search documentation

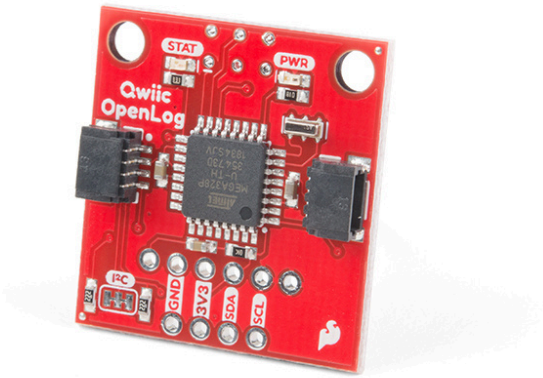
Qwiic OpenLog Hookup Guide

CONTRIBUTORS: *NATE*, *TONI_K*, *ELL C*, *ENGLANDSAURUS*

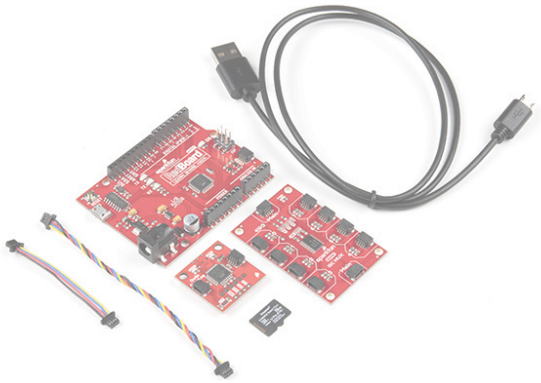
Introduction

Heads up! This tutorial is for the Qwiic Open Log for I^{2}C [DEV-15164]. If you are using the OpenLog for serial UART [DEV-13712], please refer to the OpenLog Hookup Guide.

The SparkFun Qwiic OpenLog is the smarter and better looking cousin to the extremely popular OpenLog. We've ported the original serial based interface to I^{2}C. Now you can daisy chain multiple I^{2}C devices and log them all without taking up your serial port. Pop in a microSD card, write some simple code, and your data will be recorded to a text file on the microSD card.



SparkFun Qwiic OpenLog
DEV-15164
$19.95



SparkFun Qwiic OpenLog Kit
KIT-18350
Retired

Materials Required

In order to fully work through this tutorial, you will need the following parts. 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.

Qwiic OpenLog Wishlist

microSD Card - 16GB (Class 10)
COM-15051
This is a class 10 16GB microSD memory card, perfect for housing operating systems for single board computers and a …

	microSD USB Reader COM-13004 This is an awesome little microSD USB reader. Just slide your microSD card into the inside of the USB connector, then st…
	Qwiic Cable - 50mm PRT-14426
	SparkFun RedBoard Qwiic DEV-15123
	SparkFun Qwiic OpenLog DEV-15164

Recommended Reading

If you are not familiar or comfortable with the following concepts, we recommend reading through these before continuing on with the Qwiic OpenLog Hookup Guide.

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

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

Serial Terminal Basics
This tutorial will show you how to communicate with your serial devices using a variety of terminal emulator applications.

OpenLog Hookup Guide
An introduction to working with the OpenLog data logger.

Hardware Overview

Power

The Qwiic OpenLog runs at the following settings:

OpenLog Power Ratings

VCC Input	**3.3V**
SDA	3.3V
SCL	3.3V
Idle Current Draw	~2mA-6mA
Active Writing Current Draw	~20-23mA (dep on microSD card)

The Qwiic OpenLog's current draw is about **20mA to 23mA** when writing to a microSD. Depending on the size of the microSD card and its manufacturer, the active current draw can vary when the OpenLog is writing to the memory card.

Register Map

The Qwiic Open Log implements a register map type set up if you'd like to create your own library.

