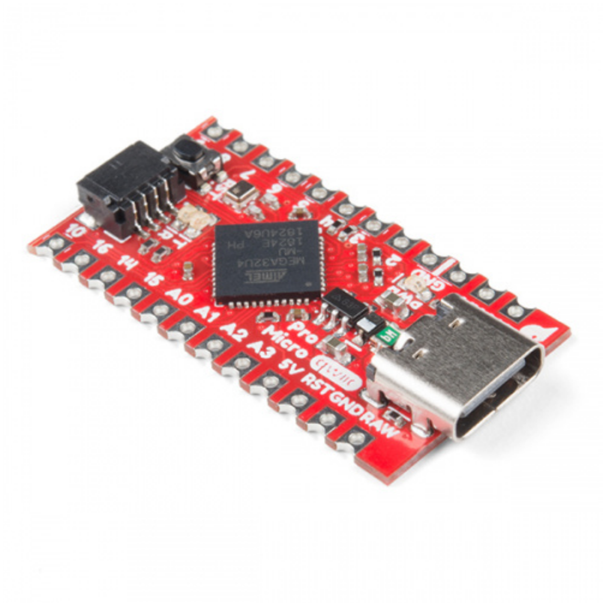
Qwiic Pro Micro USB-C (ATmega32U4) Hookup Guide

Introduction

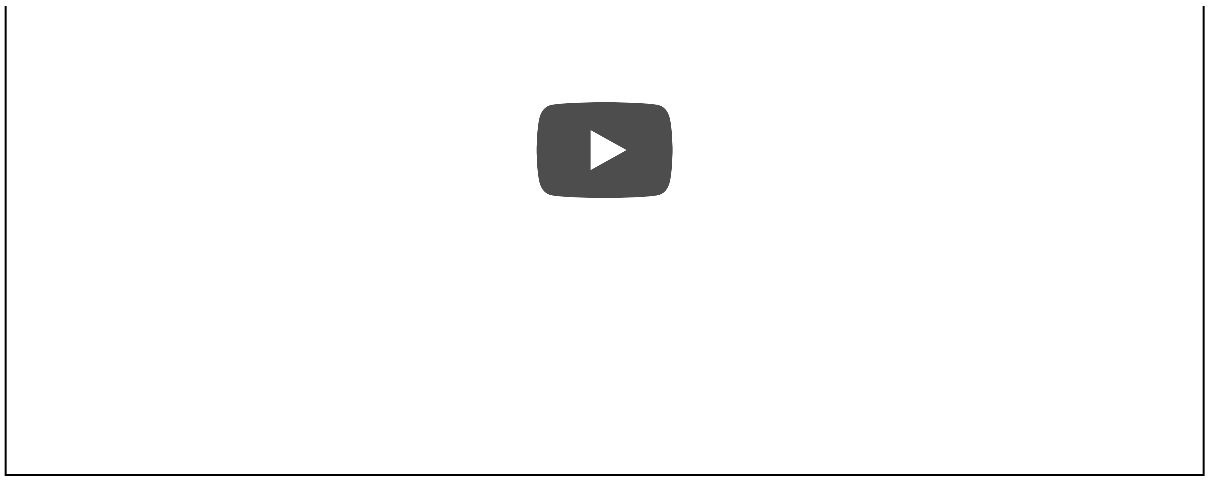
Heads up! This is for the **Qwiic Pro Micro USB-C (ATmega32U4)** version. If you are looking for information about hardware on the *Pro Micros with the micro-b USB connector*, make sure to check out the older Pro Micro & Fio V3 Hookup Guide.

The SparkFun Qwiic Pro Micro USB C is a really cool, little development board. It's an Arduino-compatible microcontroller, micro-sized, and it accomplishes with one single chip what old Arduino Unos, Duemilanoves, and Diecimeillas could never dream of: *true USB functionality*. In this tutorial, we'll check out the updated Qwiic Pro Micro and go through a few examples to get you started!



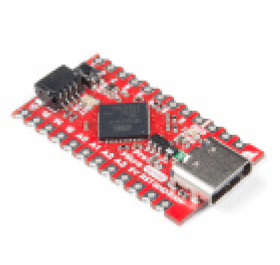
SparkFun Qwiic Pro Micro - USB-C (ATmega32U4)
● DEV-15795

Product Showcase: SparkFun Qwiic Pro Micro



Required Materials

To follow along with this tutorial, you will need the following materials at a minimum. You may not need everything though depending on what you have. Add it to your cart, read through the guide, and adjust the cart as necessary.



SparkFun Qwiic Pro Micro - USB-C (ATmega32U4)
DEV-15795

Reversible USB A to C Cable - 0.8m
CAB-15425

Tools

You will need a soldering iron, solder, and general soldering accessories for a secure connection when using the plated through hole pads.

Soldering Iron - 60W (Adjustable Temperature)
TOL-14456

Solder Lead Free - 15-gram Tube
TOL-09163

Prototyping Accessories

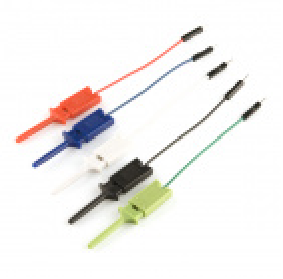
Depending on your setup, you may want to use IC hooks for a temporary connection. However, you will want to solder header pins to connect devices to the plated through holes for a secure connection.

Breadboard - Self-Adhesive (White)

◉ PRT-12002

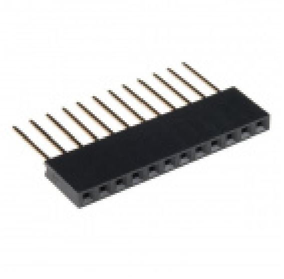
Break Away Headers - Straight

◉ PRT-00116



IC Hook with Pigtail

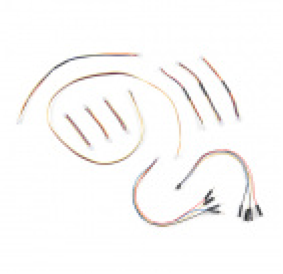
◉ CAB-09741



Photon Stackable Header - 12 Pin

◉ PRT-14322

For those that want to take advantage of the Qwiic enabled devices, you'll want to grab a Qwiic cable.



SparkFun Qwiic Cable Kit

◉ KIT-15081

Qwiic Cable - 100mm

○ PRT-14427

Qwiic Cable - Breadboard Jumper (4-pin)
○ PRT-14425

Qwiic Cable - 500mm
◉ PRT-14429

Suggested Reading

If you aren't familiar with the Qwiic system, we recommend reading here for an overview if you decide to take advantage of the Qwiic connector.

Qwiic Connect System

We would also recommend taking a look at the following tutorials if you aren't familiar with them.

Serial Communication
Asynchronous serial communication concepts: packets, signal levels, baud rates, UARTs and more!

What is an Arduino?
What is this 'Arduino' thing anyway?

Installing Arduino IDE
A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.

Hardware Overview

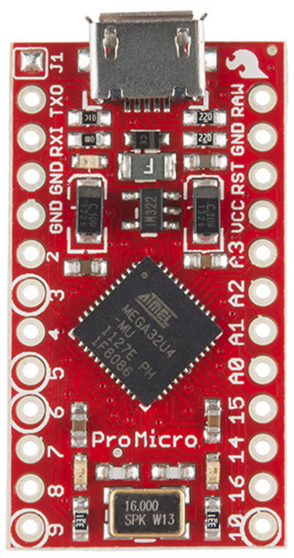
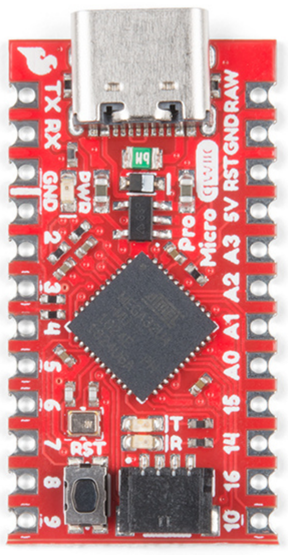
Before we get into installing and using the Qwiic Pro Micro, let's quickly look at the board -- examine its inputs, outputs, and other hardware quirks.

Old School to New School

The Qwiic Pro Micro USBC is a revision of the original Pro Micro. Overall, it is functionally the same as the previous version. The board is the same size as the original Pro Micro but we added a few additional features by shrinking down some components on the board. The major changes to the board that you will notice include:

- Reset Button
- USB C
- Castellated Pads
- Qwiic Connector
- AP2112 3.3V Voltage Regulator

The benefit of the reset button is to quickly reset the board or place it into bootloader mode without the need to take out a piece of jumper wire. The USB micro-b connector has been replaced with the USB type C connector. The through hole pads have castellated edges for each pin to add a lower profile in your projects should you decide build it into another assembly during production. Finally, a Qwiic connector is populated on the bottom of the board to easily add Qwiic enabled I^{2}C devices.

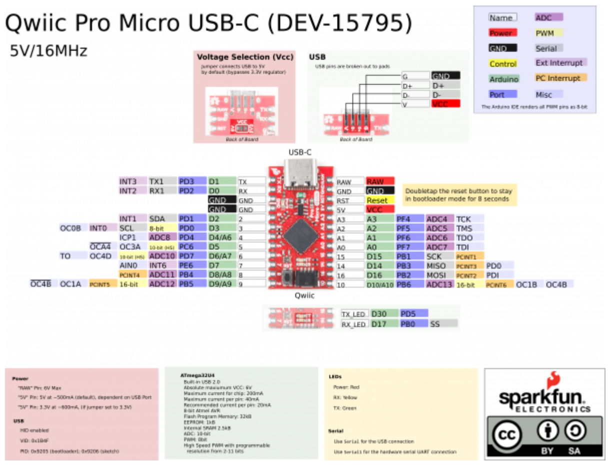
	
Older 5V/16MHz Pro Micro	*Newer 5V/16MHz Qwiic Pro Micro USB C*

A subtle difference that you may not notice is the old Pro Micro used the MIC5219, which was able to handle a maximum voltage input of 16VDC. The new Qwiic Pro Micro uses the AP2112 3.3V voltage regulator which can handle a **maximum of 6V**. The output current is slightly higher in the Qwiic Pro Micro which peaks at about 600mA. You'll need to make sure to use an appropriate voltage regulator and components and regulate the voltage down.

For more information about the differences between the boards, make sure to pull up the schematic for both boards and read further below!

The Pinout

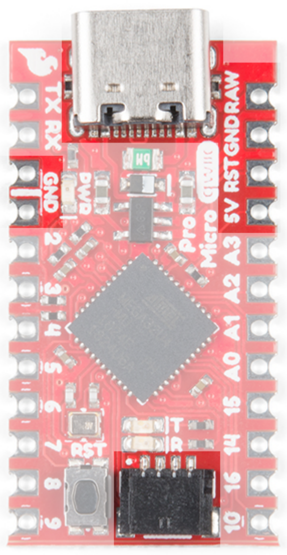
All of the Qwiic Pro Micro's I/O and power pins are broken out to two, parallel headers. Some pins are for power input or output, other pins are dedicated I/O pins. Further, the I/O pins can have special abilities, like analog input. Here's a map of which pin is where, and what special hardware functions it may have:



Click on image for a closer view

Power

There are a variety of power and power-related nets broken out to connectors and through hole pads. Each pad has a castellated edge. The back of the board also has the USB pins broken out for power.

