Pico 1.4 LED Strips Intro

LED strips are everywhere. In cars, devices and in films!
This guide will help you set up the breadboard and Pico and connect the LED strip. There are a range of commands to control individual LEDs on the strip, or the whole lot.

Contents

Which LED strip should I use

Use the WS2812B RGB LED strip.
These can be purchased from Amazon, Temu and many other online stores.
These LED strips are individually addressable. This means we can control each LED to be a variety of colours and brightnesses.

I suggest using short length of LED strip to practice. You can then develop this to create lighting for shelves, lamps, computer back lighting etc.

We control the LED strip with 3 wires.
There is a + and – wire providing power.
The third wire is the control wire that we use to send control signals to the LED strip from the Pico.

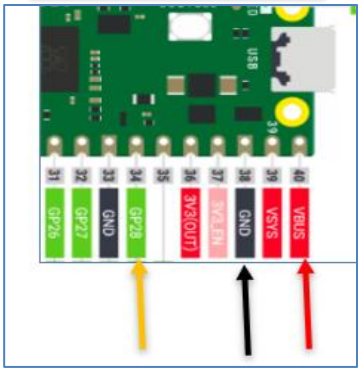
Pico and Breadboard setup:

Use the wiring diagram below to connect the LED strip to the breadboard.

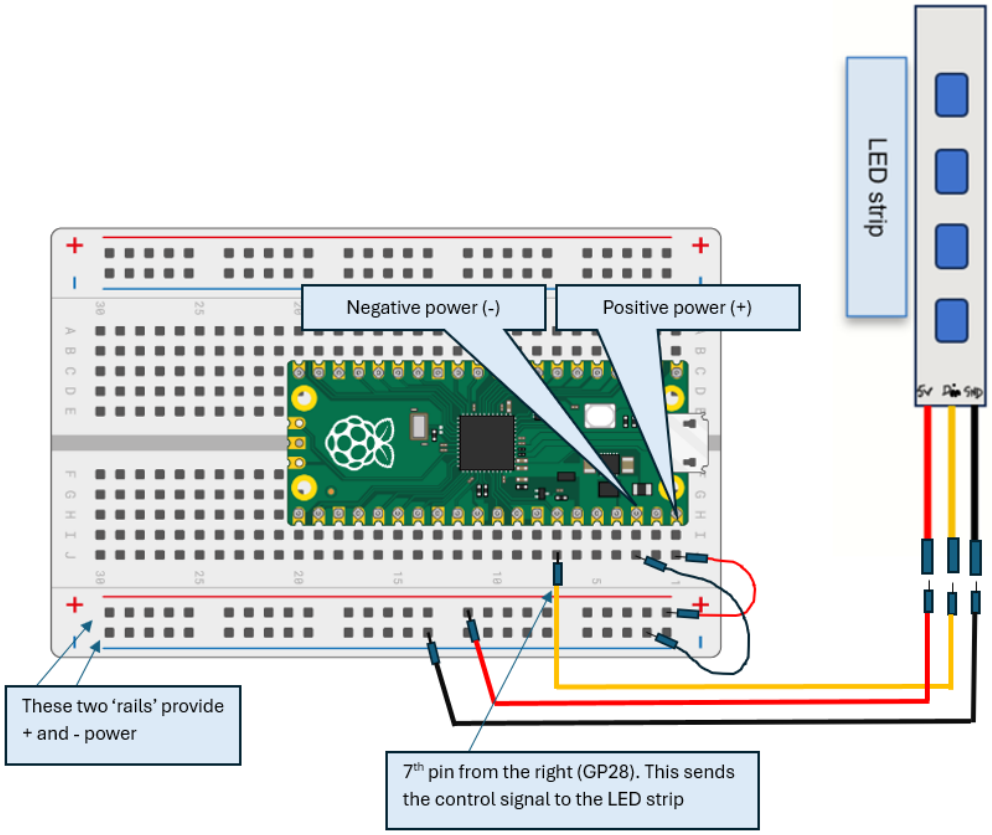
On the Pico:

- LED red wire to VBUS Pin on the Breadboard
- LED yellow wire to GP 28 on the breadboard
- LED black wire to GND Pin on the breadboard

Pico PinOut Diagram.



