Let it Glow Maker Advent Calendar Day #8: Rainbow Ring!

By The Pi Hut • Dec 8, 2023 • 0 comments

It's **day #8** of the Let it Glow Maker Advent Calendar!

Today we're playing with addressable LEDs again, but this time in the format of an **RGB LED ring** - it's another *The Pi Hut* team favourite!

These rings are a *very* popular LED format as you can make some really interesting patterns and effects with them, and even build them into projects as a multi-colour light source.

Enough talk - let's get straight into the blinky...

Warning: Some of today's activities contain fast flashing lights which may not be suitable for those with photosensitive epilepsy.

Box #8 Contents

In this box you will find:

- 1x 12-LED RGB Ring
- 3x Male to male jumper wires

Today's Activities

Today we're going to learn more about addressable LEDs and how we can chain them together to make some really fun light patterns.

We already know how to set up and use a single addressable LED (from day #6) so we'll be expanding on that whilst re-using some of the functions and tricks we've learnt so far.

First - a quick intro to the LED ring in today's box...

What is an RGB LED *ring*?

These LED rings are simply twelve addressable LEDs in a row, in a handy ring shape printed circuit board (PCB). Each LED is connected to the previous thanks to the IN and OUT data pins these LEDs have.

This means we can control all twelve of them, individually, using just a single GPIO pin from our Pico.

You already know about RGB and all that fun stuff from day #6 - **this ring is standard RGB, not GRB** - so let's jump straight into showing you how to control your fancy new LED ring!

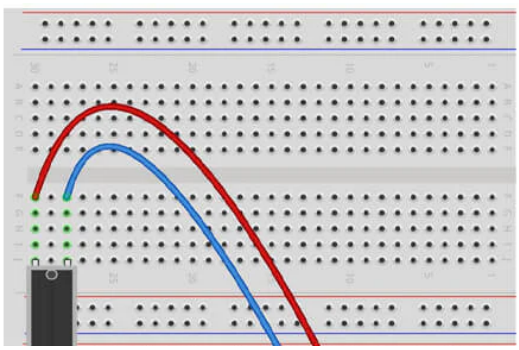
Construct the Circuit

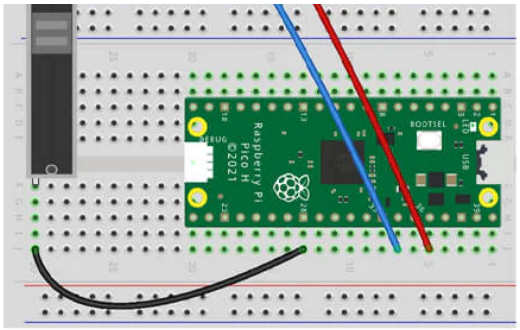
As always, make sure your Pico is disconnected from the USB cable when working on the circuit.

Prepare the breadboard

We're not using any other components today (the ring demands our full attention!) but we'll leave the slider in place from yesterday because we think you might want to play with that later...

Here's your starting point:





Connect the LED Ring

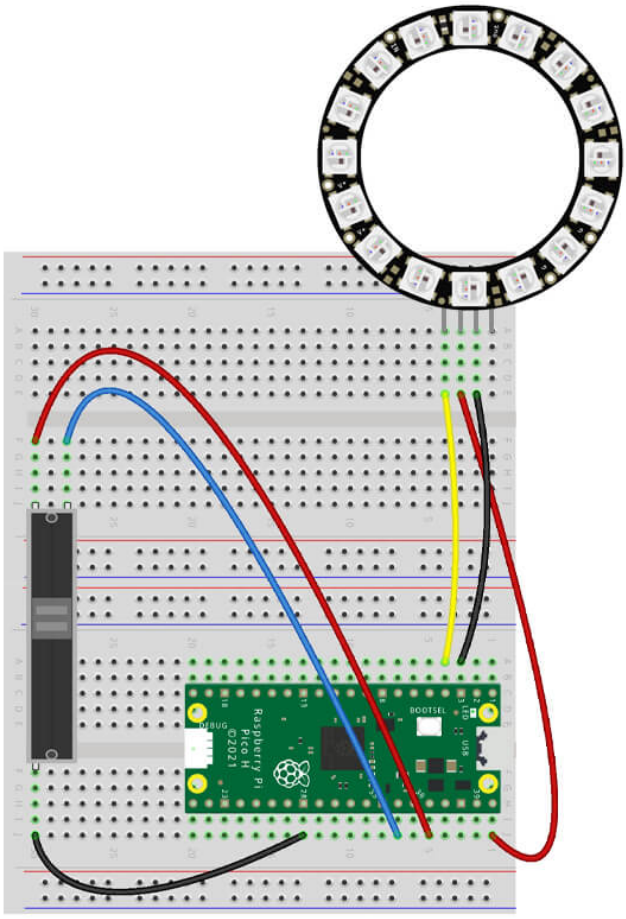
This is a nice easy component to connect, with fixed legs and only three wires to hook up.

