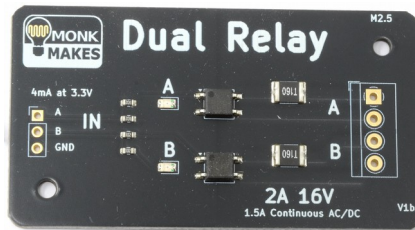


Instructions:



DUAL RELAY



This two channel relay module is ideal for switching low voltage AC and DC loads. It is compatible with 3.3V and 5V microcontroller boards such as Arduino, Raspberry Pi Pico and ESP32.

The board features:

- 2 x opto-isolated AC/DC solid state relays
- 16V maximum switching voltage
- 2A peak current, 1.5A continuous
- Indicator LEDs for each relay
- screw-terminals and header pins supplied (soldering required)
- Downloadable instruction booklet

Note that screw-terminals and header pins are supplied unsoldered, so you will need a soldering iron and solder.

WARNING

This product is for switching low voltage DC only. Under no circumstances should it be connected to AC.

Instructions version 1a.

TABLE OF CONTENTS

Warning.....	1
Assembly	3
Using the Dual Relay.....	4
Downloading the Examples.....	5
Raspberry Pi Pico.....	7
ESP32.....	11
Arduino.....	14
Using Solid State Relays.....	18
Warning: Low Voltage ONLY.....	18
Pulse Width Modulation.....	20
Troubleshooting.....	21
MonkMakes.....	22
Books.....	23

ASSEMBLY

The Dual Relay comes as a kit, with all the surface mount components already soldered onto the PCB. You just have to solder on the header pins and screw terminals. Some projects may be better with leads soldered directly to the pads.

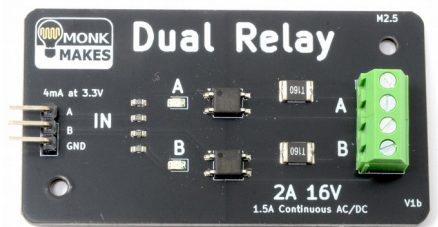
Solder the header pins first, as they are the lowest. Put the pin headers in place then flip the board over onto its back, so that the weight of the PCB holds them in position and solder the pins.

Next clip the two screw-terminals together. If you look closely they have grooves in the sides that let them lock together.



Now, you can solder the screw terminals into place, being careful to ensure that the part of the screw terminal, into which the wire fits, is towards the outside of the board.

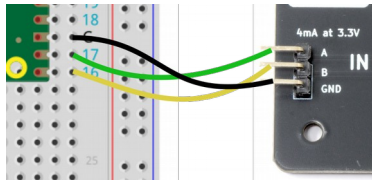
If this is the first time you have soldered, take the time to watch a few video tutorials.



USING THE DUAL RELAY

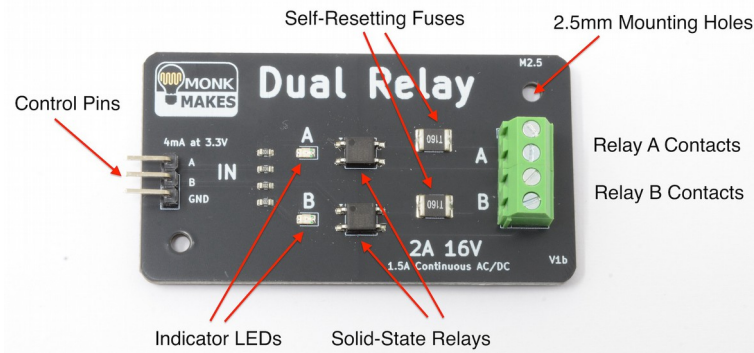
You can think of the Dual Relay as two independent switches controllable from your favourite microcontroller. You can switch DC loads, such as motors and LED lighting modules (low-voltage DC only), that use far too much current to be connected to a GPIO (General Purpose Input Output) pin directly. The Dual Relay is compatible with any 3V or 5V microcontroller, such as an Arduino, ESP32 or a Raspberry Pi Pico. You can also connect a Dual Relay to a single-board-computer with GPIO pins such as a Raspberry Pi 4.

Here's an example of how you might connect a Dual Relay to a Raspberry Pi Pico 2.



