Turn On LED using Webpage

Jul 2026

Introduction

In this example, we are going to turn on an LED on our Raspberry Pi Pico using a simple web page with two buttons. There is a second function where we can display the state of a hardware button also.

The three projects we will build for this are as follows:

- Add an LED and a Button to the Pico. This is just to test the hardware and wiring is good.
- Add a small web server and page to the Pico, this will be text only.
- Add a webserver and page that is graphical to the Pico.

This will allow us to make defined steps towards the final goal of having a web page control the LED on our Pico.

Add an LED and Button to the Pico

The image below shows what we are trying to achieve.

In this example, we have the LED being connected to GP14 (pin 19) and GND (pin 18) via a 100 ohm resistor. You can use one of the GND pins on the Pico, or a common GND rail if you have made one.

Remember, the Anode of the LED (the longer leg) goes to GP14 and Cathode goes to GND.

The button is wired between pins GP16 (pin 21) and GND (pin 23)

Micro Python Code

Here is the code that we will use:

| download

```
# Hardware Test
# Blink and LED (GP14) slowly/quickly while a button (GP16) is not-pressed/pressed

from machine import Pin          # Import Pin class to control GPIO pins
from time import sleep_ms        # Import millisecond sleep function

led = Pin(14, Pin.OUT)           # Create an output pin on GPIO14 for the LED
button = Pin(16, Pin.IN, Pin.PULL_UP)  # Create an input pin on GPIO16 with internal pull-up
resistor

while True:                      # Infinite loop
    if button.value() == 0:      # If button is pressed (input pulled LOW)
        delay = 100              # Use short delay (fast blink)
```

```
    else:                           # Otherwise (button not pressed)
        delay = 1000                # Use long delay (slow blink)

    led.toggle()                    # Flip LED state: ON → OFF or OFF → ON
    sleep_ms(delay)                 # Wait for the chosen delay before next toggle
```

Because the button uses Pin.PULL_UP, its normal (unpressed) state is HIGH (1).

Pressing the button pulls the pin to LOW (0), which is why the code checks button.value() == 0.

button-led-pico-004.mp4

Looking at the simulation above, we can see that when the button is not pressed, we get a delay of 1000mS as per the MicroPython code, so 1 flash every two seconds (1s on, 1s off).

When the button is pressed, we get a delay of only 100mS, so 5 flashes every second (100mS on, 100mS off x 5).

If you have connected the LED and Button to your Pico correctly, this is what you should observe when you run the code on your setup.

Add a Small Web Server and Page to the Pico

NOTE*** The LED status will not change on the Raspberry Pi Pico unless you refresh the web browswer when you either Press or Release the button.

For this example, we do not need to change any of the Pico hardware, so the LED and Button wiring remain as they were in the previous example. This is just a software change.

You need to load the following code in to Thonny.

| download

```
# Simple HTTP Server Example
# Control an LED and read a Button using a web browser

import time                     # Provides sleep and timing functions
import network                  # Wi-Fi control for Pico W
import socket                   # Low-level networking (TCP sockets)
from machine import Pin         # GPIO pin control

led = Pin(14, Pin.OUT)          # LED on GPIO14 as an output
ledState = 'LED State Unknown'  # Text placeholder for LED status

button = Pin(16, Pin.IN, Pin.PULL_UP)  # Button on GPIO16 with pull-up resistor

ssid = 'YourSSID'             # Wi-Fi network name
password = 'YourWifiPassword' # Wi-Fi password

wlan = network.WLAN(network.STA_IF)  # Use Wi-Fi in station mode
wlan.active(True)                    # Turn Wi-Fi hardware on
wlan.connect(ssid, password)         # Connect to the Wi-Fi network

# HTML template for the webpage
html = """<!DOCTYPE html>
<html>
<head> <title>Pico W</title> </head>
<body> <h1>Pico W HTTP Server</h1>
<p>Hello, World!</p>
<p>%s</p>
</body>
</html>
"""

# Wait for Wi-Fi connection or timeout
```

```
max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:  # Error or connected
        break
    max_wait -= 1
    print('waiting for connection...')
    time.sleep(1)

# If not connected, stop the program
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
else:
    print('Connected')
    status = wlan.ifconfig()           # Get IP configuration
    print('ip = ' + status[0])         # Print IP address

# Create a listening TCP socket on port 80 (HTTP)
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]  # Bind to all interfaces
s = socket.socket()                               # Create socket
s.bind(addr)                                       # Bind to address/port
s.listen(1)                                        # Listen for connections
print('listening on', addr)

# Main server loop
while True:
    try:
        cl, addr = s.accept()          # Accept a client connection
        print('client connected from', addr)
        request = cl.recv(1024)        # Read HTTP request
        print("request:")
        print(request)
        request = str(request)         # Convert bytes → string

        # Look for URL parameters: ?led=on or ?led=off
        led_on = request.find('led=on')
        led_off = request.find('led=off')

        print('led on = ' + str(led_on))
        print('led off = ' + str(led_off))

        # If found at position 8 (typical for GET requests)
        if led_on == 8:
            print("led on")
            led.value(1)
        if led_off == 8:
            print("led off")
            led.value(0)

        # Update LED state text
        ledState = "LED is OFF" if led.value() == 0 else "LED is ON"

        # Read button state
        if button.value() == 1:        # Button not pressed (pull-up = HIGH)
            print("button NOT pressed")
            buttonState = "Button is NOT pressed"
            led.value(0)               # Force LED OFF
        else:
            print("button pressed")
            buttonState = "Button is pressed"
            led.value(1)               # Force LED ON

        # Build webpage content
        stateis = ledState + " and " + buttonState
        response = html % stateis      # Insert text into HTML template

        # Send HTTP response
        cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
        cl.send(response)
        cl.close()                     # Close connection
```

```
    except OSError as e:
        cl.close()
        print('connection closed')
```

Before you attempt to run this code, look at the top of the code, lines 14 and 15.

```
ssid = 'YourSSID'
password = 'YourWifiPassword'
```

You need to enter your wifi details here, then you can run the code.