First we need to **fit the ring facing forwards** at the top-right corner of your breadboard. Just slot it in, easy!

Then we need to wire up the **first three pins only** (we **don't connect the fourth pin** as this is the Data OUT for daisy-chaining with other addressable LEDs):

- The **first (left) pin** is **Data IN**. We connect this to **GPIO 2 (physical pin 4)**
- The **second pin** is **5V**. We connect this to **VBUS (physical pin 40)**
- The **third pin** is **GND**. We connect this to **any GND pin** (we've used **physical pin 3)**

You should have something like this:



***Tip**: Take a look at the rear of the LED ring - it has the pins conveniently labelled on the PCB! Not all components come with such luxuries...*

Activity 1: Single LED control

A quick first activity to show you the minor difference you need to make in your LED ring code compared to the single RGB LED in day #6.

We still **import** the **neopixel library**, and we set up the ring in the same way, however the setup line now needs to be altered to reflect how many LEDs are in the chain. We'll now use **ring = NeoPixel(Pin(2), 12)** which is specifying **GPIO2** for data in, and **12** LEDs in the ring. Simple!

Then to tell our code which LED in the ring we would like to light up, we can simply send **ring[0] = (10,0,0).** This is going to light up the first LED, as the **0** is specifying the first LED in the index (remember, **indexes start at zero**).

***Note**: Do you remember from day #6 how the RGB value changed the intensity of the light? These LED rings can be VERY bright, so **for most examples we're going to stick to low intensity** (around 10) to save the strain on your eyes!*

Run the code example below, and try changing that ring[**0**] number to a different value (up to **11**):

```
# Imports
from machine import Pin
from neopixel import NeoPixel

# Define the ring pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Select the first LED (0)
# Set the RGB colour (red)
ring[0] = (10,0,0)

# Send the data to the ring
ring.write()
```

If you need to turn it off...

If you want to turn off the LEDs at any stage, just set the RGB colours to (0,0,0) for the entire ring with **ring.fill((0,0,0))** like the example below.

We'll be adding this to the start of the example programs to 'clean up' any LEDs still on from the previous program:

```
from machine import Pin
from neopixel import NeoPixel

ring = NeoPixel(Pin(2), 12)

ring.fill((0,0,0)) # 'fill' addresses all LEDs
ring.write()
```

Activity 2: Multiple LEDs

Lighting up multiple LEDs is super easy too - just add another line for each LED you want to write.

In the basic example below we light up LEDs **1, 4, 7 and 10** blue, which in our index starting at zero, is **0, 3, 6 and 9**:

```
# Imports
from machine import Pin
from neopixel import NeoPixel
```

```
import time

# Define the ring pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

# Send data to four LEDs
ring[0] = (0,0,10)
ring[3] = (0,0,10)
ring[6] = (0,0,10)
ring[9] = (0,0,10)

ring.write()
```

...or use a list!

You know how **lists** work - we learnt about them on day #4. We can create a list of the specific LEDs in the index we want to light up, then refer to that list in our code.