LED strip connections



Connecting the Pico to the PC

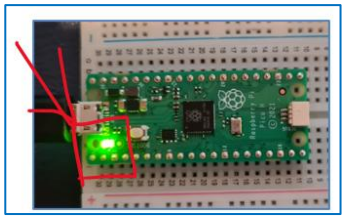
You are likely to have already connected the Pico to the PC in a previous project.

Check your connection using this test code (copy it to Thonny).
This test does not involve breadboard connections or Pin numbers at this stage, so it is a quick, simple way to check the connection.

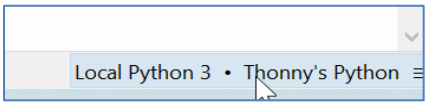
Test Code:

```
from machine import Pin
import time
led = Pin(25, Pin.OUT)
led.value(1)
time.sleep(1)
led.value(0)
```

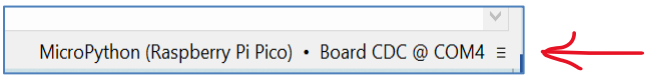
Click Run 
If the green LED on the Pico lit up for one second,
and then went off again,
we know we have a good connection between the PC and the Pico.



If your Pico is not connecting to the PC do the following…
Connect your Pico to the computer using the micro-USB cable.
In the Thonny app on your PC, by default, it shows this in bottom right corner...



...click to change it to this...

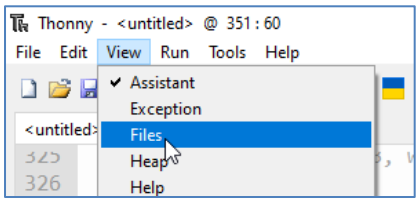


If you cannot see this in the drop-down list, try a different USB port or see guide 1.0 Getting Started to reconnect the Pico to the PC.

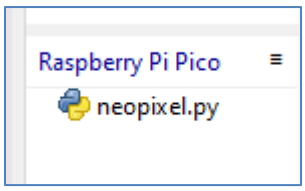
Checking if the neopixel library is installed:

You need the neopixel library installed on the Pico device if you want to control LED strips. It may already be installed!

Open up Thonny.
Click view / Files



Look down to the **bottom** left corner.
You should see a listing for the pico, and the neopixel.py file saved into it.



If you cannot see "neopixel.py" you will need to install the library. If you can see the neopixel.py file, move onto the "Programming LED strips" sub heading.

Install the neopixel Library

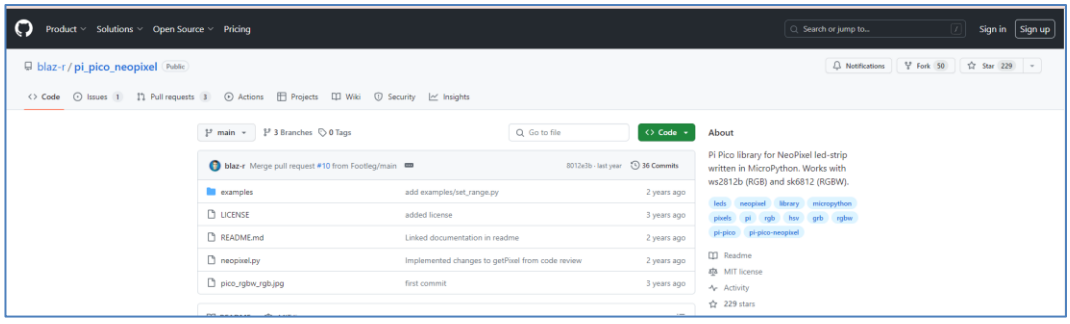
A Library allows us to use prewritten code to make our programming simpler.

- We need the **Pin** library.
- We need the **time** library.
- We need the **neopixel** library.

The **Pin** library allows the code to send and receive signals via the pico Pins.

