

C

You

how can I rewrite this program so I can have 3 leds that turn on and off in a sequence from machine

```
import Pin, Timer
led = Pin(25, Pin.OUT)
timer = Timer()
```

```
def blink(timer):
    led.toggle()
```

```
timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

ChatGPT

To modify your program to control three LEDs so they turn on and off in sequence rather than just toggling a single LED, you'll need to adjust your approach. Since you want a sequence, using a timer to toggle each LED in turn is a good strategy, but you'll manage the sequence by cycling through the LEDs in your callback function rather than just toggling one.

Here's a revised version of your program that achieves this with three LEDs. This example assumes the LEDs are connected to GPIO pins 25, 26, and 27, but you can adjust these pin numbers based on your actual setup:

python

Copy code

```
from machine import Pin, Timer

# Define LEDs
led1 = Pin(25, Pin.OUT)
led2 = Pin(26, Pin.OUT)
led3 = Pin(27, Pin.OUT)

# List to hold LEDs for easy access
leds = [led1, led2, led3]

# Index to keep track of the current LED
current_led = 0

def blink(timer):
```

```

global current_led # Use the global variable to keep track across calls

# Turn off all LEDs
for led in leds:
    led.value(0)

# Turn on the current LED
leds[current_led].value(1)

# Move to the next LED in the sequence, wrapping around if necessary
current_led = (current_led + 1) % len(leds)

# Initialize the timer
timer = Timer()

# Timer configuration
timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)

```

This program defines three LEDs and stores them in a list for easy access. The `blink` function, called by the timer, turns off all LEDs, turns on the current one, then increments the index to point to the next LED in the sequence, wrapping around to the first LED after the last one has been lit.

The `freq` parameter in the `timer.init` call determines how often the `blink` function is called. With a frequency of 2.5 Hz, the `blink` function is called 2.5 times per second, cycling through the LEDs in sequence at this rate.

You can adjust the `freq` value to make the sequence faster or slower according to your needs.

You

Can you make me have a button that when I press it, the sequence starts

ChatGPT

ChatGPT

Certainly! To start the LED sequence when a button is pressed, you'll need to add code to handle button press events. This involves setting up an interrupt on the button's pin that triggers a function to start the LED sequence. We'll assume the button is connected to GPIO pin 28, but you can adjust this to the actual pin you're using.

Here's how you can modify the program to start the LED sequence with a button press:

python

Copy code

```
from machine import Pin, Timer

# Define LEDs
led1 = Pin(25, Pin.OUT)
led2 = Pin(26, Pin.OUT)
led3 = Pin(27, Pin.OUT)
leds = [led1, led2, led3] # List of LEDs for easy access

# Initialize variables
current_led = 0 # Index for the current LED in sequence
sequence_started = False # Flag to indicate whether sequence has started

# Timer for LED sequence
timer = Timer()

def blink(timer):
    global current_led
    if sequence_started: # Only run sequence if started
        for led in leds:
            led.value(0) # Turn off all LEDs
            leds[current_led].value(1) # Turn on the current LED
            current_led = (current_led + 1) % len(leds) # Move to the next LED

def start_sequence(pin):
    global sequence_started
    sequence_started = True # Set flag to start sequence
    timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink) # Start timer for LED
sequence

# Define button
button = Pin(28, Pin.IN, Pin.PULL_DOWN)

# Interrupt for button press
button.irq(trigger=Pin.IRQ_RISING, handler=start_sequence)
```

In this version, the sequence is controlled by the `sequence_started` flag. This flag is set to `True` when the button is pressed, activating the timer's callback to begin the sequence. The `start_sequence` function is tied to the button press via an interrupt (`irq`). The interrupt is configured to trigger on a rising edge (`Pin.IRQ_RISING`), which occurs when the button is pressed (assuming the button is connected to ground and pulls the pin high when pressed).

The `Pin.PULL_DOWN` configuration for the button ensures that the pin is normally pulled low (to 0) and is considered pressed when it goes high (to 1). This setup might need to be adjusted depending on your actual button wiring.

This program will start cycling the LEDs in sequence when the button is pressed. If you want the sequence to stop when the button is pressed again or have other controls over the sequence, you would need to add additional logic to handle those cases.

You

the button is normally connected to 3v of power, is not a pull down

ChatGPT

ChatGPT

If your button is normally connected to 3V and is not configured as a pull-down, but rather it connects the input pin to ground when pressed, you should configure the button pin with an internal pull-up resistor. This means the pin is normally high (at 3V) and goes low (to ground) when the button is pressed.

Here's how to adjust the program to work with a button that is connected in this manner:

```
python
```

```
Copy code
```

```
from machine import Pin, Timer

# Define LEDs
led1 = Pin(25, Pin.OUT)
led2 = Pin(26, Pin.OUT)
led3 = Pin(27, Pin.OUT)
leds = [led1, led2, led3] # List of LEDs for easy access

# Initialize variables
current_led = 0 # Index for the current LED in sequence
sequence_started = False # Flag to indicate whether sequence has started

# Timer for LED sequence
timer = Timer()

def blink(timer):
    global current_led
    if sequence_started: # Only run sequence if started
```

```

for led in leds:
    led.value(0) # Turn off all LEDs
    leds[current_led].value(1) # Turn on the current LED
    current_led = (current_led + 1) % len(leds) # Move to the next LED

def start_sequence(pin):
    global sequence_started
    # Toggle the sequence start/stop
    sequence_started = not sequence_started
    if sequence_started:
        timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink) # Start timer for LED
sequence
    else:
        timer.deinit() # Stop the timer and thereby the sequence
        for led in leds: # Turn off all LEDs when sequence is stopped
            led.value(0)

# Define button with internal pull-up resistor
button = Pin(28, Pin.IN, Pin.PULL_UP)

# Interrupt for button press
button.irq(trigger=Pin.IRQ_FALLING, handler=start_sequence)

```

Key Changes:

- The button is configured with `Pin.PULL_UP`, ensuring the pin is normally high and becomes low when the button is pressed (since pressing the button connects it to ground).
- The interrupt trigger is changed to `Pin.IRQ_FALLING`, which means the interrupt is triggered on the falling edge (transition from high to low), corresponding to a button press in this setup.
- The `start_sequence` function now toggles the `sequence_started` flag. If the sequence is started, it initializes the timer and begins blinking the LEDs. If the button is pressed again, it stops the sequence by deinitializing the timer and turns off all the LEDs.

This setup will allow you to start and stop the LED sequence by pressing the button, with the button connected to 3V and grounding the pin when pressed.