	
Castellated Power and Reset Pins	*USB Power Pins*

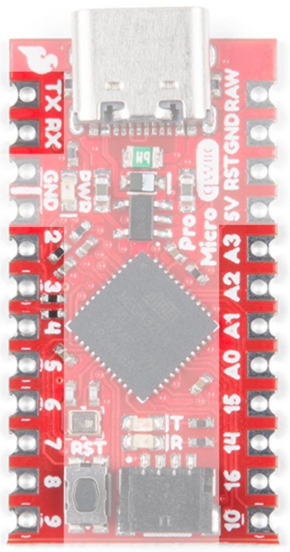
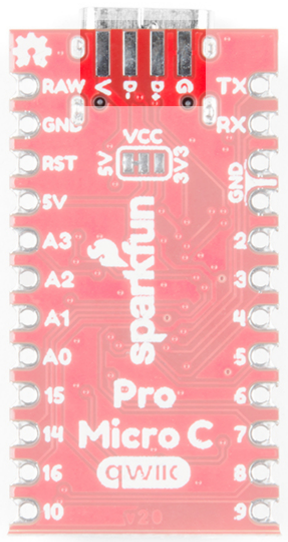
These nets consist of the following:

- **V** is the voltage provided from the USB connector.
- **RAW** is the unregulated voltage input for the Qwiic Pro Micro. If the board is powered via USB (i.e. **V**), the voltage at this pin will be about 4.8V (USB's 5V minus a Schottky diode drop). On the other hand, if the

board is powered externally, through this pin, the applied voltage can be up to **6V**.
- **5V** is the voltage supplied to the on-board ATmega32U4. This voltage will depend on whether you have the jumper set to **5V (by default)** or **3.3V**.
    - When the jumper is set to 5V, it is the voltage applied to the RAW pin. We suggest using a clean 5V power supply if you are connecting 5V devices to the I/O. If the board is powered through the 'RAW' pin (or USB), this pin can be used as an **output** to 5V supply other devices.
    - When the jumper is set to 3.3V, the output is 3.3V. If the board is powered through the 'RAW' pin (or USB), this pin can be used as an **output** to 3.3V supply other devices.
- **RST** can be used to **restart** the Qwiic Pro Micro. There is a built-in reset button to reset the Pro Micro. However, the pin is broken out if you need to access this pin externally. This pin is pulled high by a 10kΩ resistor on the board, and is **active-low**, so it must be connected to ground to initiate a reset. The Qwiic Pro Micro will remain "off" until the reset line is pulled back to high.
- **GND**, of course, is the common, ground voltage (0V reference) for the system.

I/O Pins

The Qwiic Pro Micro breaks out the I/O pins to plated through hole pads on the edge of the board. Each pad is castellated as well. On the back, there are additional pins to connect to the USB data pins.

	
Castellated I/O PTH Pads	*USB Pads*

The Pro Micro's I/O pins -- 18 in all -- are multi-talented. Every pin can be used as a **digital input or output**, for blinking LEDs or reading button presses. These pins are referenced in the Arduino IDE via an integer value between 0 and 21. (The A0-A3 pins can be referenced digitally using either their analog or digital pin number).

Nine pins feature analog to digital converters (ADCs) and can be used as **analog inputs**. These are useful for reading potentiometers or other analog devices using the `analogRead([pin])` function.

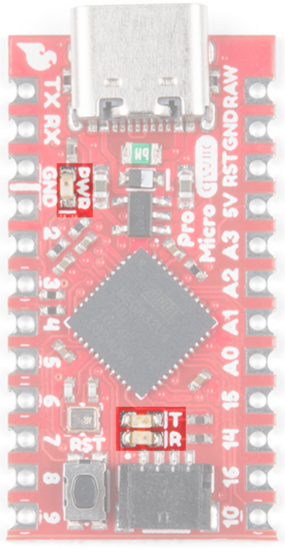
There are five pins with pulse width modulation (PWM) functionality, which allows for a form of **analog output** using the `analogWrite([pin], [value])` function. These pins are indicated on-board with a faint, white circle around the square pads.

There are hardware UART (serial), I^{2}C, and SPI pins available as well. These can be used to interface with digital devices like serial LCDs, XBees, IMUs, and other serial sensors.

The Pro Micro has five external interrupts, which allow you to instantly trigger a function when a pin goes either high or low (or both). If you attach an interrupt to an interrupt-enabled pin, you'll need to know the specific interrupt that pin triggers: pin 3 maps to interrupt 0 (INT0), pin 2 is interrupt 1 (INT1), pin 0 is interrupt 2 (INT2), pin 1 is interrupt 3 (INT3), and pin 7 is interrupt 4 (INT6).

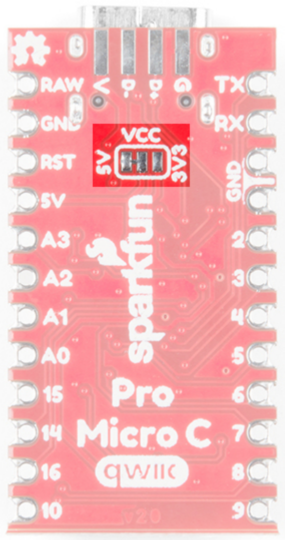
On-Board LEDs

There are three LEDs on the Pro Micro. One red LED indicates whether **power** is present. The other two LEDs help indicate when data is transferring over USB. A yellow LED represents USB data coming *into* (RX) the Pro Micro, and a green LED indicates USB data going out (TX).



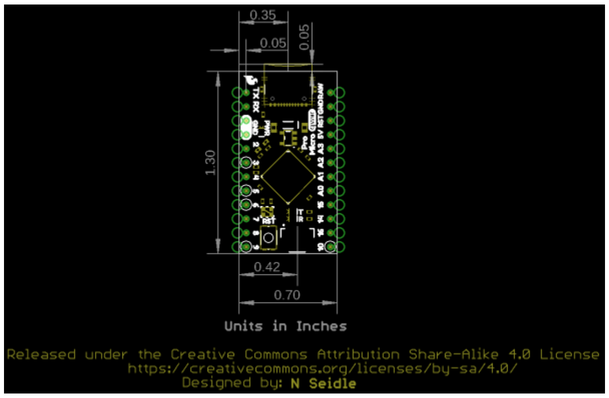
Jumpers

On the back of the board is a jumper that is able to select the system voltage. The default voltage for the board defautls to 5V. By cutting the jumper pad and adding a solder to the between the center and right jumper pads, the system voltage will be set to 3.3V.



Board Dimensions

The board measures **1.30" x 0.70"**. Not included in the image below is the PCB thickness, which is **0.8mm**. This is thinner than a majority of the PCBs used for SparkFun original designs.


0.35
0.05
0.05
1.30
0.42
0.70
Units in Inches
Released under the Creative Commons Attribution Share-Alike 4.0 License
https://creativecommons.org/licenses/by-sa/4.0/
Designed by: N Seidle

How to Power the Qwiic Pro Micro USB C

As the Pro Micro's main feature is its innate USB functionality, the most common way to power it is via **USB**. If the jumper is set to the default 5V, the Qwiic Pro Micro will be powered directly from the USB bus. If the jumper is set to 3.3V, the board will regulate the 5V supply coming in from USB down to 3.3V. The other end of the USB cable can be connected to either a computer, USB hub, USB batteries, or a USB wall adapter, which can (in most cases) provide more power.

Alternatively, if your Qwiic Pro Micro is living out in the wild, out of reach of USB cables, it can be powered through either the '**RAW**' or '**5V**' pins. A supply going into those pins will provide power to the Qwiic Pro Micro. To be safe, it shouldn't be any higher than 6V. If the jumper is set to 3.3V, the voltage will be regulated down to 3.3V.

How, exactly, you power your project is up to you and the demands of your project. If you're making something battery powered, you could use a LiPo boost converter to power the board at 5V. Or you may want to set the jumper to 3.3V, which could be powered by a LiPo battery or a couple alkalines. Make sure to check out the following tutorials for more information.

Battery Technologies

FEBRUARY 6, 2013

The basics behind the batteries used in portable electronic devices: LiPo, NiMH, coin cells, and alkaline.

How to Power a Project

FEBRUARY 7, 2013

A tutorial to help figure out the power requirements of your project.

Hardware Hookup

Header pins were left off the Qwiic Pro Micro to allow users the flexibility of connecting any type of 0.1" header to the board. For temporary connections to the I/O pins, you could use IC hooks to test out the pins. However, you'll need to solder headers or wires of your choice to the board for a secure connection. For advanced users, you could also design a PCB to take advantage of the castellated edges for a lower profile. Here are a few tutorials to connect to the pads depending on your personal preference.

How to Solder: Through-Hole Soldering

SEPTEMBER 19, 2013

This tutorial covers everything you need to know about through-hole soldering.

Working with Wire

FEBRUARY 8, 2013

How to strip, crimp, and work with wire.

How to Solder: Castellated Mounting Holes

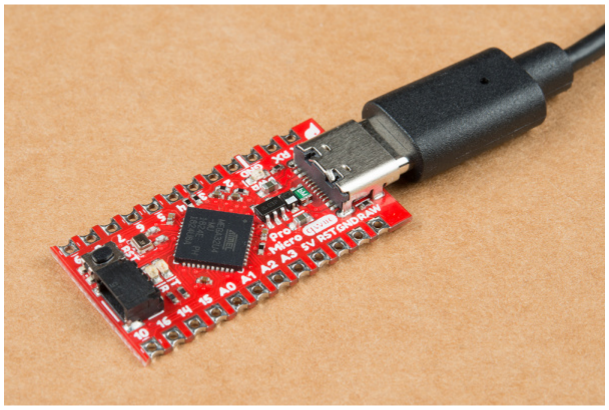
MAY 12, 2015

Tutorial showing how to solder castellated holes (or castellations). This might come in handy if you need to solder a module or PCB to another PCB. These castellations are becoming popular with integrated WiFi and Bluetooth modules.

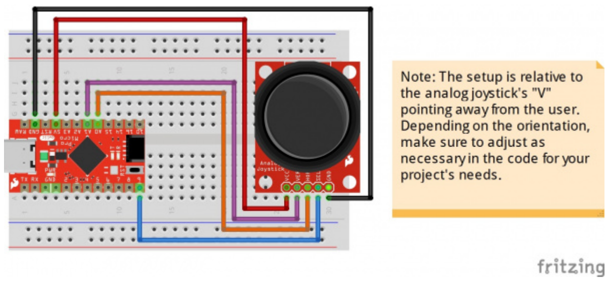
Note: For advanced users taking advantage of the castellated edges, you can design a board with the Qwiic Pro Micro Eagle footprint should you decide to make a keyboard using the Cherry MX Switch!

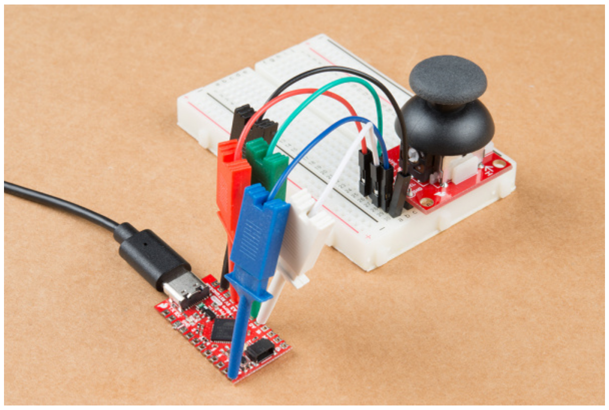
GITHUB SPARKFUN EAGLE LIBRARY: SPARKFUN-BOARDS.LBR

In order to power and upload to the board, you will simply need a USB C cable connected to your computer. For the scope of this tutorial, you'll get as far as example 1.



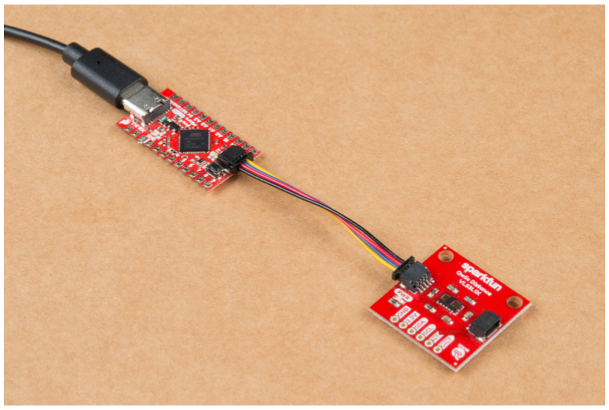
To take advantage of the board's HID, we recommend some sort of input. The analog joystick provides a few inputs to test our the keyboard and mouse in example 2.


Note: The setup is relative to the analog joystick's "V" pointing away from the user. Depending on the orientation, make sure to adjust as necessary in the code for your project's needs.
fritzing



Qwiic Enabled Device

You can also easily connect a Qwiic enabled device to the Qwiic connector. Below is an example of a distance sensor connected to the Qwiic Pro Micro. If you have an installation with a kiosk with the screensaver turned on, you could use the Qwiic Pro Micro to wake it up by moving a mouse if there is someone within a certain distance of the VL53L1X sensor.



