Using Raspberry Pi to control LED

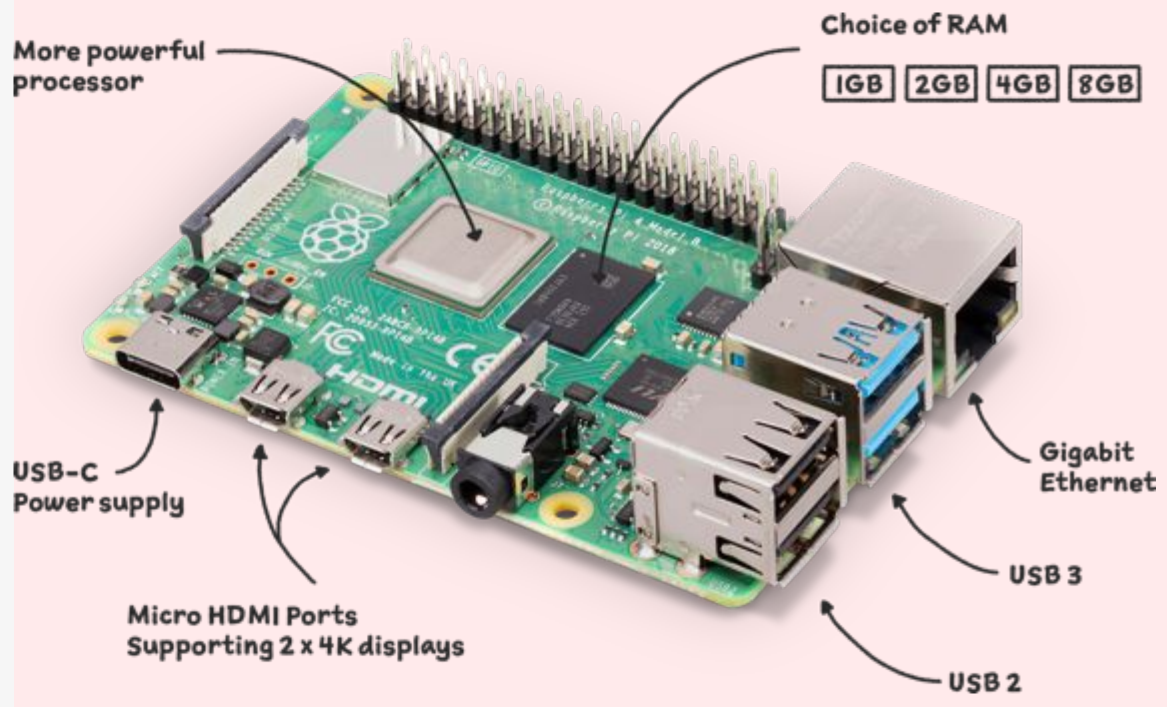
Chih-Yun Chang, Danyan Liao,
Yilin Wang, Shuwen Zhao

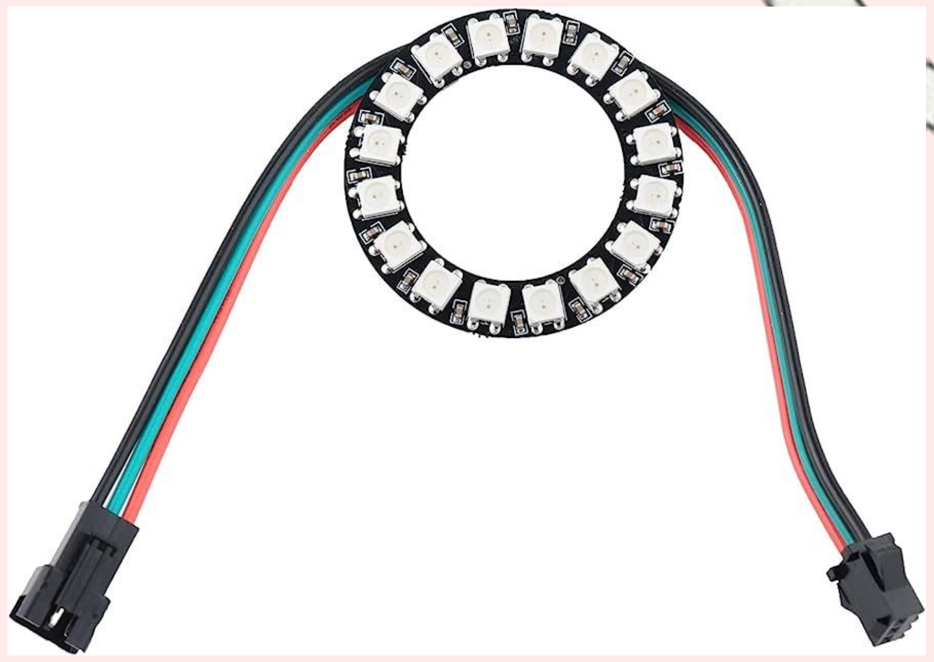
Project Abstract:

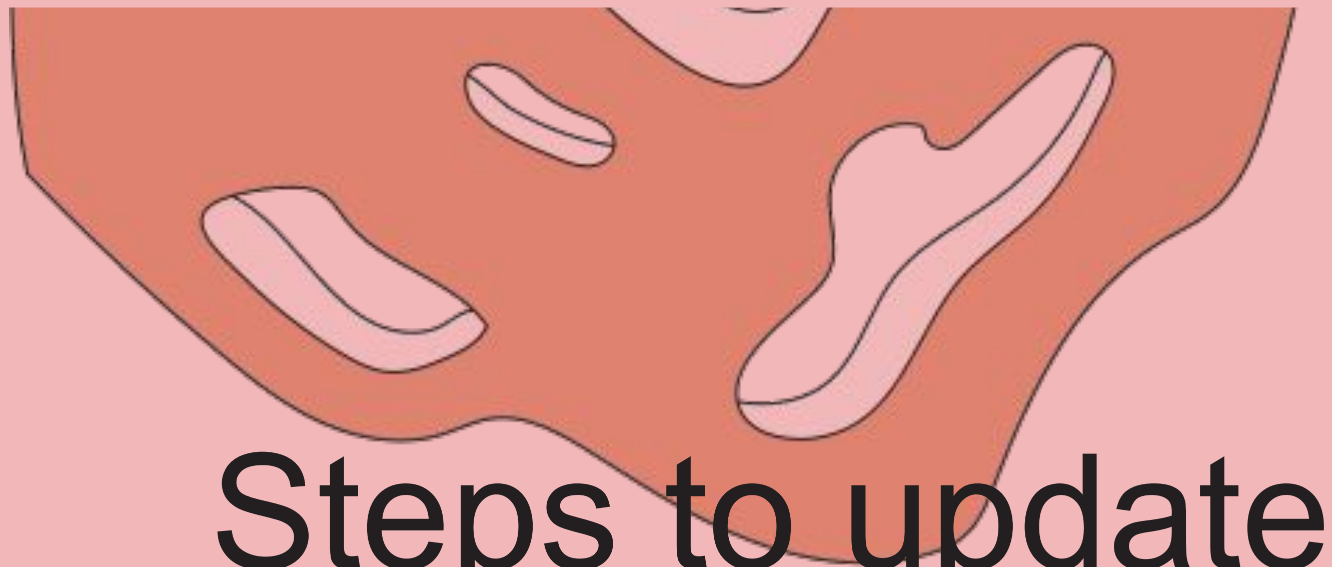
- Controlling a 16 RGB LED with a Raspberry Pi by using the C language knowledge learned.
- Project expected outcome: To control the different lights turning on and off by the power switch.

Required parts.

- Raspberry Pi : model 4
- 16 RGB LED Ring (WS2812)
- Power Switch with Indicator Light





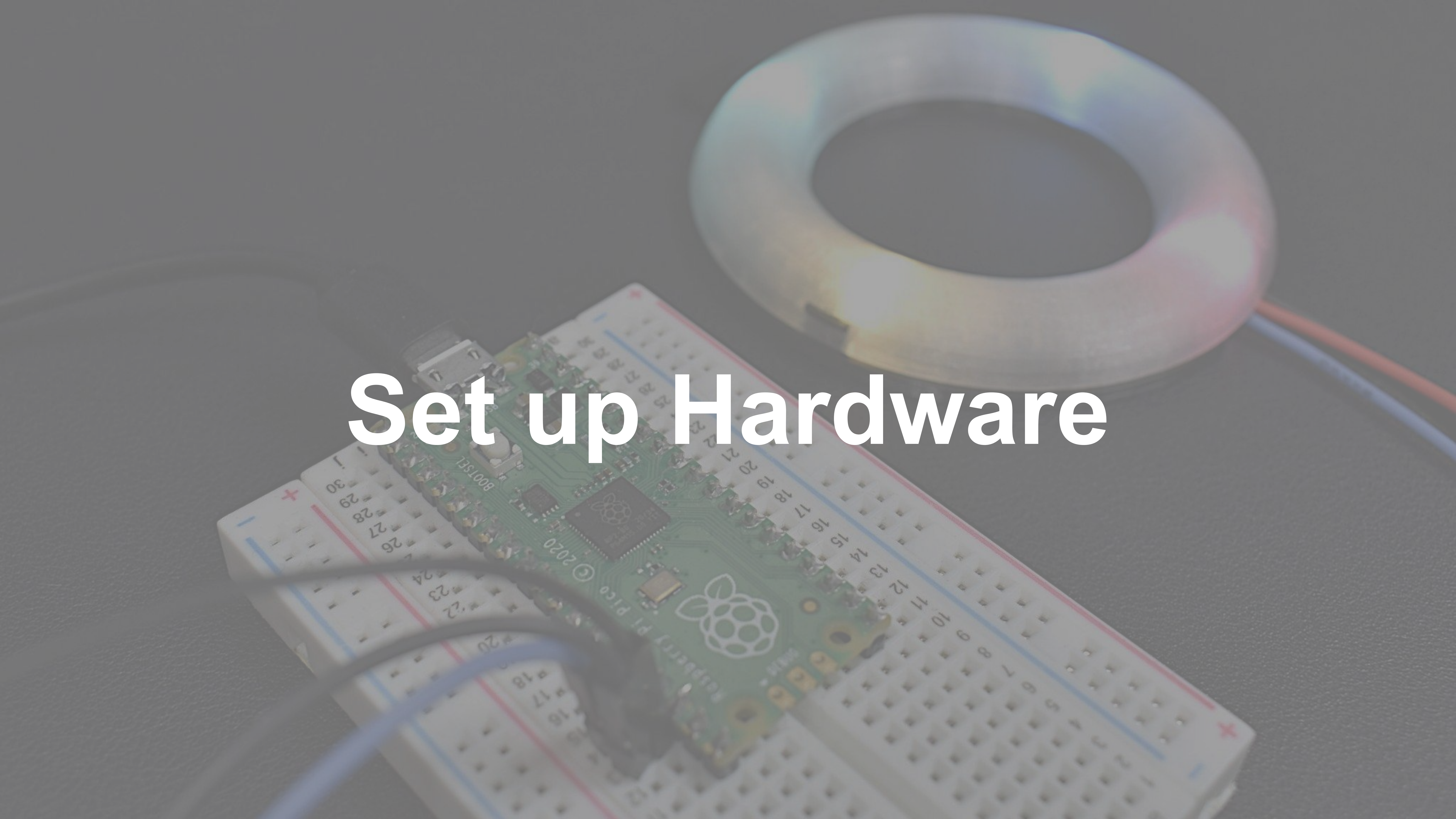


Steps to update Raspberry Pi:

- Connect the raspberry pi to a computer or access to terminal
- Open a terminal
- Update the package list

  -with the command “ sudo apt update”
- Upgrade the installed packages

  -with the command “ sudo apt upgrade”



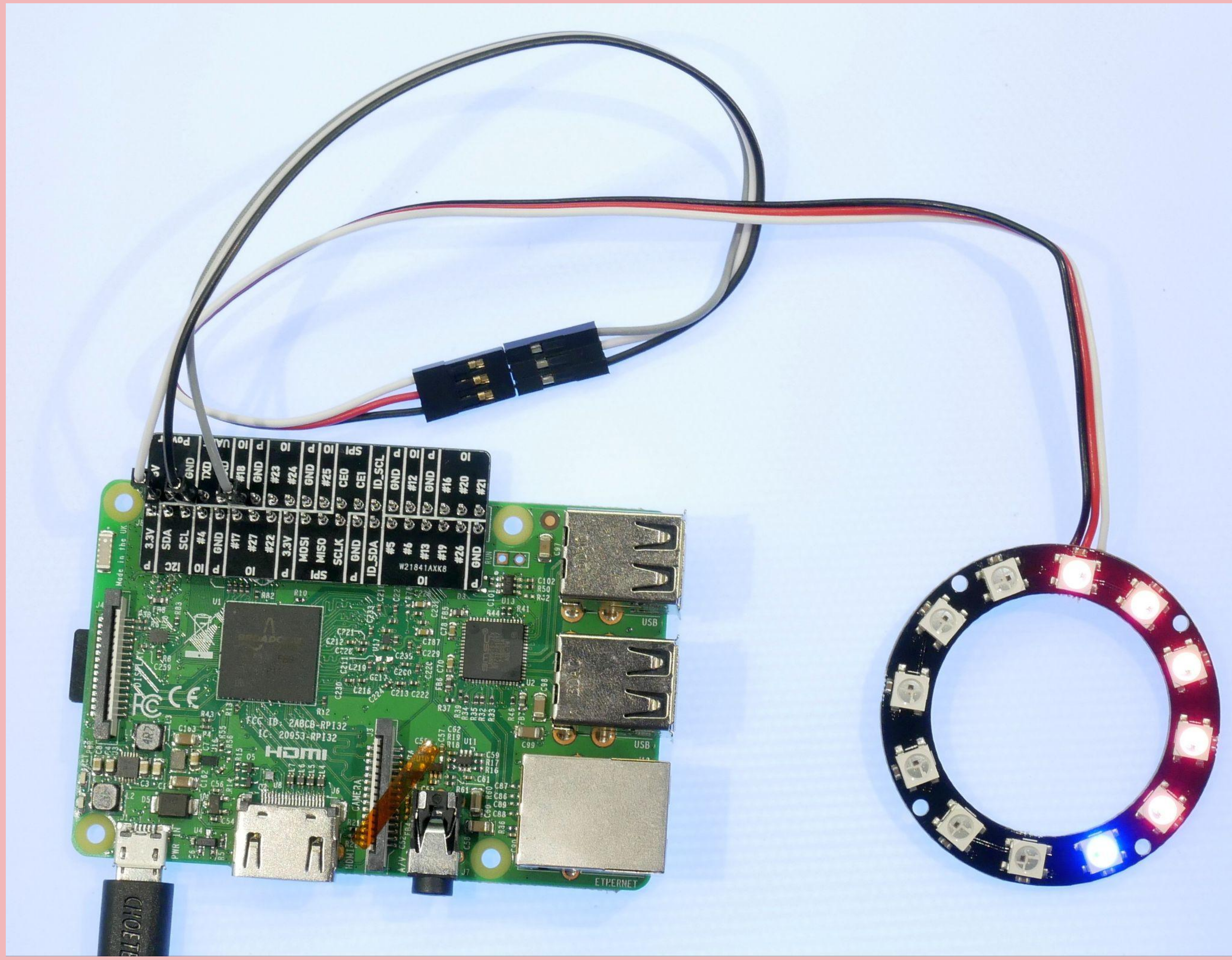
Set up Hardware

Three Steps to connect LED Ring to Raspberry Pi

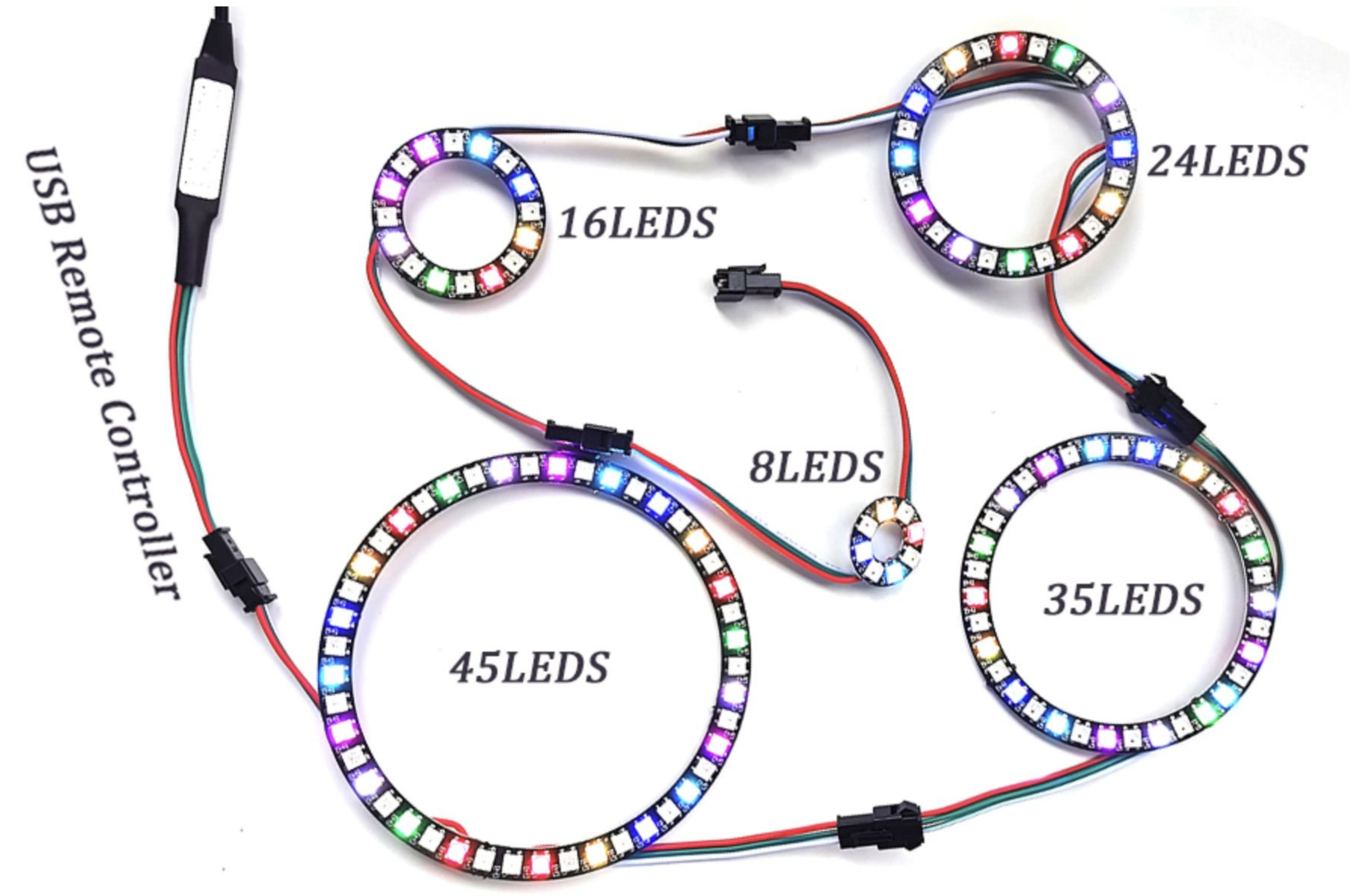
(1) Connect the 5V pin of the WS2812 ring to the 5V pin on the Raspberry Pi

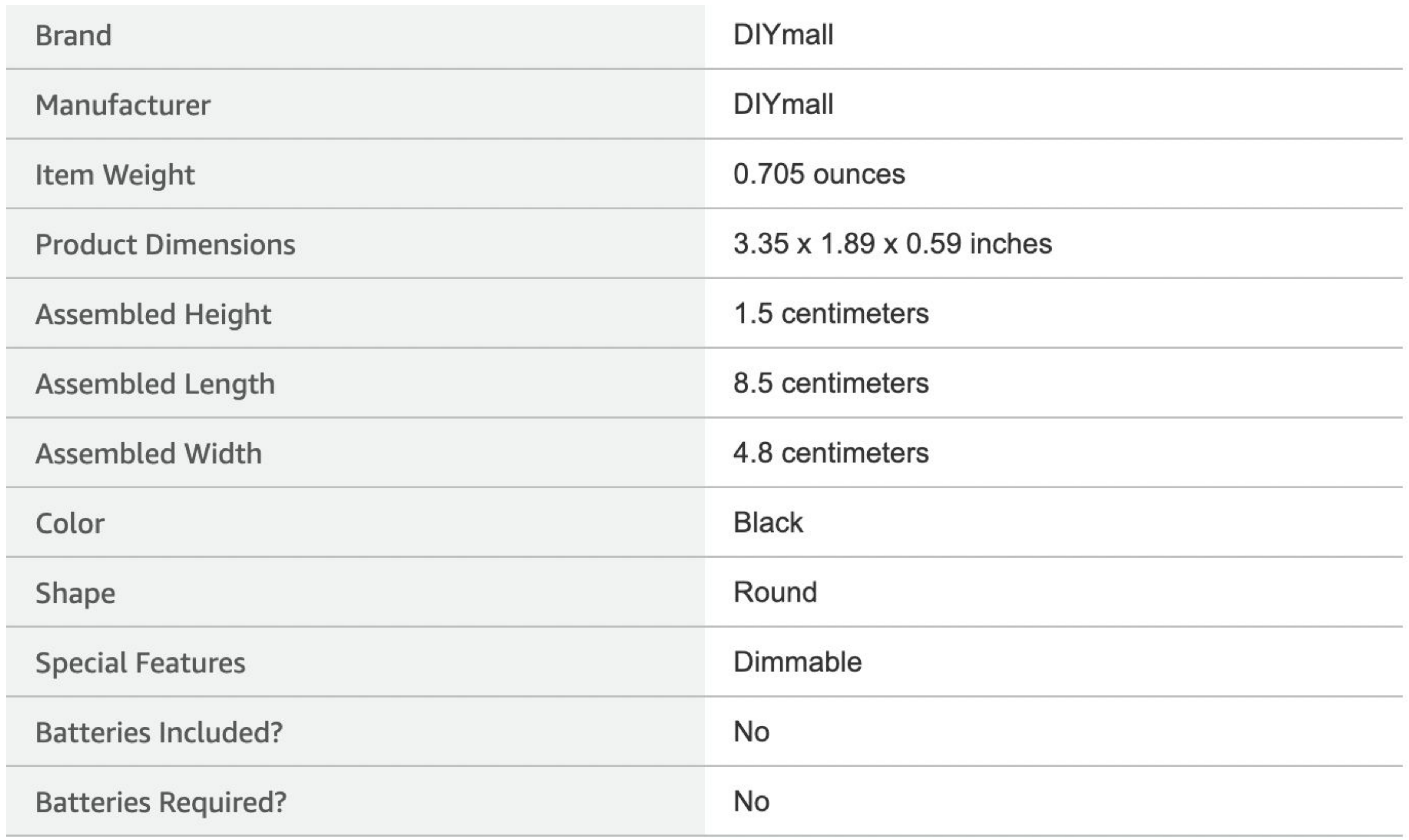
(2) Connect the **GND pin** of the WS2812 to any GND pin on the Raspberry Pi