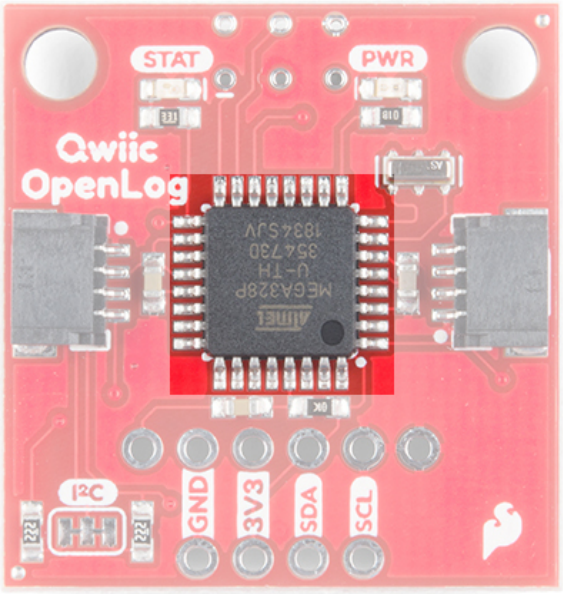
Qwiic OpenLog Register Map

Byte Number	HEX	Register Name	ead/Wri	Power On Reset	Description
0	0x00	id	R	0x78	Unique Identifier
1	0x01	status	R/W	0x00	Status bits of the Device. Bit 0: SD/Init Good Bit 1: Last Command Succeeded Bit 2: Last Command Known Bit 3: File Currently Open Bit 4: In Root Directory
2	0x02	firmware_MSB			
3	0x03	firmware_LSB	R	0x0200	Firmware Version
4	0x04	interruptEnable	R/W	0x00	There are no interrupts on the Qwiic OpenLog
5	0x05	Initialize	W	0x00	Write to this register to initialize the Log.
6	0x06	Create File	W	0x00	Write a characters into this register to create a file. This does not open the file.
7	0x07	Make Directory	W	0x00	Write a string of characters into this register to create a directory
8	0x08	Change Directory	W	0x00	Write a string of characters into this register to change directories. Writing ".." changes to root
9	0x09	Read File	R/W	0x00	Write a string of characters to this register to load the contents of this file into the response buffer, the device will respond with the file contents.
10	0x0A	Start Position	W	0x00	Write a string of characters to this register to change the start position of each new read.
11	0x0B	Append File	W	0x00	Write a string of characters to this register to create a new file and open it for writing. Will open the file if it already exists.
12	0x0C	Write File	W	0x00	Send a character to this register to write it to the current file
13	0x0D	File Size	R/W	0x00	Write a string of characters to this register to load the contents of this file into the response buffer, the device will respond with the file contents.
14	0x0E	List	R	0x00	Writing a string of characters to this register will list the contents of this folder shared with the string into the response buffer, simply writing nothing or "*0" to the register will list all file contents.
15	0x0F	Remove	R/W	0x00	Write a string of characters to this register to delete the file of the same name. Will respond with the number of files removed.
16	0x10	Remove Recursively		0x00	Write a string of characters to this register to recursively remove the file. Will respond with the number of files removed
17	0x11	Sync File	W	0x00	Write to this register to send characters to the current file. Only works in firmware version 3.0
30	0x1E	i2cAddress	R/W	0x2A	Value between 0x08 and 0x77 (inclusive) that is the address of this device. Overridden if ADR jumper is closed (address becomes 0x29). Default address is 0x2A.

Having a hard time seeing? Click the image for a closer look.

Microcontroller

Like its predecessor, the Qwiic OpenLog runs off of an onboard ATmega328, running at 16MHz thanks to the onboard crystal. The ATmega328 has the Optiboot bootloader loaded on it, which allows the OpenLog to be compatible with the "**SparkFun Redboard**" board setting in the Arduino IDE.

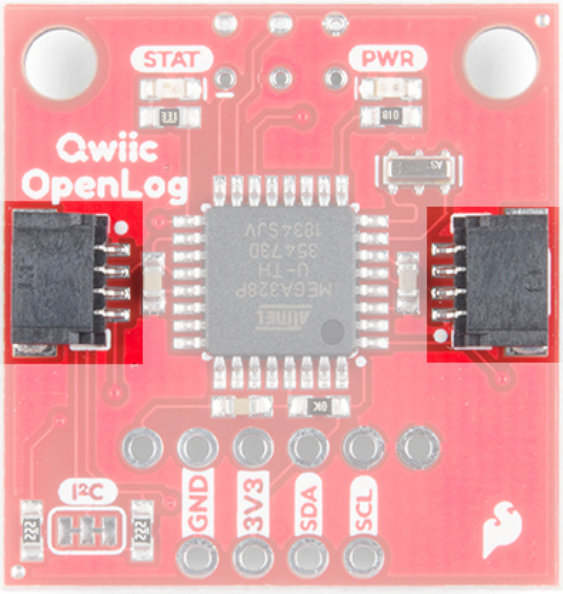


The brain of the Qwiic OpenLog.

