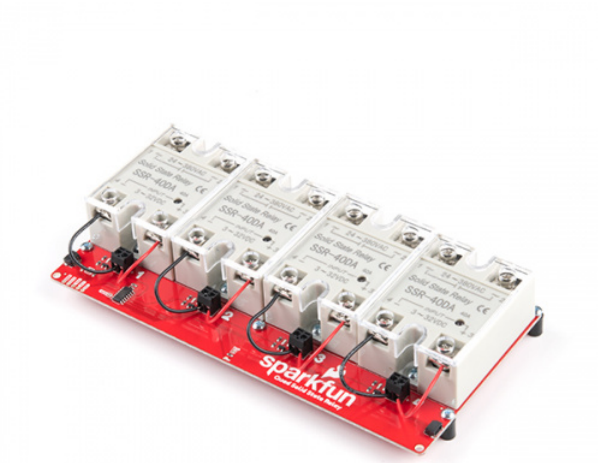
SparkFun Qwiic Quad Solid State Relay Kit Hookup Guide

Introduction

We love our Solid State Relay - 40A. We love it so much we've designed the SparkFun Qwiic Quad Solid State Relay Kit. This kit allows you to assemble up to four of these solid state relays on a single PCB and control them all via I^2C from your favorite microcontroller. The Quad Solid State Relay Kit comes with four solid state relays rated up to **40A** at voltages between **28-380 VAC** so you can switch some serious power with this board all from a single Qwiic connector attached to an Arduino or other low-powered microcontroller.



SparkFun Qwiic Quad Solid State Relay Kit
KIT-16833

Product Showcase: SparkFun Qwiic Quad Solid State Relay Kit

In this tutorial we'll cover what is included with the Quad Solid State Relay Kit along with how to assemble and finally how to connect it to an Arduino and Raspberry Pi to use the examples included in our Arduino Library and Python package.

Required Materials

You will need a microcontroller to control the Qwiic Quad Solid State Relay in order to follow along with this tutorial. Below are a few options that come Qwiic-enabled out of the box:

SparkFun Thing Plus - ESP32 WROOM
WRL-15663

SparkFun RedBoard Qwiic
DEV-15123

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

SparkFun RedBoard Artemis
DEV-15444

We also have a Python package available for this and our other Qwiic Relay boards so you can use a Raspberry Pi as your controller as well.

Raspberry Pi 3 B+
DEV-14643

Raspberry Pi 4 Model B (4 GB)
DEV-15447

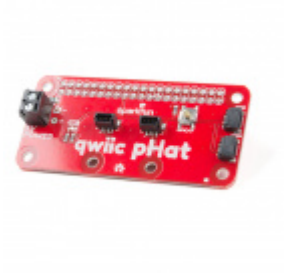
Raspberry Pi 4 Model B (2 GB)
DEV-15446

SparkFun Raspberry Pi 4 Desktop Kit - 4GB
KIT-16386

If your chosen microcontroller is not already Qwiic-enabled, you can add that functionality with one or more of the following items:

SparkFun Qwiic Shield for Arduino
DEV-14352

SparkFun Qwiic Adapter
DEV-14495



SparkFun Qwiic pHAT v2.0 for Raspberry Pi
DEV-15945

SparkFun Qwiic SHIM for Raspberry Pi
DEV-15794

You will also need at least one Qwiic cable to connect your Quad Solid State Relay Kit to your microcontroller.

Qwiic Cable - 500mm
PRT-14429

Qwiic Cable - 100mm
PRT-14427

Qwiic Cable - 200mm
PRT-14428

Qwiic Cable - 50mm
PRT-14426

Required Tools

While there is no soldering required to assemble the Qwiic Quad Solid State Relay, there is some minor assembly involved. You will need a screwdriver as well as wire strippers to assemble your kit and prepare your wires.

Self-Adjusting Wire Strippers
TOL-14872

Pocket Screwdriver Set
TOL-12891

SparkFun Mini Screwdriver
TOL-09146

Wire Strippers - 20-30AWG
TOL-15220

Suggested Reading

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

qwiic

We would also suggest reading through the following tutorials if you are not familiar with the concepts covered in them:

Serial Communication

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

Working with Wire

How to strip, crimp, and work with wire.