The Dual Relay has three control pins (A, B and GND). These pins can be connected to any microcontroller. Connect GND of the microcontroller to GND of the Dual Relay and the A and B pins should be connected to GPIO pins of the microcontroller (either 3.3V or 5V). If you only want to use one of the relays on the Dual Relay, then you only need to connect the control pin for that relay channel (A or B).

When either of the A or B pins are set to be high (1.6V up to 5V) the LED for that channel will light and the corresponding solid-state relay will activate, acting just like a switch had been closed. Any circuit, connected via the screw-terminal pair for that relay channel, will be completed.



DOWNLOADING THE EXAMPLES

To download the ZIP archive containing example programs for all platforms, visit https://github.com/monkmakes/dual_relax

The screenshot shows the GitHub repository for `dual_relax` by user `simonmonk`. The repository is public and has 0 forks and 0 stars. The file list shows folders like `examples` and `figs`, and files like `.gitignore`, `LICENSE`, and `README.md`. The `Code` button is highlighted with a red arrow, and the `Download ZIP` option in the dropdown menu is circled in red. Below the repository information, there is a section titled `MonkMakes Dual Relay` with an introduction and a photograph of the physical relay board. The board is labeled `Dual Relay` and `2A 16V 1.5A Continuous AC/DC`. It features `Control Pins` (A, B, GND), `Self-Resetting Fuses`, `2.5mm Mounting Holes`, and `Relay A Contacts` and `Relay B Contacts`.

Click on the `Code` button, select `Download ZIP` and then extract the downloaded ZIP archive.

If you are familiar with `git` and would prefer to download the examples using the command line, then you can do so with the command:

```
$ git clone https://github.com/monkmakes/dual_relax.git
```

With the extracted archive, you will find an `examples` folder that contains the following folders:

- `raspberry_pi_pico` – MicroPython examples for the Raspberry Pi Pico
- `arduino` – Examples for Arduino, and other boards that can be programmed with the Arduino IDE

- esp32 – ESP32 MicroPython examples

If you are using an ESP32 or Raspberry Pi Pico, then we recommend using the Thonny (<https://thonny.org/>) editor and the MicroPython programming language. Arduino users will probably want to use the Arduino IDE (<https://www.arduino.cc/>) and C programming language.

Since turning one of the relays of the Dual Relay on or off is just a matter of turning a microcontroller's digital output pin on or off, you will find many tutorials on controlling GPIO pins for your chosen microcontroller board.

RASPBERRY PI PICO

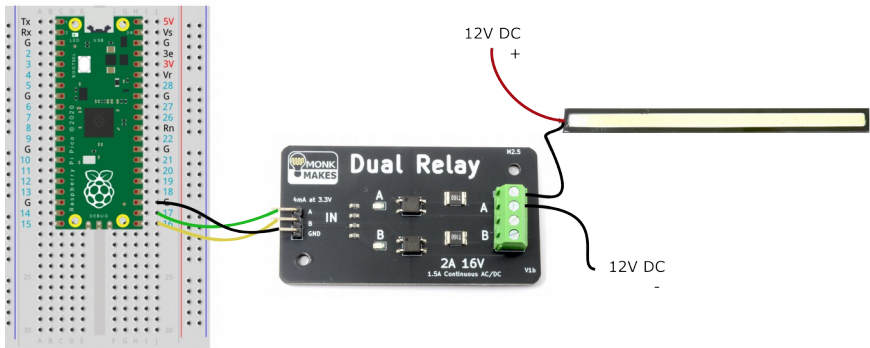
You will need

To build this example project you will need the following items:

- A MonkMakes Dual Relay Board
- A Raspberry Pi Pico
- Solderless Breadboard. The MonkMakes Breadboard for Pico has the Pico pinout printed on it, making pin identification a lot easier (https://monkmakes.com/pico_bb.html).
- 3 x female to male jumper wires
- An LED lamp module (12V). We used a 6W COB lighting strip intended for use in cars.
- 12V power supply

Note that you can also try out the example project without any load attached to the relay outputs because the indicator LED on the Dual Relay will light up when the Relay is on.

