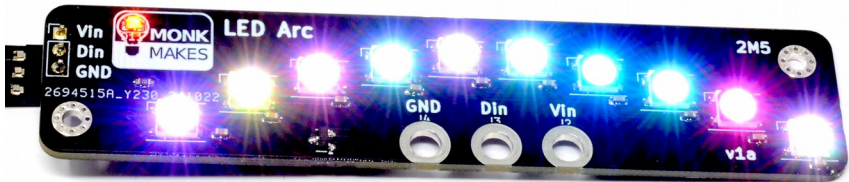


# Instructions:



## LED ARC



This 10 LED, multi-color arc with pin header and ring connectors, is designed to work with BBC micro:bit, Raspberry Pi Pico, ESP32 and Arduino.

- 10 individually controllable LEDs (brightness and color)
- 4mm ring connector for BBC micro:bit and Adafruit Circuit Playground boards
- Pin Header connector for other microcontroller boards
- Reverse polarity protection
- Power indicator LED
- Compatible with Adafruit and other addressable LED libraries
- 3.3V and 5V operation

Instructions version 1a.

# TABLE OF CONTENTS

Assembly .....	3
Using the LED Arc.....	4
Downloading the Examples.....	5
BBC micro:bit.....	7
Raspberry Pi Pico (2).....	11
ESP32.....	14
Arduino.....	17
Troubleshooting.....	20
Support.....	20
MonkMakes.....	21

## ASSEMBLY

If you are using the LED Arc with a micro:bit, or other devices that use alligator clips, there is no assembly to be done. The board is ready to use.

If, on the other hand, you plan to connect to the LED Arc using the pin headers (supplied), then you will need to solder the supplied header pins into place.



If you want to mount the LED Arc in an enclosure, and the LEDs are going to be against a front panel of some sort, then it's best to push the short end of the right-angle header pins through from the bottom of the LED Arc and solder on the top (as shown above) – as this will make it easier to attach leads.

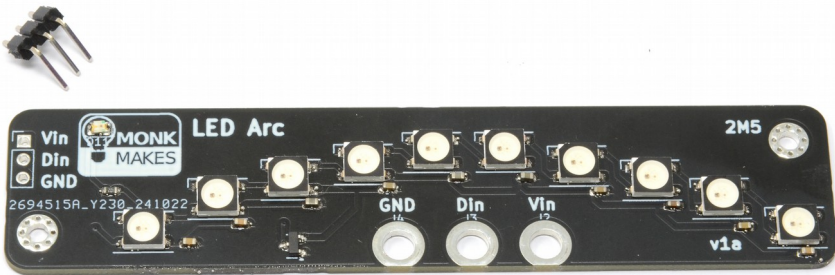
**Tip:** When soldering the pins, sometimes a piece of adhesive putty is useful to keep the pins in position until the first pin is soldered.

The mounting holes are designed to take 2M5 (2.5mm) diameter screws.

## USING THE LED ARC

The Vin and GND connections of the LED Arc can be connected directly to the voltage output pins of whatever microcontroller board you are using. This can be either 3.3V or 5V, but if you have both on your board choose 5V for optimum brightness.