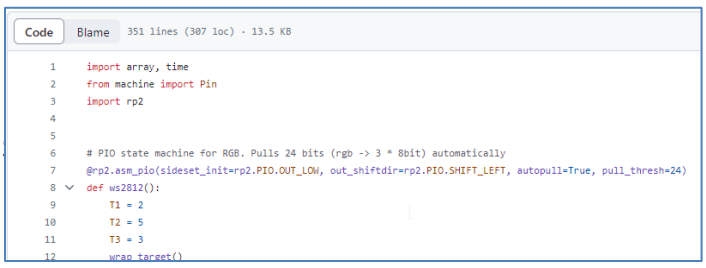
The **time** library is one we have used before.
It makes the program pause e.g. **time.sleep(1)** to pause for 1 second.
As you know, the time.sleep(1) command only works if we have already written 'import time' at the top of our program.
The **neopixel** library is code specifically written to help us control how the LED strip lights up. We now need to import the neopixel library so we can use new commands to control the LED strip.

Get the code for the neopixel library from this website:
https://github.com/blaz-r/pi_pico_neopixel
When you click the link above it will take you to a GitHub web page.
(git pages allow people to share code).

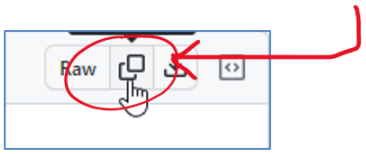


Once the webpage opens, click this link:

You will see code that starts like this:

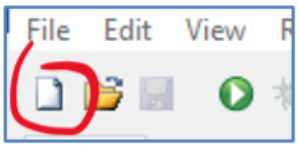


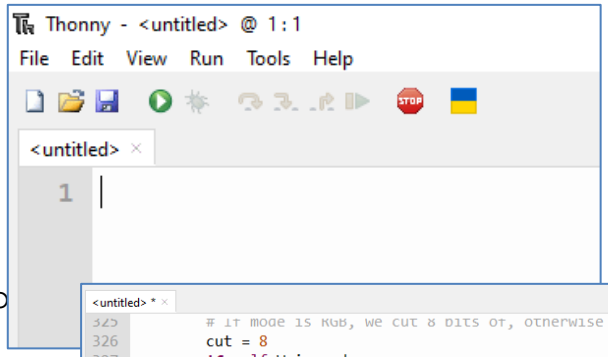
Copy the neopixel code using this button:



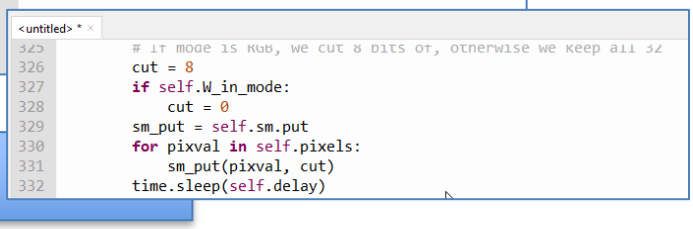
Now go back to the Thonny app 

Create a new page :

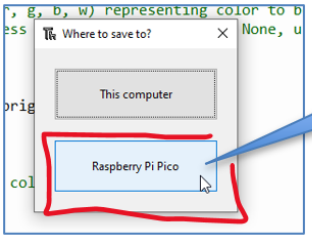




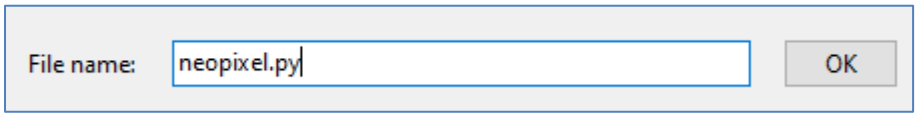
Paste in the copied co



Click File / Save as…



Save it to the Pico as neopixel.py

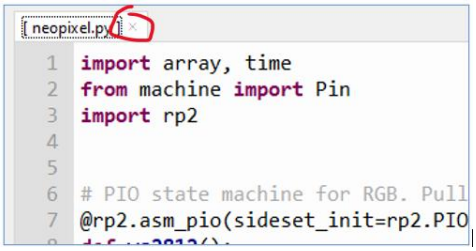


Make sure the name is spelled correctly.

Make sure you are saving it to the Pico and not the computer.

You have now installed the neopixel library onto the Pico device.