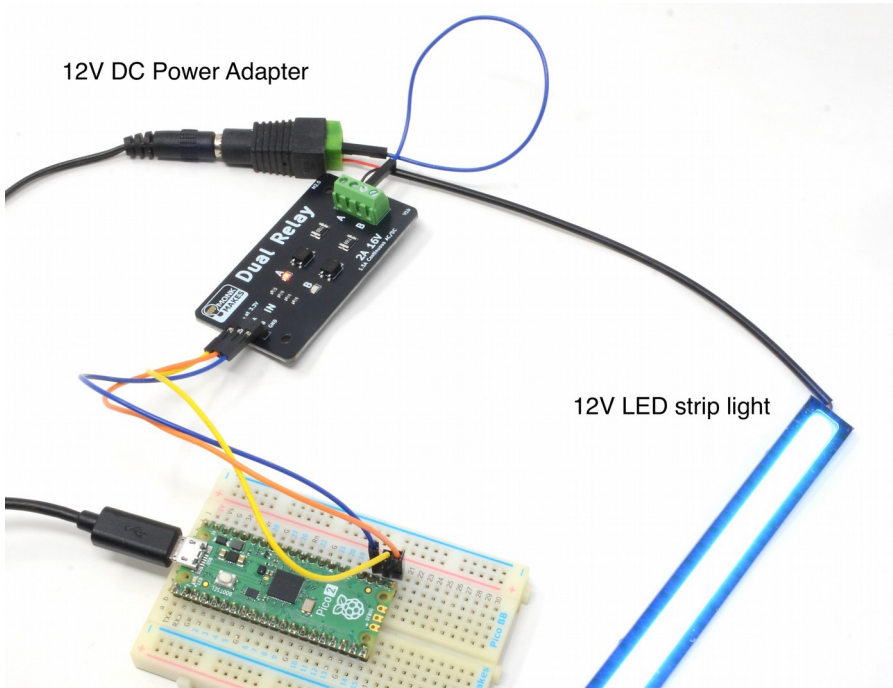
Wiring – Raspberry Pi Pico



12V DC should be connected to the positive lead of the LED strip and the Pico powered from USB. The female to male jumper leads are used to connect the Pico's GND connection to the Dual Relay's GND and two of the Pico's GPIO pins (16 and 17) are connected to the Dual Relay control pins A and B respectively.

The lower of the relay A screw-terminals should be connected to the negative lead of the 12V power source.

Note that in this example, the relay closes the connection between the LED strip and the negative side of the 12V supply, but there is no reason which you couldn't have the relay switch between the positive side of the 12V supply and the LED strip.



Example Software

In the *raspberry_pi_pico* folder of the *examples* folder that you downloaded (see Page 5) you will find the program: `blink.py`

Turning Lamps on and off

Open in Thonny (<https://thonny.org/>) and run it on your Pico. You should see the two Relay indicator LEDs light up in an alternating manner every second. If you have wired-up a load and power supply to one of the relays output (such as the 12V strip-light shown above) then that should turn on and off too.

Here is the code for blink.py.

```
from machine import Pin
from time import sleep

relayA = Pin(17, Pin.OUT)
relayB = Pin(16, Pin.OUT)

while True:
    relayA.on()
    relayB.off()
    sleep(0.5) # pause
    relayA.off()
    relayB.on()
    sleep(0.5)
```

Each of the two relays is connected with a different control pin of the Pico (17 and 16). The main while loop first turns relay A on and relay B off, pauses for half a second before reversing the relays so that A is off and B on.

Pulse Width Modulation (PWM)

For a primer on PWM, see page 20.

As well as turning things on and off, the Dual Relay is also capable of Pulse Width Modulation (PWM). This can be used to control the brightness of a lamp or the speed of a motor. The example in pwm.py illustrates this, using just Relay A of the Dual Relay.

```
from machine import Pin, PWM

relayA = PWM(Pin(14))
relayA.freq(200)

while True:
    brightness_str = input("brightness (0-100):")
    brightness = int(brightness_str)
    if brightness < 0 or brightness > 100:
        print("Brightness between 0 and 100")
    else:
        duty = int(brightness * 655.35)
        relayA.duty_u16(duty)
```

This time, the control pin (17) is a PWM pin, and the PWM frequency is set to 200 Hz (pulses per second).