The example below achieves the same outcome as the one above, just using a **list** and **for loop** instead. Sometimes one method will be better than another depending on the project/situation:

```
# Imports
from machine import Pin
from neopixel import NeoPixel
import time

# Define the ring pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Create a list of the LEDs to use
myleds = [0,3,6,9]

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

for i in myleds:
    ring[i] = (0,0,10)
    ring.write()
```

Activity 3: A little delay for effects

Adding a simple short delay can transform your LED code from simply 'lighting up' to a pattern!

Take the code above for example - if we add *just one* **time.sleep(1)** line at the end of the code inside our **for loop**, the LEDs will light one after the other.

This is because the **for loop** will run for each item in our **list**. Without a delay, it's so fast you don't realise it's happening and they look like they're tuning on all at the same time:

Give it a try:

```
# Imports
from machine import Pin
```

```
from neopixel import NeoPixel
import time

# Define the ring pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Create a list of the LEDs to use
myleds = [0,3,6,9]

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

for i in myleds:
    ring[i] = (0,0,10)
    ring.write()
    time.sleep(1)
```

Activity 4: Spinning lights with while loops

What if we add the code above into a **while True** loop that runs forever, reduce the time delay slightly, and also clear the ring after each iteration in the **for loop**?

That will give us the effect of an LED lighting one after the other in a constant cycle! Each LED in the **list** is lit up for as long as we set the delay, then cleared, then the next LED in the list does the same - over and over again (as the **while True** loop will restart the **for loop**).

Here's the code for that with some added commentary - give it a try!

```
# Imports
from machine import Pin
from neopixel import NeoPixel
import time

# Define the strip pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Create a list of the LEDs to use
myleds = [0,3,6,9]

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

while True:

    for i in myleds:

        ring[i] = (0,0,10)
        ring.write()

        # Show the light for this long
        time.sleep(0.3)

        #Clear the ring at the end of each loop
        ring.fill((0,0,0))
        ring.write()
```

Activity 5: Bouncing lights

You've used the **reverse** method a couple of times now, but let's use it again with our ring as it can help make some really cool effects.

Our light is going to bounce from one side to the other, simply by adding a second **for loop** with the LED sequence reversed.

The first **for loop** will work through the list, lighting each LED in turn, then once that loop has finished, it'll move on to the next part of the code which is our **second for loop** - it's pretty much identical apart from the **reversed function** making the LEDs run backwards.

As this is all in a **while True** loop, it'll keep bouncing back and forth - cool! Give it a try (*we also changed the colour*):

```
# Imports
from machine import Pin
from neopixel import NeoPixel
import time

# Define the strip pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

while True:

    for i in range(12):

        ring[i] = (5,0,5)
        ring.write()

        # Show the light for this long
        time.sleep(0.09)

        #Clear the ring at the end of each loop
        ring.fill((0,0,0))
        ring.write()

    for i in reversed (range(12)):

        ring[i] = (5,0,5)
        ring.write()

        # Show the light for this long
        time.sleep(0.09)

        #Clear the ring at the end of each loop
        ring.fill((0,0,0))
        ring.write()
```

Activity 6: The spinning wheel of colour!

Already sounds fun right?!

This example uses the same concepts as above, but we keep the LEDs lighting in a continuous cycle round and round again, and each time we get back to the first LED we change the colour.

We'll use the **random** method for the colours because we just think that's more fun than using a set list. We've also bumped up the max RGB intensity to **100** for this example, as it shows off the colours a bit better (*go to 255 if you want!*).

The code inside the **while loop** first sets some random numbers (**integers**) for our RGB colours (with a max range of **100**), then jumps into a **for loop** which then lights the LEDs in turn. Once the for loop is over, our **while True** loop starts the process all over again.

Here's the code:

```
# Imports
from machine import Pin
from neopixel import NeoPixel
import time
import random

# Define the strip pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

while True:

    # Create random RGB values
    r = random.randint(0,100)
    g = random.randint(0,100)
    b = random.randint(0,100)

    for i in range(12):

        # Light each LED a random colour
        ring[i] = (r,g,b)
        ring.write()

        # Show the LED for this long
        time.sleep(0.05)

        #Clear the ring at the end of each loop
        ring.fill((0,0,0))
        ring.write()
```

...or a full colour version!

