory titles, but in case you want more information, see the descriptions below.

- **Example1_Basic_Readings**

  This example connects to the Qwiic Joystick using the default I^2C address saved in the library. The sketch looks at the registers for the current joystick position and button state. The example then prints out those values over the Serial Monitor.

- **Example2_I2C_Address_and_Firmware**

  This example connects to the Qwiic Joystick using the default I^2C address set in the sketch and then prints out the I^2C address and firmware version over the Serial Monitor. The sketch then, takes an input from the Serial Monitor to change the I^2C address using a decimal value (DEC). Once the I^2C address is changed, it is stored in the EEPROM of the Qwiic Joystick. After, the sketch connects to the joystick using the new I^2C address and reads the registers for the firmware version again. The example then prints out those values over the Serial Monitor again.

- **Example3_I2C_Scanner**

  This example is from the I^2C scanner example from the Arduino Playground webpage. The sketch scans for devices on the I^2C bus and reports them on the Serial Monitor. This is useful for users who have forgotten the changed I^2C address of their boards.

- **Example4_Joystic_Serial_Output**

  This example shows a basic example of how to begin to use the Qwiic Joystick as an HID. The example sketch takes values of the joystick and converts it to directional printouts on the Serial Monitor.

Arduino Examples

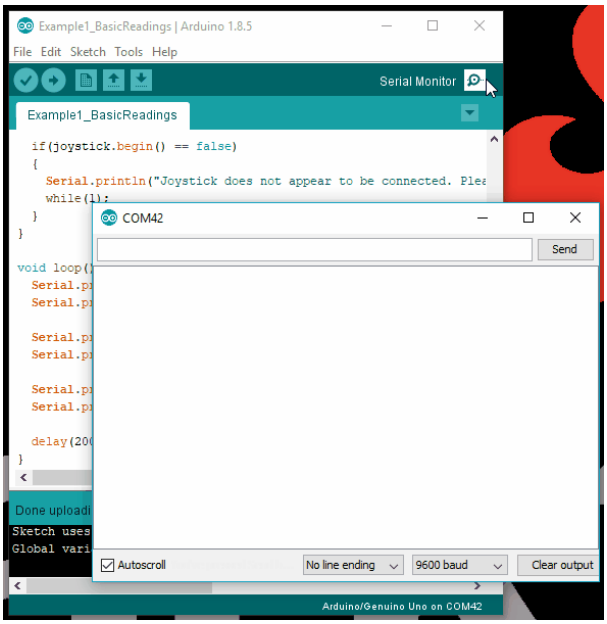
Note: This section is an example of using the Qwiic Joystick and the RedBoard Qwiic with the Arduino IDE. It is not intended to be a guide for using I^2C devices with the Arduino IDE.

Please use the following links and the internet for a better understanding of I^2C and how it works in the Arduino IDE:

- A tutorial on I^{2}C.
- An in-depth overview of the Wire (I^{2}C) library.

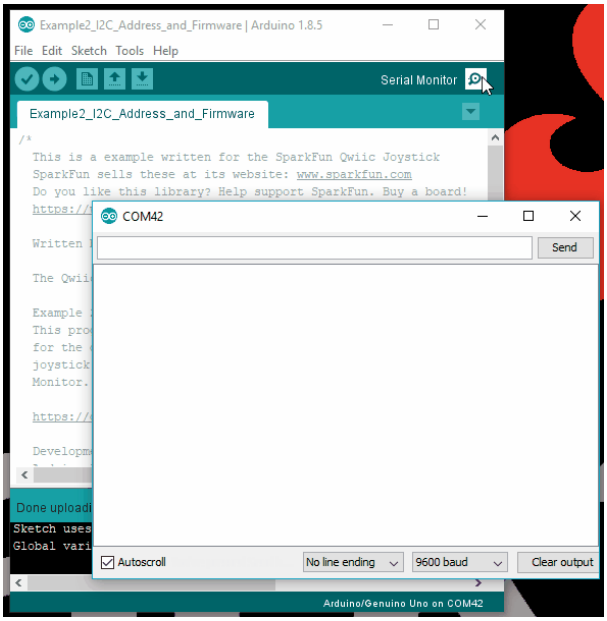
Example 1: Basic Readings

This sketch looks at the registers for the current joystick position and button state; the values are sent to the Serial Monitor. Make sure you are using a baudrate of **9600bps**.