The program's main loop uses Thonny's Shell area to prompt you to enter a brightness level between 0 and 100. It then converts the number you type into the range 0 to 65535 expected by the `duty_u16` method.

Try entering numbers and notice the brightness change on the Dual Relay's indicator LED (A) and if you have an LED lamp attached, on the lamp too.

A value of 0 should turn them off and 100 set them to maximum brightness.

ESP32

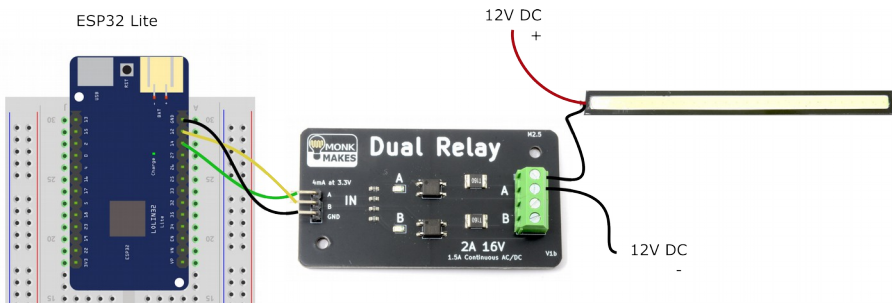
You will need

To build this example project you will need the following items:

- A MonkMakes Dual Relay Board
- An ESP32 board such as the ESP32 Lite
- Solderless Breadboard.
- 3 x female-to-male jumper wires
- An LED lamp module (12V). We used a 6W COB lighting strip intended for use in cars.
- 12V power supply

Note that you can also try out the example project without any load attached to the relay outputs because the indicator LED on the Dual Relay will light up when the Relay is on.

Wiring – ESP32



The connections between the ESP32 board (in this case an ESP32 Lite) are as follows:

ESP32 pin	Dual Relay Connection	Lead Color (in diagram)
G	GND	Black
14	A	Green
12	B	Yellow

12V DC should be connected to the positive lead of the LED strip and the ESP32 powered from USB. The female-to-male jumper leads are used to connect the ESP32's GND connection to the Dual Relay's GND and two of the ESP32's GPIO pins (16 and 17) are connected to the Dual Relay control pins A and B respectively.

The lower of the relay A screw-terminals should be connected to the negative lead of the 12V power source.

Note that in this example, the relay closes the connection between the LED strip and the negative side of the 12V supply, but there is no reason which you couldn't have the relay switch between the positive side of the 12V supply and the LED strip.

Example Software

In the *esp32* folder of the *examples* folder that you downloaded (see Page 5) you will find the program: `blink.py`

Turning Lamps on and off

Open the program in Thonny (<https://thonny.org/>) and run it on your ESP32. You should see the two Relay indicator LEDs light up in an alternating manner every second. If you have wired-up a load and power supply to one of the relays output (such as the 12V strip-light shown above) then that should turn on and off too.

Here is the code for `blink.py`.

```
from machine import Pin
from time import sleep

relayA = Pin(14, Pin.OUT)
relayB = Pin(12, Pin.OUT)

while True:
    relayA.on()
    relayB.off()
    sleep(0.5) # pause
    relayA.off()
    relayB.on()
    sleep(0.5)
```

Each of the two relays is connected with a different control pin of the ESP32 (14 and 12). The main while loop first turns relay A on and relay B off, pauses for half a second before reversing the relays so that A is off and B on.

Note that any of the ESP32's GPIO pins can be used for PWM.

Pulse Width Modulation (PWM)

For a primer on PWM, see page 20.

As well as turning things on and off, the Dual Relay is also capable of Pulse Width Modulation (PWM). This can be used to control the brightness of a lamp or the speed of a motor. The example in `pwm.py` illustrates this, using just Relay A of the Dual Relay.

```
from machine import Pin, PWM

relayA = PWM(Pin(14))
relayA.freq(200)

while True:
    brightness_str = input("brightness (0-100):")
    brightness = int(brightness_str)
    if brightness < 0 or brightness > 100:
        print("Brightness between 0 and 100")
    else:
        duty = int(brightness * 655.35)
        relayA.duty_u16(duty)
```

This time, the control pin (14) is a PWM pin, and the PWM frequency is set to 200 Hz (pulses per second).