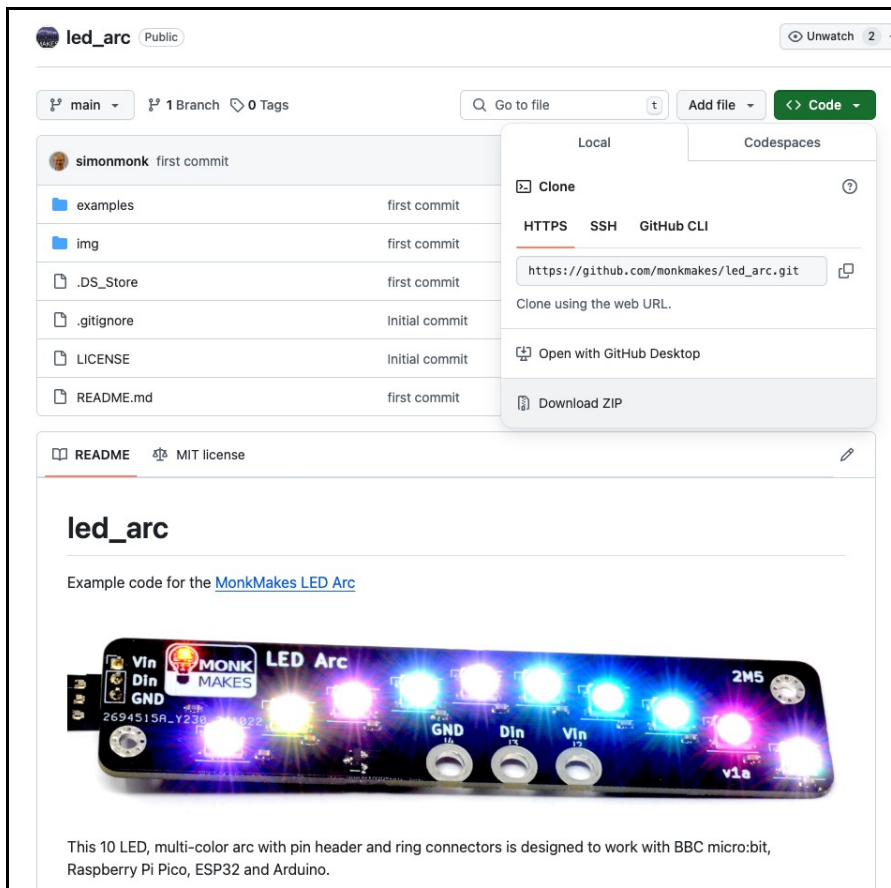
The LED Arc's blue component of light will start to dim if the supply voltage drops below 3.3V, so although the device will still function at 3.3V or even the 3.0V of a micro:bit, lower supply voltages than 3.0V will not produce a full range of colors.



# DOWNLOADING THE EXAMPLES

To download the ZIP archive containing example programs for all platforms, visit [https://github.com/monkmakes/led\\_arc](https://github.com/monkmakes/led_arc) in your browser.

Examples for the BBC micro:bit are also available, but they are in *makecode* not on github. Skip to the next section if you are using a BBC micro:bit.



The screenshot shows the GitHub repository page for `led_arc`. At the top, there's a navigation bar with the repository name, a search bar, and a `Code` button. Below this, a table lists the repository's files and folders, including `examples`, `img`, `.DS_Store`, `.gitignore`, `LICENSE`, and `README.md`. A dropdown menu is open over the `Code` button, showing options to Clone (via HTTPS, SSH, or GitHub CLI), Open with GitHub Desktop, and Download ZIP. Below the file list, there's a section for the README, which features a photo of the LED Arc board and a description: "This 10 LED, multi-color arc with pin header and ring connectors is designed to work with BBC micro:bit, Raspberry Pi Pico, ESP32 and Arduino."

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

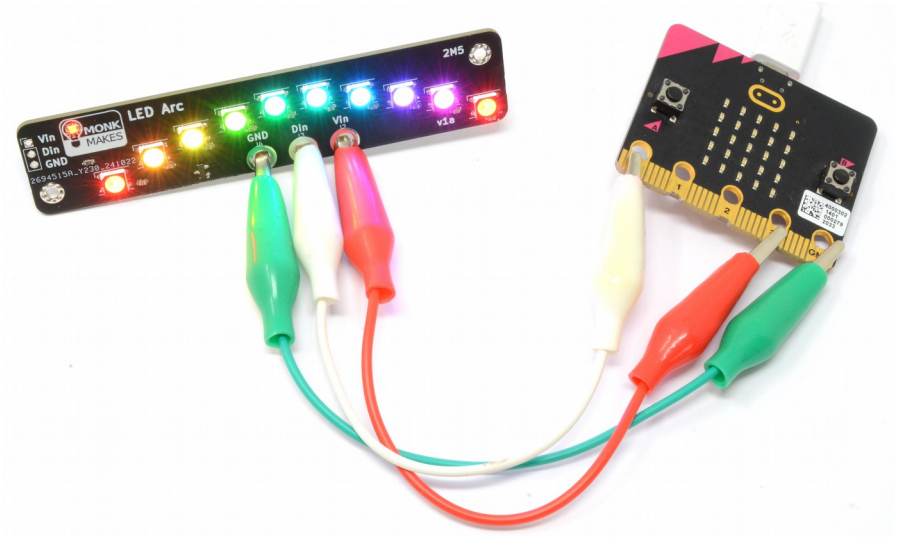
If you are familiar with git and would prefer to download the examples using the command line, then you can do so with the command:

```
$ git clone https://github.com/monkmake/led_arc.git
```

With the extracted archive, you will find a folder called *examples* containing folders called:

- pico – MicroPython examples for the Raspberry Pi Pico
- arduino – Arduino, and other boards that can be programmed with the Arduino IDE
- esp32 – ESP32 examples

# BBC MICRO:BIT



## You will need

To connect to a BBC micro:bit you will need 3 alligator clip leads.