(3) Connect the **DIN pin** of the WS2812 to a **GPIO pin** on the Raspberry Pi



45 RGB LED Ring - Descriptions

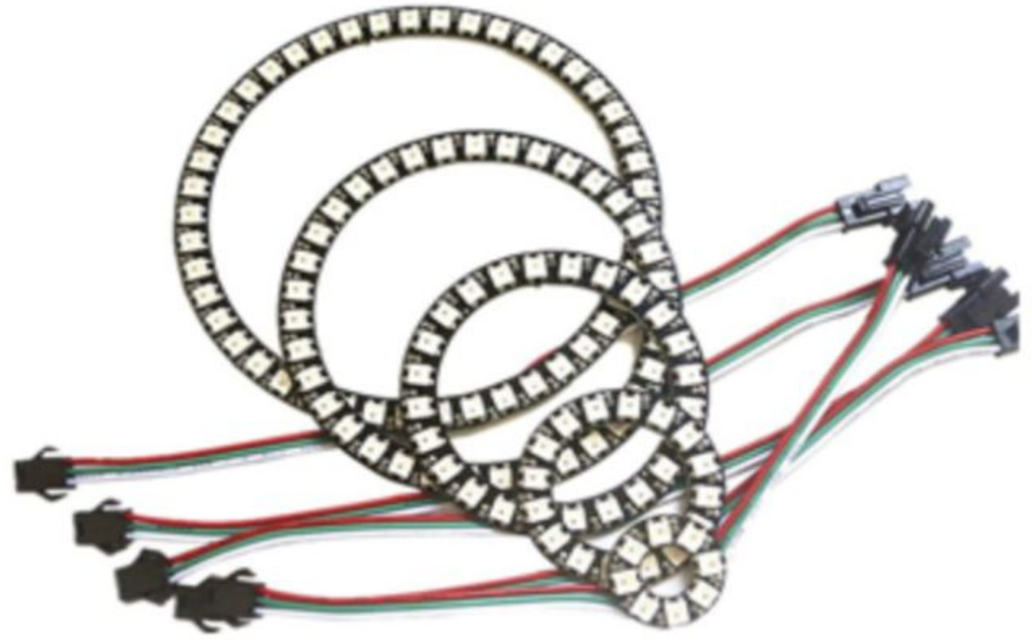


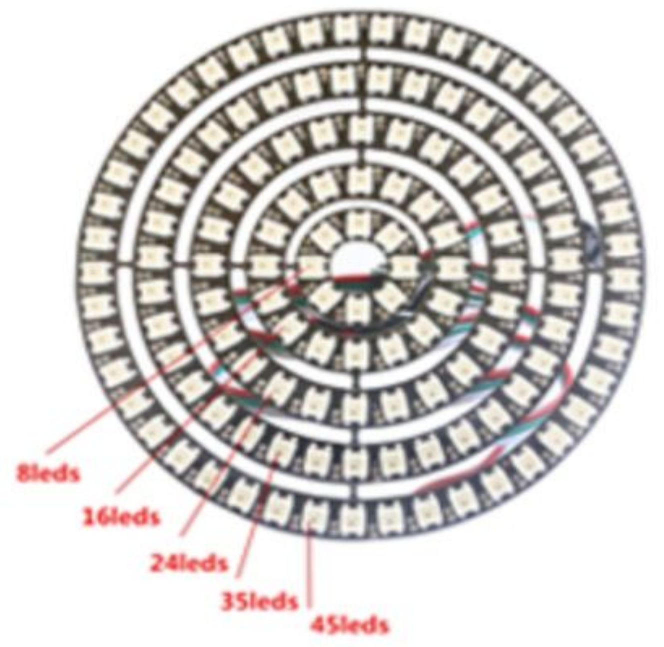


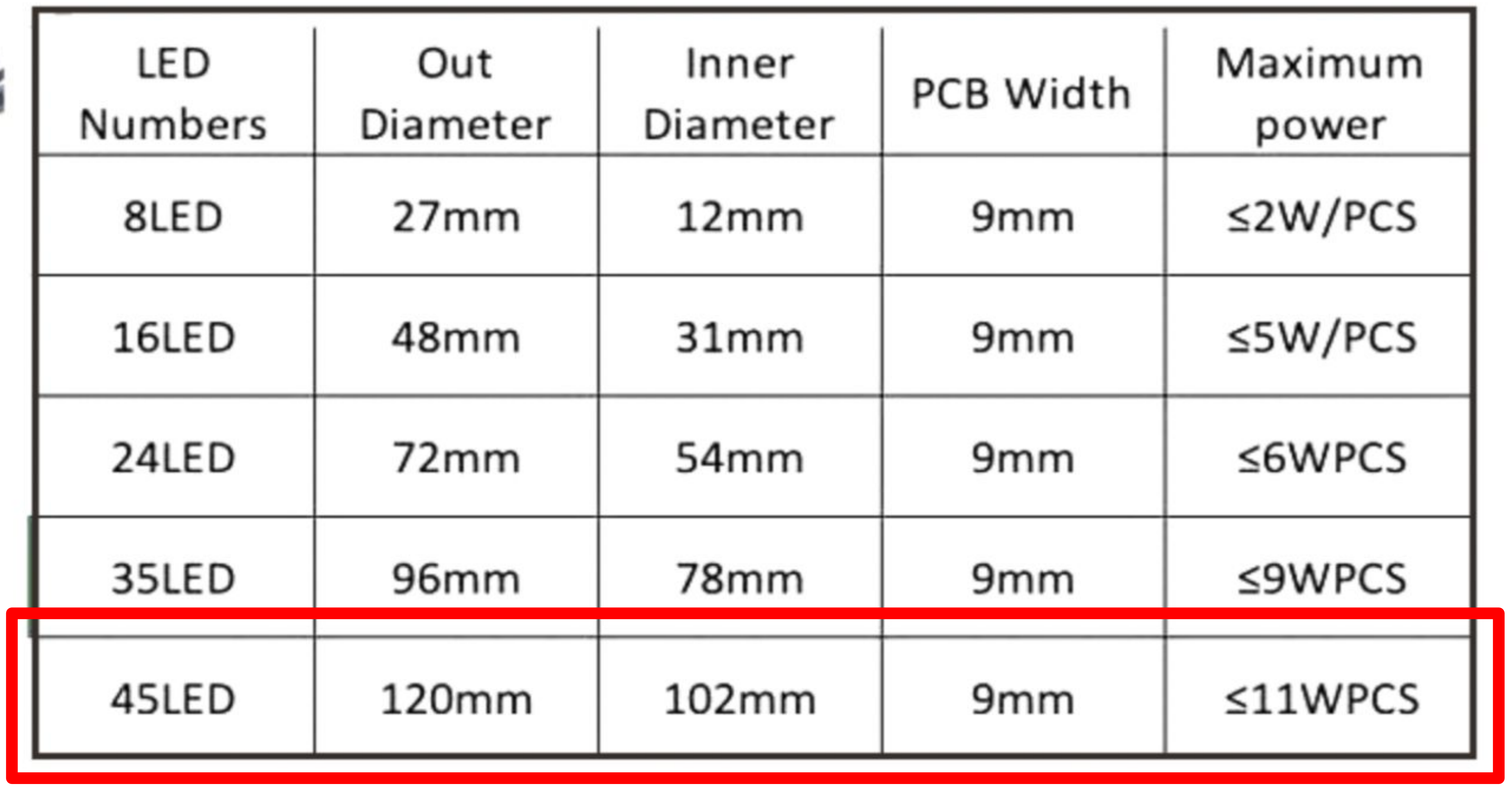
Brand	DIYmall
Manufacturer	DIYmall
Item Weight	0.705 ounces
Product Dimensions	3.35 x 1.89 x 0.59 inches
Assembled Height	1.5 centimeters
Assembled Length	8.5 centimeters
Assembled Width	4.8 centimeters
Color	Black
Shape	Round
Special Features	Dimmable
Batteries Included?	No
Batteries Required?	No