Example 2: I^{2}C Address Change and Firmware Version Read

This sketch takes an entry from the Serial Monitor, in a decimal value, and changes the I^{2}C address of the device. The firmware version is then read to confirm that the address was changed. Make sure you are using a baudrate of **9600bps**.



Example 3: I^{2}C Scanner

This sketch is from the I^{2}C scanner example from the Arduino Playground webpage. The sketch scans for devices on the I^{2}C bus and reports them on the Serial Monitor. Make sure you are using a baudrate of **9600bps**.

Example 4: Joystick Serial Output

This sketch takes values of the joystick and converts it to directional printouts on the Serial Monitor. It also reads out if the button was pressed. Make sure you are using a baudrate of **9600bps**.

Python Examples

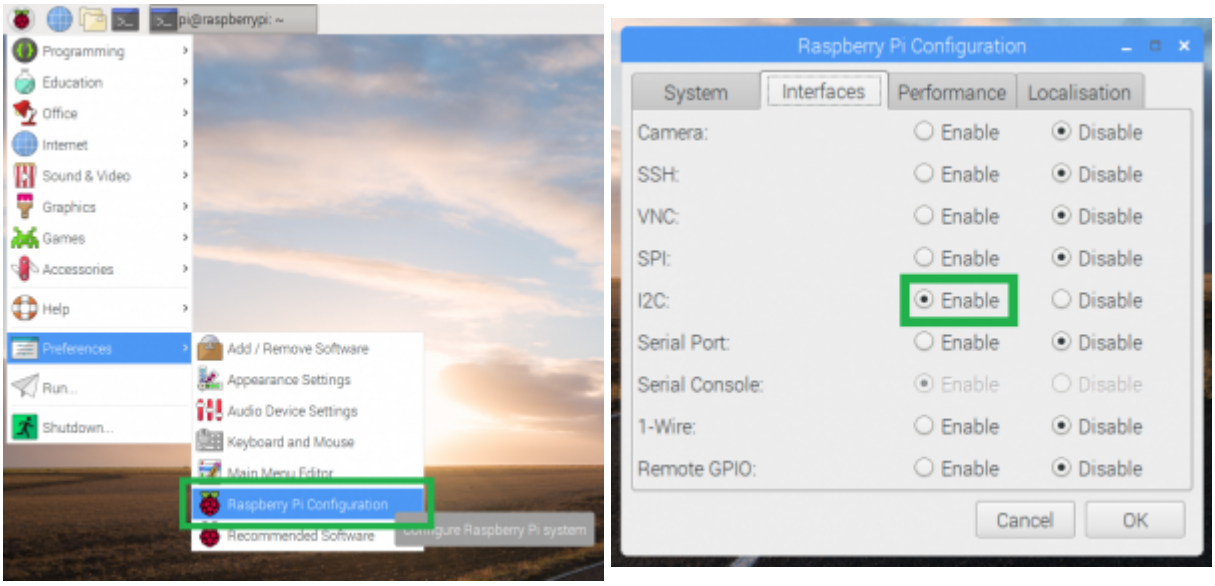
Note: This section is an example of using the Qwiic Joystick with the Raspberry Pi in Python. It is not intended to be a guide for using I^{2}C devices with a Raspberry Pi or a tutorial on Python.

Please use the following links and the internet for a better understanding of I^{2}C and how it works in the Arduino IDE:

- For a tutorial on using I^{2}C devices with a Raspberry Pi, please see our Raspberry Pi SPI and I^{2}C Tutorial.
- For a tutorial on Python, please see our Python Programming Tutorial: Getting Started with the Raspberry Pi.

Enable I^{2}C Interface

If it isn't already enabled, enable I^{2}C interface in Preferences > Raspberry Pi Configuration panel.



Enabling the I²C through the configuration panel.

Check for Device Using Command Line

On the (full) Raspbian OS, `i2ctools` should be pre-installed (if it is not, follow the instructions here). With the Qwiic Joystick and Qwiick Hat attached to the Raspberry Pi, run the following command in the terminal/commandline.