Rather than having a **single** LED spinning in a circle, you can **fill** the ring with each colour as it goes round.

All you need to do is delete the last three lines of the previous example - go on, try it!

If the ring isn't cleared at the end of each loop, it'll fill with colour instead. Easy!

Activity 7: The double random ring!

Another neat example that doesn't use a lot of code is a program that selects a **random LED** from our ring, displays a **random colour** on it, then selects the next LED and so on. If you run this fast enough it looks really good!

The best bit? It's using knowledge and functions you've already learnt - great for practice!

Our **while loop** uses the random function to select an LED from our ring using the index (**0-11**), then creates

three random RGB colour variables between **0** and **50**.

All we do then is inject those variables into our usual code to drive the LED ring, using **ring[randomled] = (r,g,b)**. Give it a try:

```
# Imports
from machine import Pin, ADC
from neopixel import NeoPixel
import time
import random

# Define the strip pin number (2) and number of LEDs (12)
ring = NeoPixel(Pin(2), 12)

# Turn off all LEDs before program start
ring.fill((0,0,0))
ring.write()
time.sleep(1)

while True:

    # Select a random LED
    randomled = random.randint(0,11)

    # Create random RGB values
    r = random.randint(0,50)
    g = random.randint(0,50)
    b = random.randint(0,50)

    # Light the random LED in a random colour
    ring[randomled] = (r,g,b)
    ring.write()

    # Show the light for this long
    time.sleep(0.05)

    #Clear the ring at the end of each loop
    ring.fill((0,0,0))
    ring.write()
```

Day #8 Complete!

Phew! *Lots* of examples there, and there's a lot more you can do with these LED rings depending on how far you want to take the code. **Feel free to bump up the examples to the full 255 RGB brightness for some really impressive blinky!**

It was another good day for re-using things you've already learnt and using them in new, fun ways. We'll be doing more with our RGB LEDs before the calendar is over.

Note: *We didn't remove the slider from yesterday's box as we thought you might like to try your hand at coding that for a speed control, colour selector or...something else! All the knowledge and examples you need can be found in the previous box articles (or maybe, just maybe, we'll show you how in a future box...)*

It's recap time – what did we cover today?

- Wiring an RGB LED ring
- Referencing individual/multiple addressable LEDs in a chain
- Different light effects:
  - Simple individual light addressing

- Light chasing (single LED/full colour)
- Spinning lights
- Bouncing lights
- Random lights and colours

Leave your circuit as it is for now, we'll give you instructions on what to do with it at the start of tomorrow's box. You'll probably want to play some more with the LED ring before then anyway...

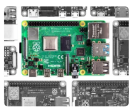
See you in the morn'!

We used Fritzing to create the breadboard wiring diagram images for this page.

Popular posts

Top 10 Raspberry Pi Pico Add-ons & Accessories

Top 10 Raspberry Pi 400 Accessories



Raspberry Pi Models

How to set up an SSD with the Raspberry Pi 4

Control your Raspberry Pi media centre with FLIRC

Leave a comment

All comments are moderated before being published.

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Name

E-mail

Message

SUBMIT

Related Posts

Coding the Waveshare RP2040 Matrix

Tony Goodhew • Sep 15, 2023

The Waveshare RP2040 Matrix development board is an inexpensive, super-compact RP2040 microcontroller with a tiny 5x5 addressable RGB LED matrix on the front. This tiny development board is packed with...