Electric Power

An overview of electric power, the rate of energy transfer. We'll talk definition of power, watts, equations, and power ratings. 1.21 gigawatts of tutorial fun!

I2C

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

Hardware Overview

The Qwiic Quad Solid State Relay Kit builds on the design for our Qwiic Quad Relay using an ATTiny84 to act as the "brain" for the board. To allow for faster and quieter switching of the loads, this board uses solid state relays instead of electromechanical relays.

In this section we'll cover the hardware present on the Quad Solid State Relay Kit and note some of the unique functionality of the components.

Solid State vs Mechanical Relays

Let's talk briefly about two types of relays before we get into the details on everything included with the kit. A relay is a special kind of switch that has a switching mechanism inside *isolated* from the switch. This allows us to control the high-voltage output of the relay from a low-power input like a microcontroller.

Now let's compare an electromechanical relay (EMR) and a solid state relay (SSR). The primary difference is an EMR uses an electromagnetic field to physically close a switch inside the relay where a SSR uses a semiconductor like a MOSFET or an opto-isolator to switch the load on and off. This allows the SSR to switch at much higher speeds than an EMR and also increases their durability. Theoretically, and if used within spec, an SSR can last forever where EMR's are designed with an expected lifetime. EMR's are typically cheaper and also can allow multiple loads on a single relay (if it has multiple contacts) where SSR's are limited to a single load.

There is a *lot* of discussion out there about which type of relay is best but it really comes down to two things: budget and versatility. This brief outline covers the very basics of mechanical and solid state relays. Take some time to do a bit more research to help you decide which type of relay will be best suited for your project. For a short yet informative overview of electromechanical relays vs solid state relays, check out this blog post.

40A Solid State Relays

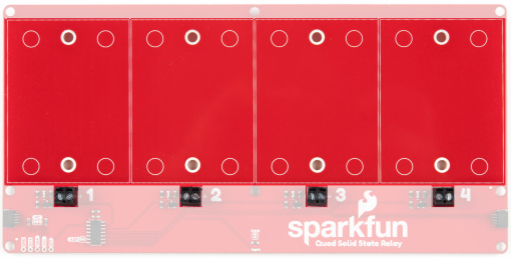
The kit includes four of our 40A Solid State Relays (SSR's) for you to assemble on to your board. Let's cover some characteristics from the relay's datasheet so you know what to expect from them and the Quad Solid State Relay.

⚡ ∼ **AC Only!** These relays are designed to switch **AC** loads only. If you attach a load powered by **DC** current to the output, you can switch it on but cannot turn the load back off again without disconnecting power elsewhere from the load.

Characteristic	Range
Input Voltage	**3V-32V**
Supply Current	7.5mA @12V (40 mA @3.3V)
Load Voltage	24-380VAC
Max Current (Through Relay)	40A (@240 VAC)

The relays use the Zero Cross trigger method so when operating on a 60Hz AC carrier signal you can switch the relays up to 120 times per second! Read on in the Arduino Examples and Python Examples sections where we'll demonstrate how to do that using a PWM signal.

The four 2-pin screw terminal blocks on the board connect the control signal inputs of the relays to the ATTiny84. The kit includes two lengths of wire for you to use to connect the relays to the board. Read on to the Hardware Hookup section for tips on assembling your kit.



Having trouble viewing the details in the image? Click on it for a larger view!

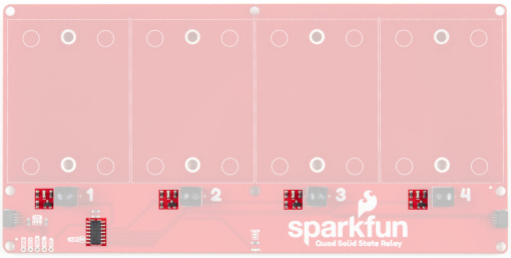
One last note before we move on from the relays. The circuit for controlling the relays is designed for a **Normally Open** circuit **only**. This means your load is normally off until the relay is switched on. For more information on how this circuit is configured, check out the schematic and the relay datasheet.

Note on Heat Sinks: For small loads (eg. ≤**10A**) a heat sink should not be required if there is air cooling available. For larger loads, a heat sink is *strongly recommended*! Review the relay's datasheet for recommendations on heat sink dimensions.