## Wiring

Make the following connections between your micro:bit and the LED Arc.

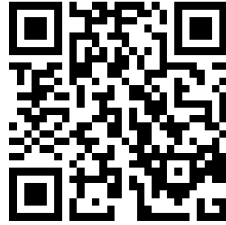
- GND on the micro:bit to GND (ground) on the LED Arc
- 3V on the micro:bit to Vin (Voltage In) on the LED Arc
- 0 on the micro:bit to Din (Data In) on the LED Arc

Once the GND and Vin connections are made, the LED Arc will be receiving power from the micro:bit and the orange LED in the MonkMakes logo will illuminate.

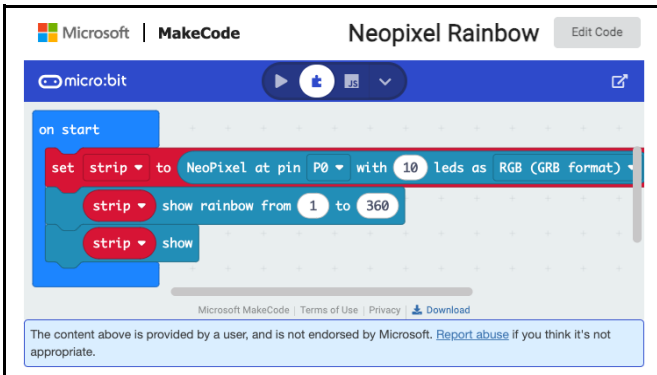
## Example Software

You will find a makecode blocks example here:  
[https://makecode.microbit.org/\\_cJHbi6f7t24K](https://makecode.microbit.org/_cJHbi6f7t24K)

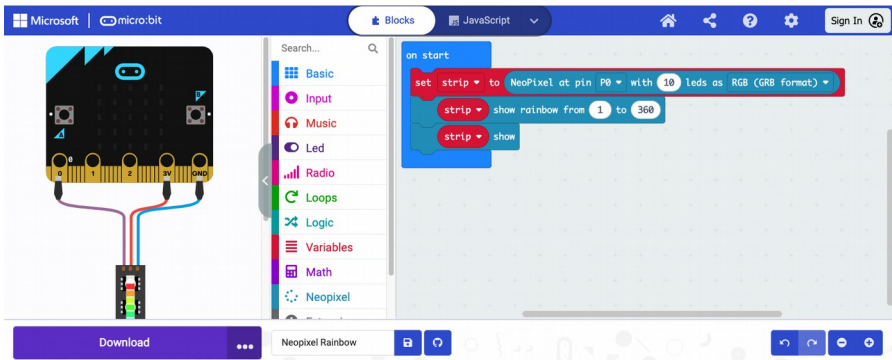
The example displays different colors on each of the LEDs.



When you follow the link or QR code, you will see a page like this.



Click on the **Edit** button, and this will load the makecode project in a normal editor.



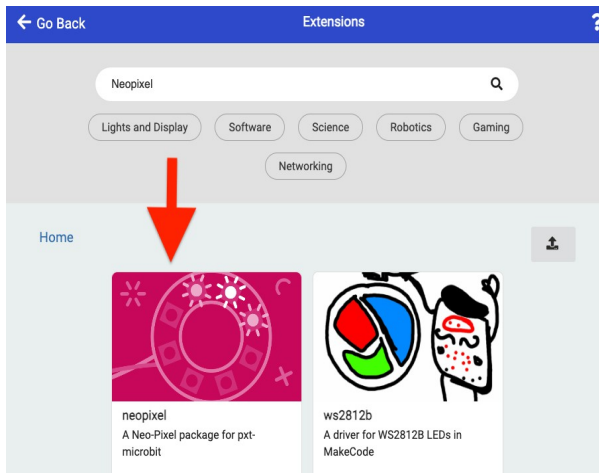
Click on the Download button and follow the instructions, that pop up, to download the project onto your micro:bit. Your LED Arc should display a pleasing array of colors.



## makecode Extension

The LED Arc is compatible with most Neopixel LED makecode libraries. I suggest you use the one indicated below. This is the library embedded in the Rainbow example that you have just run.