Interface

QWIIC CONNECTORS

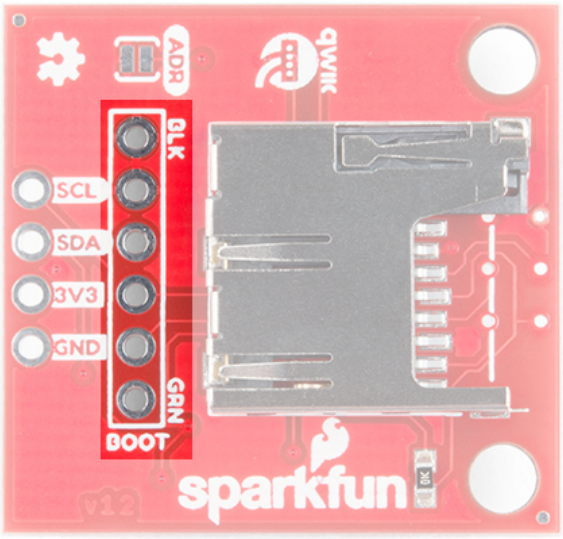
The primary interface with the Qwiic OpenLog are the Qwiic connectors on either side of the board. If you aren't familiar with the Qwiic system, you are in for a treat! The Qwiic system utilizes I^2C protocol to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips with a mere two wires. Check out all the benefits of the Qwiic System here.



Qwiic Connectors

FTDI HEADER PINS

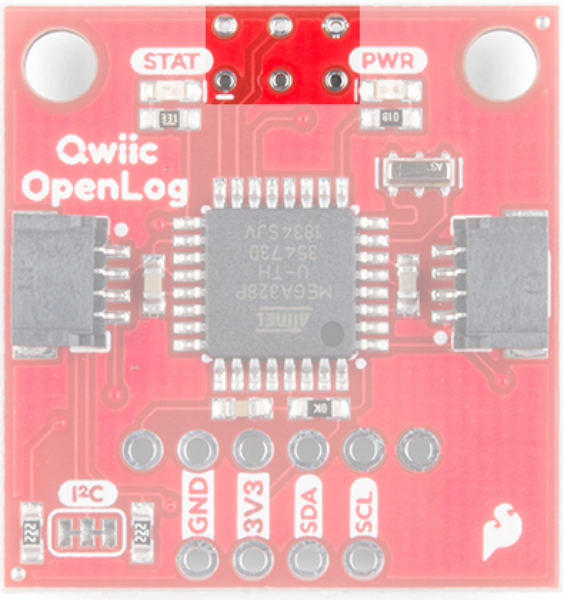
While the Qwiic OpenLog retains the broken out FTDI header pins, they pins are used specifically for reprogramming the firmware. All logging communication happens through the Qwiic lines and associated broken out pins.



FTDI Header Pins

SPI

There are also six SPI test points broken out on the opposite side of the board. You can also use these to reprogram the bootloader on the ATmega328.



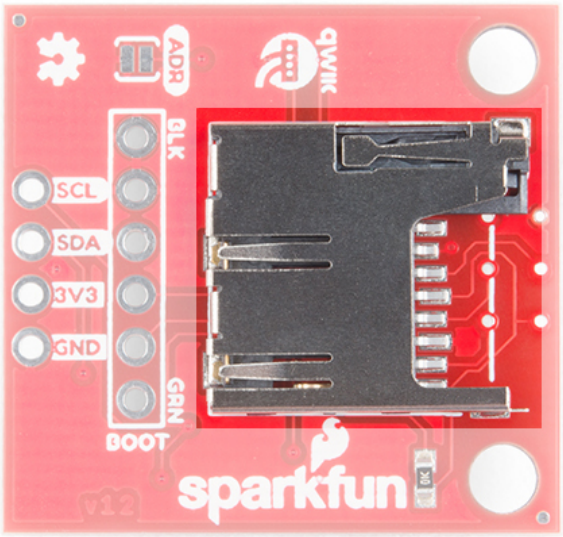
SPI Test Points

MICROSD CARD

The final interface for communicating with the Qwiic OpenLog is the microSD card itself. To communicate, the microSD card requires SPI pins. Not only is this where the data is stored by the OpenLog, but you can also update the OpenLog's configuration via the `config.txt` file on the microSD card.