We're using this 45 LEDS

45 RGB LED Ring - Descriptions

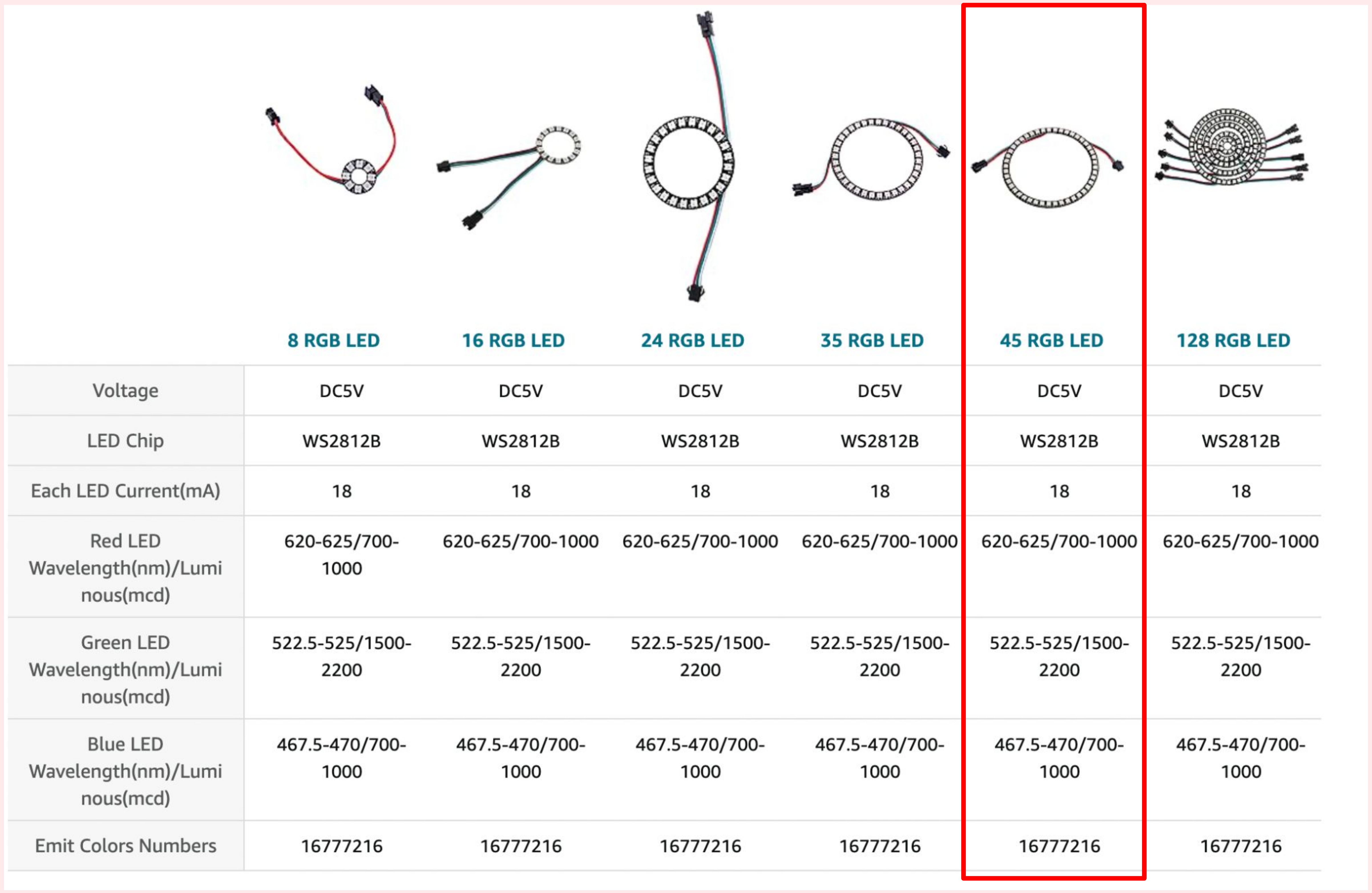






LED Numbers	Out Diameter	Inner Diameter	PCB Width	Maximum power
8LED	27mm	12mm	9mm	≤2W/PCS
16LED	48mm	31mm	9mm	≤5W/PCS
24LED	72mm	54mm	9mm	≤6WPCS
35LED	96mm	78mm	9mm	≤9WPCS
45LED	120mm	102mm	9mm	≤11WPCS

45 RGB LED Ring - Descriptions



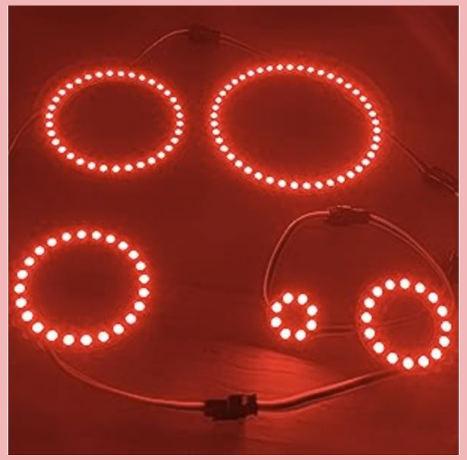
	8 RGB LED	16 RGB LED	24 RGB LED	35 RGB LED	45 RGB LED	128 RGB LED
Voltage	DC5V	DC5V	DC5V	DC5V	DC5V	DC5V
LED Chip	WS2812B	WS2812B	WS2812B	WS2812B	WS2812B	WS2812B
Each LED Current(mA)	18	18	18	18	18	18
Red LED Wavelength(nm)/Luminous(mcd)	620-625/700-1000	620-625/700-1000	620-625/700-1000	620-625/700-1000	620-625/700-1000	620-625/700-1000
Green LED Wavelength(nm)/Luminous(mcd)	522.5-525/1500-2200	522.5-525/1500-2200	522.5-525/1500-2200	522.5-525/1500-2200	522.5-525/1500-2200	522.5-525/1500-2200
Blue LED Wavelength(nm)/Luminous(mcd)	467.5-470/700-1000	467.5-470/700-1000	467.5-470/700-1000	467.5-470/700-1000	467.5-470/700-1000	467.5-470/700-1000
Emit Colors Numbers	16777216	16777216	16777216	16777216	16777216	16777216

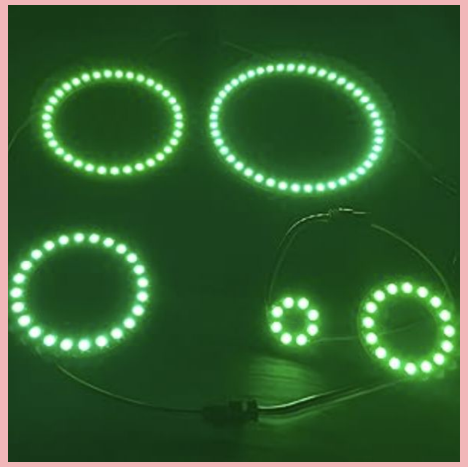
Availability: 45-LED rings is commercially available and affordable in the desired size, color, or configuration.

Compatibility: A 45-LED ring is compatible with the software or libraries being used in this project.

Power Consumption: 45 LEDs is a suitable number that doesn't draw too much power from the device.

45 RGB LED Ring - Full Color

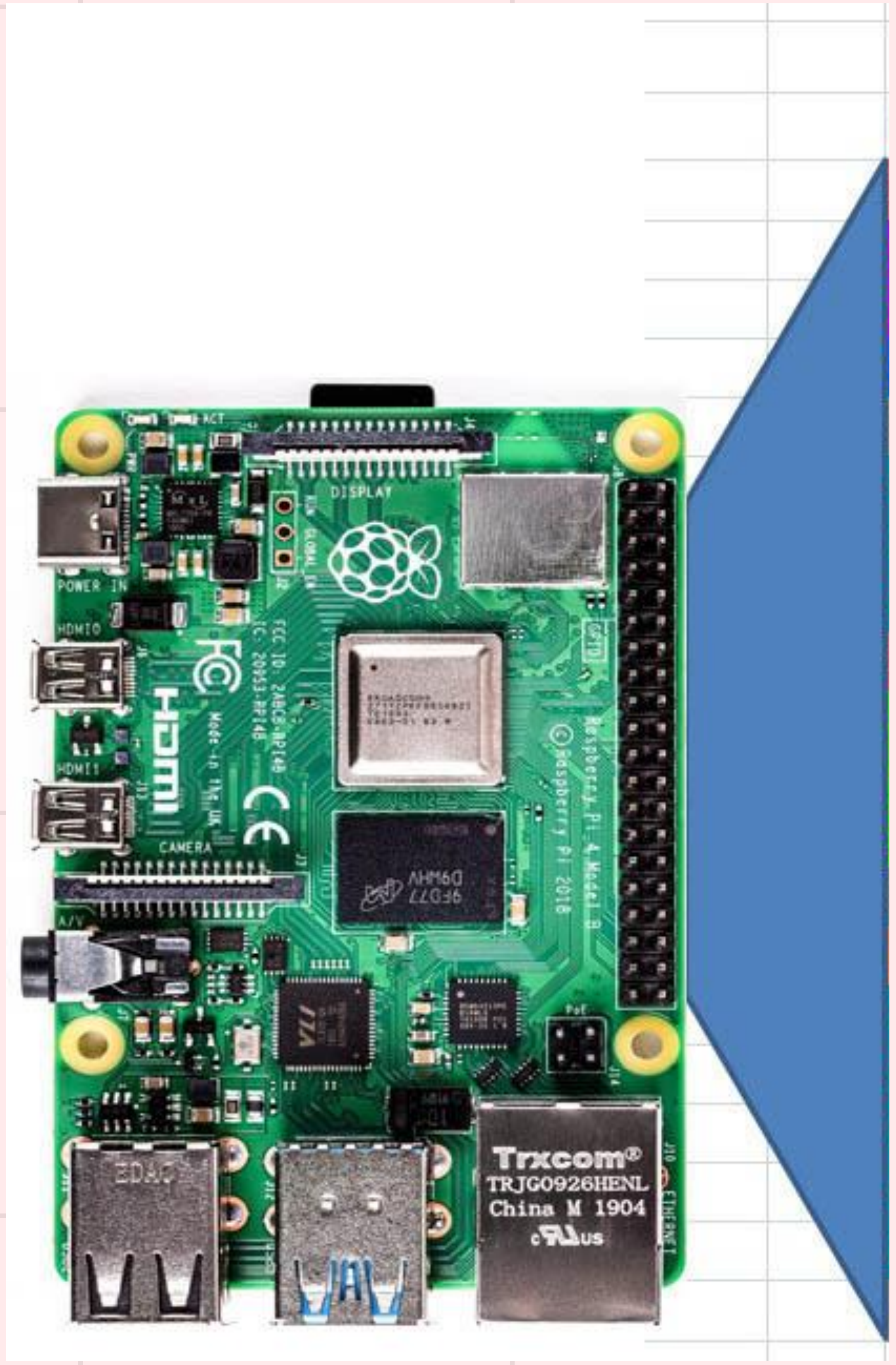


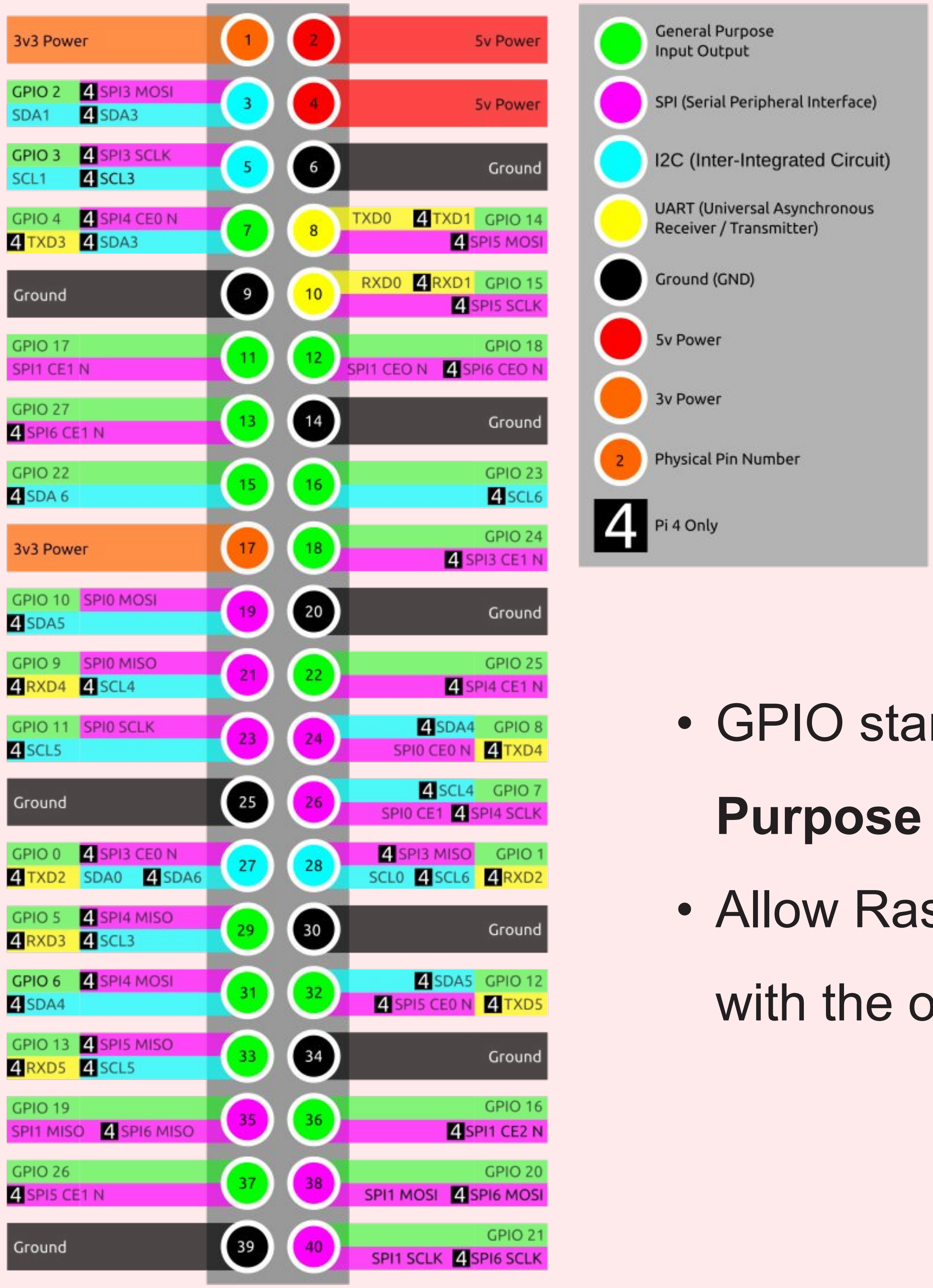


Full color refers to the ability to reproduce the entire range of colors that can be perceived by the human eye.