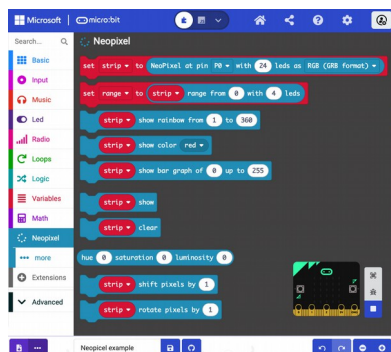
To find it, click on the *Extensions* button on the makecode editor and then type *Neopixel* into the search box.



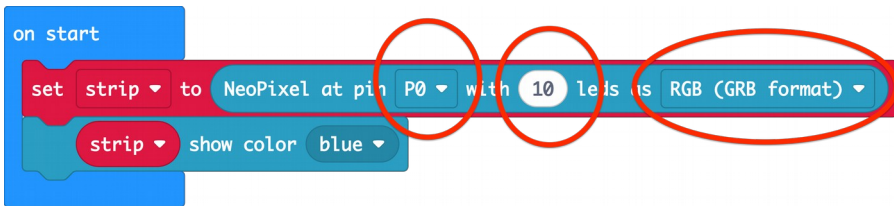
Click on the *neopixel* extension and it will be installed into your makecode editor.

This will add a new entry to the editor's pallet, with lots of blocks for using Neopixel LEDs.

The extension has a very wide selection of useful blocks for controlling the LEDs.

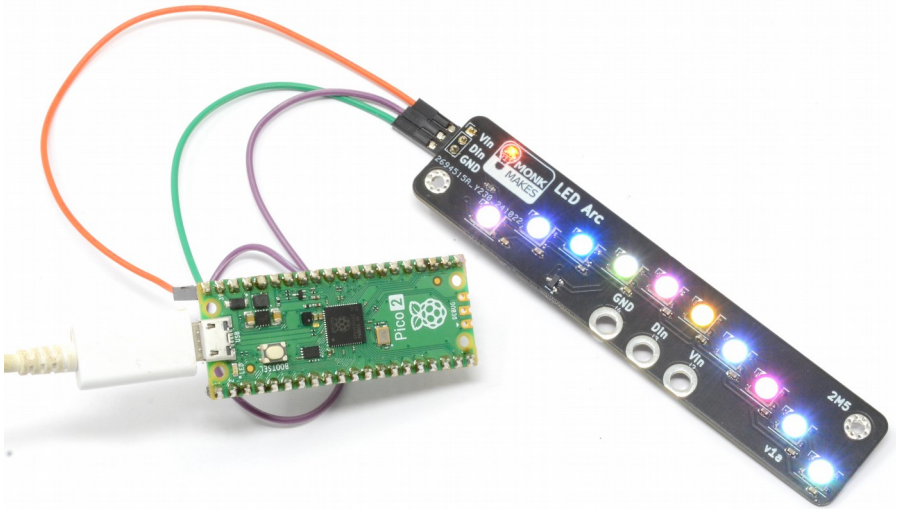


The most important block, without which nothing will be displayed, is the first one in the example code below.



This tells the micro:bit which pin to use (**P0**) how many LEDs there are on the display (**10**) and the color format of the LEDs (choose **RGB (GRB format)**)

# RASPBERRY PI PICO (2)



## You will need

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

- A Raspberry Pi Pico, Pico 2, Pico W or Pico W 2.
- 3x female to female jumper wires
- Soldering equipment to attach headers pins to the LED Arc (see Page 3)

## Wiring

Make the following connections, using female to female jumper wires, between your Pico and the LED Arc.

- GND on the Pico to GND (ground) on the LED Arc
- VSYS on the Pico to Vin (Voltage In) on the LED Arc
- 22 on the Pico to Din (Data In) on the LED Arc

Once the GND and Vin connections are made, the LED Arc will be receiving power from the Pico and the orange power LED in the MonkMakes logo will illuminate.

## Example Software

In the *pico* folder of the Examples folder that you downloaded (see Page 5) you will find a program *rainbow.py*.

Open it in Thonny, or your preferred MicroPython environment, and run it. Your LED Arc should display a pleasing array of colors.

