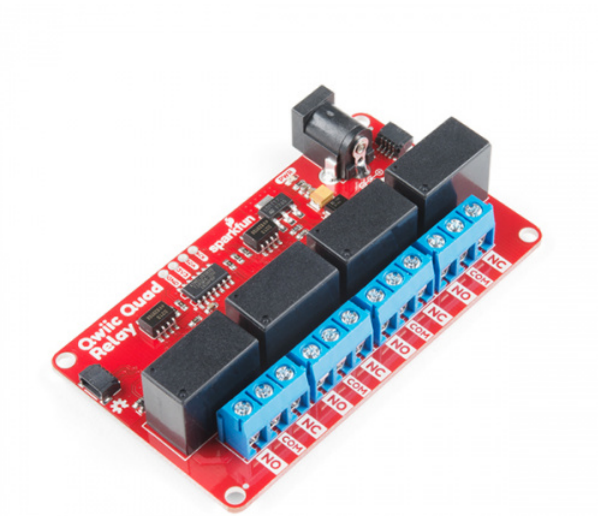
Qwiic Quad Relay Hookup Guide

Introduction

SparkFun's Qwiic Quad Relay is a product designed for switching not one but **four high powered devices** from your Arduino or other low powered microcontroller using I^2C. It has four relays rated up to 5 Amps per channel at 250VAC or 30VDC that are controlled by an ATtiny84A. Each channel has its own blue stat LED, silk for easy identification, and screw terminals for easy connection. The product is *Qwiic* enabled allowing you to easily integrate the Quad Relay with other products in the Qwiic environment, which means no solder neccessary!



SparkFun Qwiic Quad Relay
COM-15102

Product Showcase: SparkFun Qwiic Single and Quad Relay Boar...

⚡ **Before we begin!** There are a number of safety precautions included in the product, but that can not account for human inexperience and error. This product and the example below interacts with **HIGH** AC voltage and so is intended for people experienced around, and knowledgeable about **HIGH** AC voltage. If that's not quite your jam, then take a look at our IoT Power Relay! It's not I^2C but the IoT Power Relay contains shielding to prevent accidental shock.

Required Materials

For the example under **Hardware Assembly**, I used the following materials to control a load (i.e. a lamp). 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.

USB micro-B Cable - 6 Foot
CAB-10215

Jumper Wires - Connected 6" (M/M, 20 pack)
PRT-12795

Wall Adapter Power Supply - 5V DC 2A (Barrel Jack)
TOL-12889

Tactile Button Assortment
COM-10302

SparkFun BlackBoard
SPX-14669

Qwiic Cable - 100mm
PRT-14427

Breadboard - Mini Modular (Red)
PRT-12044

Additional Options

You could also use our 9 volt wall adapter if that suits your fancy and we have a number of Qwiic cable sizes to fit your needs.

Wall Adapter Power Supply - 9VDC 650mA
TOL-00298

Qwiic Cable - 100mm
PRT-14427

Qwiic Cable - 500mm
PRT-14429

Qwiic Cable - 200mm
PRT-14428

Qwiic Cable - 50mm
PRT-14426

Tools

You will need a flush cutter and wire stripper to remove the sheath and insulation from a cable. A Phillips head screwdriver will be required to connect the load's to a screw terminal.

Self-Adjusting Wire Strippers
TOL-14872

Flush Cutters - Xcelite
TOL-14782

SparkFun Mini Screwdriver
TOL-09146

Suggested Reading

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

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!

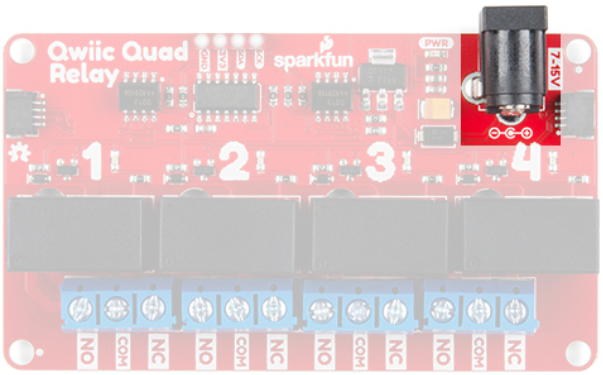
I2C
An introduction to I2C, one of the main embedded communications protocols in use today.