You can close this tab now:

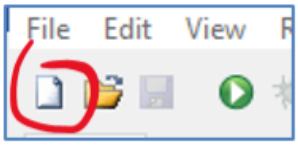


Programming the LED strip

Ok, so we have now added the neopixel.py file.

We can now start to code our LED strips.

Create a new Thonny page:



This page is where we will be writing the commands that control the LED strip.

Define Libraries

Write these commands to import the following libraries for use in your program:

```
1 from machine import Pin
2 import time
3 from neopixel import Neopixel
4
```

LED Variables

Now create a '**numpix**' variable.
This identifies how many pixels are on your LED strip:

```
5 numpix = 14
```

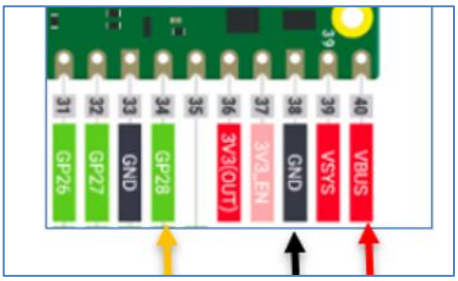
Note that I have 14 LEDs in my LED strip.
You may only have 10 LEDs on the LED strip in the test kit you have been given.
Or if you are working on a project at home you may have many more (see further information on powering longer strips at the end of this guide).

The '**strip**' variable identifies the type of LED strip you have.

The number 28 indicates the Pin sending the control signals (GP28)

```
6 strip = Neopixel(numpix, 0, 28, "RGB")
```

Here is a reminder of the PinOut diagram identifying the Pins used.



You should already have set up a jumper wire going from GP28 on the Pico to the 'Din' connection on the LED strip.

Set the strip brightness to 42.

```
7  strip.brightness(42)
```

We will keep this simple for now but, later on, we can make LEDs pulse using this variable.

This is how the program looks so far:

```
1  from machine import Pin
2  import time
3  from neopixel import Neopixel
4
5  #variables for the LED strip
6  numpix = 14
7  strip = Neopixel(numpix, 0, 28, "RGB")
8  strip.brightness(42)
```

Colour Variables

For our LED strip to flash different colours, we need to define colour variables.
This is the variable 'red' with 3 numbers assigned to it.

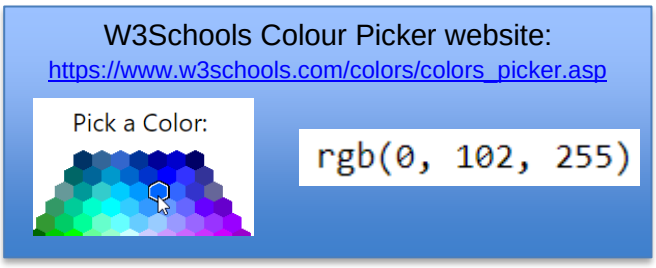
```
9  red = (255, 0, 0)
```

The three sets of numbers in the brackets indicate the Red Green Blue colour values.

You can have several colour variables.
Here I have made one for 'red' and 'blue':

```
9  red = (255, 0, 0)
10 blue = (0, 102, 255)
```

You should now add a couple more.
Use this colour picker website to add more colour variables.
Try adding colour variables such as pink, orange and yellow.



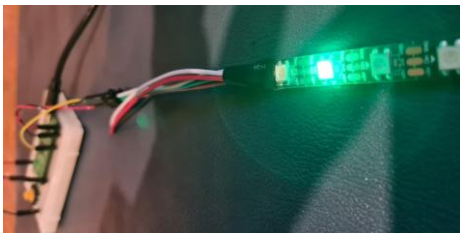
First Program for our LED strip

Ok, we are finally ready to make some LEDs light up.
Write these commands:

```
12  strip.set_pixel(1, red)
13  strip.show()
```

Click 'run':

This is what my LED strip looks like:



Yay! It lives!

Getting the wrong colour?

But wait… should it not be red?