In digital devices like LED rings, full color is typically achieved through the combination of **Red, Green, and Blue (RGB) light**, allowing for the display of thousands or even millions of distinct colors.

What is a GPIO Pin?

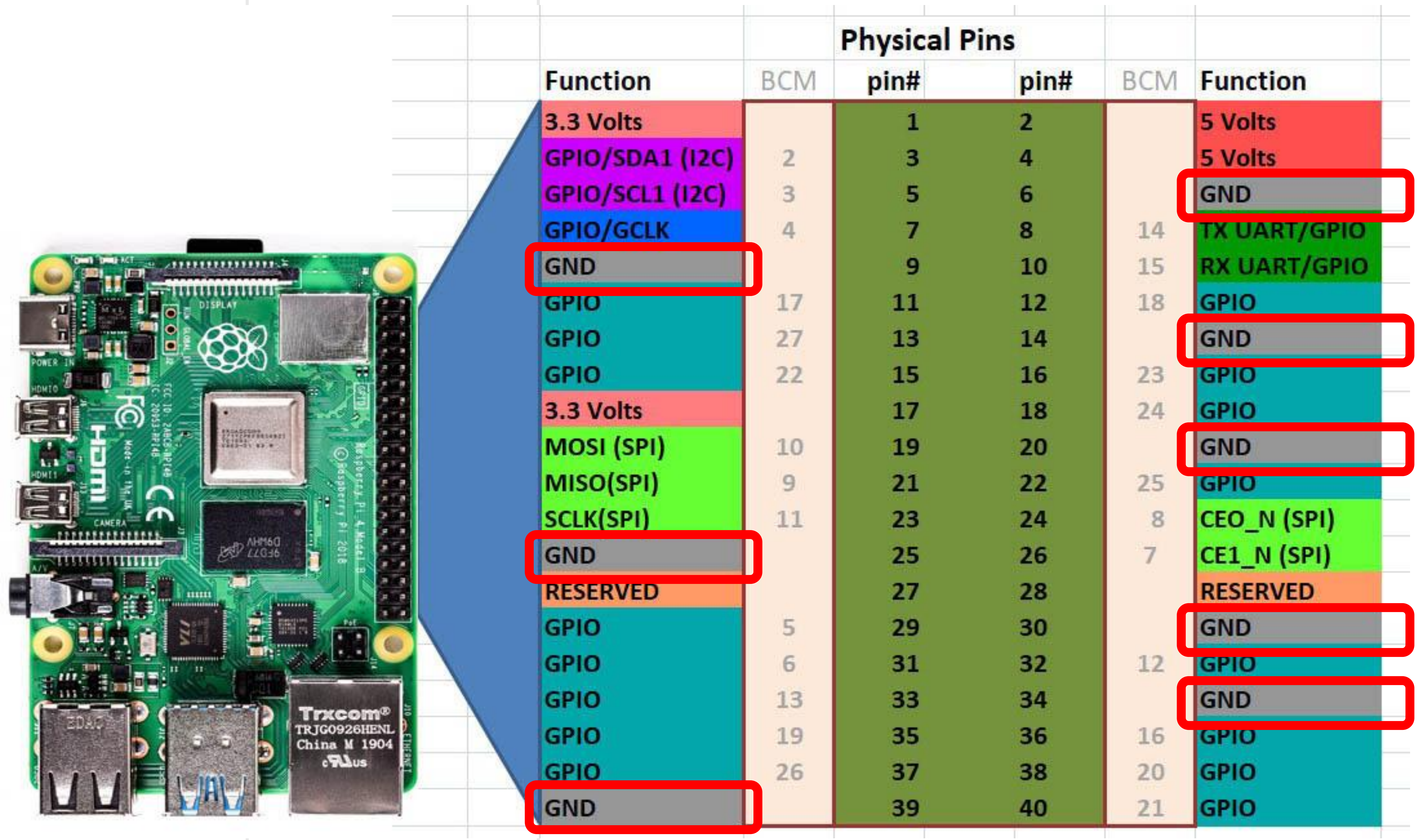



3v3 Power 1 | 2 5v Power
GPIO 2 4 SPI3 MOSI SDA1 4 SDA3 3 | 4 5v Power
GPIO 3 4 SPI3 SCLK SCL1 4 SCL3 5 | 6 Ground
GPIO 4 4 SPI4 CE0 N 4 TXD3 4 SDA3 7 | 8 TXD0 4 TXD1 GPIO 14 4 SPI5 MOSI
Ground 9 | 10 RXD0 4 RXD1 GPIO 15 4 SPI5 SCLK
GPIO 17 SPI1 CE1 N 11 | 12 GPIO 18 SPI1 CE0 N 4 SPI6 CE0 N
GPIO 27 4 SPI6 CE1 N 13 | 14 Ground
GPIO 22 4 SDA 6 15 | 16 GPIO 23 4 SCL6
3v3 Power 17 | 18 GPIO 24 4 SPI3 CE1 N
GPIO 10 SPI0 MOSI 4 SDA5 19 | 20 Ground
GPIO 9 SPI0 MISO 4 RXD4 4 SCL4 21 | 22 GPIO 25 4 SPI4 CE1 N
GPIO 11 SPI0 SCLK 4 SCL5 23 | 24 4 SDA4 GPIO 8 SPI0 CE0 N 4 TXD4
Ground 25 | 26 4 SCL4 GPIO 7 SPI0 CE1 4 SPI4 SCLK
GPIO 0 4 SPI3 CE0 N 4 TXD2 SDA0 4 SDA6 27 | 28 4 SPI3 MISO GPIO 1 SCL0 4 SCL6 4 RXD2
GPIO 5 4 SPI4 MISO 4 RXD3 4 SCL3 29 | 30 Ground
GPIO 6 4 SPI4 MOSI 4 SDA4 31 | 32 4 SDA5 GPIO 12 4 SPI5 CE0 N 4 TXD5
GPIO 13 4 SPI5 MISO 4 RXD5 4 SCL5 33 | 34 Ground
GPIO 19 SPI1 MISO 4 SPI6 MISO 35 | 36 GPIO 16 4 SPI1 CE2 N
GPIO 26 4 SPI5 CE1 N 37 | 38 GPIO 20 SPI1 MOSI 4 SPI6 MOSI
Ground 39 | 40 GPIO 21 SPI1 SCLK 4 SPI6 SCLK
General Purpose Input Output
SPI (Serial Peripheral Interface)
I2C (Inter-Integrated Circuit)
UART (Universal Asynchronous Receiver / Transmitter)
Ground (GND)
5v Power
3v Power
2 Physical Pin Number
4 Pi 4 Only

Raspberry Pi GPIO pin layout

- GPIO stands for **General Purpose Input/Output**
- Allow Raspberry Pi to interact with the outside world

What is a GND Pin?

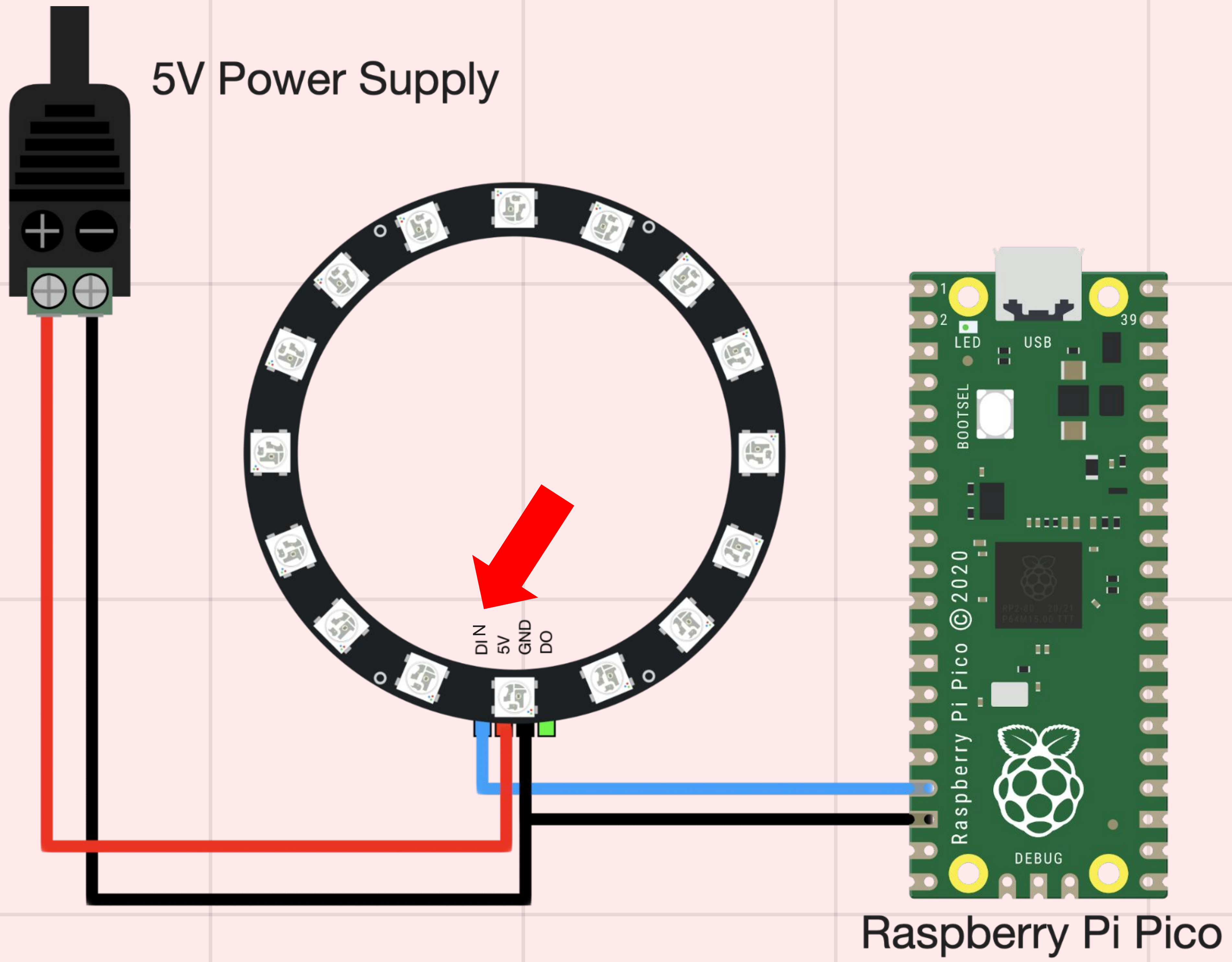


		Physical Pins			
Function	BCM	pin#	pin#	BCM	Function
3.3 Volts		1	2		5 Volts
GPIO/SDA1 (I2C)	2	3	4		5 Volts
GPIO/SCL1 (I2C)	3	5	6		GND
GPIO/GCLK	4	7	8	14	TX UART/GPIO
GND		9	10	15	RX UART/GPIO
GPIO	17	11	12	18	GPIO
GPIO	27	13	14		GND
GPIO	22	15	16	23	GPIO
3.3 Volts		17	18	24	GPIO
MOSI (SPI)	10	19	20		GND
MISO(SPI)	9	21	22	25	GPIO
SCLK(SPI)	11	23	24	8	CEO_N (SPI)
GND		25	26	7	CE1_N (SPI)
RESERVED		27	28		RESERVED
GPIO	5	29	30		GND
GPIO	6	31	32	12	GPIO
GPIO	13	33	34		GND
GPIO	19	35	36	16	GPIO
GPIO	26	37	38	20	GPIO
GND		39	40	21	GPIO