Hardware Overview

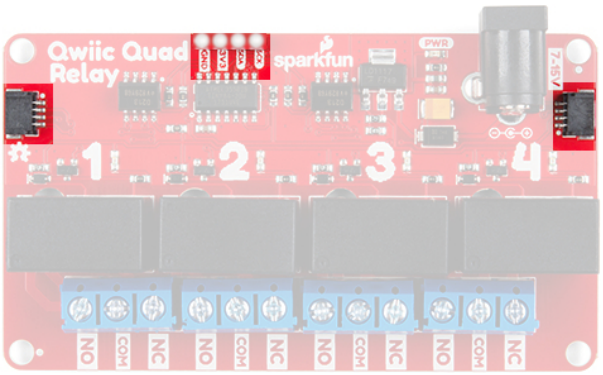
Power

There are two separate power systems on the Quad Relay: a 5V system that powers the relays and a 3.3V system that powers the on board ATtiny84A and interfaces with a microcontroller through the four pin header or Qwiic connector.

The on board barrel jack takes a power source in a range of **7-15V**. It regulates the voltage and supplies power to the 5V power system of the relays. If your wall adapter or power source is at 5 volts like our 5V/2A Wall adapter then you can close the jumper on the underside of the product labeled **5V Wall Adapter** (see **Jumpers** section below), and this will allow you to sidestep the on board regulator to power the 5V system directly. If you decide to go with a higher voltage wall adapter, be cognizant that the voltage regulator will start to heat up. With all the relay channels turned on the Quad Relay will pull ~250mA of current and at 9 Volts, that's 2.25 Watts of power (mathematical!). Over time the regulator will get hot, but will remain functional. I suggest that if you expect to have all relay channels on for extended periods of time, that you go with a 5V power supply.

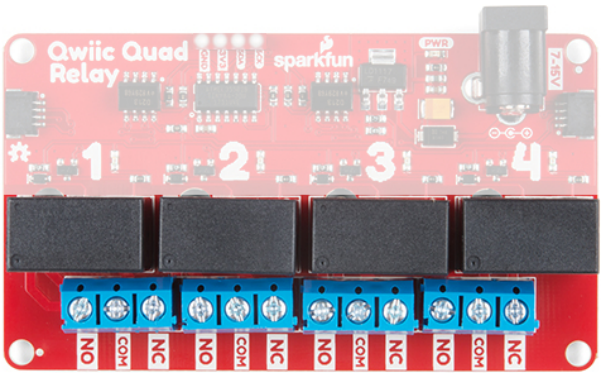


To provide 3.3V to the on board ATtiny84A you can use the plated through hole labeled **3V3** on the four pin header. Alternatively, you can plug a Qwiic connector into one of the two Qwiic connectors.



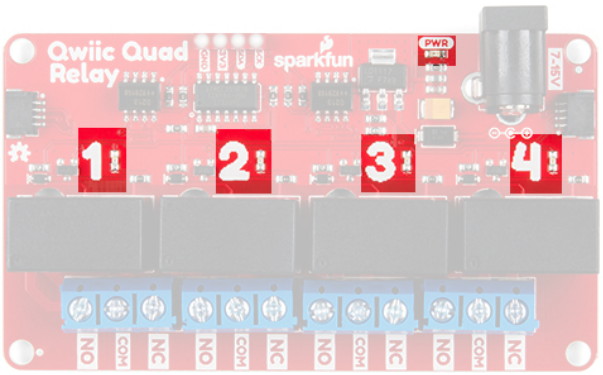
Relays

There are four single pole, double throw JZC-11F relays on the Qwiic Quad Relay. Each relay is capable of 5 Amps at 250VAC or 30VDC. These relays have an associated **blue screw pin terminals** that are aligned in order from left to right.



