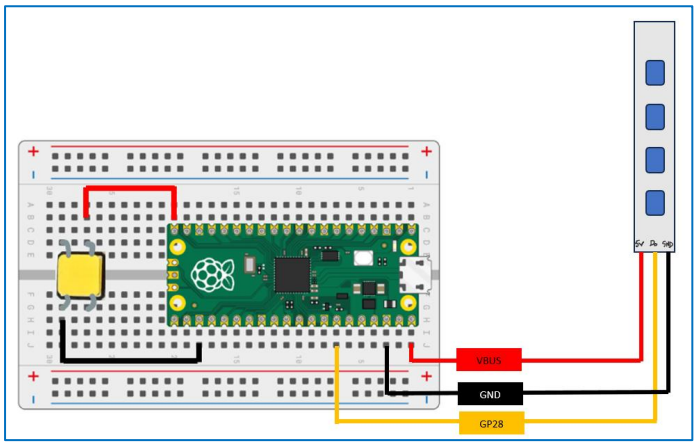
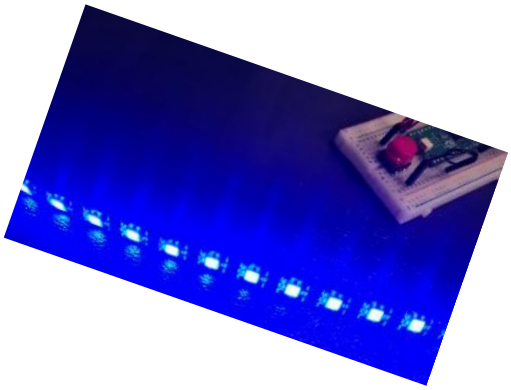
Pico 3.2 Buttons and LEDs

In the last guide we looked at adding a button to our breadboard. We connected it to the Pico and write simple code that made the green on-board LED light up.

How can we use this button to control the LED strips we were working on previously?

Contents





Hardware setup for the Button & LED strip

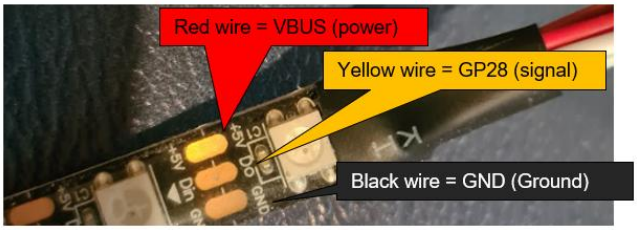
You should have the push button wired up as follows:

- The red positive wire connects to pin 20 (GP15).
- The black negative wire is connected to pin 23 (GND).

The LED strip should be wired up as follows:

- Red wire to power (VBUS)
- Black wire to a ground pin (GND)
- Yellow is the signal wire and connects to Pin GPIO28

This is a reminder of how the wires connect to the end of the LED strip.



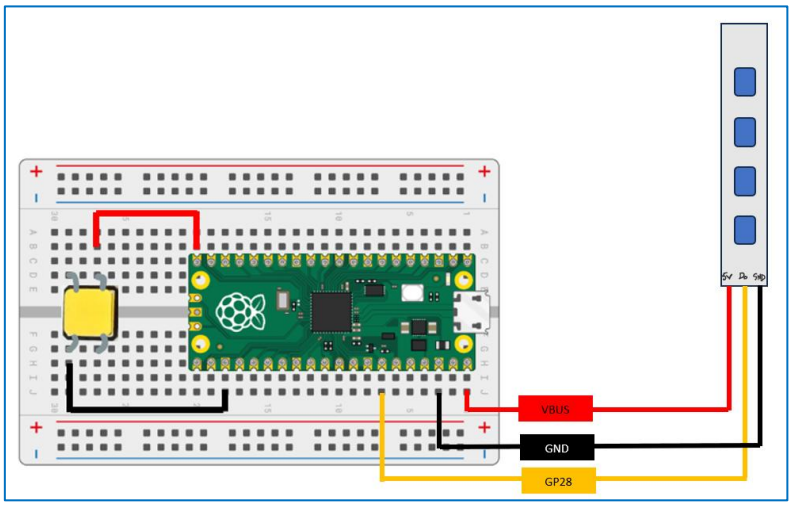
If you are working on a project at home, you may need to get someone to solder connecting wires to your LED strip.