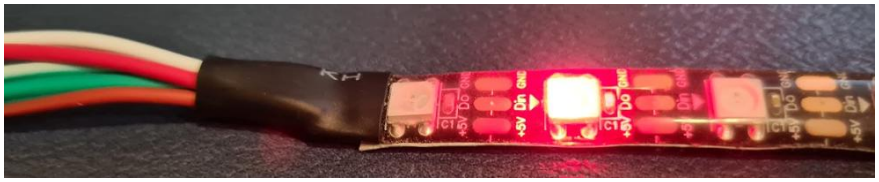
These RGB values for my 'red' variable *are* correct `red = (255, 0, 0)`
You do not have to take my word for it.
Check the colour picker website to see if the values in the brackets are correct:
https://www.w3schools.com/colors/colors_picker.asp

The reason you may be getting a green output  instead of a red one is the way the LED strip has been made. Sometimes different brands of LED strips are wired in a slightly different way.

You can easily fix this colour issue by going back to the **strip** variable line of code.
Notice that instead of "RGB" I have swopped it, so it now shows "GRB":

```
7  strip = Neopixel(numpix, 0, 28, "GRB")
```

Try stopping and re-running the program to see if the LED is now a red colour.



Not lighting up the first LED in the strip?

Wait… didn't we want the ***first*** pixel in this strip to light up?
Note that the 2nd LED in the strip has lit up, not the first in the strip.

This is because they are numbered starting from 0.
Our command identified pixel number 1, which is the 2nd in the LED strip.

```
12  strip.set_pixel(1, red)
```

Try changing it to a 0 to light up the first LED in the strip.

Test code:

If you are having issues, try copying this test code into Thonny.

```
from machine import Pin
import time
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "RGB")

#colour variables
red = (255, 0, 0)

#commands
strip.set_pixel(1, red)
strip.show()
```

Challenge 1 – light individual LEDs:

- Make additional individual LEDs light up.
- Use a range of colour variables.

You can write commands to control individual LEDs like this.

```
12  strip.set_pixel(0, red)
```

Try using different colour variables in each command.

```
strip.set_pixel(0, red)
strip.set_pixel(1, blue)
strip.set_pixel(2, green)
strip.show()
```

Troubleshooting:

- Do not forget the **strip.show()** command.
- Make sure you have set the RGB values for each colour variable

```
green = (102, 255, 153)
```

- Stop the program in Thonny, then run it again.
- If you cannot see the Pico device in the bottom right corner of Thonny, try a different USB port

Challenge 2 – using strip.set_pixel_line(1,5,red):

- Make half the LEDs in the strip light up with one colour variable.
- Use the time.sleep(1) command to add a delay
- Then output another colour variable

The **strip.set_pixel_line()** command allow

```
10 blue = (0, 102, 255)
11
12 strip.set_pixel_line(1,5, red)
13 strip.show()
```

Here I have added a delay of 1 second and then made LEDs 0 to 5 switch from red to blue:

```
12 strip.set_pixel_line(0,5, red)
13 strip.show()
14 time.sleep(1)
15 strip.set_pixel_line(0,5, blue)
16 strip.show()
```

Challenge 3 – using strip.fill(red):

- Make all the LEDs in the strip light up using a single command.
- Create a **blank** colour variable to make all of the colours disappear.
- Use time.sleep(1) to make the LED strip flash different colours

You can make the *whole* strip light up using this command:

```
12
13  strip.fill(red)
14  strip.show()
```

Sometimes you want to make pixels switch off.
I have created a 'blank' variable with 0 for RGB values:

```
 9  red = (255, 0, 0)
10  blue = (0, 102, 255)
11  blank = (0, 0, 0)
```

You can use the 'blank' variable to turn off LEDs.
Here I make the whole LED strip shine red, then go blank, then shine blue:

```
13  strip.fill(red)
14  strip.show()
15  time.sleep(0.5)
16  strip.fill(blank)
17  strip.show()
18  time.sleep(0.5)
19  strip.fill(blue)
20  strip.show()
```

You can also use the 'blank' variable in code to make a red pixel move along the strip:

```
13  strip.set_pixel(0, red)
14  strip.show()
15  time.sleep(0.5)
16  strip.set_pixel(0, blank)
17  strip.set_pixel(1, red)
18  strip.show()
19  time.sleep(0.5)
20  strip.set_pixel(1, blank)
21  strip.set_pixel(2, red)
22  strip.show()
```

Challenge 4 – using strip.set_pixel_line_gradient():

- Use the gradient command to make the LED strip blend two colours.
- Use time.sleep() to make the strip reverse after 2 seconds.