The program's main loop uses Thonny's Shell area to prompt you to enter a brightness level between 0 and 100. It then converts the number you type into the range 0 to 65535 expected by the `duty_u16` method.

Try entering numbers and notice the brightness change on the Dual Relay's indicator LED (A) and if you have an LED lamp attached, on the lamp too.

A value of 0 should turn them off and 100 set them to maximum brightness.

ARDUINO

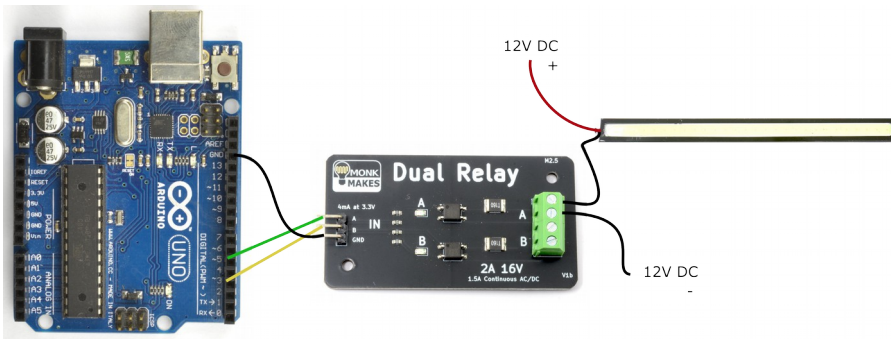
You will need

To build this example project you will need the following items:

- A MonkMakes Dual Relay Board
- An Arduino Uno board
- 3 x female to male jumper wires
- An LED lamp module (12V). We used a 6W COB lighting strip intended for use in cars.
- 12V power supply

Note that you can also try out the example project without any load attached to the relay outputs because the indicator LED on the Dual Relay will light up when the Relay is on.

Wiring – Arduino



Female-to-male header wires can be used to connect the Arduino directly to the Dual Relay. The connections are:

Arduino pin	Dual Relay Connection	Lead Color (in diagram)
GND	GND	Black
5	A	Green
3	B	Yellow

12V DC should be connected to the positive lead of the LED strip and the Arduino powered from USB. The female to male jumper leads are used to connect the Arduino's GND connection to the Dual Relay's GND and two of the Arduino's GPIO pins (5 and 3) are connected to the Dual Relay control pins A and B respectively.

The lower of the relay A screw-terminals should be connected to the negative lead of the 12V power source.

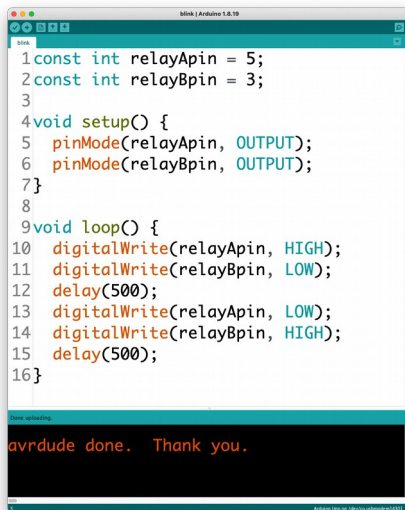
Note that in this example, the relay closes the connection between the LED strip and the negative side of the 12V supply, but there is no reason which you couldn't have the relay switch between the positive side of the 12V supply and the LED strip.

Example Software

In the *arduino* folder of the *examples* folder that you downloaded (see Page 5) you will find the sketch: *blink*.

Turning Lamps on and off

Open the *blink* sketch in the Arduino IDE and upload it to your board.



```
1const int relayApin = 5;
2const int relayBpin = 3;
3
4void setup() {
5  pinMode(relayApin, OUTPUT);
6  pinMode(relayBpin, OUTPUT);
7}
8
9void loop() {
10  digitalWrite(relayApin, HIGH);
11  digitalWrite(relayBpin, LOW);
12  delay(500);
13  digitalWrite(relayApin, LOW);
14  digitalWrite(relayBpin, HIGH);
15  delay(500);
16}
```

avrduide done. Thank you.

You should see the two Relay indicator LEDs light up in an alternating manner every second. If you have wired-up a load and power supply to one of the relays output (such as the 12V strip-light shown above) then that should turn on and off

too.