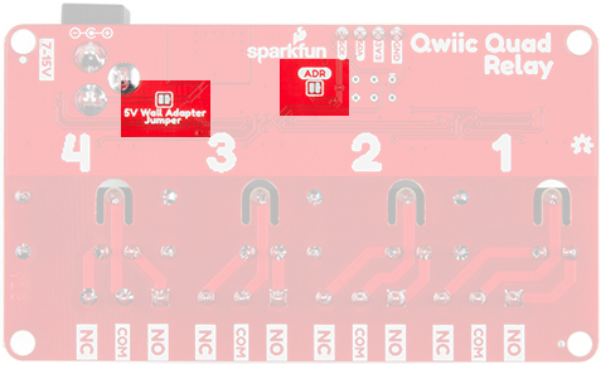
LEDs

There is a red power LED labeled **PWR** that indicates power from the barrel jack. There is also a blue stat LED for each relay labeled with their respective number **1-4**. Whenever a relay is activated (i.e. when COM is connected to NO), the respective LED will light up.



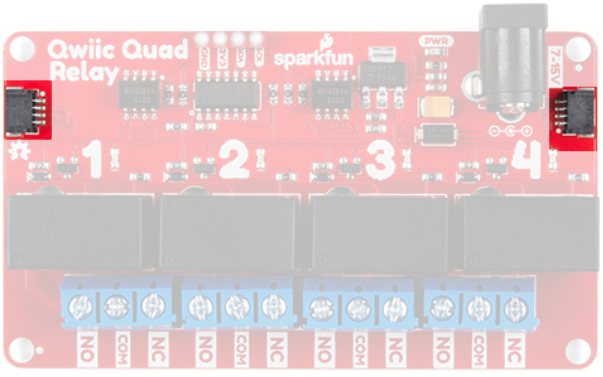
Jumpers

There are two jumpers on the underside of the Qwiic Quad Relay. The first is the address jumper that changes the default I^{2}C address from **0x6D** to **0x6C**. The second is the jumper labeled **5V Wall Adapter Jumper**. If you intend to use a wall adapter or other power source that is below 7-15V than you can close this jumper to side step the on board voltage regulator, and provide 5V directly to the 5V power system.



Qwiic Connectors

The Qwiic connectors allow you to integrate easily into our Qwiic environment and allows you to prototype *without* the need for soldering! The 3.3V provided by the Qwiic connector will power the on board ATtiny84A. If you do not power the 3.3V power system this way, you can still provide power through the four pin header.



Safety Considerations

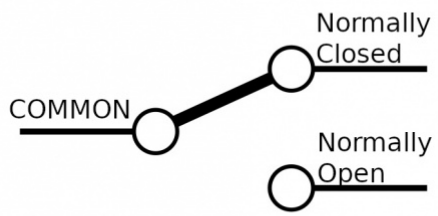
This product is designed to switch high power AC or DC and so has some inherent dangers. We've done our best to implement safety features directly into the design. To begin, the copper ground pour for the ATtiny84A circuitry is restricted to an area apart from the relays. In regards to the microcontroller, there are opto-isolators that isolate the 3.3V power system that it utilizes from the 5V power system of the relays. Next, the common pin of the relays have an air gap surrounding the pin on three sides to prevent any high voltage arcing. Finally, the traces on the relays are extra wide to handle the high amperage carrying potential of the relays.

Hardware Assembly

Introduction to Relays

Let's walk through how to setup the relay to switch on a lamp or other device, but let's begin with a short introduction into relays. A relay is a **switch**. However, unlike most switches, within the relay's housing there is also a switching mechanism that is *isolated* from the switch. This is the relay's defining feature because this separation between switching mechanism and switch, as well as the switching mechanism's low-power requirements, allows for low-power microcontrollers to activate the switching mechanism without interfacing with whatever is getting "switched". Shmow-zow!

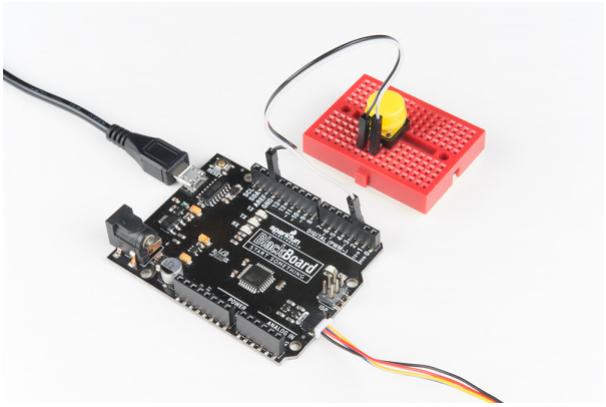
We have three channels per relay broken out to blue screw pin terminals. The channels are labeled for their function. One is considered *normally open* or **NO**, the next channel is *common* or **COM**, and the final is *normally closed* or **NC**. The names explain the state of the channel with relation to the switch at rest. The *normally closed* channel is where the switch sits before the switching mechanism has been activated and conversely the *normally open* channel is where the switch would sit after. The *common* channel is, as the name implies, what the other two channels have in common. This is known as a single pole, double throw switch (**SPDT**). The image below helps to illustrate this characteristic of our particular relay.