ATTiny84

The ATTiny84 on this board comes pre-programmed with firmware to control up to four relays via I^2C commands. The ATTiny84's I^2C address is **0x08** by default. Each relay control input is connected to an I/O pin on the ATTiny84 via a transistor circuit to protect the I/O pins as each relay takes **~40mA** to toggle **@3.3V**. Each relay also has a blue STAT LED tied to the I/O circuit to indicate when the relay is on.

There is also a 2x3 header broken out on the back of the board for programming the ATTiny84. This is primarily used for programming during manufacturing but you can re-program the IC if you would like. You can download and modify the firmware from the Hardware GitHub Repository.



Having trouble viewing the details in the image? Click on it for a larger view!

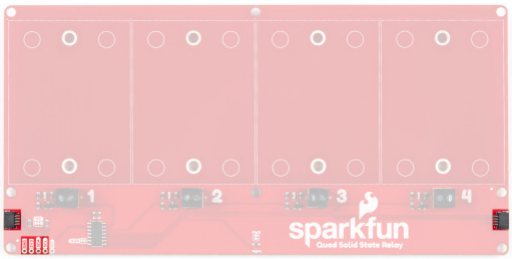
Need help re-programming the ATTiny84 on your Qwiic Quad Solid State Relay? Take a look at our Tiny AVR Programmer Hookup Guide and our Re-Programming the LilyTiny/LilyTwinkle Tutorial.

Power

Power for the board is provided either via the Qwiic interface or through the labeled **3.3V** and **GND** pins broken out on the bottom left of the board.

Qwiic and I^{2}C Interface

The I^{2}C interface is broken out to two Qwiic connectors on either side of the board as well as 0.1"-spaced header pins for those who prefer a more traditional soldered connection.

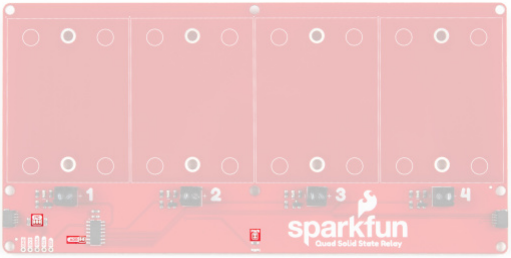


Having trouble viewing the details in the image? Click on it for a larger view!

Jumpers

If you have never worked with solder jumpers and PCB traces before or would like a quick refresher, check out our How to Work with Solder Jumpers and PCB Traces tutorial for detailed instructions and tips.

There are three jumpers on the Qwiic Quad Relay labeled PWR, I2C and ADR. The PWR jumper enables the Power LED, the I^{2}C jumper pulls the SDA and SCL lines to **3.3V** and the ADR jumper sets the I^{2}C address of the ATTiny84.



Having trouble viewing the details in the image? Click on it for a larger view!

Power Jumper

The power jumper (labeled PWR) controls voltage to the power LED on the board. This jumper is **closed** by default. To disable the power LED, simply open the jumper by severing the trace in between the two pads. Disabling the power LED can help reduce the total current draw of the board.

Address Jumper

The address jumper (labeled ADR) sets the I^2C address for the ATTiny84. The jumper is **closed** by default with the address set to **0x08**. Opening the jumper will change the address to **0x09**.

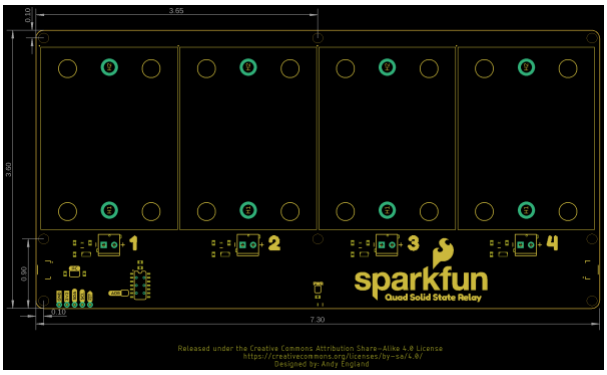
I^2C Jumper

The I^2C Jumper pulls the SDA and SCL lines to **3.3V** via two **4.7K Ohm** resistors. The jumper is **closed** by default. Open this jumper if you have multiple devices connected to the bus with their own pull-up resistors enabled to avoid causing the parallel equivalent resistance creating too strong of a pull-up for the bus to operate correctly. As a general rule, disable all but one pair of pull-up resistors if multiple devices are connected to the bus.