All data logged by the OpenLog is stored on the microSD card. The OpenLog works with microSD cards that involve the following features:

- 64MB to 32GB
- FAT16 or FAT32

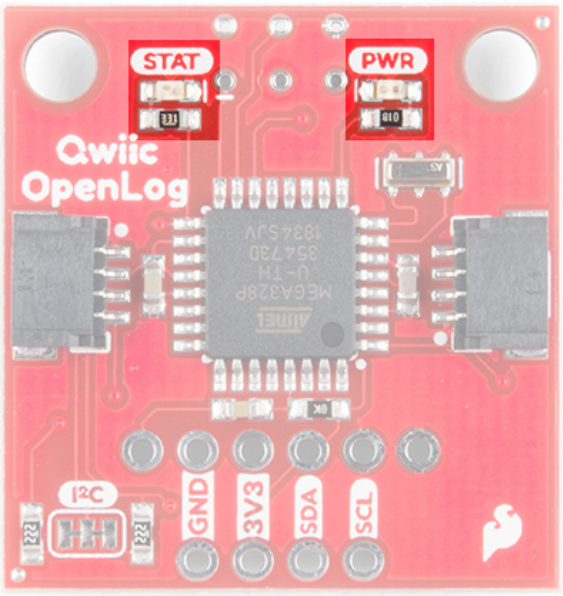


microSD Slot on the bottom of the Qwiic OpenLog

Status LED

There are two LEDs on the OpenLog to help you with troubleshooting.

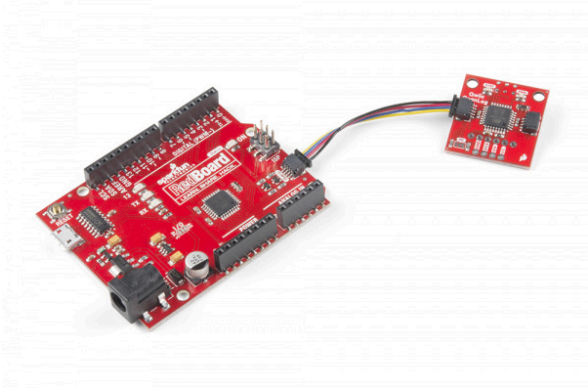
- **STAT** - This green LED is connected to Arduino **D13** (Serial Clock Line/ ATmega328 PB5). This LED only blinks when the SPI interface is active. You will see it flash when the OpenLog records data to the microSD card.
- **PWR** - This red indicator LED is attached to Arduino **D5** (ATmega328 PD5) and lights when the board is active and functioning.



Status and Power LEDs on the Qwiic OpenLog

Hardware Hookup

If you've purchased the SparkFun RedBoard Qwiic, hardware hookup is as simple as plugging in your Qwiic cable!



If, however, you are using an older SparkFun RedBoard, you'll need an assembled Qwiic Shield, so if you haven't done that yet, now would be the time to head on over to that tutorial.

Qwiic Shield for Arduino & Photon Hookup Guide
OCTOBER 19, 2017
Get started with our Qwiic ecosystem with the Qwiic shield for Arduino or Photon.

Arduino Sketches

Note: This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

Note: The Qwiic OpenLog utilizes clock stretching, which some I^2C master devices like ESP32 can't handle. To avoid this, you may have to only send/read bytes one at a time to receive proper data.

There are eleven examples in the Qwiic OpenLog Arduino Library that can be used when connected to a Qwiic OpenLog. These Arduino functions replace the serial command communication that occurred on the previous version of the OpenLog. These example files are named such that their functions should be self-explanatory, but

in case you want more information, see the descriptions below. The easiest way to install the library is by searching **SparkFun Qwiic OpenLog** is within the Arduino library manager. To manually install, head on over to the GitHub Page or feel free to download the library here!