Here's the code:

```
import array, time
from machine import Pin
import rp2
from random import randint

NUM_LEDS = 10
LED_PIN = 22

# Start of Magic programmable IO code
@rp2.asm_pio(sideset_init=rp2.PIO.OUT_LOW,
out_shiftdir=rp2.PIO.SHIFT_LEFT, autopull=True, pull_thresh=24)
def ws2812():
    T1 = 2
    T2 = 5
    T3 = 3
    wrap_target()
    label("bitloop")
    out(x, 1)                .side(0)    [T3 - 1]
    jmp(not_x, "do_zero")    .side(1)    [T1 - 1]
    jmp("bitloop")          .side(1)    [T2 - 1]
    label("do_zero")
    nop()                    .side(0)    [T2 - 1]
    wrap()

sm = rp2.StateMachine(0, ws2812, freq=8_000_000,
sideset_base=Pin(LED_PIN))
sm.active(1) # Start the StateMachine
pixels = array.array("I", [0 for _ in range(NUM_LEDS)])
# End of Magic programmable IO code

def set_led(led, red, green, blue):
    pixels[led] = (green << 16) + (red << 8) + blue

def show():
    sm.put(pixels, 8)

def clear():
    for i in range(NUM_LEDS):
        set_led(i, 0, 0, 0)
```

```

show()

def randomize():
    clear()
    for i in range(NUM_LEDS):
        set_led(i, randint(0, 50), randint(0, 50))
        show()
        time.sleep(0.1)

clear()
randomize()

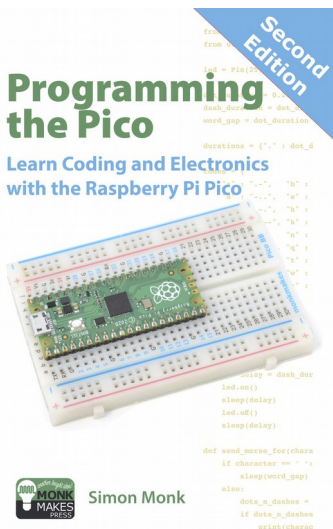
```

The code makes use of the Pico's programmable GPIO feature to generate the pulses to set the color and brightness of the LEDs. This code is all contained between the comments: # Start of Magic programmable IO code and # End of Magic programmable IO code

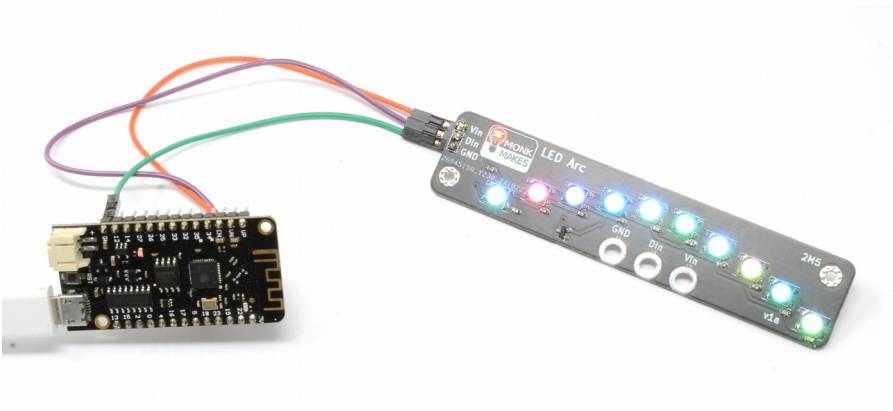
You don't need to understand this code (which is just as well) to use it, you can simply add it to your program along with the functions `set_led` and `show`, which set the color of a particular LED and tell the LED to update what it's displaying respectively.

If you want to learn more about programming the Raspberry Pi Pico, then you may be interested in this book, which is available from Amazon, as well as many Hobby Electronics suppliers. [https://monkmakes.com/book\\_prog\\_pico2](https://monkmakes.com/book_prog_pico2)

The book is also available in Spanish and Italian.



# ESP32



## You will need

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

- An ESP32 development board such as the ESP32 Lite shown above
- 3x female to female jumper wires
- Soldering equipment to attach headers pins to the LED Arc (see Page 3)

## Wiring

Make the following connections, using female to female jumper wires, between your Pico and the LED Arc.