> **Note:** If you disable the pull-up resistors on your Qwiic Quad Relay, make sure the rest of your I^2C bus is operating at **3.3V Logic**. If your bus is running at a different logic, you will need to shift that logic to **3.3V** to avoid damaging the ATTiny84 on this board.

Board Dimensions

The Qwiic Quad Solid State Relay is quite large to fit all four relays and measures 7.30" x 3.30" (185.42mm x 91.44mm). There are eight mounting holes that fit a 4-40 screw.

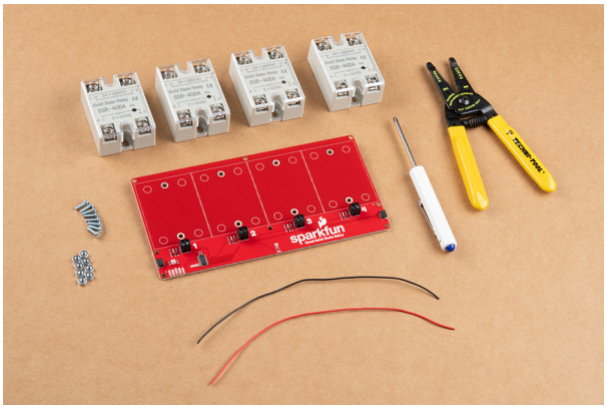


Having trouble seeing the details? Click on the image for a larger view.

With all the hardware included with the Qwiic Quad Solid State Relay covered, it's time to start assembling it!

Hardware Assembly

While the Qwiic Quad Solid State Relay does not require any soldering, there is some minor assembly needed prior to using your board. In this section we'll provide a brief set of instructions to assemble your board and afterward you can start switching some high power loads.



Wire Strippers and Screwdriver sold separately.

Preparation and Initial Relay Assembly

First, we want to attach our SSR's to the Qwiic Quad Solid Relay board. All you need here is a screwdriver, the screws included with the kit and your relays. Place one of the relays in the area outlined on the PCB and secure it in place from the opposite side of the PCB. Make sure the relays are oriented properly with the side labeled `Input` closest to the screw terminals.

We just used hand-tightening for the relays here but if your installation is going to be more permanent you'll want to use some pliers or a socket to hold the nut into place while securing the relay.

Next, locate the lengths of wire included with your kit as we'll need them to connect our relays to their respective screw terminal block. Cut each wire into four even lengths and strip away a small portion of the insulation on each end. With everything **unpowered**, take one of your prepared wires, secure one end to the terminal labeled "**-**" and the other to the terminal on your relay labeled "**4/-**". Next, take another of your wires and secure one end to the terminal labeled "**+**" and the other to the terminal on your relay labeled "**3/+**".

For the purpose of this tutorial we're only fully assembling one of the included SSR's but feel free to attach the other three at this point.

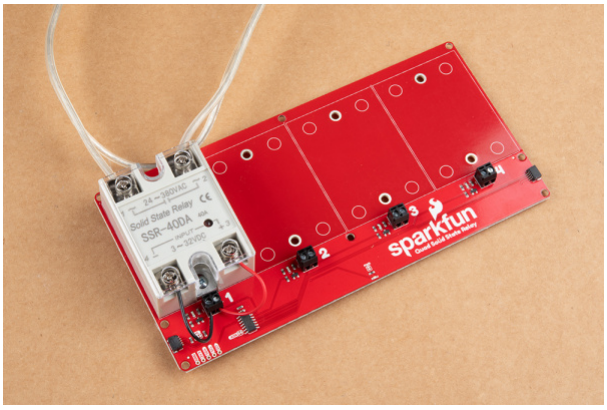
Relay Output Load Assembly

Next up we'll want to prepare our AC load we are controlling with the relays.

Note: Not sure about what color insulation wiring is used in you region? Check out the standard wire insulation colors listed online for reference. If you are unsure about the standard wiring color in your region, please consult a certified electrician to connect to the AC input voltage side.