Installing Drivers (Windows)

Getting the Qwiic Pro Micro set up on your computer and in your Arduino environment can be difficult. Follow along on this page for a step-by-step guide through the driver installation and Arduino-enabling process.

Windows Driver Installation

Note: On a Windows 10 OS, the driver should automatically install. You may not need to download the driver for the Atmega-32U4-based Arduino. If that is the case, you can move to installing the board add-on for the Arduino IDE.

Step 1: Download the Driver

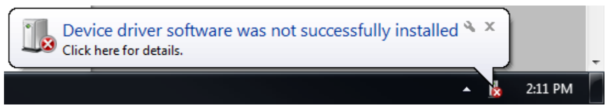
Before plugging your board in, get a head start by downloading the drivers. Check the GitHub Repository for the latest files. The same driver file works for both the Pro Micro and the Fio v3. The drivers for both the Fio and the Pro Micro are signed for Windows users. You can download them directly using the link below.

FIO AND PRO MICRO DRIVERS (ZIP)

Unzip that zip file, and don't forget where you've left its contents. In that zip file, you should find the **.inf** and **.cat** files, which contains all the information Windows needs to install the Pro Micro's driver. The *sparkfun.inf* driver and *sparkfun.cat* digitally signed catalog file will be found in **...Arduino_Boards-master/sparkfun/avr/signed_driver** .

Step 2: Plug in the Qwiic Pro Micro

When you initially **plug the board in**, an "Installing device driver software" bubble notification should pop up in the lower-right corner of your taskbar. After the green dot circles the grey box a number of times, you'll probably get a sad bubble like this:


Device driver software was not successfully installed
Click here for details.

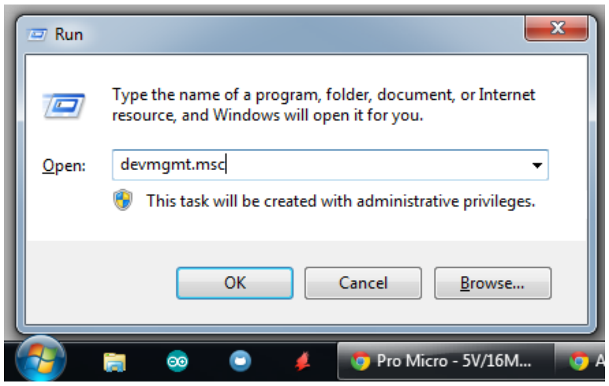
Never fear! Windows just doesn't know where to find our driver.

Note: Some users have experienced issues when plugging the Pro Micro into a USB 3.0 port. If you experience issues on USB 3.0 ports, try switching to use a USB 2.0 port.

Step 3: Open the Device Manager

From here, the most straightforward way to install the driver is through the Device Manager. To get to the Device Manager, click the **Start** button, then open the **Control Panel**. In the Control Panel, click **System and Maintenance**, and then open the **Device Manager**.

Alternatively, you can open the **Run prompt** (Windows key+R) and type 'devmgmt.msc' and click OK.

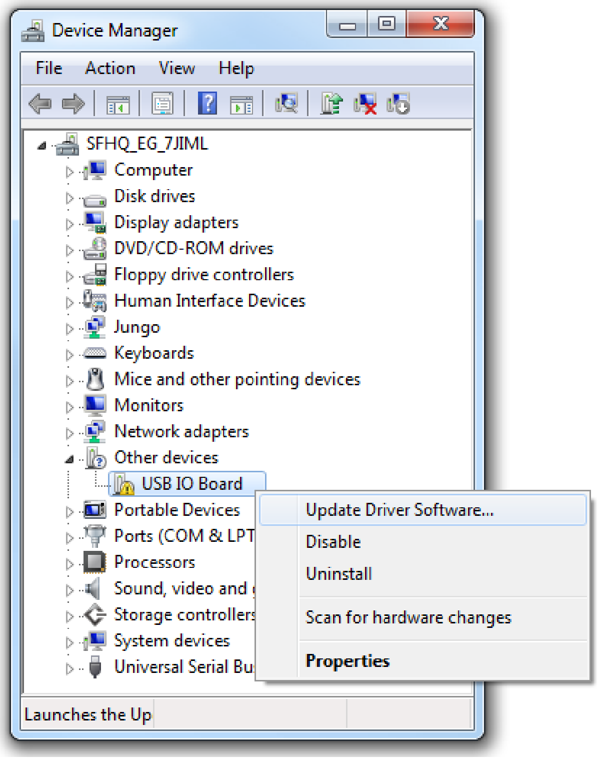

Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: devmgmt.msc

This task will be created with administrative privileges.

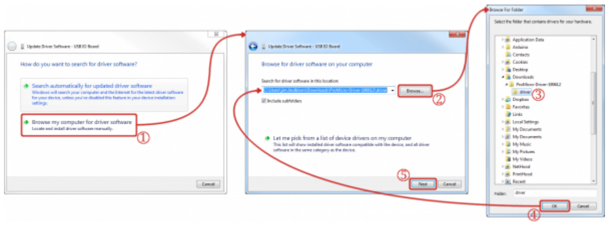
In the Device Manager, expand the 'Other devices' tree, where you should find a **'USB IO Board'** with a yellow warning sign over its icon. Right-click the 'USB IO Board' and select **Update Driver Software...**.



This should spawn an 'Update Driver Software - USB IO Board' window.

Step 4: Finding the Driver

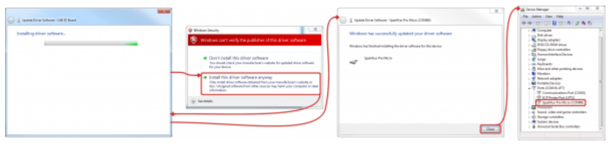
In the first window that pops up, click **'Browse my computer for driver software'**. On the next window, click **'Browse...'** to search for the driver you just downloaded. It should be a folder named '**Arduino_Boards-master**', in a subdirectory noted in step 1. After you've selected the 'driver' folder, click **OK**, then select **Next**.



Click on image for a closer view.

Windows will attempt to do its driver install thing, but not before complaining about the driver being unsigned. It's safe to select **'Install this driver software anyway'** on the warning dialog.

After watching the progress bar beam by a few times, you should get a happy 'Windows has successfully updated your driver software' window. And the 'Device Manager' should have a new entry for the '**SparkFun Pro Micro (COM ##)**' if you have one of those) under the '**Ports (COM & LPT)**' tree.



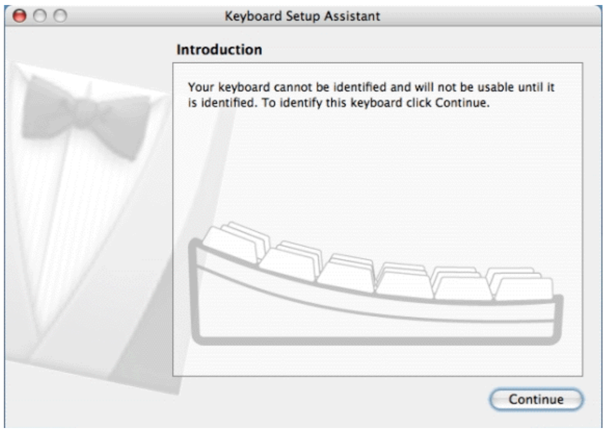
Click on image for a closer view.

Take note of which COM port your Qwiic Pro Micro was assigned. We'll need it soon.

Installing Drivers (Mac and Linux)

If you're using Mac or Linux, follow the steps below to get your Pro Micro ready to go on your computer. We're not going to name names here, but installing the Qwiic Pro Micro on Mac OS X and Linux is *a lot* easier than on other OS's...

When you initially plug your Qwiic Pro Micro into a Mac, it'll pop up a "Keyboard Setup Assistant" window. This stems from the Qwiic Pro Micro's ability to emulate an HID USB device (e.g. keyboards and mice) -- the Mac thinks your Pro Micro is a human input device (which it could be! but isn't yet).



Keyboard Setup Assistant

Introduction

Your keyboard cannot be identified and will not be usable until it is identified. To identify this keyboard click Continue.

There's nothing to configure in this window, so just click the big, red, '**X**' to close it.

That's all there is to it! The CDC (communication device class) portion of your Qwiic Pro Micro (the part that handles USB-to-serial conversion) should automatically install on your computer.

Setting Up Arduino

Note: If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE.

We're still not completely ready for Arduino, but this is the final stretch. Before you can use the Qwiic Pro Micro in the Arduino IDE, you'll need to install the board (*.brd) files for the Qwiic Pro Micro so the Arduino IDE will know how to communicate with your board. Jump to the section based on your operating system for instructions on how to install the board files so that you can select the **SparkFun Pro Micro, ATmega32U4 (5V, 16MHz)**.

- Windows
- Mac and Linux

Installing the Arduino Addon Windows

Using the Board Manager

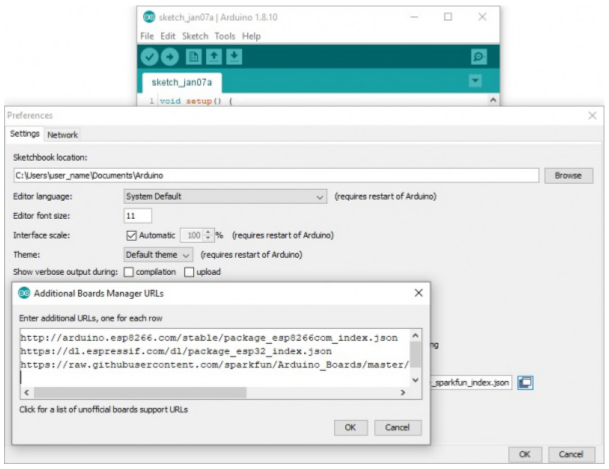
Since the release of Arduino 1.6.4, adding third party boards to the Arduino IDE is easily achieved through the board manager. If you're running an older version of Arduino (v1.6.3 or earlier), we recommend upgrading now. As always, you can download the latest version of Arduino from arduino.cc.

DOWNLOAD THE ARDUINO IDE

To begin, you'll need to point the Arduino IDE board manager to a custom URL. Open up Arduino, then go to the Preferences (**File** > **Preferences**). Then, towards the bottom of the window, paste this URL into the "Additional Board Manager URLs" text box:

```
https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_spark
fun_index.json
```

You can add multiple URLs by clicking the window icon, and pasting in one URL per line.


sketch_jan07a | Arduino 1.8.10
File Edit Sketch Tools Help
sketch_jan07a
void setup() {
Preferences
Settings
Network
Browse
Additional Boards Manager URLs
OK
Cancel

Sketchbook location: C:\Users\user_name\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 11

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

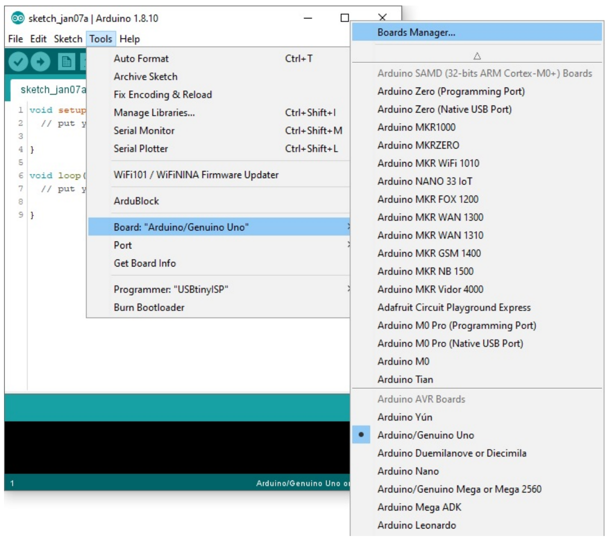
Show verbose output during: compilation upload