As mentioned before, you can power up to 30 LEDs on a strip through the breadboard. If your project involves using more than 30 LEDs, you need a separate 5v power supply. The 1.4 guide explains further.

If you are in school you have been given a LED strip like this:

We need to connect this LED strip to the breadboard in the same way we did in guide 1.4.

This wiring diagram shows the layout for the push button and the LED strip.



This wiring diagram is the same setup as used in the 'Pico 1.4 LED strips Intro' and 'Pico 1.7 Buttons' tutorials.

Test Code for LED strip:

We are going to run a simple program to check the LED strip is connected to the breadboard. The button is not involved at this stage.

You should already have the neopixel library on your Pico. This was explained in guide 1.4. You can copy/paste the neopixel code from this website:

https://github.com/blaz-r/pi_pico_neopixel

Libraries and Variables

These are the libraries and variables we have used in previous tutorials.

```
###Testcode - make LED strip light up###

#import the libraries
#you will also need neopixel.py saved onto the pico
from machine import Pin
from neopixel import Neopixel
import time
#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)
#LED variables
numpix = 14
strip = Neopixel(numpix, 0, 28, "RGB")
strip.brightness(42)
#colour variables
red = (0, 255, 0)
green = (255, 0, 0)
blue = (0, 0, 255)
blank = (0,0,0)
```

Commands

These commands will test the connection to your LED strip.

```
blank = (0,0,0)
#commands
strip.fill(red)
strip.show()
time.sleep(2)
strip.fill(blank)
strip.show()
```

You can copy and paste this code to make sure your LED strip is connected to the Pico properly:

Click the green run button on Thonny to test this code.

Hopefully the LEDs strip lit up.

Test Code:

```
###Testcode - make LED strip light up###

#import the library
#you will also need neopixel.py saved onto the pico
from machine import Pin
from neopixel import Neopixel
import time
#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)
#LED variables
numpix = 14
strip = Neopixel(numpix, 0, 28, "RGB")
strip.brightness(42)
#colour variables
red = (0, 255, 0)
green = (255, 0, 0)
blue = (0, 0, 255)
blank = (0,0,0)
#command
strip.fill(red)
strip.show()
time.sleep(2)
strip.fill(blank)
strip.show()
```

Troubleshooting

Does your LED strip have 14 LEDs?

*If not, change the value assigned to **numpix**.*

Have you saved neopixel.py onto the pico?

See guide 1.4 to check this.

Have you checked the jumper wires are connected to the correct Pins?

See the wiring diagram.

If you still cannot get the LED strip to light up, try a different LED strip.

Sometimes the wires get twisted and broken.

Challenge 1: Change the Colour

- ✓ Make the LED strip light up a different colour.

Challenge 2: Flashing

- ✓ Make the LED strip flash red 3 times, the blue 3 times.

Ok, so we now know the LED strip is connected to the Pico.

But the button does not do anything yet.

Let's see if we can make the LEDS only light up when the button triggers them.

Using the Button to trigger the LED strip

The test code provided above includes the variables needed. This includes the button variable:

```
4  #you will also need neopixel.py saved onto the pico
5  from machine import Pin
6  from neopixel import Neopixel
7  import time
8  #button variable
9  button = Pin(15, Pin.IN, Pin.PULL_UP)
10 #LED variables
```

The values assigned to the button variable link the physical button to GP15 on the Pico.