⚡ **Warning!** Make sure the cable is **not** plugged into the wall as you cut into the wire in the following section.

You'll have to cut and strip your live AC line (usually black or red) and connect one end of the cut wire to the terminal on your relay labeled `1` and the other end to the terminal labeled `2`.

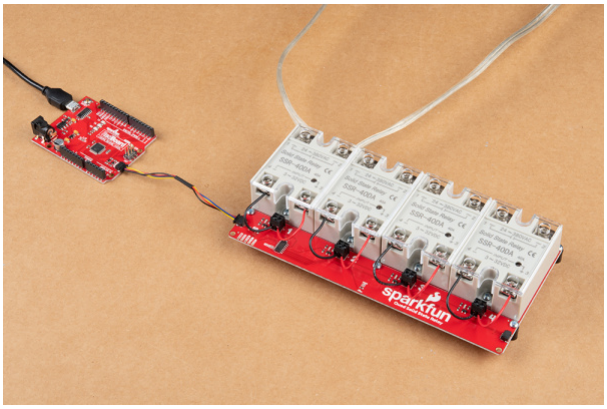


⚡ **Warning!** Make sure that your wires connecting to the wall outlet are secure and are rated to handle the current of your load! Please be careful when handling the contacts when the cable is plugged into a wall outlet. **Touching the contacts while powered could result in injury.**

If you need tips or help on safety and insulation for high voltage circuits, check out the notes about Safety and Insulation from our Beefcake Relay Control Kit.

Connecting to Microcontroller

Now that your Quad SSR Kit is fully assembled and connected to your load you can connect it to your development board using a Qwiic cable or adapter cable. If you would prefer to not use the Qwiic connectors, you can connect to the 0.1" header pins broken out on the bottom left of the board.



*Note: The black standoffs and screws securing them to the relay board pictured here are **not** included with the kit.*

If you decide to use the PTH pins broken out on the board you will need to either solder to them. Alternatively, if you want a temporary connection for prototyping, these IC Hooks are a great option to make that connection. If you are not familiar with through-hole soldering take a look at this tutorial:

How to Solder: Through-Hole Soldering
SEPTEMBER 19, 2013
This tutorial covers everything you need to know about through-hole soldering.

With the Qwiic Quad SSR Kit assembled and connected to your microcontroller it's time to get some code uploaded and start taking measurements!

Qwiic Relay Arduino Library

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.

The SparkFun Qwiic Relay Arduino Library works with all SparkFun Qwiic Relay boards to help you use your Qwiic Relay to its full functionality. You can click the link below to download the file or navigate through the Arduino Library Manager by searching **SparkFun Qwiic Relay**. You can also go the library's GitHub repo and download it from there.

SPARKFUN QWIIC RELAY ARDUINO LIBRARY (ZIP)

Library Functions Overview

The list below outlines all of the functions of the Qwiic Relay library along with short descriptions of what they do. The examples cover nearly all of the functions so take a look at those for help integrating them into your own code.

These functions will work with all SparkFun Qwiic Relay boards.

- `bool begin(TwoWire &wirePort = Wire);` - Initialize the Qwiic Relay on the I^{2}C bus.
- `float singleRelayVersion();` - Returns the version number of the relay.
- `void turnRelayOn(uint8_t relay);` - Turn the given relay on. For example, `turnRelayOn(1);` will toggle the first relay. If using the single Qwiic Single Relay leave the unsigned integer blank.
- `void turnRelayOff(uint8_t relay);` - Turn the selected relay off. Similar to the above function, select values between 1 and 4 to turn the chosen relay off. Leave the integer blank if using the Qwiic Single Relay.
- `void toggleRelay(uint8_relay)` - Toggles the selected relay to the opposite state. The function first checks the status of the relay and is toggled to either `on` or `off` depending on what the status check returns.
- `void turnAllRelaysOn();` - Turns all relays on the board on.
- `void turnAllRelaysOff();` - Turns all relays on the board off.
- `void toggleAllRelays();` - Checks the state of each relay and toggles all relays on the board to the opposite state.
- `uint8_t getState(uint8_t relay);` - Returns the status of the selected relay. Returns `1` if on or `0` if off.
- `bool changeAddress(uint8_t newAddress); - Changes the I^{2}C address of the Qwiic Relay. The new address is written to the memory location in EEPROM that determines the address.