Here is the code for blink.py.

```
const int relayApin = 5;
const int relayBpin = 3;

void setup() {
  pinMode(relayApin, OUTPUT);
  pinMode(relayBpin, OUTPUT);
}

void loop() {
  digitalWrite(relayApin, HIGH);
  digitalWrite(relayBpin, LOW);
  delay(500);
  digitalWrite(relayApin, LOW);
  digitalWrite(relayBpin, HIGH);
  delay(500);
}
```

Each of the two relays is connected with a different control pin of the Arduino (5 and 3).

The main `loop` first turns relay A on and relay B off, pauses for half a second before reversing the relays so that A is off and B on.

Note that on some Arduino's not all pins can be used for PWM.

Pulse Width Modulation (PWM)

For a primer on PWM, see page 20.

As well as turning things on and off, the Dual Relay is also capable of Pulse Width Modulation (PWM). This can be used to control the brightness of a lamp or the speed of a motor. The example sketch *pwm* illustrates this, using just Relay A of the Dual Relay.

```
const int relayApin = 5;

void setup() {
  Serial.begin(9600);
  Serial.println("brightness (0-255):");
}

void loop() {
  if (Serial.available()) {
    int brightness = Serial.parseInt();
```



```

    if (brightness < 0 || brightness > 255) {
        Serial.println("Brightness between 0 and 255");
    }
    else {
        analogWrite(relayApin, brightness);
        Serial.print("Brightness set to: ");
        Serial.println(brightness);
    }
    Serial.println("brightness (0-255):");
}
}

```

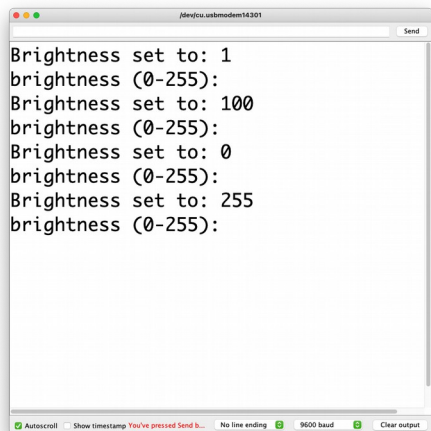
This time, the control pin (5) is a PWM pin.

The Arduino IDE's Serial Monitor is used to tell the Arduino the brightness. To do this, open the Serial Monitor.

Make sure that the Serial Monitor is set to *9600 baud* and *No line ending*.

Try entering numbers and notice the brightness change on the Dual Relay's indicator LED (A) and if you have an LED lamp attached, on the lamp too.

A value of 0 should turn them off and 100 set them to maximum brightness. Note that the default PWM frequency of the Arduino Uno R3 is 976Hz for pin 5, which is a bit fast for the relay chips on the Dual Relay. This means that the strip light may not light at all until you have entered a value of 50 or more.

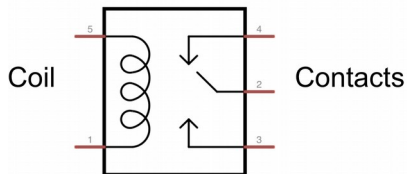


USING SOLID STATE RELAYS

WARNING: LOW VOLTAGE ONLY

A traditional relay is an electromechanical device that uses an electromagnet to actually move a pair of contacts.

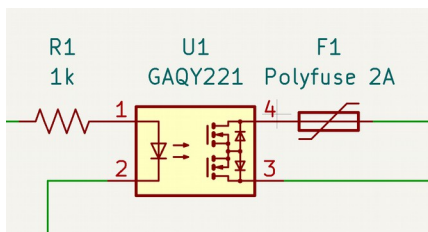
When a current of a few tens of mA (milliamps) flows through the coil, it pulls the contacts together making an electrical connection, just as if you had flipped a switch.



The important thing here is that there is NO electrical connection between the coil and the contacts. The contacts are free to operate at a different voltage and switch a completely different circuit to whatever the coil is controlled by (typically a microcontroller like the Pico using some extra circuitry).