This is how our programming command section looks:

```
#commands
strip.fill(red)
strip.show()
time.sleep(2)
strip.fill(blank)
strip.show()
```

We are now going to replace all of these commands with a while loop.

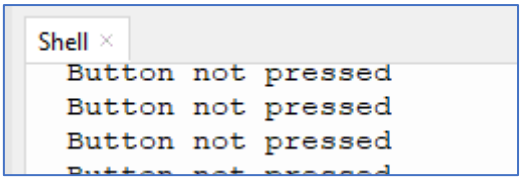
Delete the old command code and replace it with this while loop
(we have used this before in guide 1.7 Button Intro).

This code will check the button connections are working.

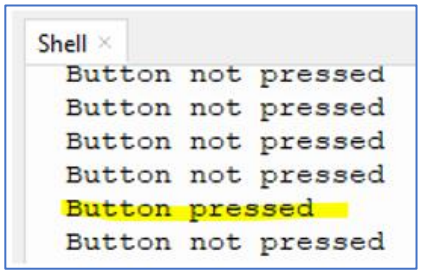
```
#commands
while True:
    if button.value() == 0:
        print("Button pressed")
    else:
        print("Button not pressed")

    time.sleep(0.1)
```

Click the green run button on Thonny to test this code. 

When the button is ***not*** pressed you should see this in the shell:



When the button ***is*** pressed, you should see this in the shell:



Ok, we now know the button is working as it should.

Now we have proved the button is talking to the Pico we can use it to trigger the LED strip.

Test Code:

```
###Testcode - make LED strip light up###

#import the libraries
#you will also need neopixel.py saved onto the pico
from machine import Pin
from neopixel import Neopixel
import time
#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)
#LED variables
numpix = 14
strip = Neopixel(numpix, 0, 28, "RGB")
strip.brightness(42)
#colour variables
red = (0, 255, 0)
green = (255, 0, 0)
blue = (0, 0, 255)
blank = (0,0,0)
#commands
while True:
    if button.value() == 0:
        print("Button pressed")
    else:
        print("Button not pressed")

    time.sleep(0.1)
```

We need to amend the while loop to trigger the LED strip.

```
19 #commands
20 while True:
21     if button.value() == 0:
22         print("Button pressed")
23         strip.fill(red)
24         strip.show()
25     else:
26         print("Button not pressed")
27         strip.fill(blank)
28         strip.show()
29     time.sleep(0.1)
```

This listens out for the button press.

If the button IS pressed, the LED strip lights up.

If the button is NOT pressed, the LED strip is blank.

The **time.sleep(0.1)** code on line 23 makes the loop pause for a 10th of a second so a button press does not accidentally register two or more clicks.

Click the green run button on Thonny to test this code. 

When the user holds the button down, the LED strip should light up.

When the user lets go of the button, the LED strip should go blank.

Hopefully you have now confirmed that when the button is pressed the LED strip lights up.

Challenge 3 – different colours:

- ✓ Make the LED strip light up a different colour.

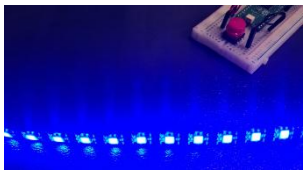
Challenge 4 – trigger a For Loop on button press:

- ✓ Use a For Loop to make the LED strip flash different colours.
- ✓ Use a For Loop to make a single LED sweep from left to right (see guide 1.4).

Ok, now we have that working, let's see if we can make the LED strip change colour with a 2nd button press.

2nd button press = new colour:

Add a new variable to count the number of button presses. This can then be used to make the LED strip light up different colours.



```
5  from machine import Pin
6  from neopixel import Neopixel
7  import time
8  #variable to count button presses
9  clicks = 0
10 #button variable
11 button = Pin(15, Pin.IN, Pin.PULL_UP)
12 #LED variables
13 numpix = 14
```

This is the variable that will store how many times the button has been pressed.

Now add this new line of code.