The below function is intended for the Qwiic Solid State Relay boards **only**. It will not work on either of the Qwiic Mechanical Relay boards.

- `bool setSlowPWM(uint8_t relay, uint8_t pwmValue);` - Starts a slow PWM (1Hz) with a range from 0-120 for the selected relay. The PWM range is limited to 0-120 as this is the maximum PWM resolution for Zero-crossing SSR's at 60Hz. Setting `pwmValue` to 120 will give you 100% duty cycle and the selected relay will be always on.

Arduino Examples

The examples included with the SparkFun Qwiic Relay Arduino Library work with all Qwiic Relay boards with **one** exception. Example 7 - Slow PWM was added specifically for the Quad Solid State Relay Kit to demonstrate how to use the `setSlowPWM(uint8_t relay, uint8_t pwmValue);` function. As mentioned in the previous section, this function will **only** work on the SSR boards and not the EMR boards. In this section we'll explore this example and explain how it works and how to modify it.

Example 7 - Slow PWM

> If you are not familiar with how Pulse Width Modulation works or what we mean by "duty cycle", head on over to our Pulse Width Modulation tutorial for a detailed introduction to how this type of signal works.

This example generates a Pulse Width Modulation (PWM) signal from the ATTiny84 on the board to quickly switch a selected relay on and off. Open the example in Arduino by navigating to **File** > **Examples** > **SparkFun Qwiic Relay Arduino Library** > **Example7_Slow_PWM**. Next, open the Tools menu and select your board (in our case,

Arduino Uno) and the correct Port your board enumerated on.

Upload the code and open your serial monitor with the baud set to **115200**. The code prints out whether or not the initialization of the Quad SSR Kit was successful and then prints the PWM value for each relay set using the `setSlowPWM();` function. Let's take a closer look at the code:

First, in all code for the Quad SSR Kit, you will need to declare the I^2C address:

```
Qwiic_Relay quadRelay(QUAD_SSR_DEFAULT_ADDRESS);
```

If you have altered the ADR jumper or changed the address using Example 6 - Change I2C Address you will need to adjust this to the correct value.

The code sets up each of the four relays to run at a PWM value of 75 (62.5% duty-cycle). Try playing around with the PWM values set in the `setSlowPWM();` function for each relay to see how it affects the behavior of your load(s).

```
// To turn on a relay give the function the number you want to turn on (or
// off).
quadRelay.setSlowPWM(1, 75);
quadRelay.setSlowPWM(2, 75);
quadRelay.setSlowPWM(3, 75);
quadRelay.setSlowPWM(4, 75);
```

Note, as we mentioned in the previous section, the PWM resolution is capped at 120 since the relay cannot switch more times than that in one second on a 60Hz signal.

The code also demonstrates how to use the `getSlowPWM();` function by printing out the PWM value set for each relay:

```
Serial.println(quadRelay.getSlowPWM(1));
Serial.println(quadRelay.getSlowPWM(2));
Serial.println(quadRelay.getSlowPWM(3));
Serial.println(quadRelay.getSlowPWM(4));
```

Check out that pulsing desk lamp action!

If Arduino isn't your jam, head on to the next section where we cover the Python package we wrote for this and our other Qwiic Relay boards.

Qwiic Relay Python Package

Note: This example assumes you are using the latest version of Python 3. If this is your first time using Python or I^{2}C hardware on a Raspberry Pi, please checkout our tutorial on Python Programming with the Raspberry Pi and the Raspberry Pi SPI and I2C Tutorial.

We've written a Python package to control the Qwiic Relay family including the Quad Solid State Relay Kit. You can install the `sparkfun-qwiic-relay` Python package hosted by PyPi. However, if you prefer to manually download and build the libraries from the GitHub repository, you can grab them here (**Please be aware of any package dependencies. You can also check out the repository documentation page, hosted on Read the Docs.*):

DOWNLOAD THE SPARKFUN RELAY PYTHON PACKAGE (ZIP)

Installation

Note: Don't forget to double check that the hardware I^{2}C connection is enabled on your Raspberry Pi or other single board computer.