- GND (labeled G) on the ESP32 to GND (ground) on the LED Arc
- 3.3V on the ESP32 to Vin (Voltage In) on the LED Arc
- 5 on the Pico to Din (Data In) on the LED Arc

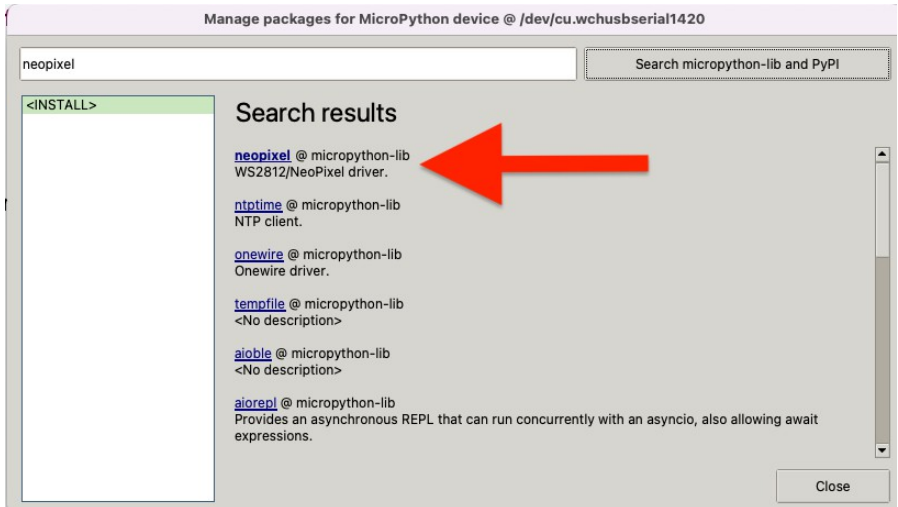
Once the GND and Vin connections are made, the LED Arc will be receiving power from the ESP32 board and the orange power LED in the MonkMakes logo will illuminate.

## Example Software

In the *ESP32* folder of the Examples folder that you downloaded (see Page 5) you will find a program `rainbow.py`.

Open it in Thonny, or your preferred MicroPython environment.

Before you can run it, you will need to install the *neopixel* library onto your ESP32 board. To do this in Thonny, open the package manager from the Tools->Manage Packages.. menu option and then type *neopixel* into the search field.



Select the search result *neopixel* and follow the prompts to install the library.

You should now be able to run *rainbow.py* -- your LED Arc should display a pleasing array of colors.

Here's the code:

```
from time import sleep
from machine import Pin
from random import randint
from neopixel import NeoPixel

NUM_LEDS = 10
pixels = NeoPixel(Pin(5), NUM_LEDS)

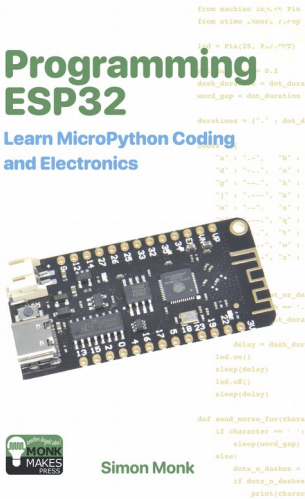
def clear():
    pixels.fill((0, 0, 0))
    pixels.write()

def randomize():
    clear()
    for i in range(NUM_LEDS):
        pixels[i] = (randint(0, 50), randint(0, 50), randint(0, 50))
        pixels.write()
        sleep(0.1)
```

randomize()

The code makes use of *neopixel* library to generate the pulses needed to control the LEDs. You can read more about the library here: <https://docs.micropython.org/en/latest/esp8266/tutorial/neopixel.html>

If you want to learn more about programming ESP32 boards in MicroPython, then you may be interested in this book, which is available from Amazon, as well as some Hobby Electronics suppliers. [https://monkmakes.com/book\\_prog\\_esp32](https://monkmakes.com/book_prog_esp32)