Here is an example of the gradient command:

```
11  blank = (0, 0, 0)
12
13  strip.set_pixel_line_gradient(0, 13,red, blue)
14  strip.show()
```

This test code is not the full solution.
How could you develop this to make the gradient reverse?

Challenge 5 – while loop:

- Use a While Loop to make a series of commands flash up colours one after the other.

```
14 #commands
15 while True:
16     strip.fill(red)
17     strip.show()
18     time.sleep(0.5)
19     strip.fill(blue)
20     strip.show()
21     time.sleep(0.5)
```

While loops will loop forever (or until the condition has been met).

Challenge 6 – For Loop:

- Use a For Loop to make the LED strip fill with one colour, then another

Here I have used a for loop to make the LED strip light up red for a second, then go blank.
It loops 5 times.

```
13 for x in range(5):
14     strip.fill(red)
15     strip.show()
16     time.sleep(1)
17     strip.fill(blank)
18     strip.show()
19     time.sleep(1)
```

Challenge 7 – For Loop, using X to select LED:

- Use a For Loop to make each LED light up one after the other

In the first challenge we used commands like this to control individual LEDs.

```
strip.set_pixel(0, red)
```

We can use the **x** in the for loop to change which LED lights up on each loop.

```
14  #commands
15  for x in range(5
16      strip.set_pixel(x, red)
17      strip.show()
18      time.sleep(0.5)
19
```

As X increments, this variable is used to select the next LED in the strip.

Can you develop to use a colour other than red?
Can you make the entire strip light up, one after the other?
Then, can you make each LED go blank?

Challenge 8 – LED sweep from left to right:

- Use a For Loop to increment which individual LED lights up.
- As the 2nd LED lights up, make the first LED go blank.
- When it gets to the end try to reverse the sequence.

Test Code:

```
from machine import Pin
import time
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")

#colour variables
red = (255, 0, 0)
blue = (0, 0, 255)
blank = (0, 0, 0)

#commands
for x in range(5):
    strip.set_pixel(x, red)
    strip.show()
    time.sleep(0.5)

    strip.set_pixel(x, blank)
    strip.show()

for y in range(5):
    strip.set_pixel(5-y, red)
    strip.show()
    time.sleep(0.5)

    strip.set_pixel(5-y, blank)
    strip.show()
```

Components used:

It may be you want to develop your own projects at home. Here is some information on the components used.

WS2812b type LED strip (also known as GlowBits or NeoPixels).
You can order from Amazon
https://www.amazon.co.uk/dp/B0BTCN9TZS?psc=1&ref=ppx_yo2ov_dt_b_product_details

You should be able to get a waterproofed strip of around 300 LEDs (5-meters) for approximately £25. Non waterproofed LED strips are available and may be appropriate depending on the application.

If the link above becomes "no longer available", you can order from other sources, but make sure they are the WS2812b type.

We will also be using a breadboard and a Raspberry Pico and some jumper wires, all which can be bought at The Pi Hut store.

https://thepihut.com/products/raspberry-pi-pico?variant=41925332566211

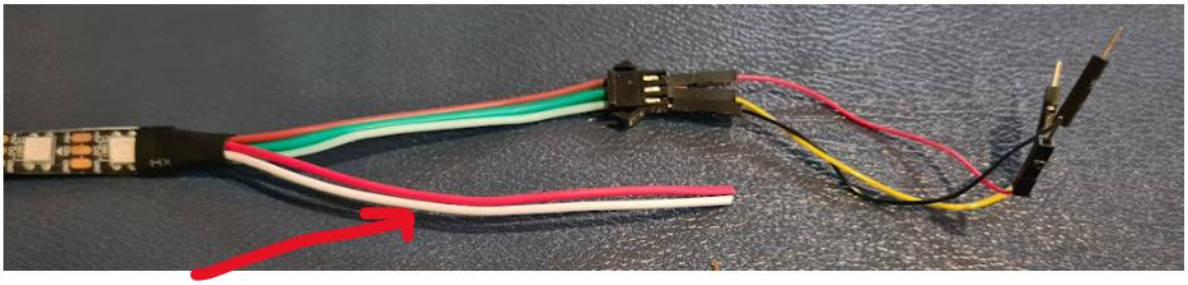
LED strip wire connections:

On the LED strip:

- red to 5v power wire
- yellow to signal wire
- black to GND wire

You can power up to 30 LEDs directly via the breadboard. If your LED strip is longer than this, see the sub heading further on in this guide.