When the switching mechanism is activated the thicker bar in the image above that connects *normally closed* to *common* flips over to connect *normally open* and *common*.

Assembly

Onto the assembly. First, I'm using a BlackBoard for it's Qwiic capabilities and it's powered via micro-USB. I have a button plugged into a breadboard, straddling the gap in the center, and jumper wires connecting it to pin 2 and GND on the blackboard.

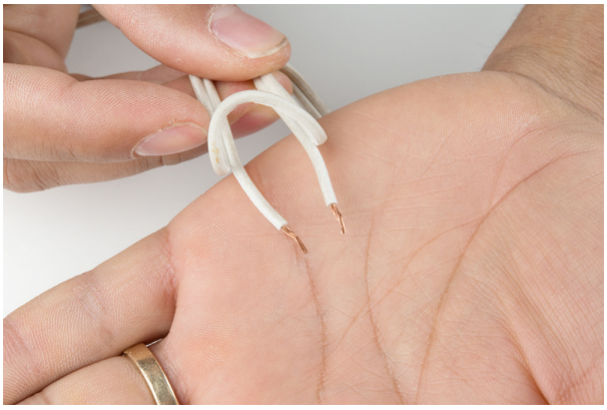


On the tail end is a Qwiic connector leading to the Quad Relay.

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

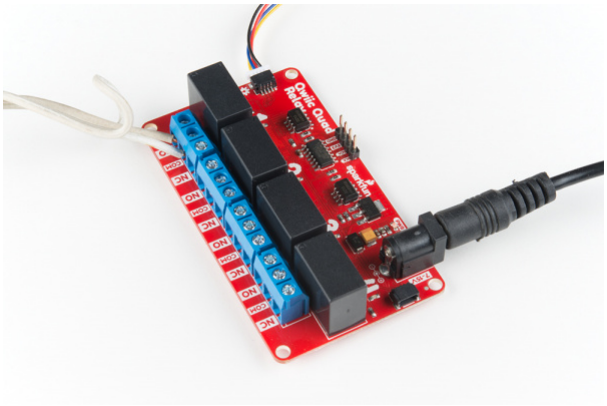
Let's take a quick look at the lamp wire, before we look at the Quad Relay. Our goal here is to sever one of the two lamp wires, and plug the two ends of the cut wire into the relay which will reconnect the wire when we activate the switching mechanism. First, I've cut one of the two wires as shown to create a break in the connection.

I then peeled the wire apart and stripped the two ends.



We'll put one end of our wire in the **COM** channel, and the other we'll have to decide upon. For this project we want our switch to act intuitively: when you activate the switching mechanism, the light switches on. There could be a case where you want the switching mechanism activated as its "rest" state. Since we're going with a more normal approach we'll cut our wire and place one end in *common* and the other in the *normally open* channel. Now when we activate the switching mechanism, the severed wire will be reconnected when the switch flips to the *normally open* channel connecting it and the *common* channel.

For the quad relay, I'm powering the 5V system (the relays), with a 5V Wall Adapter, and the **5V Wall Adapter** jumper closed underneath. The Qwiic cable from the black board is providing power to the 3.3V system as seen at the top of the picture below, and we have the lamp cable plugged into and the screw terminals tightened down on channels **COM** and **NO**.



⚡ **Warning!** Make sure that your wires connecting to the wall outlet are secure and are rated to handle the current! 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.**

Looking for information about safety and insulation? Check out the notes about Safety and Insulation from our Beefcake Relay Control Kit.

Now that our hardware is all set up, let's take a look at the code that turns the lamp on. Remember to not touch the relay's contacts when the system is powered.