Raspberry Pi GPIO pin layout

- GND stands for **Ground**
- A common return path for electric current
- Prevent electrical damage
- Give a common reference point for all components in the circuit

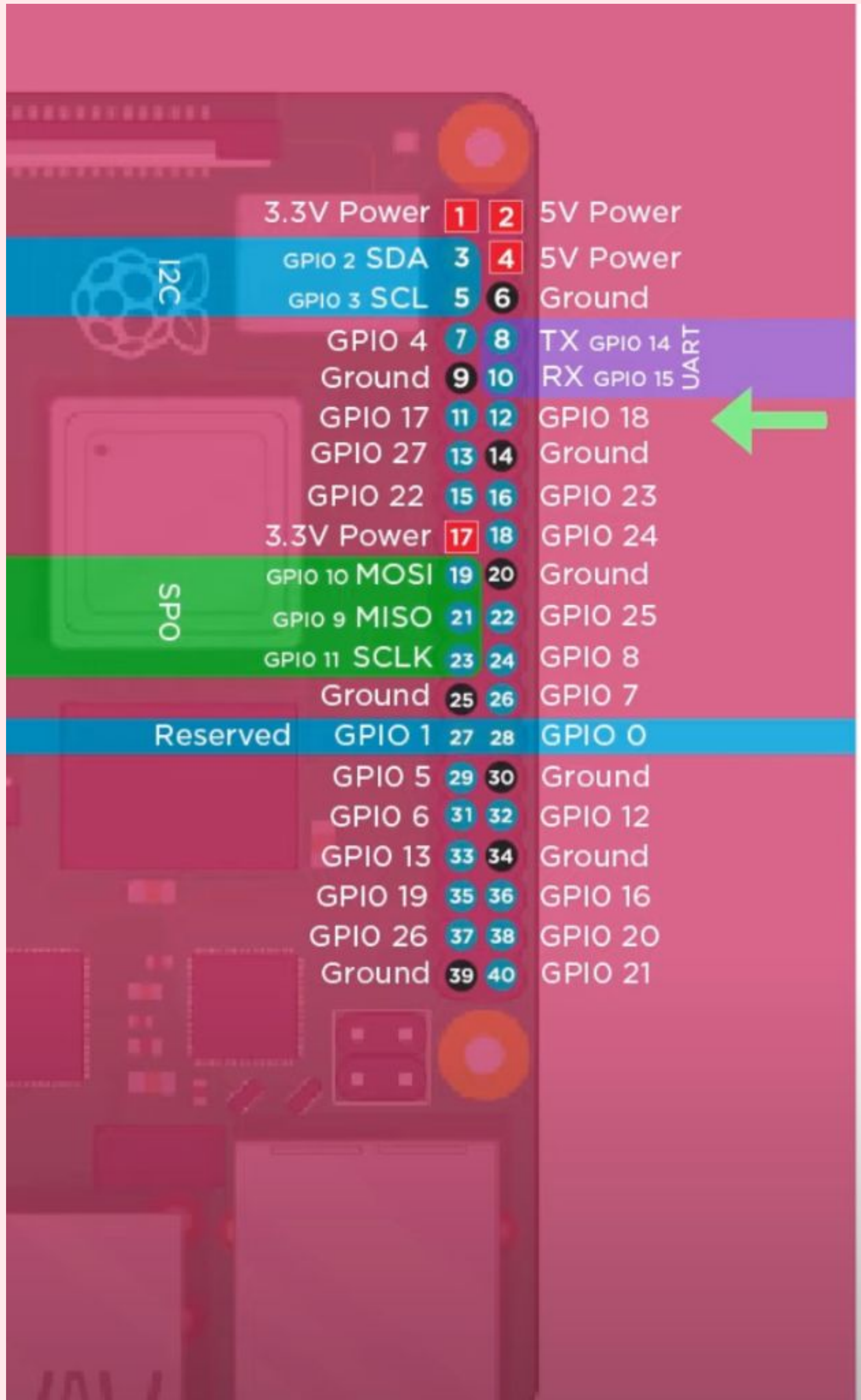
What is a DIN Pin (on the LED Ring)?

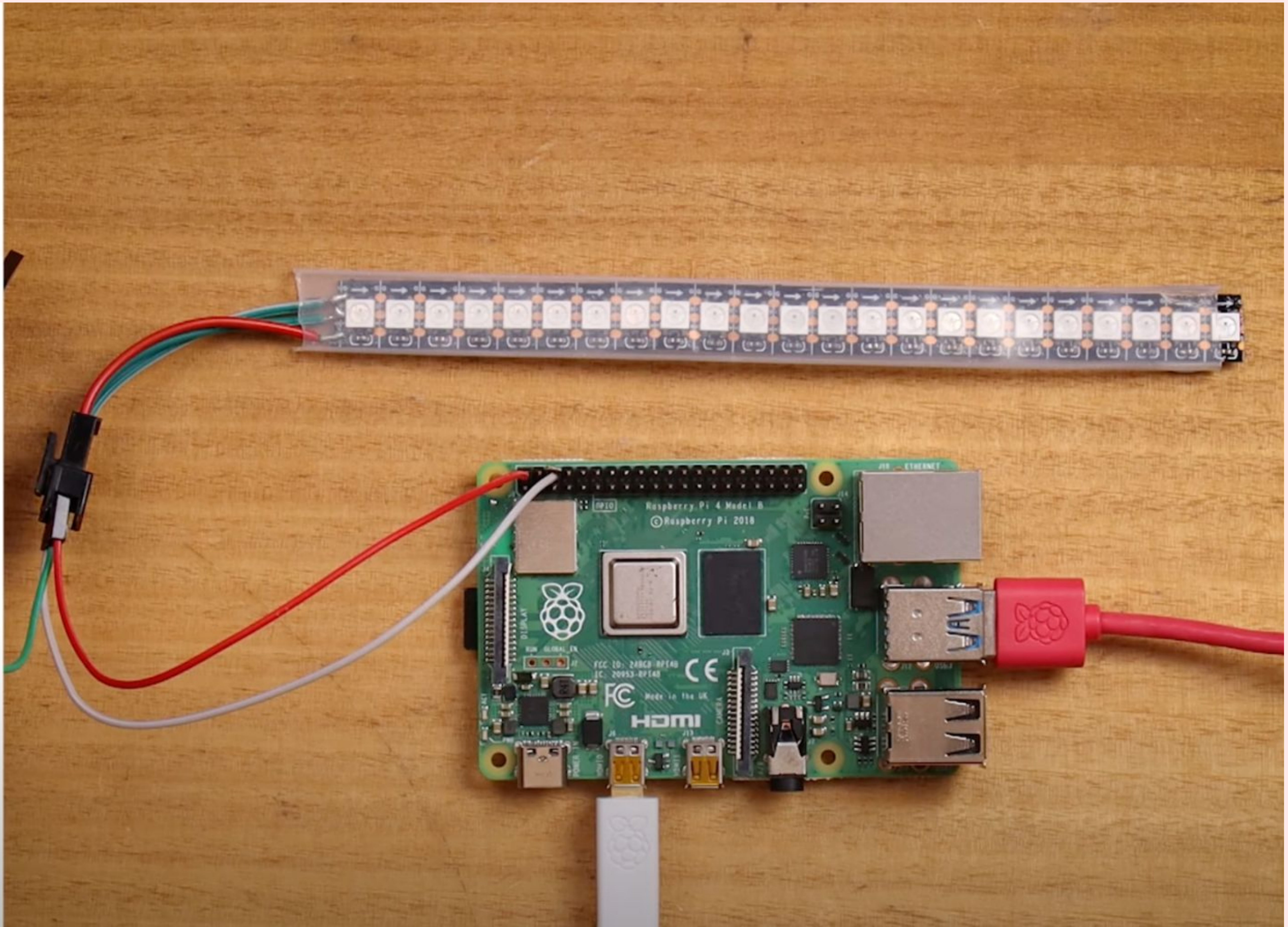


- DIN stands for **Data IN**
- DIN pin is where it **receives data**
- Color and brightness information for each LED