Enter additional URLs, one for each row

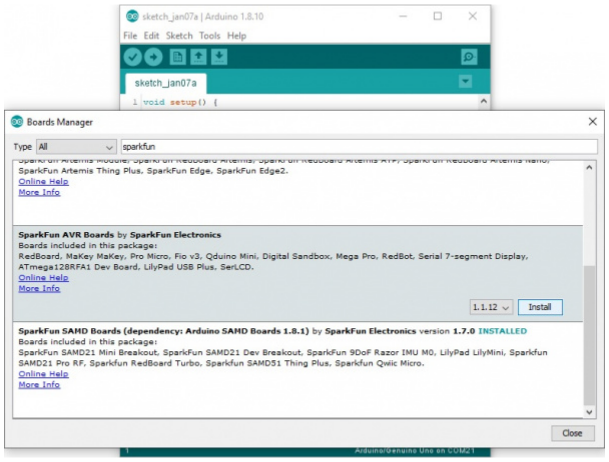
```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
https://dl.espressif.com/dl/package_esp32_index.json
https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/
```

Click for a list of unofficial boards support URLs

Click OK. Then open the Board Manager by clicking Tools, then hovering over the Board selection tab and clicking Board Manager.


sketch_jan07a | Arduino 1.8.10
File Edit Sketch Tools Help
sketch_jan07a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
Boards Manager...
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
Arduino Zero (Programming Port)
Arduino Zero (Native USB Port)
Arduino MKR1000
Arduino MKRZERO
Arduino MKR WiFi 1010
Arduino NANO 33 IoT
Arduino MKR FOX 1200
Arduino MKR WAN 1300
Arduino MKR WAN 1310
Arduino MKR GSM 1400
Arduino MKR NB 1500
Arduino MKR Vidor 4000
Adafruit Circuit Playground Express
Arduino M0 Pro (Programming Port)
Arduino M0 Pro (Native USB Port)
Arduino M0
Arduino Tian
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino/Genuino Uno

Search for '**sparkfun**' in the Board Manager. You should see the **SparkFun AVR Boards** package appear. Click install, wait a few moments, and all the ***.brd** files you'll need should be installed, indicated by the blue 'Installed' that is printed next to the package.

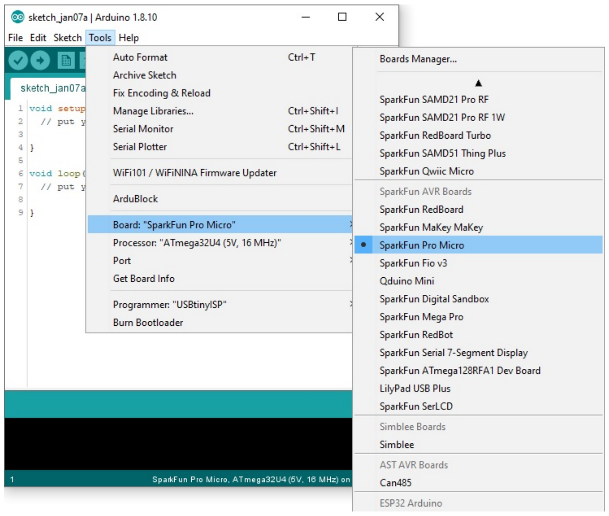

sketch_jan07a | Arduino 1.8.10
File Edit Sketch Tools Help
sketch_jan07a
Boards Manager
Type All
sparkfun
1.1.12
Install
Close
Arduino/Genuino Uno on COM21

SparkFun Artemis Thing Plus, SparkFun Edge, SparkFun Edge2.
Online Help
More Info

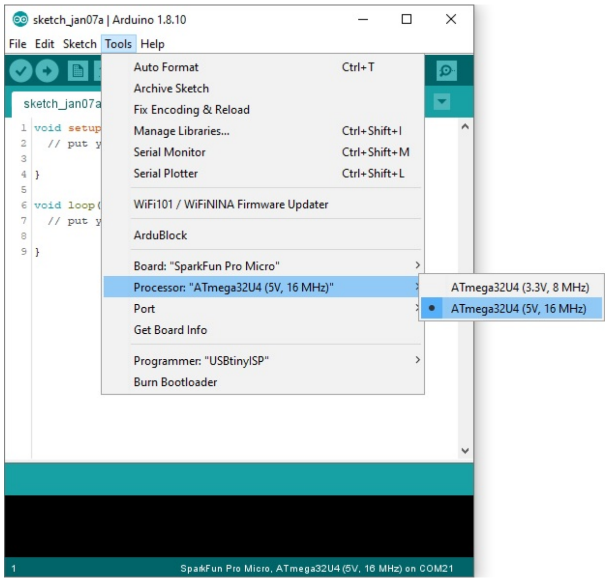
SparkFun AVR Boards by **SparkFun Electronics**
Boards included in this package:
RedBoard, MaKey MaKey, Pro Micro, Fio v3, Qduino Mini, Digital Sandbox, Mega Pro, RedBot, Serial 7-segment Display, ATmega128RFA1 Dev Board, LilyPad USB Plus, SerLCD.
Online Help
More Info

SparkFun SAMD Boards (dependency: Arduino SAMD Boards 1.8.1) by **SparkFun Electronics** version **1.7.0** INSTALLED
Boards included in this package:
SparkFun SAMD21 Mini Breakout, SparkFun SAMD21 Dev Breakout, SparkFun 9DoF Razor IMU M0, LilyPad LilyMini, Sparkfun SAMD21 Pro RF, Sparkfun RedBoard Turbo, Sparkfun SAMD51 Thing Plus, Sparkfun Qwiic Micro.
Online Help
More Info

You should now be able to upload code to a number of SparkFun Arduino-compatible products, including the **SparkFun Pro Micro**.



Notice there are two options for the Pro Micro - 8MHz and 16MHz. It's very important that you **select the Pro Micro option that matches your board's voltage and speed**. This should be listed under the **Tools** > **Processor**. For the Qwiic Pro Micro, you will need to ensure that you select the **ATmega32U4 (5V, 16MHz)**.


sketch_jan07a | Arduino 1.8.10
File Edit Sketch Tools Help
sketch_jan07a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ArduBlock
Board: "SparkFun Pro Micro"
Processor: "ATmega32U4 (5V, 16 MHz)"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
ATmega32U4 (3.3V, 8 MHz)
● ATmega32U4 (5V, 16 MHz)
SparkFun Pro Micro, ATmega32U4 (5V, 16 MHz) on COM21

You should also see your Pro Micro's **COM port** under the '**Tools** > **Serial Port**' menu. Select it, and head over to the Example 1 page where we'll upload our first piece of code.

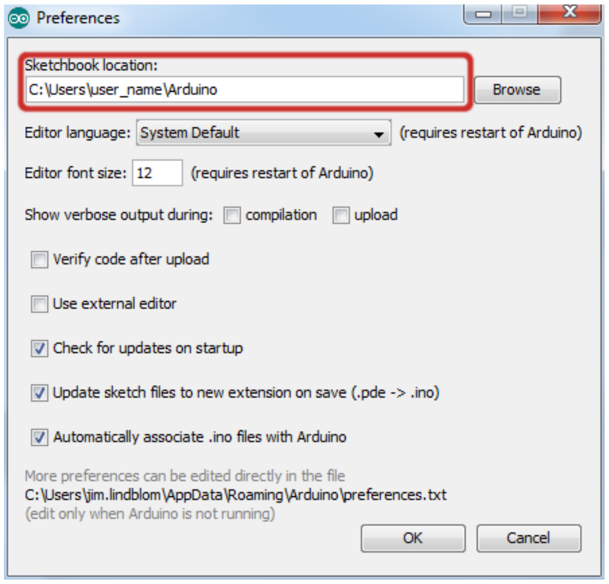
Manually Installing the *.brd Files

If you are using an older version of the Arduino IDE and do not have access to the Arduino Board Manager, then you'll need to install the *.brd files the old fashioned way.

To begin, **download this zip folder**, and unzip its contents into a **'hardware' directory** within your Arduino sketchbook.

> **Note:** These Arduino addon files only work with **Arduino 1.5** and up. If you're using an earlier version of Arduino, either update (and get some cool new features), or download the older version of the Addon.

Where's your Arduino sketchbook? Well, by default, it should an '**Arduino**' folder in your **home directory**, but to double check you can go to **File** > **Preferences** within Arduino and check the '*Sketchbook location*' text box. Just make sure you close all Arduino windows once you're done.


Preferences

Sketchbook location:
C:\Users\user_name\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12 (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

☐ Verify code after upload

☐ Use external editor

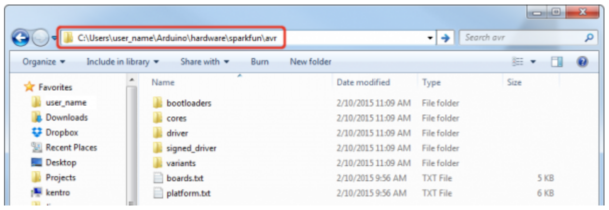
☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Automatically associate .ino files with Arduino

More preferences can be edited directly in the file
C:\Users\jim.lindblom\AppData\Roaming\Arduino\preferences.txt
(edit only when Arduino is not running)

Once you've unzipped that folder into the '**hardware**' folder within your Arduino sketchbook (you may actually have to create a hardware folder), your directory structure should look something like this:


C:\Users\user_name\Arduino\hardware\sparkfun\avr

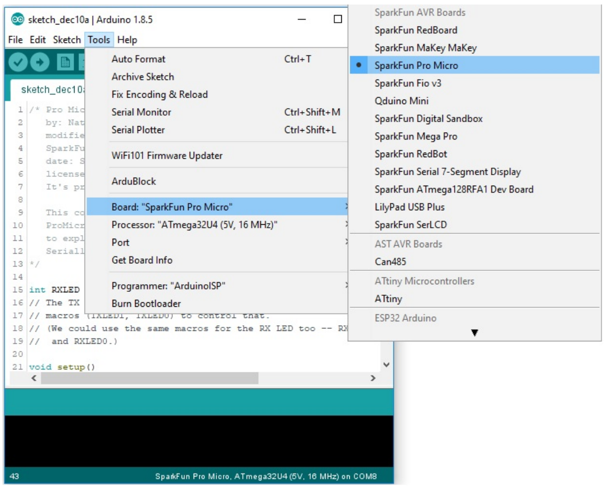
Name	Date modified	Type	Size
bootloaders	2/10/2015 11:09 AM	File folder	
cores	2/10/2015 11:09 AM	File folder	
driver	2/10/2015 11:09 AM	File folder	
signed_driver	2/10/2015 11:09 AM	File folder	
variants	2/10/2015 11:09 AM	File folder	
boards.txt	2/10/2015 9:56 AM	TXT File	5 KB
platform.txt	2/10/2015 9:56 AM	TXT File	6 KB

The structure of this directory is critical -- it should look something like "**.../Arduino/hardware/[manufacturer]/[architecture]**", in this case `[manufacturer]` is "**sparkfun**", and `[architecture]` is "**avr**."

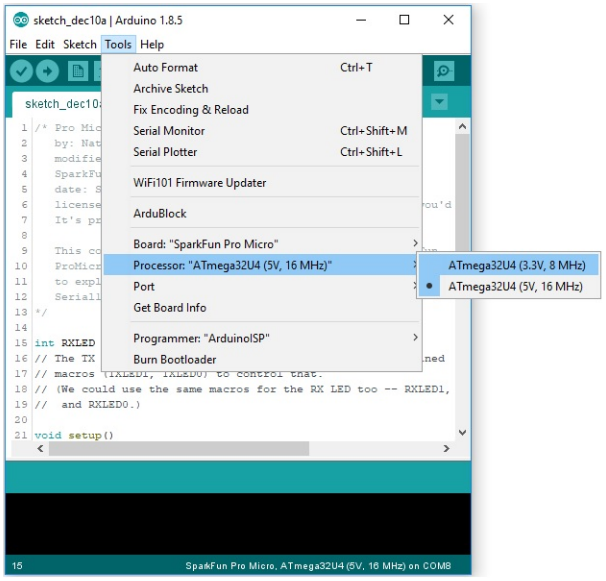
```
.../Arduino/hardware/sparkfun/avr
```

There's a lot going on in that addon, but one of the most important files is '**boards.txt**', which will add a few new entries to your '**Tools** > **Board**' menu.