**Programming ESP32**  
Learn MicroPython Coding and Electronics

Simon Monk

```
from machine import Pin
from time import sleep

led = Pin(27, Pin.OUT)

dash_dur = 0.2
word_gap = dot_duration

messages = ["." + dot_d

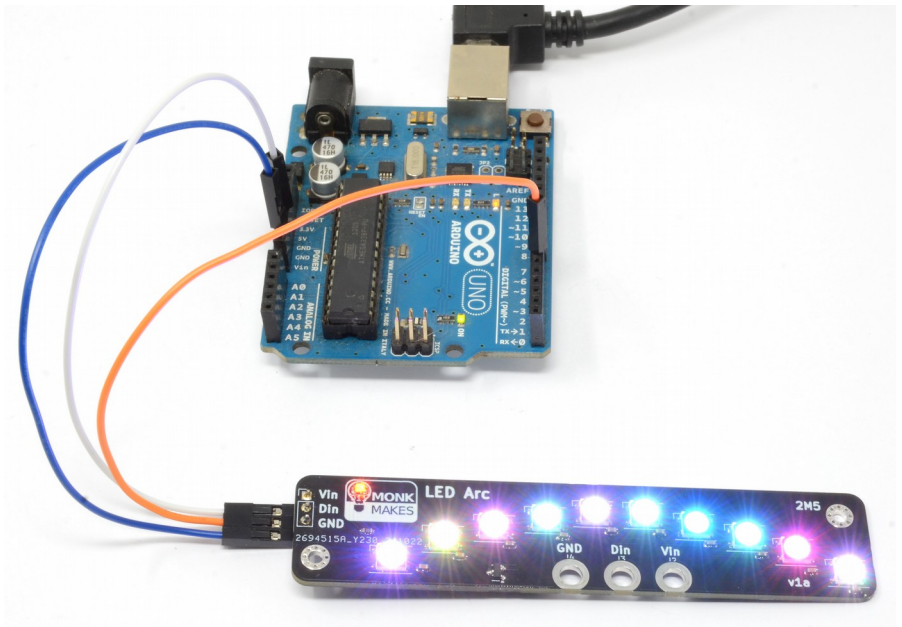
"0" + "0000", "0" +
"1" + "0000", "0" +
"2" + "0000", "0" +
"3" + "0000", "0" +
"4" + "0000", "0" +
"5" + "0000", "0" +
"6" + "0000", "0" +
"7" + "0000", "0" +
"8" + "0000", "0" +
"9" + "0000"

dot_dur = dash_dur
led.on()
sleep(dash_dur)
led.off()
sleep(dash_dur)

def send_message_for(char)
if character == ".":
sleep(word_gap)
else:
dots_n_dashes =
if dots_n_dashes:
print(char)
```



# ARDUINO



## You will need

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

- An Arduino Uno or other Arduino board
- 3x female to male jumper wires
- Soldering equipment to attach headers pins to the LED Arc (see Page XX)

Make the following connections, using female-to-female jumper wires, between your Arduino and the LED Arc.

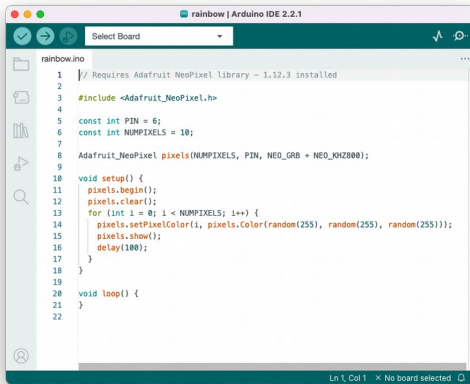
- GND on the Arduino to GND (ground) on the LED Arc
- 5V on the Arduino to Vin (Voltage In) on the LED Arc
- 5 on the Pico to Din (Data In) on the LED Arc

Once the GND and Vin connections are made, the LED Arc will be receiving power from the Arduino and the orange power LED in the MonkMakes logo will illuminate.

## Example Software

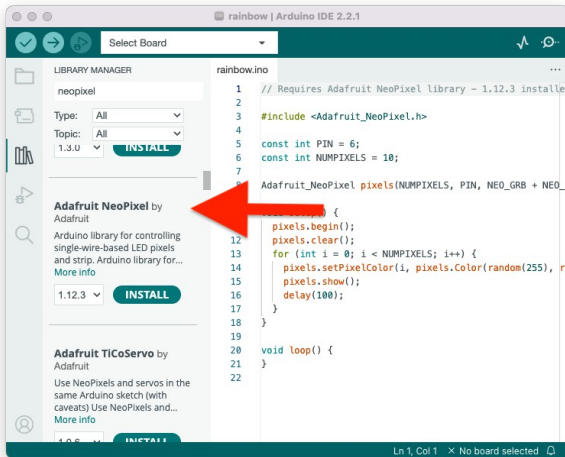
In the *Arduino* folder of the *examples* folder that you downloaded (see Page 5) you will find a program *rainbow.py*.

Open it in the Arduino IDE.



```
1 // Requires Adafruit NeoPixel library - 1.12.3 installed
2
3 #include <Adafruit_NeoPixel.h>
4
5 const int PIN = 6;
6 const int NUMPIXELS = 10;
7
8 Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
9
10 void setup() {
11   pixels.begin();
12   pixels.clear();
13   for (int i = 0; i < NUMPIXELS; i++) {
14     pixels.setPixelColor(i, pixels.Color(random(255), random(255)));
15     pixels.show();
16     delay(100);
17   }
18
19
20 void loop() {
21 }
22
```

Before you can run this example, you will need to install the *Adafruit Neopixel* library. To do this, use the *Tools->Manage Libraries* option to open the Library Manager.