PyPi Installation

This repository is hosted on PyPi as the `sparkfun-qwiic-relay package` . On systems that support PyPi installation via `pip3` (use `pip` for Python 2) is simple, using the following commands:

For **all users** (note: the user must have **sudo** privileges):

```
sudo pip3 install sparkfun-qwiic-relay
```

For the **current user**:

```
pip3 install sparkfun-qwiic-relay
```

Local Installation

To install, make sure the `setuptools` package is installed on the system.

Direct installation at the command line (use `python` for Python 2):

```
python3 setup.py install
```

To build a package for use with `pip3` :

```
python3 setup.py sdist
```

A package file is built and placed in a subdirectory called dist. This package file can be installed using `pip3` .

```
cd dist
pip3 install sparkfun_qwiic_relay-<version>.tar.gz
```

Dependencies

This Python package has a few dependencies in the code, listed below:

```
from __future__ import print_function
import math
import qwiic_i2c
```

Function Overview

For a full overview of all the functions included with the Qwiic Relay Py package, head on over to the ReadtheDocs page.

Python Examples

Now that you have the Qwiic Relay Py package installed, it's time to check out the examples included with the package.

Example 1 - The Basics

This example demonstrates the basics of turning relays on and off. The code initializes the Qwiic Quad Solid State Relay on the I^2C bus, turns on relays 1 and 3 and prints the status of all relays.

```
 if myRelays.begin() == False:
    print("The Qwiic Relay isn't connected to the system. Please check your connection", \
        file=sys.stderr)
    return

#Turn on relays one and three
myRelays.set_relay_on(1)
myRelays.set_relay_on(3)
time.sleep(1)

#Print the status of all relays
for relayNum in range(4):
    current_status = None
    if myRelays.get_relay_state(relayNum) is True:
        current_status = "On"
    else:
        current_status = "Off"
    print("Status 1: " + current_status + "\n")
```

Next, the code turns off relays 1 and 3 and turns on relays 2 and 4.

```
#Turn off 1 and 3, turn on 2 and 4
myRelays.set_relay_off(1)
myRelays.set_relay_on(2)
myRelays.set_relay_off(3)
myRelays.set_relay_on(4)
time.sleep(1)
```

Finally, the code turns all relays on for one second and then turns them all off.

```
#Turn all relays on, then turn them all off
myRelays.set_all_relays_on()
time.sleep(1)

myRelays.set_all_relays_off()
```

With the basics of toggling all the relays, it's time to move on to using the Slow PWM function to pulse our Solid State Relays.

Example 2 - Slow PWM

The second example included with the Python package demonstrates how to use the `set_slow_pwm()` function. The example starts just like the first by initializing the Quad SSR Kit on the I^2C bus and then sets the PWM duty-cycle for each relay and uses the `get_slow_pwm()` function to print the PWM value set for each relay:

```
myRelays.set_slow_pwm(1, 30) #25% duty cycle
myRelays.set_slow_pwm(2, 60) #50% duty cycle
myRelays.set_slow_pwm(3, 90) #75% duty cycle
myRelays.set_slow_pwm(4, 120) #100% duty cycle

for relay_num in range(4):
    pwm_value = myRelays.get_slow_pwm(relay_num)
    print("PWM Value for relay ")
    print(relay_num)
    print(": ")
    print(pwm_value)
#Let the slow PWM run for a while
time.sleep(15)
```

Each relay is set to a different duty cycle to demonstrate how that affects the behavior of the load and will run for 15 seconds. Want your relay to pulse at a 50% duty cycle? Set the slow PWM value to `60`. Note: Just like the Arduino Library, the PWM resolution is limited to 0-120 since there are only 120 times where the zero crossing relay can switch in one second.

After our 15 seconds is up, the code turns all relays off by setting the PWM value to 0 and prints out the new PWM values:

```
myRelays.set_slow_pwm(1, 0)
myRelays.set_slow_pwm(2, 0)
myRelays.set_slow_pwm(3, 0)
myRelays.set_slow_pwm(4, 0)


for relay_num in range(4):
    pwm_value = myRelays.get_slow_pwm(relay_num)
    print("PWM Value for relay ")
    print(relay_num)
    print(": ")
    print(pwm_value)
```

That's all for the basics of our Python package! Try using the various functions included here to write your own code for your next relay project.