To double-check that the board definitions have been added to Arduino, **open up Arduino**, and check under the '**Tools** > **Board**' menu. There should be some new entires for 'SparkFun Pro Micro', 'SparkFun FioV3', 'Qduino Mini', and other 32U4-based boards.



Notice there are two options for the Pro Micro - 8MHz and 16MHz. It's very important that you **select the Pro Micro option that matches your board's voltage and speed**. This should be listed under the **Tools** > **Processor**. For the Qwiic Pro Micro, you will need to ensure that you select the **ATmega32U4 (5V, 16MHz)**.


sketch_dec10a | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ArduBlock
Board: "SparkFun Pro Micro"
Processor: "ATmega32U4 (5V, 16 MHz)"
Port
Get Board Info
Programmer: "ArduinoISP"
Burn Bootloader
ATmega32U4 (3.3V, 8 MHz)
● ATmega32U4 (5V, 16 MHz)
SparkFun Pro Micro, ATmega32U4 (5V, 16 MHz) on COM8

You should also see your Pro Micro's **COM port** under the '**Tools** > **Serial Port**' menu. Select it, and head over to the Example 1 page where we'll upload our first piece of code.

Installing the Arduino Addon Mac and Linux

Following these directions is critical to getting your Pro Micro supported within your Arduino environment!

Using the Board Manager

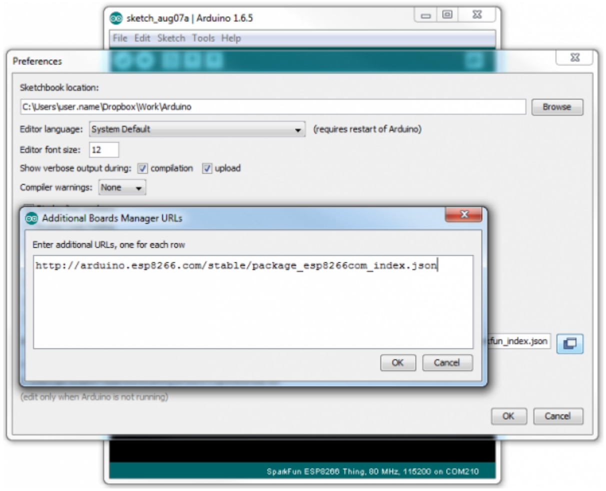
Since the release of Arduino v1.6.4, adding third party boards to the Arduino IDE is easily achieved through the board manager. If you're running an older version of Arduino (v1.6.3 or earlier), we recommend upgrading now. As always, you can download the latest version of Arduino from arduino.cc.

DOWNLOAD THE ARDUINO IDE

To begin, you'll need to point the Arduino IDE board manager to a custom URL. Open up Arduino, then go to the Preferences (**File** > **Preferences**). Then, towards the bottom of the window, paste this URL into the "Additional Board Manager URLs" text box:

```
https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_spark
fun_index.json
```

You can add multiple URLs by clicking the window icon, and pasting in one URL per line.


sketch_aug07a | Arduino 1.6.5

Preferences

Sketchbook location: C:\Users\user.name\Dropbox\Work\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Show verbose output during: [✓ compilation] [✓ upload]

Compiler warnings: None

Additional Boards Manager URLs

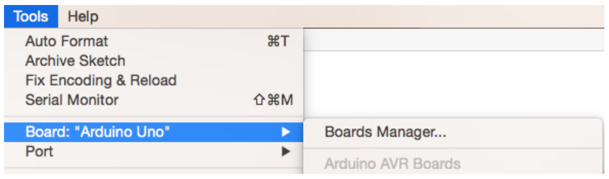
Enter additional URLs, one for each row

```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

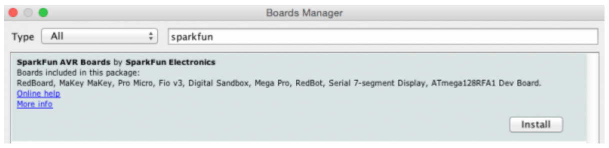
(edit only when Arduino is not running)

SparkFun ESP8266 Thing, 80 MHz, 115200 on COM210

Click OK. Then open the Board Manager by clicking Tools, then hovering over the Board selection tab and clicking Board Manager.



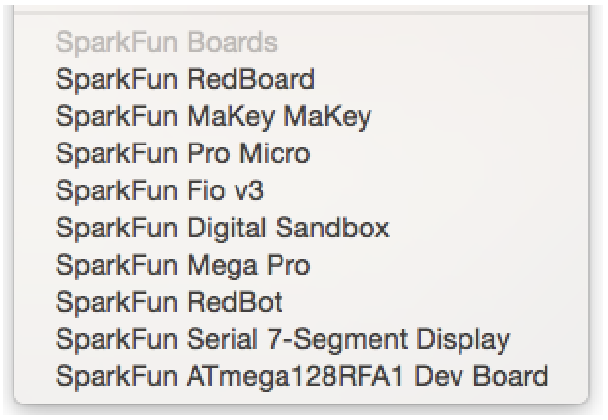
Search for '**sparkfun**' in the Board Manager. You should see the **SparkFun AVR Boards** package appear. Click install, wait a few moments, and all the *.brd files you'll need should be installed, indicated by the blue 'Installed' that is printed next to the package.


Boards Manager

Type: All — sparkfun

SparkFun AVR Boards by **SparkFun Electronics**
Boards included in this package:
RedBoard, MaKey MaKey, Pro Micro, Fio v3, Digital Sandbox, Mega Pro, RedBot, Serial 7-segment Display, ATmega128RFA1 Dev Board.
Online help
More info

You should now be able to upload code to a number of SparkFun Arduino-compatible products, including Pro Micro.



Manually Installing the *.brd Files

If you are using an older version of the Arduino IDE and do not have access to the Arduino Board Manager, then you'll need to install the *.brd files the old fashioned way.

In order to use the Pro Micro in your Arduino IDE, you need to add a few board definition files to it. That's what we'll do in this section. Begin by **downloading the Pro Micro addon files**.

Note: These Arduino addon files only work with **Arduino 1.5** and up. If you're using an earlier version of Arduino, either update (and get some cool new features), or download the older version of the Addon.

With that downloaded, follow these steps to enable the Pro Micro in your Arduino environment:

1. The addon files are supplied in a zip folder, so you'll need to **extract the files** within first.
2. Find your **Arduino sketchbook folder**. If you don't know where it is, you can locate your sketchbook by looking at the preferences dialog in your Arduino IDE.
3. If there isn't already one, create a folder in your sketchbook called '**hardware**'.
4. Copy the 'sparkfun' folder that was unzipped in the first step into the '**hardware**' folder.
    - Your directory structure should look something like "**.../Arduino/hardware/sparkfun/avr**."
5. Restart Arduino, and look under the **Tools** > **Board** menu. You should see a few new options, including SparkFun Pro Micro, SparkFun Fio V3, and Qduino Mini.

If the boards are visible, select the option that matches your board. If you have a Pro Micro, **make sure you select the correct operating speed and voltage** under **Tools** > **Processor**! At this point, select the COM port that the Pro Micro enumerated on. Then head over to the next page where we'll upload our first sketch!

Example 1: Blinkies and Hello World!

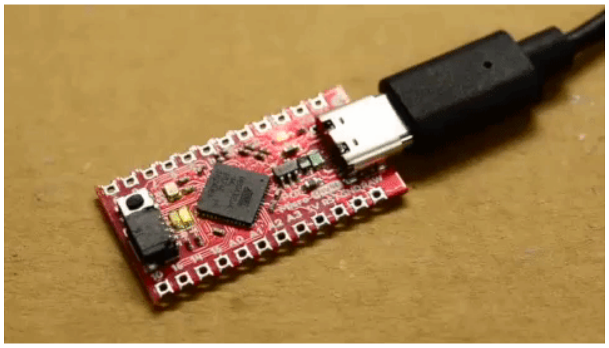
The Arduino-standard Blink sketch won't have any visible effect on the Pro Micro -- **there's no LED on pin 13**. In fact, the only LEDs on the board are the power indicator, and RX/TX blinkies. Unlike other Arduino boards, though, we can control the RX/TX LEDs in our sketch. So let's get blinking!

Upload the RX/TX Blinky, Hello World Sketch

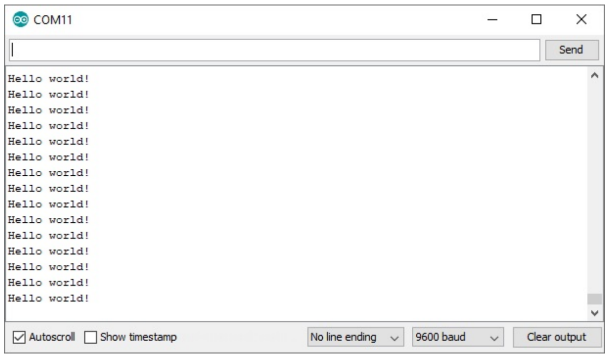
Copy and paste the code below into the Arduino IDE. In the **Tools**, make sure to select **Board** > **SparkFun Pro Micro** and **Processor** > **ATmega32U4 (5V, 16MHz)**. Also, don't forget to select the COM port that your board enumerated to. Finally, upload [1] it to your Qwiic Pro Micro USB-C.

```
/* Pro Micro Test Code
   by: Nathan Seidle
   modified by: Jim Lindblom
   SparkFun Electronics
   date: September 16, 2013
   license: Public Domain - please use this code however you'd like.
   It's provided as a learning tool.

   This code is provided to show how to control the SparkFun
   ProMicro's TX and RX LEDs within a sketch. It also serves
   to explain the difference between Serial.print() and
   Serial1.print().
*/

int RXLED = 17;  // The RX LED has a defined Arduino pin
// Note: The TX LED was not so lucky, we'll need to use pre-defined
// macros (TXLED1, TXLED0) to control that.
// (We could use the same macros for the RX LED too -- RXLED1,
//  and RXLED0.)

void setup()
{
  pinMode(RXLED, OUTPUT);  // Set RX LED as an output
  // TX LED is set as an output behind the scenes

  Serial.begin(9600); //This pipes to the serial monitor
  Serial.println("Initialize Serial Monitor");

  Serial1.begin(9600); //This is the UART, pipes to sensors attached to board
  Serial1.println("Initialize Serial Hardware UART Pins");
}

void loop()
{
  Serial.println("Hello world!");  // Print "Hello World" to the Serial Monitor
  Serial1.println("Hello! Can anybody hear me?");  // Print "Hello!" over hardware UART

  digitalWrite(RXLED, LOW);   // set the RX LED ON
  TXLED0; //TX LED is not tied to a normally controlled pin so a macro is needed, turn LED OFF
  delay(1000);              // wait for a second

  digitalWrite(RXLED, HIGH);    // set the RX LED OFF
  TXLED1; //TX LED macro to turn LED ON
  delay(1000);              // wait for a second
}
```

With the code uploaded you should see the RX and TX LEDs take turns blinking on and off every second.



You can also open up the Arduino IDE's serial monitor (set to **9600 bps**) and see every programmer's favorite two-word phrase.



```
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
```

Understanding the Sketch

RX LED

The RX LED is tied to Arduino's pin 17. You can control it just as you would any other digital pin. Set it as an `OUTPUT`, and `digitalWrite([pin], [level])` it `HIGH` to turn the LED off or `LOW` to turn the LED on. Here's part of the code highlighted.

```
int RXLED = 17;  // The RX LED has a defined Arduino pin
void setup(){
  pinMode(RXLED, OUTPUT);  // Set RX LED as an output
}
.
.
.
  digitalWrite(RXLED, LOW);   // set the RX LED ON
  digitalWrite(RXLED, HIGH);    // set the RX LED OFF
```

TX LED

The TX LED was not provided as an Arduino-defined pin, unfortunately, so you'll have to use a pair of macros to control it. `TXLED1` turns the LED on, and `TXLED0` turns the LED off. We could use the same macros for the RX LED too -- `RXLED1` and `RXLED0`. Here's part of the code highlighted.