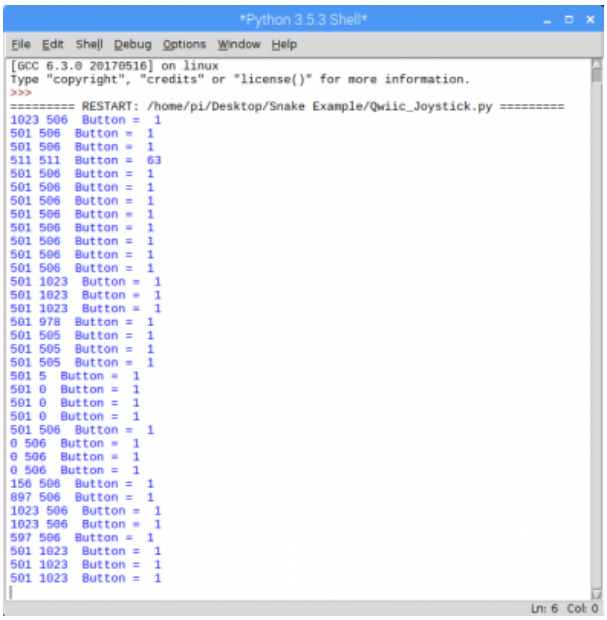
```
i2cdetect -y 1
```

Download and Run Python Examples

Download the Python Examples from the GitHub repository. All you need to do is open an example python file and execute the module. You can do this in Python 3 or another compatible, programming text editor like Geany.

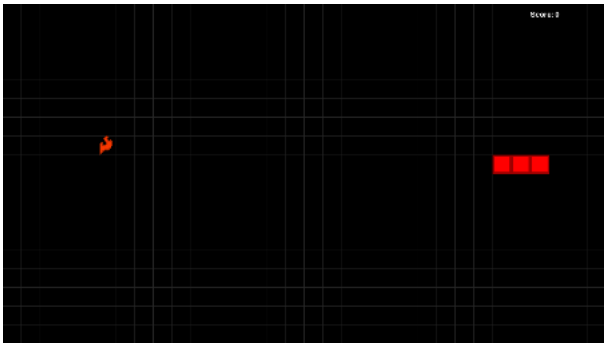
Basic Readings

Below is a screenshot of the python shell from the **Basic Readings** example file.



Snake Game

When you run the **Snake Game** example code, you only need to click the button to start the game. (**Shout out to Al Sweigart for his **Wormy** example, which was adapted to work with the Qwiic Joystick.*)



Troubleshooting Tip: I did run into an error with the python example codes. Unfortunately, I am not an *expert* in Python and the Raspberry Pi. However, I got around this by having the error print out instead of terminating the script. From what I could deduce based on the available information on the internet, the error `[Errno 121] Remote I/O error` is an issue with the I^{2}C bus and how SMBUS interacts with the slave device.

You will also get an `[Errno 121] Remote I/O error` if your I^{2}C address is defined wrong.

Resources and Going Further

For more information, check out the resources below:

Product Information

- GitHub Product Repo
- ATtiny85 Firmware
- Schematic (PDF)
- Eagle Files (ZIP)
- SFE Product Showcase

Example Code and Arduino Library

- Qwiic Joystick Arduino Library
- Qwiic Joystick Python Example Code

Need help getting started with Arduino and I^{2}C? Check out these resources here:

- Arduino I^{2}C Scanner Example
- Arduino Wire Library Reference Page
- Arduino Wire Library (In-Depth) Reference

Need help getting started with your Raspberry Pi? Check out these resources here:

- Setting up your Raspberry Pi
- Using your Raspberry Pi
- Documentation:
    - Setup Documentation
    - Installation Documentation
    - Raspbian Documentation

- SD card Documentation

Need help getting started with Python and I^{2}C? Check out these resources here:

- Python Programming Tutorial: Getting Started with the Raspberry Pi
- Raspberry Pi SPI and I^{2}C Tutorial

Need some inspiration for your next project? Check out some of these other Qwiic product tutorials:

Qwiic Micro OLED Hookup Guide

Get started displaying things with the Qwiic Micro OLED.

Qwiic Distance Sensor (VL53L1X) Hookup Guide

The Qwiic VL53L1X time of flight sensor is capable of several modes, as well as having a range of 4M. Let's hook it up and find out just how far away that thing over there is.

ESP32 LoRa 1-CH Gateway, LoRaWAN, and the Things Network

Using the ESP32 LoRa 1-CH Gateway as a gateway and device, and pushing data to The Things Network.

GPS-RTK Hookup Guide

Find out where you are! Use this easy hook-up guide to get up and running with the SparkFun high precision GPS-RTK board.