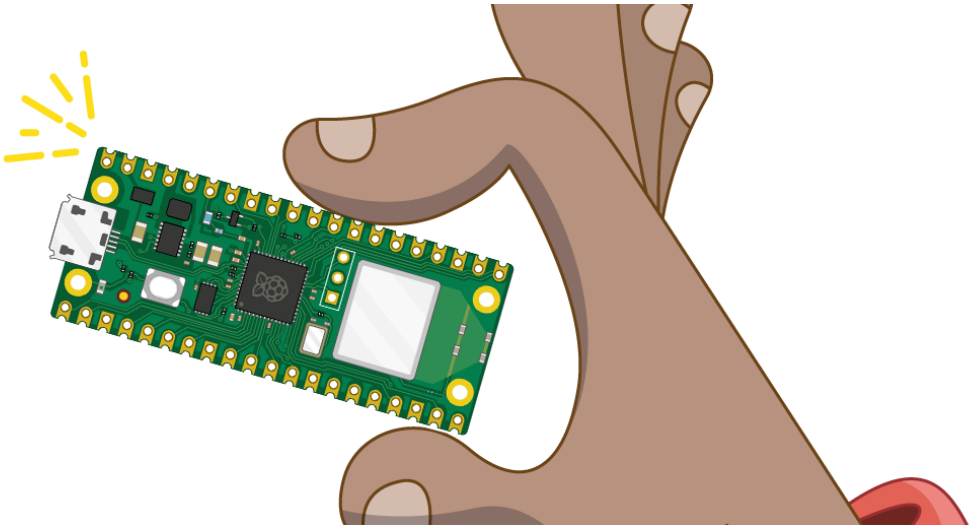
Raspberry Pi Foundation

inglés

Menu

Primeros pasos con su Raspberry Pi Pico W

Frambuesa Pi Pico Pitón



Contents

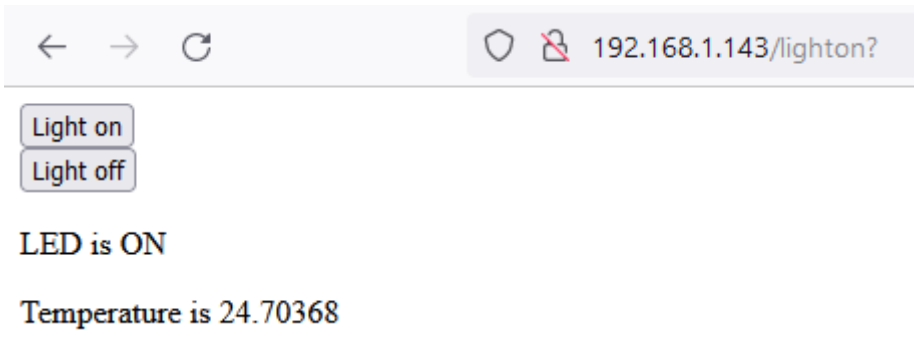
¡Guarda tu progreso!

Si desea volver a este proyecto más tarde, puede crear una cuenta de Raspberry Pi para guardar su progreso hasta el momento. En tu cuenta también verás todos los proyectos que completes.

Iniciar sesión o registrarse

Sirve tu página web

En este paso, iniciará su servidor web para que un cliente pueda conectarse a él, controlar su LED y leer la temperatura.



Cree una función que iniciará su servidor web, utilizando el `connection`objeto que guardó como parámetro. Las variables `state`y `temperature`deben configurarse para sus datos HTML. El estado comenzará como establecido en `'OFF'`, y la temperatura en `0`, lo que significa que también debe asegurarse de que el LED esté apagado cuando se inicie el servidor.

servidor_web.py
```
def serve(connection):
    #Start a web server
    state = 'OFF'
    pico_led.off()
    temperature = 0
```
Cuando su navegador web solicita una conexión a su Raspberry Pi Pico W, se debe aceptar la conexión. Después de eso, los datos que se envían desde su navegador web deben realizarse en fragmentos específicos (en este caso, 1024 bytes). También necesita saber qué solicitud está realizando su navegador web: ¿está solicitando solo una página simple? ¿Está pidiendo una página que no existe?