I suspect the one thing that will cause issues for you will be connecting to the Wifi, so take your time and get this right.

Running the Code

When you run the code, it will initially do three things:

1. Connect to your SSID
2. Login using your credentials
3. Get an IP Address

Once all three of these criteria are met, you will see the IP Address in the bottom window of Thonny.

| download

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
Connected
ip = 10.194.1.206
listening on ('0.0.0.0', 80)
```

Above we can see we got the IP Address 10.194.1.206 and we are listening on 0.0.0.0:80 - (0.0.0.0 means we are listening on port 80 on ALL IP interfaces).

On your computer (that is on the same network) run a web browser, and browse to the IP Address that you were given.

If all is well, you should have a webpage showing that looks like the one below (I only captured the top corner).

The page clearly states that the Button is NOT pressed, and the LED is off.

Now, on the Pico, press AND HOLD the button, while pressing and holing the button, refresh the webpage.

You should see now that the page states that the button IS pressed and that the LED is on.

When you let go of the button, to get the update, and to turn off the LED, you have to refresh the webpage.

Add a Webserver and Page with Buttons to the Pico.

In this section, all we are doing is adding some buttons that you can click to turn the LED on and OFF. The only thing we have to do is replace the HTML section in our previous code, with a new HTML section.

When looking at the code, we can see a section called html that starts at line 21 and goes to line 29, like so.

| download

```
html = """<!DOCTYPE html>
<html>
<head> <title>Pico W</title> </head>
<body> <h1>Pico W HTTP Server</h1>
<p>Hello, World!</p>
<p>%s</p>
</body>
</html>
"""
```

What we need to do, is to replace the lines above, with the following block of code:

| download

```
html = """<!DOCTYPE html><html>
<head><meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" href="data:,">
<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align:
center;}
.buttonGreen { background-color: #4CAF50; border: 2px solid #000000;; color: white; padding: 15px
32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin:
4px 2px; cursor: pointer; }
.buttonRed { background-color: #D11D53; border: 2px solid #000000;; color: white; padding: 15px
32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin:
4px 2px; cursor: pointer; }
text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}
</style></head>
<body><center><h1>Control Panel</h1></center><br><br>
<form><center>
<center> <button class="buttonGreen" name="led" value="on" type="submit">LED ON</button>
<br><br>
<center> <button class="buttonRed" name="led" value="off" type="submit">LED OFF</button>
</form>
<br><br>
<br><br>
<p>%s<p></body></html>
"""
```

All we have to do now is to refresh the webpage (check your IP has not changed).

You can now change the LED status on or off by simply clicking the buttons on the webpage, clicking either button also refreshed the webpage. This means, if you hold the button down (or not) and click a button to change the LED, the LED status will change, and the text at the bottom of the webpage will also change to reflect the button status.

Full Code

| download

```
# Simple HTTP Server Example
# Control an LED and read a Button using a web browser

import time
import network
import socket
from machine import Pin

led = Pin(14, Pin.OUT)
ledState = 'LED State Unknown'
```

```
button = Pin(16, Pin.IN, Pin.PULL_UP)

ssid = 'YourSSID'
password = 'YourWifiPassword'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)

# replace the "html" variable with the following to create a more user-friendly control panel
html = """<!DOCTYPE html><html>
<head><meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" href="data:,">
<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align:
center;}
.buttonGreen { background-color: #4CAF50; border: 2px solid #000000;; color: white; padding: 15px
32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin:
4px 2px; cursor: pointer; }
.buttonRed { background-color: #D11D53; border: 2px solid #000000;; color: white; padding: 15px
32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin:
4px 2px; cursor: pointer; }
text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}
</style></head>
<body><center><h1>Control Panel</h1></center><br><br>
<form><center>
<center> <button class="buttonGreen" name="led" value="on" type="submit">LED ON</button>
<br><br>
<center> <button class="buttonRed" name="led" value="off" type="submit">LED OFF</button>
</form>
<br><br>
<br><br>
<p>%s<p></body></html>
"""

# Wait for connect or fail
max_wait = 10
while max_wait > 0:
    if wlan.status() < 0 or wlan.status() >= 3:
        break
    max_wait -= 1
    print('waiting for connection...')
    time.sleep(1)

# Handle connection error
if wlan.status() != 3:
    raise RuntimeError('network connection failed')
else:
    print('Connected')
    status = wlan.ifconfig()
    print( 'ip = ' + status[0] )


# Open socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)
print('listening on', addr)

# Listen for connections, serve client
while True:
    try:
        cl, addr = s.accept()
        print('client connected from', addr)
        request = cl.recv(1024)
        print("request:")
        print(request)
        request = str(request)
        led_on = request.find('led=on')
        led_off = request.find('led=off')
```

```
        print( 'led on = ' + str(led_on))
        print( 'led off = ' + str(led_off))

        if led_on == 8:
            print("led on")
            led.value(1)
        if led_off == 8:
            print("led off")
            led.value(0)

        ledState = "LED is OFF" if led.value() == 0 else "LED is ON" # a compact if-else statement

        if button.value() == 1: # button not pressed
            print("button NOT pressed")
            buttonState = "Button is NOT pressed"
        else:
            print("button pressed")
            buttonState = "Button is pressed"

        # Create and send response
        stateis = ledState + " and " + buttonState
        response = html % stateis
        cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
        cl.send(response)
        cl.close()

    except OSError as e:
        cl.close()
        print('connection closed')
```