SparkFun Qwiic OpenLog Arduino Library (ZIP)

- **Example1_WritingLog** -- This example shows how to record various text and variables to Qwiic OpenLog.
- **Example2_AppendFile** -- Arduino sketch showing how to append text to the end of *File*. If *File* does not exist when this function is called, the file will be created.
- **Example3_CreateFile** -- This example shows how to create a new file named *File* in the current directory. Standard 8.3 filenames are supported. For example, "87654321.123" is acceptable, while "987654321.123" is not.
- **Example4_ReadFileSize** -- This example shows how to record some strings to a default log, check the size of a given file name, and if the given file doesn't exist, say so.
- **Example5_ReadFile** -- This example shows how to record some strings to a default log, check the size of a given file name, if that given file doesn't exist, create it with random characters, and read back the contents of the given file (containing random characters)
- **Example6_MakeDirectory** -- This example shows how to create a directory, move into that directory called MONDAY, create a sub directory within MONDAY called LOGS, and create and write to a file inside MONDAY.
- **Example7_ReadDirectoryContents** -- This example shows how to read the files in a given directory. You can use wildcards if desired. This is handy for listing a certain type of file such as *.LOG or LOG01*.TXT.
- **Example8_RemoveDirectory** -- This example shows how to create a directory, create some files there, delete a specific file, delete *.TXT, and remove the directory we created.
- **Example9_ReadVersion** -- This example shows how to read the firmware version of Qwiic OpenLog.
- **Example10_CheckStatus** -- This example shows how to read the status byte of the OpenLog.
- **Example11_ChangeAddress** -- This example shows how to change the I^{2}C address of the Qwiic OpenLog from the default address **0x2A** to *0x1E* (or **42** and *30* in decimal, respectively. It's easy to change the I^{2}C address. If you forget what the address is you can use the I2CScanner Example to re-discover it. You can also close the ADR jumper on the board. This will force the I^{2}C address to *0x29* regardless of any other setting or command.
    - Valid I^{2}C addresses are *0x08* to *0x77* (inclusive)

Note: Example 11 is useful if there is another I^{2}C device that uses the same address. For example, the Qwiic OpenScale uses the same address (**0x2A**). You would need to configure the I^{2}C address of either the Qwiic OpenLog or OpenScale.

For the scope of this tutorial, let's say you modified the address of the OpenLog using **Example11_ChangeAddress.ino** to `0x1E` . To continue using the examples in the Qwiic OpenLog library, you will need to make sure to include the address as a parameter when making a connection to the OpenLog. The Qwiic OpenLog examples will default value of 0x2A if you do not provide an

address. To include the new address you will need to change this function in the `setup()` function from:

```
myLog.begin(); //Open connection to OpenLog (no pun intended)
```

To:

```
myLog.begin(0x1E); //Open connection to OpenLog (no pun intended)
```

Firmware

The Qwiic OpenLog has two primary pieces of software on board: the bootloader and the firmware.

Arduino Bootloader

You can treat the Qwiic OpenLog just like a **SparkFun RedBoard** when uploading example code or new firmware to the board. When uploading code, you will simply select the board in the Arduino IDE's menu under **Tools** > **Board** > **Arduino/Genuino Uno**.

Compiling and Loading Default Firmware onto the Qwiic OpenLog

Note: If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide to manually install the libraries.

If for any reason you need to update or reinstall the default firmware on your Qwiic OpenLog, the following process will get your board up and running. First, download the Qwiic OpenLog default firmware. You can go to the Qwiic OpenLog GitHub Page or download via the button here:

Download Qwiic OpenLog Default Firmware (ZIP)

Once you have the firmware downloaded, you will need to also install the following libraries into Arduino. If you are unsure how to manually install the libraries in the IDE, please refer to the link above.

Download SerialPort (ZIP) **Download SdFat v1.1.4 (ZIP)**

If you haven't yet, connect your Qwiic OpenLog to the computer via a 3.3V FTDI board using the silkscreen labels as reference. Please double check the example circuit if you are not sure how to do this properly. For a secure connection, you will need solder headers to the FTDI pins. See our soldering tutorial if you need help here. Otherwise, you can simply apply pressure to the male headers at an angle. As long as you have contact between the FTDI and Qwiic OpenLog, code can still be uploaded.

If you have not already, unzip the default firmware folder and open it in the Arduino IDE. It will probably be in your Downloads folder under the following path:

```
...Downloads\Qwiic_OpenLog-master\Qwiic_OpenLog-master\Firmware\Qwiic_OpenLog\Qwiic_OpenLog.ino
```

Once open, select the board **Tools** > **Board** > **Arduino/Genuino Uno** and select the proper *COM port* for your FTDI board under **Tools** > **Port**. Upload the code by hitting the upload button.

That's it! Your Qwiic OpenLog is now programmed with new firmware.