```
TXLED0; //TX LED is not tied to a normally controlled pin so a macro is needed, turn LED OFF
TXLED1; //TX LED macro to turn LED ON
```

Serial Monitor (`Serial`) vs Hardware Serial UART (`Serial1`)

In that sketch, you'll also notice a pair of **Serial** initialization statements:

```
  Serial.begin(9600); //This pipes to the serial monitor
  Serial.println("Initialize Serial Monitor");

  Serial1.begin(9600); //This is the UART, pipes to sensors attached to board
  Serial1.println("Initialize Serial Hardware UART Pins");
```

That " `1` " makes a huge difference. Think of the Qwiic Pro Micro having two separate serial ports. The one *without* the " `1` " is for communication to and from the computer over USB; this is what is visible in the Serial Monitor. The `Serial1` port is a bonafide, **hardware UART**, where your Qwiic Pro Micro can talk to any serial-enabled piece of hardware.

If you open up the **Serial Monitor**, you should only see '`Hello world!`' printed. '`Hello! Can anybody hear me?`' is being sent out over the hardware UART, where, presumably, nothing is listening. This begs the age-old question: "If a Qwiic Pro Micro is saying 'Hello!' over the hardware serial port, and nothing is there to hear it, does the Qwiic Pro Micro really say anything at all?"

Why Does My Board Re-Enumerate Every Upload?

In order to communicate serially, the Qwiic Pro Micro emulates a **virtual serial port**. Actually, it emulates two different serial ports -- one for the bootloader, and one for the sketch. Since the bootloader and sketch run individually. Only one of these serial ports is visible at any one time.

When you click 'Upload' in the Arduino IDE, the Qwiic Pro Micro resets itself and starts its **bootloader** program. (The bootloader is a low-level program on the Pro Micro which enables self-programming via serial.) To our operating system, the bootloader looks like a completely different device, so it gets its own serial port number. While the Qwiic Pro Micro is being programmed, the bootloader serial port will be open. When the sketch upload is finished, the bootloader will exit, that serial port will be closed, and the regular Qwiic Pro Micro serial port will open up.

What this all boils down to is the fact that you have to **be patient** with Pro Micros. Every time you upload a new sketch, your OS will need to work its driver magic before you can open up the COM port. This can take a few seconds after the code has finished uploading.

---

[1] **Note for Windows users**: The first time you upload a sketch, it may fail and give you an error. On top of that, Windows will pop up that familiar '*Device driver software was not successfully installed*' notification. Don't let this worry you too much. If you get the error, wait about a minute, and try uploading again.

Hopefully the upload will succeed the second time, but if it continues to fail, check out the how to enter the bootloader section of the FAQ. Windows needs to install the same driver we've already installed for the Qwiic Pro Micro's bootloader, but it's unable to get everything set up before the bootloader exits.

Example 2: HID Mouse and Keyboard

When the Pro Micro's were released, the most revolutionary feature (as far as Arduinos go) is its true USB functionality. The Pro Micro can be programmed to emulate any USB device you could imagine. You can even program it to act just like a mouse, keyboard, or other HID-class USB device.

What is **HID**?

It's one of the many defined USB device classes. Every USB device is assigned a class, which defines what its general purpose is. There are loads of classes -- printers, hubs, speakers, and webcams to mention a few, but in this example we'll be emulating HID -- **Human Interface Device**. The ATmega32U4 takes care of the USB-hardware hurdle, but we've still got to clear the firmware one. Time for some example code! We broke up the HID example into two parts.

- Example 2a: USB Keyboard Made Simple
- Example 2b: USB Mouse Functionality

Example 2a: USB Keyboards Made Simple

To emulate a USB keyboard, we'll be making use of the `Keyboard` class. Here are some of the functions made available to use by this class:

- **`Keyboard.write(char)`** - This function will send a single character over USB. The character passed can be any standard, printable, ASCII-defined character: 0-9, a-z, A-Z, space, symbols, etc. Here's an example line of code:

  ```
  Keyboard.write('z') // This will send a single 'z' character to your computer.
  ```

- **`Keyboard.print(string)`** - If you need to perform a series a `Keyboard.write()` 's, consider using just a single `Keyboard.print()` . This works similar to `Serial.print()` – give it a string of characters and it'll send that stream of characters over USB. **`Keyboard.println(string)`** is also defined, if you want a newline/linefeed to close out your string. An example:

  ```
  Keyboard.print("Hello, world"); // This'll send your computer an 'H', followed by 'e', fol
  lowed by...
  ```

- **`Keyboard.press(byte)`** and **`Keyboard.release(byte)`** give you more precise control over key presses. They do exactly what you'd expect. One presses a button down, the other releases a button. Make sure you release any buttons you press, otherwise you'll encounter some wiggyness on your computer.

That's it. You don't need to include any libraries or anything, just invoke any of those functions. Here's an **example sketch** to try it out:

```
/* HID KeyBoard Example
   by: Jim Lindblom
   date: 1/12/2012
   license: MIT License - Feel free to use this code for any purpose.
   No restrictions. Just keep this license if you go on to use this
   code in your future endeavors! Reuse and share.

   This is very simplistic code that allows you to send a 'z' with
   a momentary pushbutton.
*/

#include <Keyboard.h>
int buttonPin = 9;  // Set a button to any pin

void setup()
{
  pinMode(buttonPin, INPUT);  // Set the button as an input
  digitalWrite(buttonPin, HIGH);  // Pull the button high

  Keyboard.begin(); //Init keyboard emulation
}

void loop()
{
  if (digitalRead(buttonPin) == 0)  // if the button goes low
  {
    Keyboard.write('z');  // send a 'z' to the computer via Keyboard HID
    delay(1000);  // delay so there aren't a kajillion z's
  }
}
```

In this sketch, **connecting pin 9 to ground** will make the Qwiic Pro Micro spit out a '`z`' character. If you have a simple, **momentary button** handy, tie one end to pin 9 and the other to ground. Otherwise, just use a wire to short 9 to GND.

After a `Keyboard.write()` or `Keyboard.print()` function has been performed by the Qwiic Pro Micro, your computer will have to decide what to do with it. What your computer does with that character, or string of characters, is entirely dependent on what program it's running at the time. If you have a text editor (Notepad, Word, TextEdit, etc.) open and active, it'll print it out there. Or you can try using the textbox below to test.

SparkFun Qwiic Pro Micro USB C Test Area

SparkFun Electronics
Start Something!
SparkFun Qwiic Pro Micro USB C Testing Area

Warning! Any text written in this textbox will be erased when you refresh this webpage!

Example 2b: USB Mouse Functionality

That covers about half of USB HID library. How about we add a mouse to the mix now? Implementing a USB HID mouse requires a few more functions, but it's still crazy simple. There are five functions provided by Arduino's HID class that can be used to implement a mouse:

- **`Mouse.move(x, y, wheel)`** tells the computer to move the mouse a certain number of pixels along either the x, y and/or wheel axis. Each variable can be any value between `-128` and `+127`, with negative numbers moving the cursor down/left, positive numbers move the right/up.

- **`Mouse.press(b)`** sends a down-click on a button or buttons. The button(s) will remain "pressed" until you call **`Mouse.release(b)`**. The `b` variable is a single byte, each bit of which represents a different button. You can set it equal to any of the following, or OR (|) them together to click multiple buttons at once:

    - `MOUSE_LEFT` - Left mouse button
    - `MOUSE_RIGHT` - Right mouse button
    - `MOUSE_MIDDLE` - Middle mouse button
    - `MOUSE_ALL` - All three mouse buttons

- **`Mouse.click(b)`** sends a down-click (press) followed immediately by an up-click (release) on button(s) `b`. For example, to click the left and right buttons simultaneously, try this:

    ```
    Mouse.click(MOUSE_LEFT | MOUSE_RIGHT); // Press and release the left and right mouse butto
    ns
    ```

Here's some **example code** to show off these functions:

```
/* HID Joystick Mouse Example
   by: Jim Lindblom
   date: 1/12/2012
   license: MIT License - Feel free to use this code for any purpose.
   No restrictions. Just keep this license if you go on to use this
   code in your future endeavors! Reuse and share.

   This is very simplistic code that allows you to turn the
   SparkFun Thumb Joystick (http://www.sparkfun.com/products/9032)
   into an HID Mouse. The select button on the joystick is set up
   as the mouse left click.
*/
#include <Mouse.h>
int horzPin = A0;  // Analog output of horizontal joystick pin
int vertPin = A1;  // Analog output of vertical joystick pin
int selPin = 9;  // select button pin of joystick

int vertZero, horzZero;  // Stores the initial value of each axis, usually around 512
int vertValue, horzValue;  // Stores current analog output of each axis
const int sensitivity = 200;  // Higher sensitivity value = slower mouse, should be <= about 500
int mouseClickFlag = 0;

//int invertMouse = 1;        //Invert joystick based on orientation
int invertMouse = -1;         //Noninverted joystick based on orientation

void setup()
{
  pinMode(horzPin, INPUT);  // Set both analog pins as inputs
  pinMode(vertPin, INPUT);
  pinMode(selPin, INPUT);  // set button select pin as input
  digitalWrite(selPin, HIGH);  // Pull button select pin high
  delay(1000);  // short delay to let outputs settle
  vertZero = analogRead(vertPin);  // get the initial values
  horzZero = analogRead(horzPin);  // Joystick should be in neutral position when reading these

  Mouse.begin(); //Init mouse emulation
}

void loop()
{
  vertValue = analogRead(vertPin) - vertZero;  // read vertical offset
  horzValue = analogRead(horzPin) - horzZero;  // read horizontal offset

  if (vertValue != 0)
    Mouse.move(0, (invertMouse * (vertValue / sensitivity)), 0); // move mouse on y axis
  if (horzValue != 0)
    Mouse.move((invertMouse * (horzValue / sensitivity)), 0, 0); // move mouse on x axis

  if ((digitalRead(selPin) == 0) && (!mouseClickFlag))  // if the joystick button is pressed
  {
    mouseClickFlag = 1;
    Mouse.press(MOUSE_LEFT);  // click the left button down
  }
  else if ((digitalRead(selPin)) && (mouseClickFlag)) // if the joystick button is not pressed
  {
    mouseClickFlag = 0;
    Mouse.release(MOUSE_LEFT);  // release the left button
  }
}
```

This sketch is set up so that an analog joystick connected to analog pins A0 and A1 can be used to move your mouse cursor.

The `loop()` of this code continuously monitors the horizontal and vertical analog values of the joystick and sends the `Mouse.move()` command based on what it reads. It'll move the mouse in steps, depending on what the sensitivity variable is set to. With sensitivity set to 2, the cursor will move in either 1 or 2 pixel steps. Depending on the orientation of the joystick, there is also an option to invert the mouse based on how the "V" is pointing by adjusting `invertMouse`.

The select switch on the joystick is used to control the mouse left click. Notice this code is using `Mouse.press()` and `Mouse.release()`, rather than just calling a single `Mouse.click()`. This requires a bit more coding, but it allows you to do things like drag-and-drop, double click, etc.

For more HID example code, check out the Arduino-supplied examples under the **File** > **Examples** > **09.USB** menu. Or check out Arduino's reference language under USB for more information.

ARDUINO LANGUAGE REFERENCE > USB

Example 3: Qwiic Enabled HID Mouse and Keyboard

Note: If you have not previously installed an Arduino library, please check out our installation guide.

As another option, you can try to use a Qwiic device to take advantage of the HID class. We broke up the Qwiic HID examples into three parts. Depending on your application and personal preference, you could essentially use a different Qwiic enabled device.

- Example 3a: HID Mouse with Qwiic Joystick
- Example 3b: HID Keyboard with Qwiic Keypad
- Example 3c: Qwiic HID Mouse and Keyboard Combined

Example 3a: HID Mouse with Qwiic Joystick

For the scope of this tutorial example, we will be using the Qwiic Joystick so that we do not need to solder any pins. Make sure to check out the Qwiic Joystick Hookup Guide before continuing on with the examples below.

Qwiic Joystick Hookup Guide

FEBRUARY 21, 2019

Looking for an easy way to implement a joystick to your next Arduino or Raspberry Pi project? This hookup guide will walk you through using the Qwiic Joystick with the Arduino IDE on a RedBoard Qwiic and in Python on a Raspberry Pi.

Copy and paste the code below into the Arduino IDE. Make sure to select the correct board and COM port that your respective board enumerated to.