Scroll down the list of search results until you find the library called just *Adafruit Neopixel* and install the library.

You should now be able to upload the rainbow sketch to your Arduino -- your LED Arc should display a pleasing array of colors.

The Adafruit Neopixel library is fantastically useful library that works with most types of board that can be programmed using the Arduino IDE. Adafruit have their own wide selection of NeoPixel displays (<https://www.adafruit.com/category/168>).

Here's the code:

```
#include <Adafruit_NeoPixel.h>

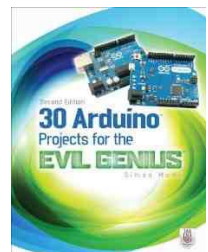
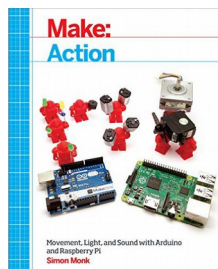
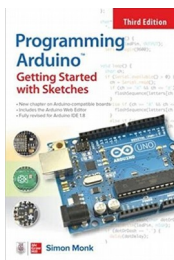
const int PIN = 6;
const int NUMPIXELS = 10;

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pixels.begin();
  pixels.clear();
  for (int i = 0; i < NUMPIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(random(255), random(255),
                                           random(255)));
  }
  pixels.show();
  delay(100);
}

void loop() {
}
```

If you want more resources to help you on your Arduino and electronics journey, you might enjoy some of these books, from the author of these instructions.



## TROUBLESHOOTING

**Problem:** The orange power LED in the MonkMakes logo on the LED Arc does not light.

**Solution:** Make sure that the power connections are good. If you have a multimeter, set it to it's 20V range and test the voltage between GND and Vin on the LED Arc's ring connectors.

**Problem:** The Power LED lights, but none of the Neopixels are lit.

**Solution:** Check that the lead between your LED Arc and microcontroller is going to the GPIO pin that you are using in the code. Sometime jumper and alligator leads fail, so try a different lead.

## SUPPORT

You can find the Product's information page here: [https://monkmakes.com/led\\_arc](https://monkmakes.com/led_arc) including a datasheet for the product.

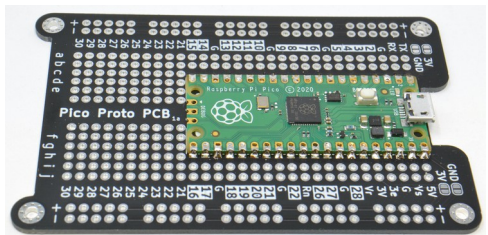
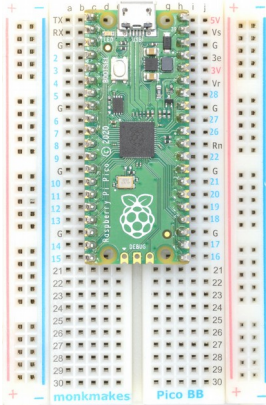
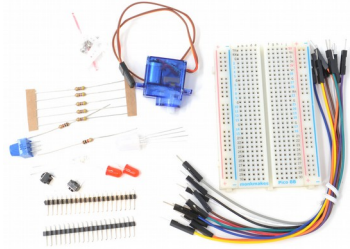
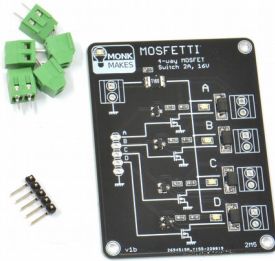
If you need further support, please email [support@monkmakes.com](mailto:support@monkmakes.com).

# MONKMAKES

As well as this board, 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>

You can also follow MonkMakes on Instagram where we are [monk\\_makes\\_ltd](https://www.instagram.com/monk_makes_ltd) and on our Facebook page: <https://www.facebook.com/monkmakes/>



Mosfetti: <https://monkmakes.com/mosfetti.html>

Electronics Kit 1 for Pico: [https://monkmakes.com/pico\\_kit1.html](https://monkmakes.com/pico_kit1.html)

Breadboard for Pico: [https://monkmakes.com/pico\\_bb.html](https://monkmakes.com/pico_bb.html)

Illuminata: <https://monkmakes.com/illuminata.html>

Pico Proto PCB: [https://monkmakes.com/pico\\_proto.html](https://monkmakes.com/pico_proto.html)

# NOTES