Configuration File

The configuration file is not as relevant with the updated Qwiic OpenLog as it was with its predecessor. When you open the config file, you will see the I^2C address, escape character, the number of escape characters, and the mode. You can edit the I^2C address and the mode, but ignore the escape character and number of escape characters. Here's what you will see when opening the **config.txt** file in a text editor. As you can see, the `i2c_address` is `42`. This is the default address of the Qwiic OpenLog in decimal form. Converting the value, this matches our default address which is **0x2A**.

```
42,26,3,0
i2c_address,escape,esc#,mode
```

Troubleshooting

There are several different options to check if you are having issues with your Qwiic OpenLog.

Check STAT1 LED Behavior

STAT1 LED shows different behavior for two different common errors.

- 3 Blinks: The microSD card failed to initialize. You may need to format the card with FAT/FAT16 on a computer.
- 5 Blinks: OpenLog has changed to a new baud rate and needs to be power cycled.

Double Check Subdirectory Structure

If you are using the default OpenLog.ino example, OpenLog will only support two subdirectories. You will need to change `FOLDER_TRACK_DEPTH` from `2` to the number of subdirectories you need to support. Once you've done this, recompile the code, and upload the modified firmware.

Verify the Number of Files in the Root Directory

OpenLog will only support up to 65,534 log files in the root directory. We recommend reformatting your microSD card to improve logging speed.

Verify the Size of your Modified Firmware

If you are writing a custom sketch for the OpenLog, verify that your sketch is not larger than 32,256. If so, it will cut into the upper 500 bytes of Flash memory, which is used by the Optiboot serial bootloader.

Double Check File Names

All file names should be alpha-numeric. `MyLOG1.txt` is ok, but `Hi !e _.txt` may not work.

Format your MicroSD Card

Remember to use a card with few or no files on it. A microSD card with 3.1GB worth of ZIP files or MP3s has a slower response time than an empty card.

If you did not format your microSD card on a Windows OS, reformat the microSD card and create a DOS filesystem on the SD card.

Swap MicroSD Cards

There are many different types of card manufacturers, relabeled cards, card sizes, and card classes, and they may not all work properly. We typically use a 16GB class 10 microSD card, which works well at 9600bps. If you are using an older card or anything less than class 6, you may want to try upgrading your SD card.

Add Delays Between Character Writes

By adding a small delay between Serial.print() statements, you can give OpenLog a chance to record its current buffer.

For example:

```
Serial.begin(115200);
for(int i = 1 ; i < 10 ; i++) {
    Serial.print(i, DEC);
    Serial.println(":abcdefghijklmnopqrstuvwxyz-!#");
}
```

may not log properly, as there are a lot of characters being sent right next to each other. Inserting a small delay of 15ms between large character writes will help OpenLog record without dropping characters.

```
Serial.begin(115200);
for(int i = 1 ; i < 10 ; i++) {
    Serial.print(i, DEC);
    Serial.println(":abcdefghijklmnopqrstuvwxyz-!#");
    delay(15);
}
```

Check with the Community

If you are still having issues with your Qwiic OpenLog, please check out the current and closed issues on our GitHub repository here. There is a large community working with the OpenLog, so chances are that someone has found a fix for the problem you are seeing.

Resources and Going Further

Now that you've successfully logged data with your Qwiic OpenLog, you can set up remote projects and monitor all the possible data coming in. Consider creating your own Citizen Science project, or even a pet tracker to see what Fluffy does when out and about!

For more information, check out the resources below:

- Eagle Files (ZIP)
- Schematic (PDF)
- Qwiic OpenLog GitHub
- Qwiic OpenLog Arduino Library GitHub
- SdFat GitHub
- SerialPort GitHub

Need some inspiration for your next project? Check out some of these related tutorials:

Blynk Board Project Guide

A series of Blynk projects you can set up on the Blynk Board without ever re-programming it.

Hazardous Gas Monitor

Build a portable gas monitor to check for dangerous levels of hazardous gases.

Mini GPS Shield Hookup Guide

A hookup guide for the SparkFun Mini GPS Shield.

MicroMod Weather Carrier Board Hookup Guide

A quick guide to help to create your own MicroMod weather station using the MicroMod Weather Carrier Board and Processor of your choice.

If you have any tutorial feedback, please visit the comments or contact our technical support team at TechSupport@sparkfun.com.