Electro-mechanical relays are cheap, but switch slowly and require quite a high current through the coil to activate them. The Dual Relay uses a type of relay called a solid state relay (SSR). Instead of a coil of wire and contacts, the SSRs used in the Dual Relay use an LED and phototransistors all contained within a small package. The LED only needs a few milliamps to switch the phototransistors on, and is still electrically isolated from the *contacts*.

Here is a section of the Dual Relay's schematic diagram showing how the SSR is used.

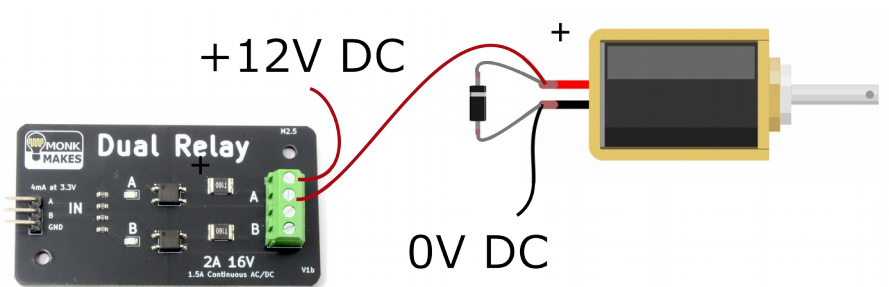


The SSR behaves just like an electromechanical relay, except that it switches much faster, and does not have any moving parts in it. The SSR is protected against over-current by a separate self-resetting polyfuse, that cuts off the current if it exceeds 2A for more than a few seconds.

Switching Inductive Loads

If you plan to use any of the relays to switch inductive loads, such as solenoids or motors, then there is a risk that 'back EMF' voltage spikes may damage the Dual Relay.

When driving inductive loads, use a 'flyback' or 'kickback' diode across the terminals of the solenoid or motor, as shown below. The Dual Relay comes supplied with two of these, just in-case you need them. Note the stripe on the diode must go to the positive connection of the motor or solenoid.



PULSE WIDTH MODULATION

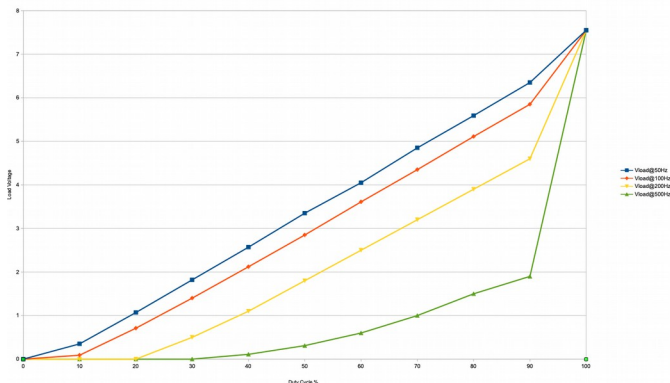
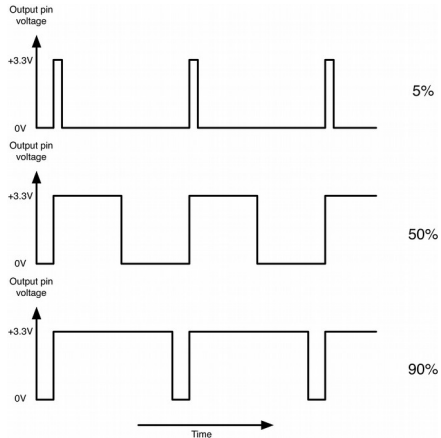
The SSRs used in the Dual Relay are capable of more than just on/off control. They can also be used say to control the brightness of an LED (such as the LED strip-light we we will use in the examples) or the speed of a motor using a process called Pulse Width Modulation or PWM.

A PWM output supplies a train of pulses to the control pin. The duration of these pulses is varied to vary the apparent brightness of a lamp or the speed of a motor. So a short pulse will make the lamp appear dim or a motor slow but, if the pulse is long (3.3V for most of the cycle) then the lamp will appear bright or the motor rotate quickly.

Although the SSRs used in the Dual Relay are much faster than electromechanical relays, they are still relatively slow in electronics terms.

The graph below shows the response of the relay (at full load of 1.5A) to PWM at various frequencies.