This makes the variable **clicks** increase by 1 when the button is pressed.

```
22 while True:
23     if button.value() == 0:
24         clicks = clicks + 1
25         print("Button pressed")
26         strip.fill(red)
27         strip.show()
28     else:
29         print("Button not pressed")
```

Now we add a nested If Statement to respond differently, depending on the value of **clicks**.

```
#commands
while True:
    if button.value() == 0:
        clicks = clicks + 1
        print("Button pressed")
        if clicks == 1:
            strip.fill(red)
            strip.show()
        elif clicks == 2:
            strip.fill(blue)
            strip.show()
    else:
        print("Button not pressed")
    time.sleep(0.1)
```

If **clicks** is 1, the LED strip lights up red.

If **clicks** is 2, the LED strip lights up blue.

Run your code to test it. Did the LED strip light up when you pressed the button?

Did the LED strip change colour when you pressed the button again?

Trouble shooting:

Is your indentation correct?

Remember, the second if statement is nested.

Have you set the colour variables for all the extra colours you need for all these new button presses?

You have to set the RGB colour values for any new colour variable you use. The W3Schools website is a good source for finding the RGB values for new colours.

If you continue to have a problem, use this test code to check your hardware is working as it should:

Test Code:

```
###Testcode - make LED strip light up- multi button presses###

#import the libraries
#you will also need neopixel.py saved onto the pico
from machine import Pin
from neopixel import Neopixel
import time
#variable to count button presses
clicks = 0
#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)
#LED variables
numpix = 14
strip = Neopixel(numpix, 0, 28, "RGB")
strip.brightness(42)
#colour variables
red = (0, 255, 0)
green = (255, 0, 0)
blue = (0, 0, 255)
blank = (0,0,0)
#commands
while True:
    if button.value() == 0:
        clicks = clicks + 1
        print("Button pressed")
        if clicks == 1:
            strip.fill(red)
            strip.show()
        elif clicks == 2:
            strip.fill(blue)
            strip.show()
    else:
        print("Button not pressed")
    time.sleep(0.1)
```

Challenge 5 –more colours and button presses:

- ✓ Make the LED strip light up different colours if the button is pressed a 3rd or 4th time.

Challenge 6 – Use a For Loop

- ✓ Use a For Loop instead of strip.fill(colour) for one of the button presses.

Using a For Loop means we can make a single LED sweep from left to right. Or you could run a series of different flashing commands.

Question – while loop problem:

Why would using a while loop be a problem with this button press program?

Resetting clicks to 0:

The problem with our current program is, after a few clicks, the program does not know how to go back to the beginning.

In the test code above I have programmed outputs for up to 2 clicks of the button.

```
22 while True:
23     if button.value() == 0:
24         clicks = clicks + 1
25         print("Button pressed")
26         if clicks == 1:
27             strip.fill(red)
28             strip.show()
29         elif clicks == 2:
30             strip.fill(blue)
31             strip.show()
32     else:
33         print("Button not pressed")
34     time.sleep(0.1)
```

Let's develop this so, on the 3rd click, it resets the variable **clicks** back to 0.

Add another **elif** to the nested If Statement.

```
22  while True:
23      if button.value() == 0:
24          clicks = clicks + 1
25          print("Button pressed")
26          if clicks == 1:
27              strip.fill(red)
28              strip.show()
29          elif clicks == 2:
30              strip.fill(blue)
31              strip.show()
32          elif clicks == 3:
33              clicks = 0
34      else:
35          print("Button not pressed")
36      time.sleep(0.1)
```

This resets **clicks** to 0

The dodgy click issue:

Sometimes, if the user presses the button for too long, it registered as two clicks.

This can be fixed by adding a small delay.

```
while True:
    if button.value() == 0:
        print("Button is Pressed")
        clicks = clicks + 1
        time.sleep(0.2)
        if clicks == 1:
            strip.fill(red)
            strip.show()
```

A small delay stops a single press making **clicks** increase by 2.