Hardware Set up Instruction

https://www.youtube.com/watch?v=aNlaj1r7NKc



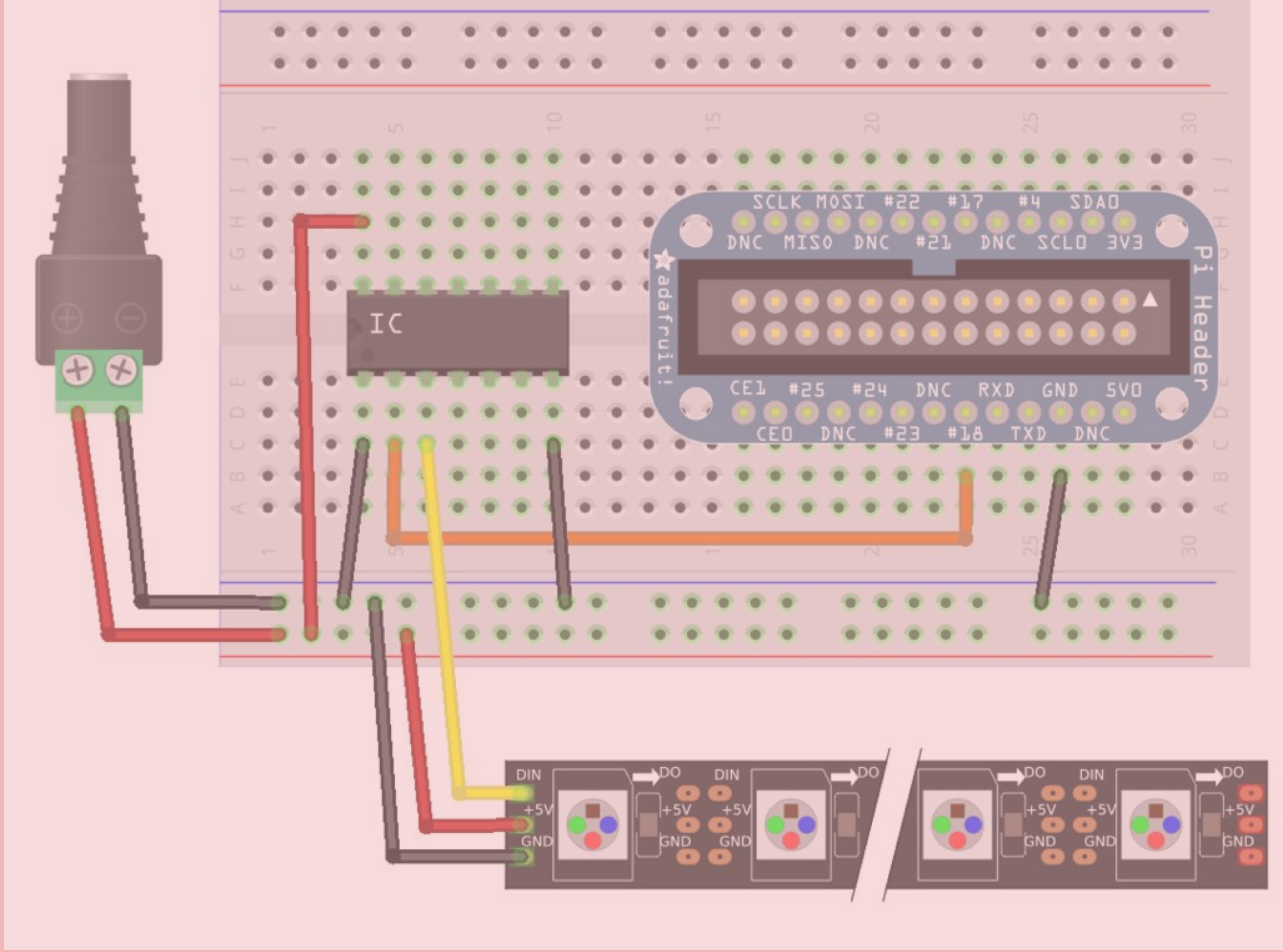


Required Libraries Installation

Main Library

rpi_ws281x

- git clone https://github.com/jgarff/rpi_ws281x.git
- cd rpi_ws281x
- scons
- sudo scons install



IC
SCLK MOSI #22 #17 #4 SDA0
DNC MISO DNC #21 DNC SCL0 3V3
Pi Header
adafruit!
CE1 #25 #24 DNC RXD GND 5V0
CE0 DNC #23 #18 TXD DNC
DIN
DO
+5V
GND

Data Structures and Functions in Main Library

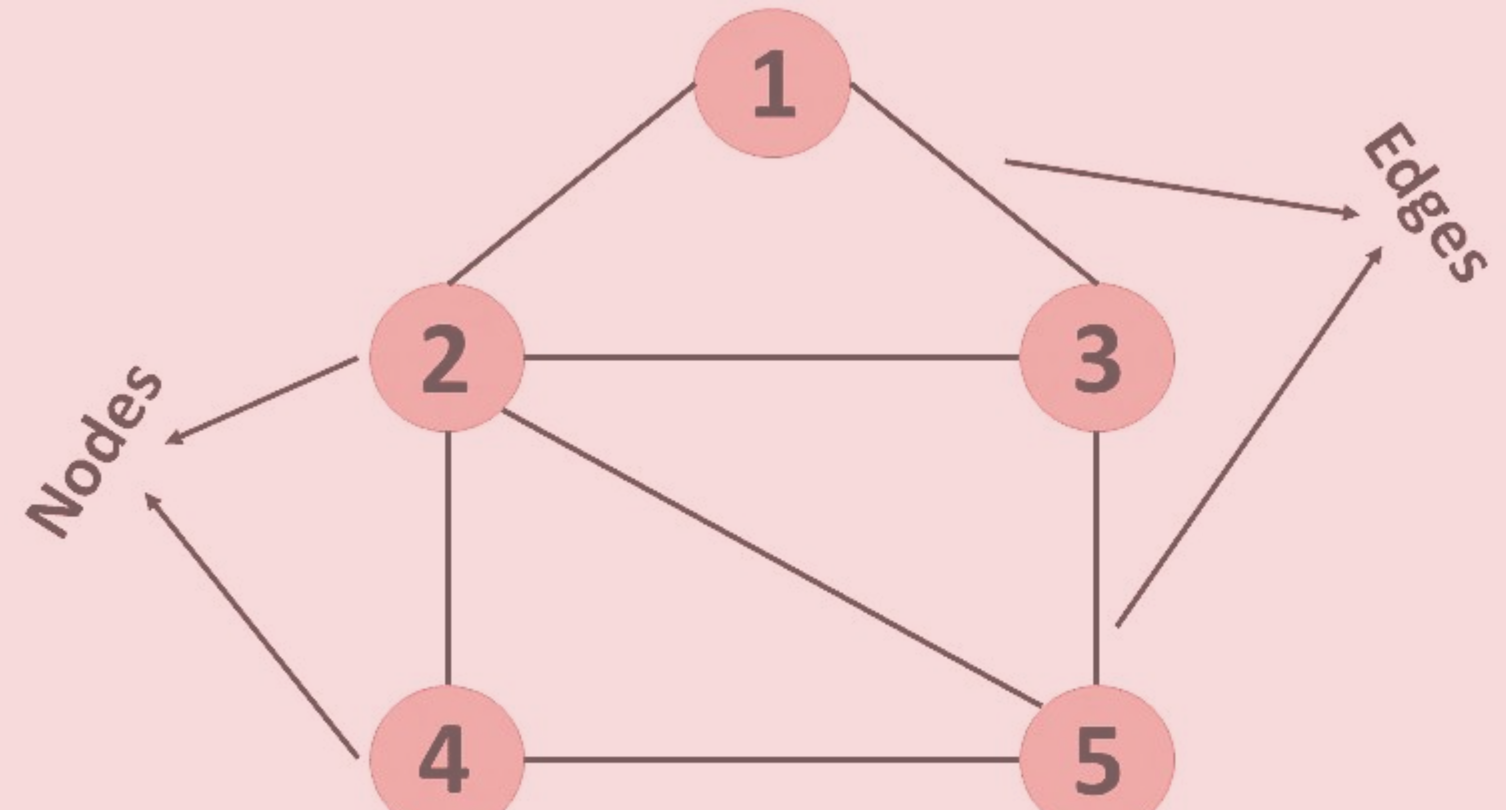
rpi_ws281x

Data Structures

- initializing the parameters of the LED strip.
- setting the parameters of each channel of the LED strip.

Functions

- ws2811_init()
- ws2811_render()
- ws2811_fini()



Other Libraries

WiringPi

- ‘wringPiISR()’ function

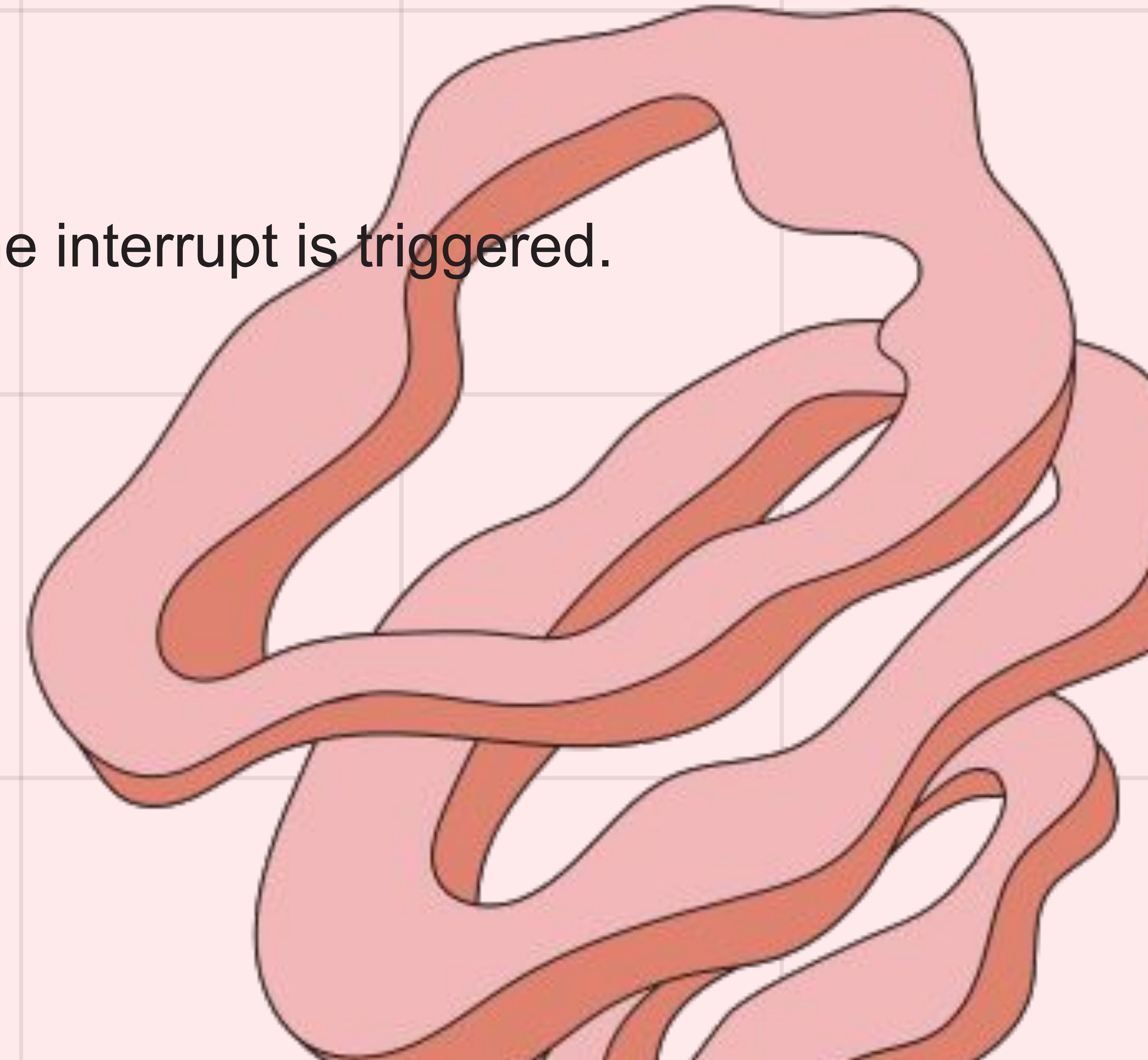
  example in C: int wiringPiISR(int pin, int edgeType, void (*function)(void))

  ‘pin’: The GPIO pin number to set up the ISR for.

  ‘edgeType’: The type of edge triggering the interrupt.

  ‘function’: A pointer to the function that will be called when the interrupt is triggered.
- Install dependencies in terminal:

  ```
  sudo apt-get update
  sudo apt-get install git-core
  ```
- git clone https://github.com/WiringPi/WiringPi.git
- cd WiringPi
- ./build



Main reasons for using WiringPi

- **Simplicity and Abstraction:** abstracts the low-level hardware operations, making it easier for developers to control the GPIO pins without having to deal with the intricacies of direct hardware manipulation.