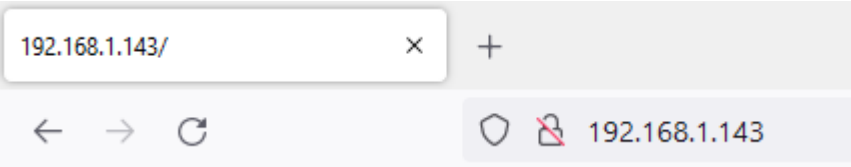
▢

Desea mantener el servidor web activo y escuchando todo el tiempo, para que cualquier cliente pueda conectarse a él. Puede hacer esto agregando un `while True:`bucle. Agregue estas cinco líneas de código para que pueda aceptar una solicitud y `print()`ver cuál era la solicitud. Agregue una llamada a su `serve`función en sus llamadas en la parte inferior de su código.

servidor_web.py
```
def serve(connection):
    #Start a web server
    state = 'OFF'
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        print(request)
        client.close()


try:
    ip = connect()
    connection = open_socket(ip)
    serve(connection)
except KeyboardInterrupt:
    machine.reset()
```

Prueba: ejecute su programa y luego escriba la dirección IP en la barra de direcciones de un navegador web en su computadora.



You should see something like this in the shell output in Thonny.

```
>>> %Run -c $EDITOR_CONTENT
Waiting for connection...
Waiting for connection...
Waiting for connection...
Connected on 192.168.1.143
b'GET / HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5.0 (Windo
b'GET /favicon.ico HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla
```

Next, you need to send the HTML code you have written to the client web browser.

web_server.py

```
def serve(connection):
    #Start a web server
    state = 'OFF'
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        print(request)
        html = webpage(temperature, state)
        client.send(html)
        client.close()


try:
    ip = connect()
    connection = open_socket(ip)
    serve(connection)
except KeyboardInterrupt:
    machine.reset()
```

Refresh your page when you've run the code again. Click on the buttons that are displayed. In Thonny, you should then see that there are two different outputs from your shell.

```
b'GET /lighton? HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5.(
```

and

```
b'GET /lightoff? HTTP/1.1\r\nHost: 192.168.1.143\r\nUser-Agent: Mozilla/5
```

Notice that you have `/lighton?` and `lightoff?` in the requests. These can be used to control the onboard LED of your Raspberry Pi Pico W.

- [ ]

Split the request string and then fetch the first item in the list. Sometimes the request string might not be able to be split, so it's best to handle this in a `try/except`.

If the first item in the split is `lighton?` then you can switch the LED on. If it is `lightoff?` then you can switch the LED off.

web_server.py

```
def serve(connection):
    #Start a web server
    state = 'OFF'
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        try:
            request = request.split()[1]
        except IndexError:
            pass
        if request == '/lighton?':
            pico_led.on()
        elif request =='/lightoff?':
            pico_led.off()
        html = webpage(temperature, state)
        client.send(html)
        client.close()
```

- [ ]

Ejecute su código de nuevo. Esta vez, cuando actualice la ventana de su navegador y haga clic en los botones, el LED integrado debería encenderse y apagarse.

- [ ]

También puede decirle al usuario de la página web cuál es el estado del LED.

servidor_web.py

```
def serve(connection):
    #Start a web server
```

```
    state = 'OFF'
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        try:
            request = request.split()[1]
        except IndexError:
            pass
        if request == '/lighton?':
            pico_led.on()
            state = 'ON'
        elif request =='/lightoff?':
            pico_led.off()
            state = 'OFF'
        html = webpage(temperature, state)
        client.send(html)
        client.close()
```

Ahora, cuando ejecute el código, el texto del estado del LED también debería cambiar en la página web actualizada.

Por último, puede usar el sensor de temperatura integrado para obtener una lectura aproximada de la temperatura de la CPU y mostrarla también en su página web.

servidor_web.py

```
def serve(connection):
    #Start a web server
    state = 'OFF'
    pico_led.off()
    temperature = 0
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)
        try:
            request = request.split()[1]
        except IndexError:
            pass
        if request == '/lighton?':
            pico_led.on()
            state = 'ON'
```

```
        elif request =='/lightoff?':
            pico_led.off()
            state = 'OFF'
        temperature = pico_temp_sensor.temp
        html = webpage(temperature, state)
        client.send(html)
        client.close()
```

Prueba: puede sostener su mano sobre su Raspberry Pi Pico W para aumentar su temperatura, luego actualice la página web en su computadora para ver el nuevo valor que se muestra.

¿Qué sigue?

¿Detectó un error? ¿Disfrutando del proyecto? ¿Alguna opinión sobre la web? ¡Haznos saber!

Enviar comentarios