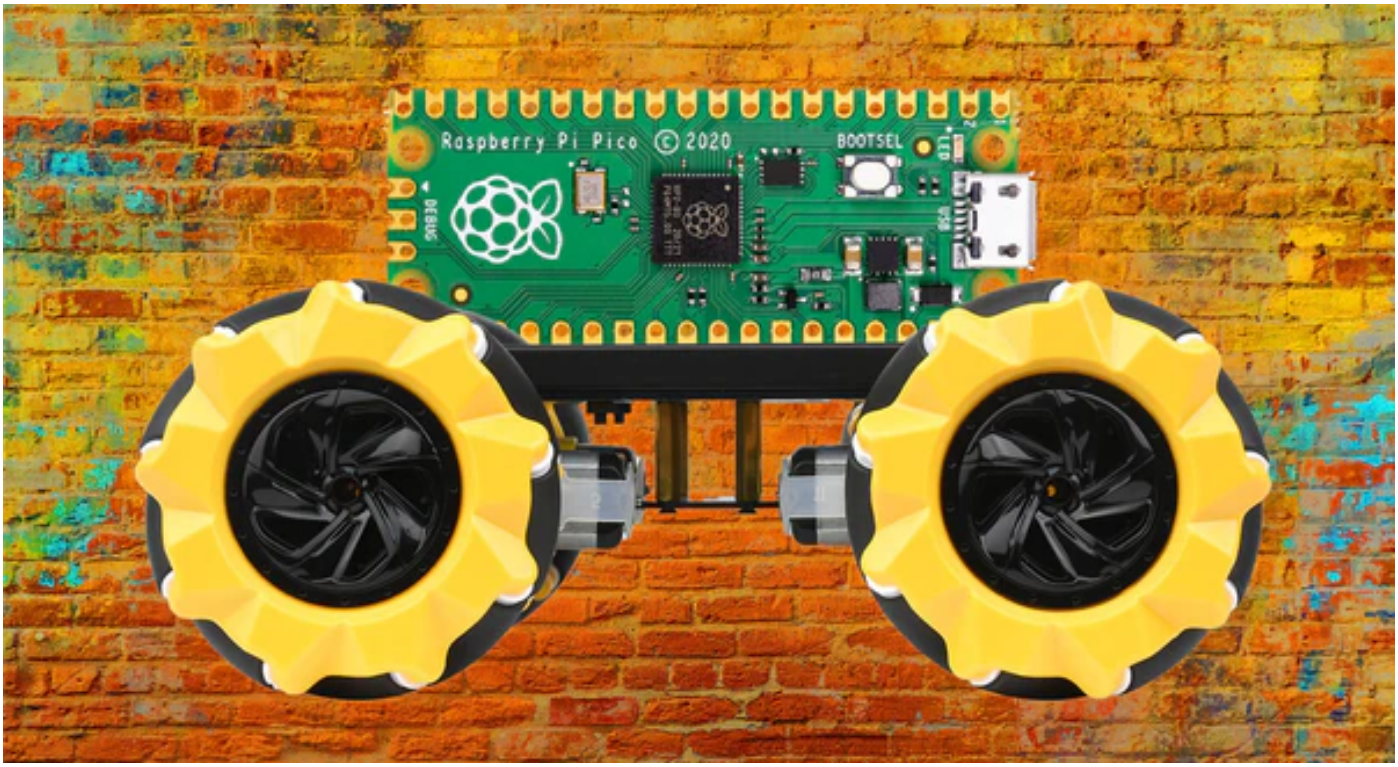
Read more

MicroPython Skill Builders - #6 Temperature & Humidity Sensors

Tony Goodhew • Aug 24, 2023

Welcome to another instalment in our MicroPython Skill Builders series by Tony Goodhew, aiming to improve your coding skills with MicroPython whilst introducing new components and coding techniques - using a Raspberry...

Read more



Build a Raspberry Pi Pico Robot with Mecanum Wheels

Tony Goodhew • Jul 24, 2023

In this tutorial, we'll show you how to build and code a Raspberry Pi Pico robot using mecanum wheels! Normal wheels allow the robot to move forwards, backwards, spin and...

Read more

Terms & Conditions

Delivery

Lithium Shipping

Pre-Orders

Privacy Statement

Policies

Terms of Service

Company Info

FAQ

Klarna FAQ

Quick Start Guide

Search

Support Portal

Our Store Sections

Raspberry Pi

Maker Store

micro:bit

Arduino

Gifts

Follow us

Raspberry Pi
Approved Reseller

We accept