- **Consistency:** provides a consistent API across different models of Raspberry Pi, allowing developers to write code that works across multiple hardware revisions.

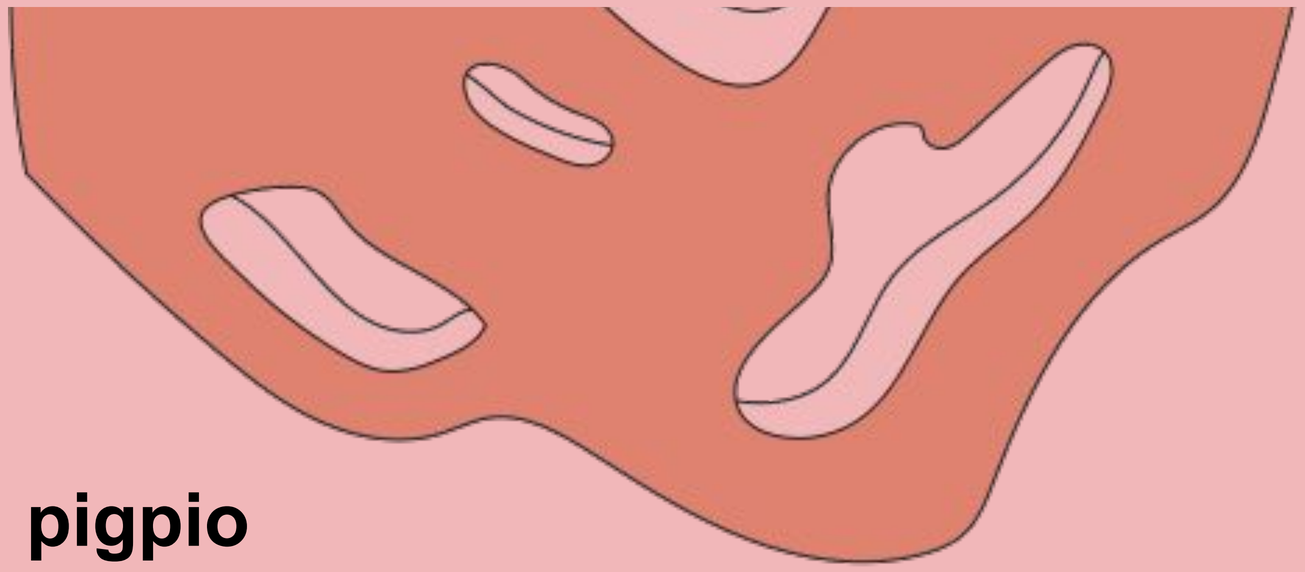
- **Community Support:** gained popularity and has an active community, which means that you can find plenty of resources, tutorials, and examples to help you get started and troubleshoot any issues.

- **Flexibility:** supports various programming languages, including C, C++, and Python, making it accessible to a wide range of developers with different language preferences.

- **Compatibility:** provides functions for working with I2C, SPI, and serial communication, in addition to GPIO, making it suitable for projects that require multiple types of communication protocols.

- **Speed and Performance:** it is optimized for performance, allowing developers to achieve efficient GPIO operations even in time-critical applications.

Other Libraries



pigpio

- It is an alternative to WiringPi and allows access to Raspberry Pi GPIO pins from user space.
- Offers similar functionality as WiringPi and is chosen by some users due to its advanced features and capabilities.

unistd.h

- unistd.h is a C standard library header file.
- Provides access to POSIX operating system APIs, enabling interaction with functions like read, write, and sleep for tasks such as file I/O and process control.

Write C Program to control odd lights on or even lights on

Firstly, we need to import the libraries:

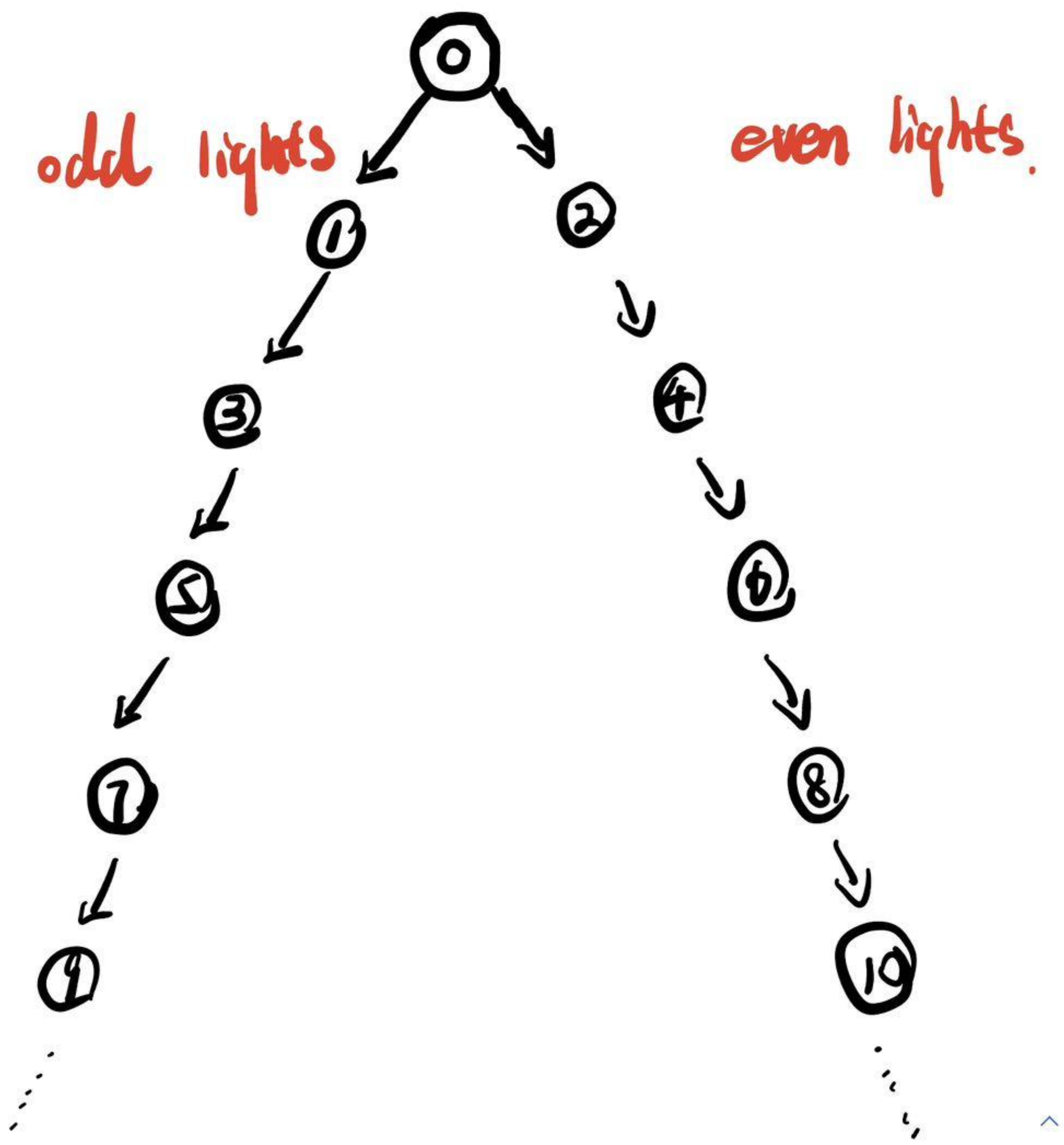
```
#include <ws2811.h>
#include <unistd.h>
```

typedef struct TreeNode

TreeNode* createNode(int led_index)

void traverse Tree

void freeTree(TreeNode root)


odd lights
even lights.
0
1
2
3
4
5
6
7
8
9
10

Challenges

When install WiringPi:

The terminal says that :unable to determine board type

Why?

WiringPi may not support the specific Raspberry Pi model

How to solve this problem:

Install pigpio: A GPIO interface library for the Raspberry Pi which provides a wide range of functionality including hardware timed PWM suitable for LEDs, motors and other hardware capture.

When install rpi_w281x:

The terminal said command not found

How to solve this problem:

`scons` command is not installed or not available in your system's PATH.

`scons` is a software construction tool (i.e., a build tool) that you're trying to use to build the `rpi_ws281x` library.

When we tried to build the our code:

It said : ws2811.h No such file or directory
the compiler cannot find the `ws2811.h` header file. This file is part of the `rpi_ws281x` library, and it should have been installed on our system when we installed that library.

There are several reasons for that:

1. The `rpi_ws281x` library wasn't installed correctly.
2. The `ws2811.h` file isn't in a directory that the compiler is looking in. By default, the compiler only looks in certain directories for header files.
3. If `ws2811.h` is in a different directory, we 'll need to tell the compiler where to find it.

```
gcc -o led_test led_test.c -lws2811
```

the terminal said /usr/bin/ld: cannot find -lws2811.   and collect2: ld returned 1 exist status

Provide the correct library path: If our library is not in a standard location, we need to tell the linker where to find it using the `-L` flag.

```
gcc -o led_test led_test.c -I/home/pi/rpi_ws281x/ -lws2811
```

How do I know if my LED light is really working?

Is it possible the program work but my LED is not working? I bought Two LED lights. and the problem is that we need to a C program to make the light bright. and luckily, I found that there is a main in ws281x library can make it work.

but it is still not working!!!!!

well remove ws281x and install it again. check if we have gccm scons, and we install ws281x again
the led is lighting after I type ./main in rpi_ws281x
main.c in rpi_ws281x it said undefined reference to ws2911_init
he `-L` flag tells the linker where to find the library files.
he `-l` flag is followed by the name of the library you want to link against. In this case, it's `ws2811`.

```
gcc -o main main.c -I/path/to/rpi_ws281x -L/path/to/rpi_ws281x -lws2811
```

but the program I wrote is still not working
check the directory and also set up the Geany
the terminal said: can't open /dev/mem: permission denied   failed to initialize the library. exiting

t's trying to access the memory-mapped I/O to control the LEDs. This requires superuser privileges because of the potential for causing harm to the system if used incorrectly.

To run the program with superuser privileges, you need to use the `sudo` command

```
sudo ./main
```

```
gcc -o led led.c -lws2811 -L/path/to/rpi_ws281x -I/path/to/rpi_ws281x
```

it said ws2811.c 1320 undefined reference to 'pow'

```
gcc -o led led.c -lws2811 -lm -I/path/to/rpi_ws281x -L/path/to/rpi_ws281x
```

also, the keyboard is not very compatible with raspberry pi

In summary:
I have come across a lot problem and hiccups, at some point, I felt very frustrating.
However, as long as I kept trying and solve the problem one by one. I succeed and gained a sense of achievement.
This is an interesting project, and I appreciate the help from Shiva and TAs.
I have chance to know raspberry pi and and know more about it by this project.

Project Timeline

7/3-7/9

1st Week

- Project Kickoff
- Research
- Define requirements and objectives
- Select hardwares
- Prepare the presentation

7/10-7/16

2nd Week

- Purchase hardwares
- Set Up Development Environments

7/17-7/23

3rd Week

- Develop basic LED Control Code
- Test LED patterns

7/24-7/30

4th Week

- Optimize and refine code

7/31-8/9

5th Week

- Prepare final presentation