Example Code

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.

Let's take a look at some example code fort the Qwiic Quad Relay. You can download all example codes from the GitHub repo by clicking the link below.

QWIIC QUAD RELAY EXAMPLE CODE (ZIP)

Example 1 - Relay Control

This is the code used for the lamp example above. Unzip and open up example one under **...** > **Qwiic_Quad_Relay-master** > **Example Code** > **Example1_Relay_Control** to follow along. At the top, let's break down the defines that are provided. While all of them might not be necessary for your project and they weren't for the lamp example, they are still displayed here to elucidate the functionality of the Quad Relay. First the default address **0x6D** is included and it's followed by the second I^2C address **0x6C**, which is commented out. If you'd rather use the latter then close the jumper on the underside of the product labeled **ADDR**.

```
#include <Wire.h>

#define RELAY_ADDR 0x6D // Default address - open jumper.
//#define RELAY_ADDR 0x6C // Address when jumper is closed.
```

The rest of the defines list all of the possible I^{2}C commands possible. You can toggle each individual relay, *toggle* them all, turn them all on, or turn them all off. There is also a list of commands to check the state of each relay; whether the relay is *on* or *off* represented numerically by **zero** and **15** respectively.

```
// Here are the commands to turn on and off individual relays.
#define TOGGLE_RELAY_ONE 0x01
#define TOGGLE_RELAY_TWO 0x02
#define TOGGLE_RELAY_THREE 0x03
#define TOGGLE_RELAY_FOUR 0x04

// Here are the commands to turn them all off or on.
#define TURN_ALL_ON  0xB
#define TURN_ALL_OFF 0xA

//Here is the command to toggle every relay.
#define TOGGLE_ALL 0xC

// Here are the commands to check on the 'status' of the relay i.e. whether the
// relay is on or off.
#define RELAY_ONE_STATUS 0x05
#define RELAY_TWO_STATUS 0x06
#define RELAY_THREE_STATUS 0x07
#define RELAY_FOUR_STATUS 0x08

//Four buttons
const uint8_t yellow_btn = 2;
const uint8_t blue_btn = 3;
const uint8_t red_btn =  4;
const uint8_t green_btn =  5;
```

Let's move on. In the following code, we define four buttons that when pressed, will send an I^{2}C command to the associated relay. We have a small delay for debounce of 400 milliseconds associated with each button press. In the setup is a call to a function that gets the status of the relays: **On** or **Off** represented by **zero** and **15** respectively. This is detailed more extensively just after the next code block.

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

    //Use internal resitors to keep them in a known high state.
  pinMode(yellow_btn, INPUT_PULLUP);
  pinMode(blue_btn, INPUT_PULLUP);
  pinMode(red_btn, INPUT_PULLUP);
  pinMode(green_btn, INPUT_PULLUP);

  get_relays_status();
}

void loop()
{
    // Since we'll only ever want the relay to be on or off,
    // the logic is handled by the product. Here we're just pressing buttons and
  // putting a small 400 ms debounce.

    //button one, relay one!
  if(digitalRead(yellow_btn) == LOW){
    Serial.println("Yellow Button");

    Wire.beginTransmission(RELAY_ADDR);
    Wire.write(TOGGLE_RELAY_ONE);
    Wire.endTransmission();
        delay(400);
  }

    //button two, relay two!
  if(digitalRead(blue_btn) == LOW){
    Serial.println("Blue Button");
    Wire.beginTransmission(RELAY_ADDR);
    Wire.write(TOGGLE_RELAY_TWO);
    Wire.endTransmission();
        delay(400);
  }

    //button three, toggle every relay: on -> off and off -> on.
  if(digitalRead(red_btn) == LOW){
    Serial.println("Red Button");
    Wire.beginTransmission(RELAY_ADDR);
    Wire.write(TOGGLE_ALL);
    Wire.endTransmission();
        delay(400);
  }

    //button four, turn off all the relays!
  if(digitalRead(green_btn) == LOW){
    Serial.println("Green Button");
    Wire.beginTransmission(RELAY_ADDR);
    Wire.write(TURN_ALL_OFF);
    Wire.endTransmission();
```