Register Map

Below is the I^{2}C register map if you would prefer to write your own code package or use a different language to control the Qwiic Quad Solid State Relay Kit.

Qwiic Quad Solid State Relay Register Map

Byte Number	HEX	Register Name	Type	Read/Write	Power On Reset	Description
1	0x01	TOGGLE_RELAY_ONE	byte	Write Only		Toggles relay one to the opposite state.
2	0x02	TOGGLE_RELAY_TWO	byte	Write Only		Toggles relay two to the opposite state.
3	0x03	TOGGLE_RELAY_THREE	byte	Write Only		Toggles relay three to the opposite state.
4	0x04	TOGGLE_RELAY_FOUR	byte	Write Only		Toggles relay four to the opposite state.
5	0x05	RELAY_ONE_STATUS	byte	Read Only		Returns 1 if relay one is on, 0 if off.
6	0x06	RELAY_TWO_STATUS	byte	Read Only		Returns 1 if relay two is on, 0 if off.
7	0x07	RELAY_THREE_STATUS	byte	Read Only		Returns 1 if relay three is on, 0 if off.
8	0x08	RELAY_FOUR_STATUS	byte	Read Only		Returns 1 if relay four is on, 0 if off.
10	0x0A	TURN_ALL_OFF	byte	Write Only		Turn all relays off
11	0x0B	TURN_ALL_ON	byte	Write Only		Turn all relays on.
12	0x0C	TOGGLE_ALL	byte	Write Only		Toggle all relays to the opposite state.
16	0x10	RELAY_ONE_PWM	byte	R/W		PWM value for relay one. Valid range is 0 to 120. Writing this register sets the current PWM value. Reading returns value set.
17	0x11	RELAY_TWO_PWM	byte	R/W		PWM value for relay two. Valid range is 0 to 120. Writing this register sets the current PWM value. Reading returns value set.
18	0x12	RELAY_THREE_PWM	byte	R/W		PWM value for relay three. Valid range is 0 to 120. Writing this register sets the current PWM value. Reading returns value set.
19	0x13	RELAY_FOUR_PWM	byte	R/W		PWM value for relay four. Valid range is 0 to 120. Writing this register sets the current PWM value. Reading returns value set.
199	0xC7	I2cAddress	byte	R/W	NVM/User Set	Value between 0x08 and 0x77 (inclusive) that is the address of this device. Overridden if ADR jumper is opened (address becomes ADR + 1). Default address is 0x08.

You can also download the PDF.

Resources and Going Further

Now that you've successfully assembled your SparkFun Qwiic Quad Solid State Relay and gotten it up and running, it's time to incorporate it into your own project!

For more information, check out the resources below:

- Schematic (PDF)
- Eagle Files (ZIP)
- Board Dimensions (PNG)
- Relay Datasheet (PDF)
- Hardware GitHub Repo
- Arduino Library GitHub Repo
- Qwiic Relay Python Package
    - ReadtheDocs
- Register Map (PDF)

Need some inspiration for your next project? Check out some of these other relay and power control related tutorials.

Photon Remote Water Level Sensor

Learn how to build a remote water level sensor for a water storage tank and how to automate a pump based off the readings!

Blynk Board Project Guide

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

ESP8266 Powered Propane Poofer
Learn how Nick Poole built a WiFi controlled fire-cannon using the ESP8266 Thing Dev Board!

Blynk Board Bridge Widget Demo
A Blynk project that demonstrates how to use the Bridge widget to get two (or more) Blynk Boards to communicate.

Beefcake Relay Control Hookup Guide
This is a guide for assembling and basic use of the Beefcake Relay Control board

How to Build a Remote Kill Switch
Learn how to build a wireless controller to kill power when things go... sentient.

IoT Power Relay
Using the ESP32 to make a web-configured timed relay.

Qwiic Single Relay Hookup Guide
Get started switching those higher power loads around with the Qwiic Single Relay.

Or you can check out these blog posts for inspiration.

Enginursday: Qwiic Escape Room
MARCH 7, 2019

Automatic Light Switch on the Internet of Things
MARCH 21, 2019

Enginursday: Internet of Terror
JULY 11, 2019

Enginursday: Secure DIY Garage Door Opener
JANUARY 16, 2020

Switching to Different Relays
JULY 9, 2020