```
/******************************************************************************
  Example_3a_Qwiic_Joystick_HID_Mouse.ino
  Written by: Ho Yun "Bobby" Chan
  Date: January 13, 2020
  Development Environment Specifics:
    Arduino IDE 1.8.9

  Description:
    Based on the Jim's Pro Micro "HID Mouse" and Wes' Qwiic Joystick "basic reading"
    examples, this example moves your computer's mouse based on the joystick's
    position. Pressing down on the joystick's will enable a mouse's left click.
    The left click will relese as soon as you stop pressing down on the joystick.

  Libraries:
    Mouse.h (included with Arduino IDE)
    Wire.h (included with Arduino IDE)
    SparkFun_Qwiic_Joystick_Arduino_Library.h (included in the src folder) http://librarymanage
r/All#SparkFun_joystick

  License:
    This code is released under the MIT License (http://opensource.org/licenses/MIT)

  ******************************************************************************/

#include <Mouse.h>
#include <Wire.h>

#include "SparkFun_Qwiic_Joystick_Arduino_Library.h" //Click here to get the library: http://lib
rarymanager/All#SparkFun_joystick
JOYSTICK joystick;                                   //Create instance of this object

int vertZero, horzZero;       // Stores the initial value of each axis, usually around 512
int vertValue, horzValue;     // Stores current analog output of each axis
const int sensitivity = 200;  // Higher sensitivity value = slower mouse, should be <= about 500
int mouseClickFlag = 0;

//int invertMouse = 1;        //Invert joystick based on orientation
int invertMouse = -1;         //Noninverted joystick based on orientation

//Debug mode, comment one of these lines out using a syntax
//for a single line comment ("//"):
#define DEBUG 0       //0 = HID only
//#define DEBUG 1     //1 = HID with serial output

void setup() {
#if DEBUG
  Serial.begin(9600);
  Serial.println("Example 3: HID Mouse w/ Qwiic Joystick");
#endif

  if (joystick.begin() == false)
  {
#if DEBUG
```

```
    Serial.println("Joystick does not appear to be connected. Please check wiring. Freezing..."
);
#endif
    while (1);
  }

  delay(1000);  // short delay to let outputs settle
  vertZero = joystick.getVertical();  // get the initial values
  horzZero = joystick.getHorizontal();  // Joystick should be in neutral position when reading t
hese

  Mouse.begin(); //Init mouse emulation
}

void loop() {
#if DEBUG
  Serial.print("X: ");
  Serial.print(joystick.getHorizontal());

  Serial.print(" Y: ");
  Serial.print(joystick.getVertical());

  Serial.print(" Button: ");
  Serial.println(joystick.getButton());
#endif

  vertValue = joystick.getVertical() - vertZero; // read vertical offset
  horzValue = joystick.getHorizontal() - horzZero;  // read horizontal offset

  if (vertValue != 0)
    Mouse.move(0, (invertMouse * (vertValue / sensitivity)), 0); // move mouse on y axis
  if (horzValue != 0)
    Mouse.move((invertMouse * (horzValue / sensitivity)), 0, 0); // move mouse on x axis

  if ((joystick.getButton() == 0) && (!mouseClickFlag))  // if the joystick button is pressed
  {
    mouseClickFlag = 1;
    Mouse.press(MOUSE_LEFT);  // click the left button down
  }
  else if ((joystick.getButton()) && (mouseClickFlag)) // if the joystick button is not pressed
  {
    mouseClickFlag = 0;
    Mouse.release(MOUSE_LEFT);  // release the left button
  }

  //delay(200); //remove "//" on this line if you need a small delay
}
```

Moving the Qwiic joystick will move your mouse on the screen. Pressing down on the button will cause a left mouse click. Once you stop pressing down on the joystick, it will release the left mouse click. Depending on the orientation for the vertical, you can invert the mouse's movement using the `invertMouse` variable at the beginning of the example code.

Example 3b: HID Keyboard with Qwiic Keypad

For the scope of this tutorial example, we will be using the Qwiic Keypad so that we do not need to solder any pins. Make sure to check out the Qwiic Keypad Hookup Guide before continuing on with the examples below.

Qwiic Keypad Hookup Guide

APRIL 25, 2019

If you are tired of taking up GPIO pins, wiring a bunch of pull up resistors, and using firmware that scans the keys taking up valuable processing time... check out the new Qwiic Keypad.

Copy and paste the code below into the Arduino IDE. Make sure to select the correct board and COM port that your respective board enumerated to.

```
/******************************************************************************
  Example_3b_Qwiic_Keypad_HID_Keyboard.ino
  Written by: Ho Yun "Bobby" Chan
  Date: February 6, 2020
  Development Environment Specifics:
    Arduino IDE 1.8.9

  Description:
    Based on the Jim's Pro Micro "HID Mouse" and Pete' Qwiic Keypad "read button"
    examples, this example outputs keyboard presses associated with the keypad.

  Libraries:
   Keyboard.h (included with Arduino IDE)
   Wire.h (included with Arduino IDE)
   SparkFun_Qwiic_Keypad_Arduino_Library.h (included in the src folder) http://librarymanager/Al
l#SparkFun_keypad

  License:
    This code is released under the MIT License (http://opensource.org/licenses/MIT)

******************************************************************************/

#include <Keyboard.h>
#include <Wire.h>
#include "SparkFun_Qwiic_Keypad_Arduino_Library.h" //Click here to get the library: http://libra
rymanager/All#SparkFun_keypad
KEYPAD keypad1;                                    //Create instance of this object

void setup() {
  Serial.begin(9600);
  Serial.println("Qwiic KeyPad Example");

  if (keypad1.begin() == false)   // Note, using begin() like this will use default I2C address,
0x4B.
    // You can pass begin() a different address like so: keypad1.begin(Wire, 0x4A).
  {
    Serial.println("Keypad does not appear to be connected. Please check wiring. Freezing...");
    while (1);
  }
  Serial.print("Initialized. Firmware Version: ");
  Serial.println(keypad1.getVersion());
  Serial.println("Press a button: * to do a space. # to go to next line.");

  Keyboard.begin(); //Init keyboard emulation
}

void loop() {
  keypad1.updateFIFO();  // necessary for keypad to pull button from stack to readable register
  char button = keypad1.getButton();

  if (button == -1)
  {
    Serial.println("No keypad detected");
    delay(1000);
  }
  else if (button != 0)
  {
    if (button == '0') {//note that this is a keypad '0' not the button press itself
      Keyboard.write('0');
    }
    else if (button == '1') {
      Keyboard.write('1');
    }
    else if (button == '2') {
      Keyboard.write('2');
    }
    else if (button == '3') {
      Keyboard.write('3');
    }
    else if (button == '4') {
      Keyboard.write('4');
    }
    else if (button == '5') {
      Keyboard.write('5');
    }
    else if (button == '6') {
      Keyboard.write('6');
    }
    else if (button == '7') {
      Keyboard.write('7');
    }
    else if (button == '8') {
      Keyboard.write('8');
    }
    else if (button == '9') {
      Keyboard.write('9');
    }
    else if (button == '#') {
      Keyboard.write('#');
    }
    else if (button == '*') {
      Keyboard.write('*');
    }
  }

  //Do something else. Don't call your Keypad a ton otherwise you'll tie up the I2C bus
  delay(25); //25 is good, more is better
}
```

If you have a text editor (Notepad, Word, TextEdit, etc.) open and active from Example 2, press a button on the Qwiic Keypad. Or you can try using the textbox below to test. The outputs should be associated with each of the Qwiic keypad button press.

SparkFun Qwiic Pro Micro USB C Test Area

SparkFun Electronics
Start Something!
SparkFun Qwiic Pro Micro USB C Testing Area

Warning! Any text written in this textbox will be erased when you refresh this webpage!

Example 3c: Qwiic HID Mouse and Keyboard Combined

This example combines 3a and 3b into one sketch. Copy and paste the code below into the Arduino IDE. Make sure to select the correct board and COM port that your respective board enumerated to.

```
/******************************************************************************
  Example_3c_Qwiic_HID_Mouse_Keyboard_Combined.ino
  Written by: Ho Yun "Bobby" Chan
  Date: February 6, 2020
  Development Environment Specifics:
    Arduino IDE 1.8.9

  Description:
    Based on the following examples:
      - Jim's Pro Micro "HID Mouse/Keyboard"
      - Wes' Qwiic Joystick "basic reading"
      - Pete's Qwiic Keypad "read button"

    This example controls the mouse with the Qwiic Joystick
    and keyboard presses associated with the Qwiic Keypad.
    The serial output is turned off so that Qwiic Pro Micro
    can respond faster.

  Libraries:
   Mouse.h (included with Arduino IDE)
   Keyboard.h (included with Arduino IDE)
   Wire.h (included with Arduino IDE)
   SparkFun_Qwiic_Joystick_Arduino_Library.h (included in the src folder) http://librarymanager/
All#SparkFun_joystick
   SparkFun_Qwiic_Keypad_Arduino_Library.h (included in the src folder) http://librarymanager/Al
l#SparkFun_keypad

  License:
    This code is released under the MIT License (http://opensource.org/licenses/MIT)

******************************************************************************/

#include <Mouse.h>
#include <Keyboard.h>
#include <Wire.h>

#include "SparkFun_Qwiic_Joystick_Arduino_Library.h" //Click here to get the library: http://lib
rarymanager/All#SparkFun_joystick
JOYSTICK joystick;                                   //Create instance of this object

#include "SparkFun_Qwiic_Keypad_Arduino_Library.h" //Click here to get the library: http://libra
rymanager/All#SparkFun_keypad
KEYPAD keypad1;                                    //Create instance of this object

int vertZero, horzZero;       // Stores the initial value of each axis, usually around 512
int vertValue, horzValue;     // Stores current analog output of each axis
const int sensitivity = 150;  // Higher sensitivity value = slower mouse, should be <= about 500
int mouseClickFlag = 0;

//int invertMouse = 1;        //Invert joystick based on orientation
int invertMouse = -1;         //Noninverted joystick based on orientation

//Debug mode, comment one of these lines out using a syntax
```

```
//for a single line comment ("//"):
#define DEBUG 0       //0 = HID only
//#define DEBUG 1     //1 = HID with serial output

void setup() {
#if DEBUG
  Serial.begin(9600);
  Serial.println("Qwiic KeyPad Example");
#endif

  if (joystick.begin() == false) {
#if DEBUG
    Serial.println("Joystick does not appear to be connected. Please check wiring. Freezing..."
);
#endif
    while (1);
  }
  if (keypad1.begin() == false) {  // Note, using begin() like this will use default I2C addres
s, 0x4B.
    // You can pass begin() a different address like so: keypad1.begin(Wire, 0x4A).
#if DEBUG
    Serial.println("Keypad does not appear to be connected. Please check wiring. Freezing...");
#endif
    while (1);
  }

  delay(1000);  // short delay to let outputs settle
  vertZero = joystick.getVertical();  // get the initial values
  horzZero = joystick.getHorizontal();  // Joystick should be in neutral position when reading t
hese

#if DEBUG
  Serial.print("Initialized. Firmware Version: ");
  Serial.println(keypad1.getVersion());
  Serial.println("Press a button: * to do a space. # to go to next line.");
#endif

  Mouse.begin(); //Init mouse emulation
  Keyboard.begin(); //Init keyboard emulation

}//end setup

void loop() {
  keypad1.updateFIFO();  // necessary for keypad to pull button from stack to readable register
  char button = keypad1.getButton();

#if DEBUG
  Serial.print("X: ");
  Serial.print(joystick.getHorizontal());

  Serial.print(" Y: ");
  Serial.print(joystick.getVertical());