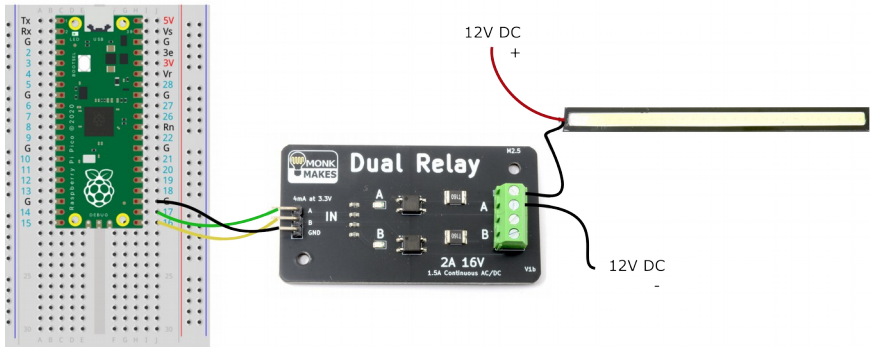
The ideal SSR would have a completely straight line and, at lower currents than the test current here, you get close to that ideal. But here, you can see that you get quite acceptable results up to a frequency of about 100Hz.



TROUBLESHOOTING

Problem: I've connected a motor/light to the relay outputs, but they do not turn on even though the orange indicator LEDs on the Dual Relay light up.

Solution: The screw-terminal relay outputs do not supply an output voltage, they act as an electronic switch. This means that to operate they must act as a switch in a circuit that has both its own power supply and a suitable load. Here is an example.



Problem: The orange indicator LEDs do not light, even though they are connected to the control pins of the Dual Relay.

Solution: Here are a few tips:

Check that all the jumper wires used are correctly wired for the board that you are using, and that the all-important GND wire is connected. Jumper wires will occasionally fail silently, so try swapping out the jumper wires for other wires.

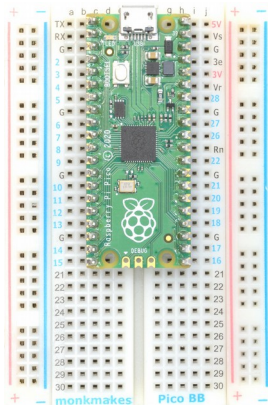
Make sure that the GPIO pin that you are using has been set to operate as a digital output.

You can also check that the Dual Relay is functional by connecting the A or B control pins to 3.3V or 5V on your microcontroller rather than using a GPIO pin.

MONKMAKES

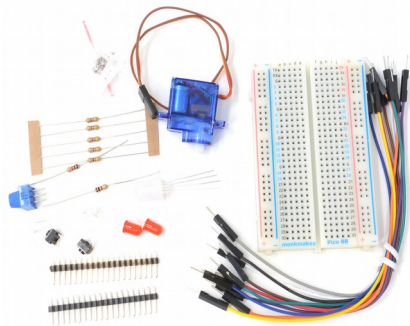
As well as this kit, MonkMakes makes all sorts of kits and gadgets to help with your electronics projects. Find out more, as well as where to buy here:

<https://monkmakes.com/products> you can also follow MonkMakes on Twitter @monkmakes and Instagram as @monk_makes_ltd.



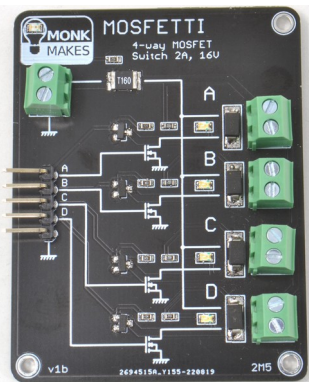
Breadboard for Pico (a solderless breadboard with Pico Pin names printed on it)

https://monkmakes.com/pico_bb



Electronics Kit 1 for Pico.

https://monkmakes.com/pico_kit1



MOSFETI – 4 channel MOSFET switch.

<https://monkmakes.com/mosfetti>



A 4-channel low-voltage Solid State Relay controller designed for the Raspberry Pi Pico.

<https://monkmakes.com/picocontroller>

Books

Simon Monk (the author of this guide) has also written a number of books about hoppy electronics and programming. You can find a full list at: <https://simonmonk.org>, but here are a few that may be of interest.