If you are buying your own LED strip (pictured below) there are wires pre attached. Male to male jumper leads just connect to the end of the red, green and white wires.

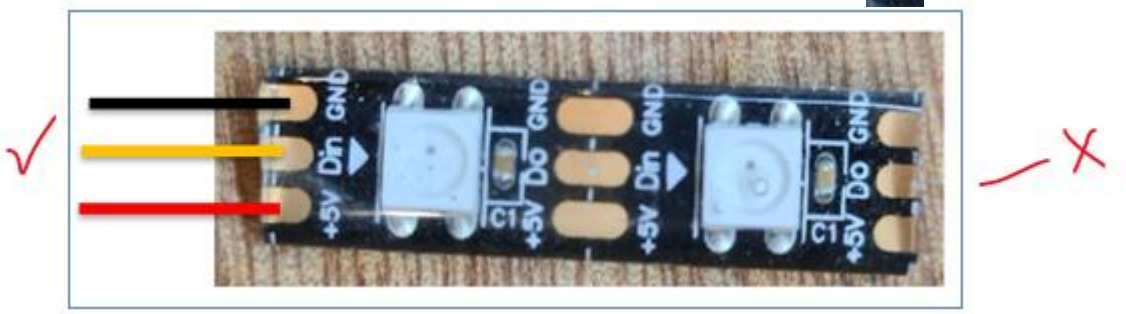


Note these unused red and white wires are for external power.

If you have been provided with a school project kit you will have a section of the LED strip with soldered wires:

Soldering your own connections:

If you are soldering connections, be careful to connect to the correct end of your LED strip. Look for the "Din" connection with the arrow next to it. 



If you connect your signal wire to the "Do" connection on the other end of the LED strip it will not light up.

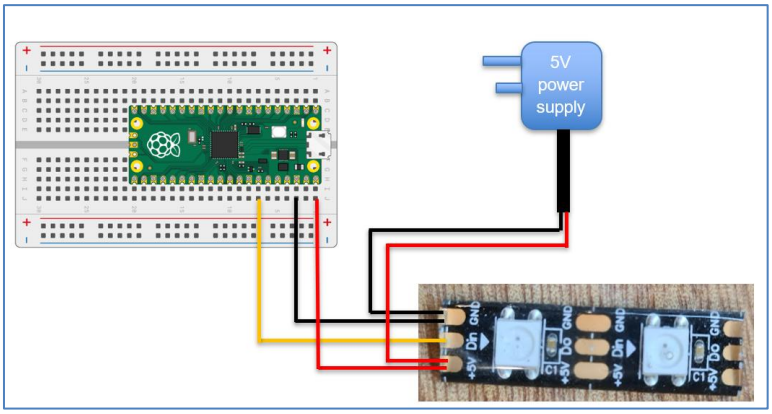
The + and – power connections can be connected anywhere along the strip.
If you have a strip length of 30 LEDs or less you can power it from the breadboard.

What if I have more than 30 LEDs?

If you are working with more than 30 LEDs in your strip you may not be able to provide enough power through the breadboard. I have used up to 100 LEDs without an issue though. It may be that prolonged use could burn out the components.

You will need to wire in an external power source for the LED strip.

If you have a strip length of *more* than 30 LEDs, it is recommended you connect a separate 5v power supply.



For really big projects you can have multiple power connections along the strip length. The LEDs do not care which direction the power is coming from.
But the signal pin does care! The yellow wire will follow the direction of the arrow.

If using a soldering iron, be careful of the fire risk, potential damage to tables and to yourself. This is a good kit I have used. It has an auto off feature and a good holder for when you need to put it down.
https://www.amazon.co.uk/gp/product/B077GB7CSZ/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1