  Serial.print(" Button: ");
  Serial.println(joystick.getButton());
#endif

  vertValue = joystick.getVertical() - vertZero; // read vertical offset
  horzValue = joystick.getHorizontal() - horzZero;  // read horizontal offset

  if (vertValue != 0)
    Mouse.move(0, (invertMouse * (vertValue / sensitivity)), 0); // move mouse on y axis
  if (horzValue != 0)
    Mouse.move((invertMouse * (horzValue / sensitivity)), 0, 0); // move mouse on x axis

  if ((joystick.getButton() == 0) && (!mouseClickFlag))  // if the joystick button is pressed
  {
    mouseClickFlag = 1;
    Mouse.press(MOUSE_LEFT);  // click the left button down
  }
  else if ((joystick.getButton()) && (mouseClickFlag)) // if the joystick button is not pressed
  {
    mouseClickFlag = 0;
    Mouse.release(MOUSE_LEFT);  // release the left button
  }

  if (button == -1) {
#if DEBUG
    Serial.println("No keypad detected");
#endif
    delay(1000);
  }

  else if (button != 0) {
#if DEBUG
    Serial.print("Qwiic Keypad Button:  ");
    Serial.println(button);
#endif
    if (button == '0') {//note that this is a keypad '0' not the button press itself
      Keyboard.write('0');
    }
    else if (button == '1') {
      Keyboard.write('1');
    }
    else if (button == '2') {
      Keyboard.write('2');
    }
    else if (button == '3') {
      Keyboard.write('3');
    }
    else if (button == '4') {
      Keyboard.write('4');
    }
    else if (button == '5') {
      Keyboard.write('5');
    }
    else if (button == '6') {
      Keyboard.write('6');
    }
    else if (button == '7') {
      Keyboard.write('7');
    }
    else if (button == '8') {
      Keyboard.write('8');
    }
    else if (button == '9') {
      Keyboard.write('9');
    }
    else if (button == '#') {
      Keyboard.write('#');
    }
    else if (button == '*') {
      Keyboard.write('*');
    }
  }

  //Do something else. Don't call your Keypad a ton otherwise you'll tie up the I2C bus
  //Uncomment this if necessary but since we check the Qwiic Joystick it does not
  // appear to be an issue
  //delay(25); //25 is good, more is better

}//end loop
```

The functionality should be the same when moving the joystick or pressing on the keypad's key. The serial output is turned off so that Qwiic Pro Micro can respond faster.

Troubleshooting and FAQ

On this page you'll find troubleshooting tips and FAQs. Here's a directory of the subjects covered:

Serial Port Not Showing Up in 'Tools > Board' Menu

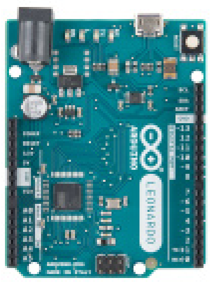
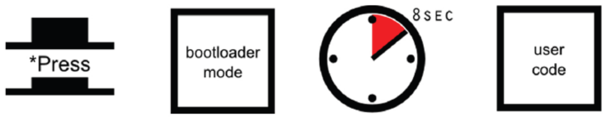
The Qwiic Pro Micro can be a finicky little thing. There are a few series of events that can lead to its serial port being removed from the Arduino IDE's Serial Port selection menu. If you can't see your Pro Micro's serial port, give these steps a try:

1. Close *all* Arduino windows. (Don't forget to save!)
2. Unplug Qwiic Pro Micro from your computer.

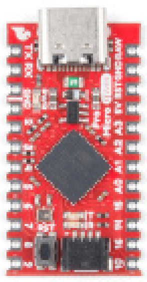
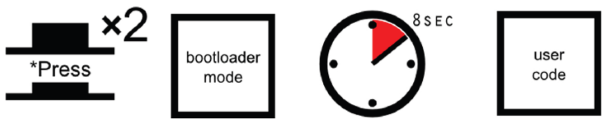
3. Wait a few seconds for the device to be detached.
4. Plug Qwiic Pro Micro back in.
5. Open Arduino back up, check the Serial Ports menu again.

Reset to Bootloader

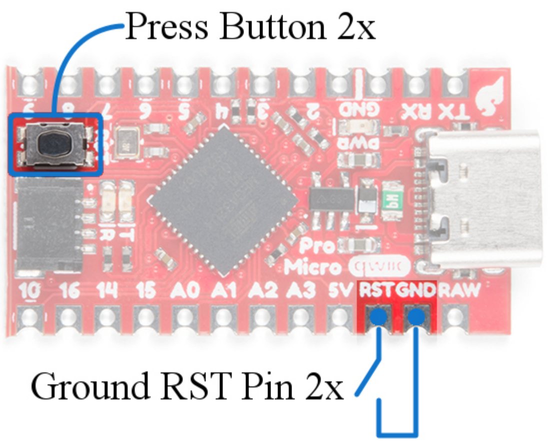
We ship the Qwiic Pro Micro with a modified version of the Arduino Leonardo bootloader, with one major enhancement. When a Leonardo (or any device using the "stock" bootloader) is externally reset, it goes back into the bootloader...and waits there eight seconds before it starts running the sketch. For some embedded projects, waiting eight seconds before a program runs isn't acceptable, so we modified the bootloader run time.

	
Arduino Leonardo	*Leonardo bootloader on reset functionality.*

When a Qwiic Pro Micro is externally reset (by pulling the RST pin low), it'll only briefly (<750ms) start the bootloader before continuing on to the sketch. If you need the bootloader to run longer, **resetting twice quickly** will get the Pro Micro to enter bootloader mode for eight seconds.

	
Qwiic Pro Micro	*Qwiic Pro Micro reset functionality. Press reset twice, quickly to enter bootloader mode.*

Resetting the Qwiic Pro Micro is now easier to reset with the built in reset button. Just press the reset button located next to the Qwiic connector two times. You can also connected the RST pin to ground to initiate a reset. This can be done with a small piece of wire, or an externally connected button.

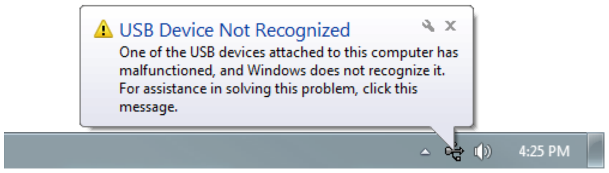


Why would you need to enter bootloader mode in the first place. Glad you asked...

How to Revive a "Bricked" Pro Micro

Incorporating all of the USB tasks on a single chip is an awesome feature that makes the Pro Micro and boards like it truly unique. But it also places more stress on a single chip, and if anything goes wrong with that chip, the board becomes nearly unusable. It's not uncommon for Qwiic Pro Micro's to become "bricked" and unprogrammable. But, in most cases, the bricking is reversible!

The most common source of Qwiic Pro Micro "bricking" is uploading code to it with an **incorrectly set board** (e.g. programming a 16MHz/5V Pro Micro with the board set to 8MHz/3.3V). Also, make sure your sketch doesn't mess with the ATmega32U4's PLLCSR register, or any other register that sets up USB functionality on the ATmega32U4. The Pro Micro will actually take code compiled for the wrong operating speed, but when it tries to re-enumerate, you'll be greeted with a notification like this:


USB Device Not Recognized
One of the USB devices attached to this computer has malfunctioned, and Windows does not recognize it. For assistance in solving this problem, click this message.

4:25 PM

To revive the Pro Micro, you'll need to find a way to upload a sketch to it with the board option correctly set. We can do this with a little help from the bootloader.

First, you'll need to **set the serial port to the bootloader**. But that port is only visible when the board is in bootloader mode, so hit the reset button or pull the reset line low twice quickly to invoke the bootloader reset feature discussed above. **While the Pro Micro is in the bootloader** change the '**Tools** > **Serial Port**' menu to the bootloader COM port. Quick! You've only got eight seconds. On Windows, the bootloader's COM port number is usually one number higher than the Qwiic Pro Micro's regular port number.

With the serial port set, we're just about ready to re-upload our sketch. But first, **double check that the board is correctly set**. Then **reset to bootloader again**, and quickly upload your sketch. Again, you'll have to be quick...you've only got eight seconds. It may help to press the Upload keybind -- `CTRL + U` / `CMD + U` -- immediately after resetting.

It can take a few tries to get the timing right. Since the code has to compile first, it may help to **hit upload first** and then reset.

Code Runs After Upload But Fails to Start After Power Cycle

We found that an ATmega32U4 (like the Pro Micro 3.3V/8MHz) can brown out when outputting power to a boost converter. While code can run after uploading, a power cycle from the initial current draw to a boost converter is enough to cause the Pro Micro brown out. Thus causing the sketch to not run. This requires the user to toggle the reset button after a power cycle.

Frequently Asked Questions

If you're having technical difficulties with your Qwiic Pro Micro, see if any of the answers to these FAQs help. If not, please get in touch with our tech support team.

What are VID and PIDs?

VID is short for 'Vender Identification' and PID is short for 'Part Identification'. In other words, this pair of IDs defines the device. This is how your computer knows what you've plugged in, what drivers to use with it, what COM port is assigned to it, etc. All native USB devices have a VID/PID.

All SparkFun ATmega32U4 boards share the same VID -- 0x1B4F, and they all have unique PIDs. 5V Pro Micros lay claim to PIDs 0x9205 and 0x9206 (one for the bootloader, one for the sketch). 3.3V Pro Micros will show up as 0x9203 and 0x9204 for bootloader and sketch, respectively. And the Fio v3 has 0xF100 and 0xF101.

How Can I Change the VID and PID on an ATMega32U4 Board?

Every time you upload code the VID and PID are uploaded to the device. These values are located in the '**boards.txt**' file and will therefore be determined by the board you have selected. Keep in mind that if you select the wrong board you will get the wrong VID/PID uploaded which means the computer can’t recognize, and program the board. The VID/PID for the bootloader is part of the bootloader file. To change this you will need to recompile the bootloader with the new VID/PID, and upload it.

Why Does my ATMega32U4 Board Show up Twice in Device Manager?

Both the bootloader and the sketch have their own VID/PIDs. When you plug in a board the bootloader starts running for a few seconds, and you will see the board show up in Device Manager based on those VID/PIDs. After a few seconds, the sketch will start running, and you will see Device Manager disconnect from the bootloader and connect to the sketch.

How Does the IDE Know Which COM Port to Use?

When the IDE resets the board, the COM port is disconnected from the computer. The IDE then looks for a new COM port. This is the port it uses. This is one of those weird things Arduino did to get things to work on this chip.

How Do I Reinstall the Bootloader?

Check out or reinstalling the bootloader tutorial, which should work for both ATMega32U4 and ATMega328 boards. If you have the tools to do so, reinstalling the bootloader is often easier then trying to stay in the bootloader. Since reinstalling the bootloader puts the board back in factory settings this will reset the VID/PID numbers allowing your board to work again.

Resources and Going Further

Thanks for checking out our Qwiic Pro Micro Hookup Guide! If you're looking for more resources related to these boards, here are some links:

- Schematic (PDF)
- Eagle Files (ZIP)
- Board Dimensions

- Graphical Datasheet (PDF)
- ATmega32U4 Firmware Notes (PDF)
- ATmega32U4 Datasheet (PDF)
- GitHub Repo
  - Arduino Board Add-on
  - Hardware Repo
- SFE Product Showcase

Thanks for reading along with our Qwiic Pro Micro hookup guide! Hopefully now you're fully prepared to begin using the Pro Micro in a project of your own. Here are some tutorials that might be worth checking out as you continue down the rabbit hole:

Connecting Arduino to Processing
Send serial data from Arduino to Processing and back - even at the same time!

Cherry MX Switch Breakout Hookup Guide
How to assemble and use the Cherry MX Switch Breakout, allowing you to turn a matrix of mechanical switches into a full-size keyboard!

LilyPad ProtoSnap Plus Hookup Guide
The LilyPad ProtoSnap Plus is a sewable electronics prototyping board that you can use to learn circuits and programming with Arduino, then break apart to make an interactive fabric or wearable project.

Tech Prank: Hardware Mouse Jiggler
Create an innocuous-looking USB stick with an Arduino Pro Micro and a 3D printed case that moves your mouse pointer randomly every few seconds. Sure to anger your coworkers and friends!

Feeling ambitious? Try making your own custom keyboard! Check out the custom keyboard builds from Martin Knobel using Pro Micros and Cherry MX switches:

- Nospace like Home

Or check out these blog posts for inspiration.

Turning Your Pro Micro Into a Keyboard
FEBRUARY 13, 2012

Enginursday: Pressing Our Buttons
JUNE 15, 2017

A Qwiic Upgrade for a DIY Keyboard
FEBRUARY 11, 2020