```
        delay(400);
  }
}
```

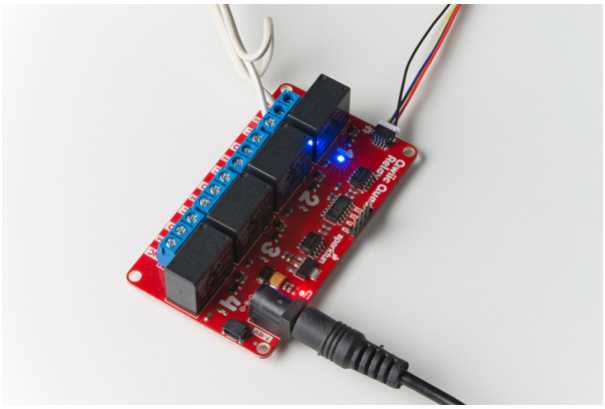
The **get_relays_status** function is of particular interest. Getting the status of the relays works like an I^2C buffer. Depending on the number of bytes requested, you will get the status of that many relays. For example, if you request the status of relay one and four bytes, then you'll get the status of relay one plus the other three. If you request relay one and two bytes, then you'll get the status of relay one and two. If you request the status of relay three and three bytes, then you'll get the status of relay three, relay four, and relay one.

```
void get_relays_status(){
  int i = 1;
  Wire.beginTransmission(RELAY_ADDR);
  // Change the Wire.write statment below to the desired relay you want to check on.
  Wire.write(RELAY_ONE_STATUS);
  Wire.endTransmission();
  //  We want the states of all the relays so we're requesting four bytes, i.e.
  //  four relays. If you request 2 bytes, it will give you the state of two relays!
  Wire.requestFrom(RELAY_ADDR, 4);

  // Print it out, 0 == OFF and 15 == ON.
  while(Wire.available()){
    Serial.print("Relay ");
    Serial.print(i);
    Serial.print(": ");
    Serial.println(Wire.read());
    i++;
  }

}
```

Now let's upload some code via the Arudino IDE. Before uploading, be sure to remove power to the load when uploading to safely handle the relay. Then connect the Arduino to your computer to upload. Select the board (in this case the **Arduino/Genuino Uno**) and COM port that your Arduino has enumerated to. Click the upload button. When the code has finished uploading, place the Arduino and relay on a non-conductive surface to test. Remember to not touch the relay's contacts when the system is powered.

Let There Be Light!

After we load up the code, and press the button we should see the relay one LED light up.



If your relay LED is on and the lamp doesn't turn on, make sure you have the lamp turned on. We'll let the relay handle turning it off and on from now on. Now if all is correctly assembled:

Algebraic!!

> **Note:** Did you notice that there were more examples in the **Example Code** folder? If you need, there is an example using interupts (i.e. **Example2_Relay_Using_Interrupts**) if you need your Arduino to stop whatever it is processing to toggle the relays. Or if you need to control more than one Qwiic quad relay on the same bus? Check out the third example (i.e. **Example3_ChangeI2CAddress**) to adjust an I^2C address.

> **Heads up!** The circuit used in this tutorial is a temporary connection so you will need to secure the circuit and place the relay in an enclosure. For more ideas, check out some of the projects in resources and going further.

Resources and Going Further

Now that you've successfully got your SparkFun Qwiic Quad Relay 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)
- Datasheet (PDF)
- Default Firmware
- Example Code (ZIP)
- GitHub Repo
- SFE Product Showcase

Need some inspiration for your next project? Check out some of these other awesome related tutorials using relays. Be sure to check your current rating when handling the Qwiic Single Relay when browsing some of the other tutorials using relays.

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.

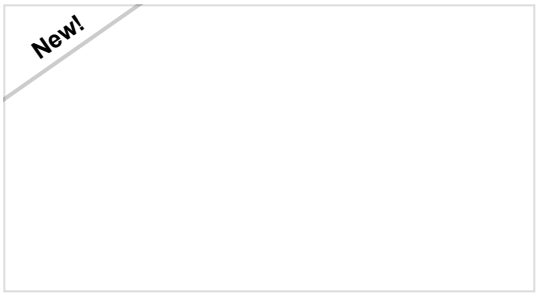
Beefcake Relay Control Hookup Guide

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



Qwiic Single Relay Hookup Guide

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

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