12/6/24 Raspberry Pi Pico W Wifi

**Introduction**

By now, you should be able to run Thonny from your Raspberry Pi 5 and download a program to get the Raspberry Pi Pico W (picow) to blink its onboard LED and send back the message “Hello, World!” This is an essential milestone because Wi-Fi programming has a tendency to get weird if everything isn’t just so.

And then there’s ChatGPT—your programming buddy, equal parts genius and goofball. Before tackling Wi-Fi, I thought it was the best thing since sliced bread for coding. Afterward, I realized it’s more of an idiot savant: brilliant at solving problems but hilariously incapable of remembering its own mistakes. Maybe by the time you read this, ChatGPT will have gained a memory and learned how to consult a list of rules it’s broken 20 times already. But for now, you’ll want to approach Wi-Fi programming in simple, repeatable steps. These “sanity checks” will save you when things inevitably get messy.

That said, I still think ChatGPT is amazing. It’s helped me write in an afternoon what would have taken me a week, and it’s fantastic at admitting its screw-ups and helping you fix them. Just be patient—it’s a quirky teacher, but a helpful one.

So let’s dive into Wi-Fi programming, starting with the very first step: keeping secrets. Not the spy kind, but the kind that protects your Wi-Fi credentials while letting you share your code with the world.

**Keeping Secrets**

You may recall that in order to use wifi you have to first identify the name of the wifi network you want to use, this is known as the SSID. Look at your RPi 5 desktop and on the upper right you see that icon (figure 1) that shows you how powerful your wifi signal is. At the moment mine is two ‘bars’ and not so hot, but it's what it is.



Figure 1: Wifi signal strength

Click on this icon and a drop down menu will show the wifi networks available for you to use. After verifying that, indeed, the network you think you will be using is, in fact, available, then write your secrets.py file.

I’m going to assume that you know how to check for your wifi network and log onto it using a password. We will write software to do this for us since we don’t have a keyboard and a monitor for the picow.. We will refer to the network name as SSID (Service Set IDentifier - just because that is so much simpler than ‘network name’). In Thonny, write the following program and save it to your picow as secrets.py

| secrets = {  "ssid": "Your\_SSID",  "password": "Your\_PASSWORD", }    Figure 2: Thonny with ‘secrets.py |
| --- |

Check your picow directory to make sure it is there as shown in figure 3.

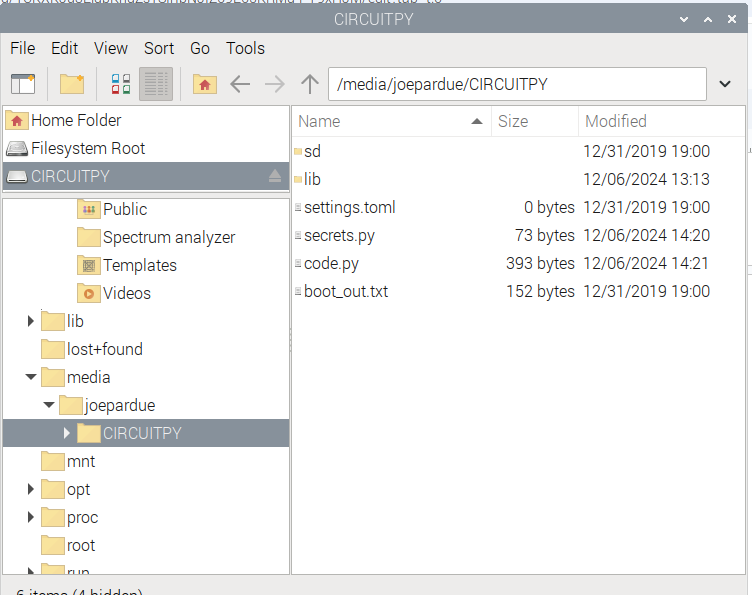


Figure 3: secrets.p in CIRCUITPY

Now we are ready to write the absolute minimal program for the picow that shows it can get on the wifi network.

| import wifi from secrets import secrets # Import the secrets dictionary print("SSID: ", secrets["ssid"]) print("SSID: ", secrets["password"])  print("Connecting to Wi-Fi...") try:  wifi.radio.connect(secrets["ssid"], secrets["password"])  print("Connected to Wi-Fi!")  print(f"IP Address: {wifi.radio.ipv4\_address}") except Exception as e:  print(f"Wi-Fi connection failed: {e}") |
| --- |

Save this as your code.py file. As a refresher, press the red circle with the white square in it to make sure nothing is running on the pico. Then make sure you’ve downloaded the most recent code.py file to the picow. Finally, cross your fingers and take a deep breath,hod it and press the green circle with the right facing white triangle.

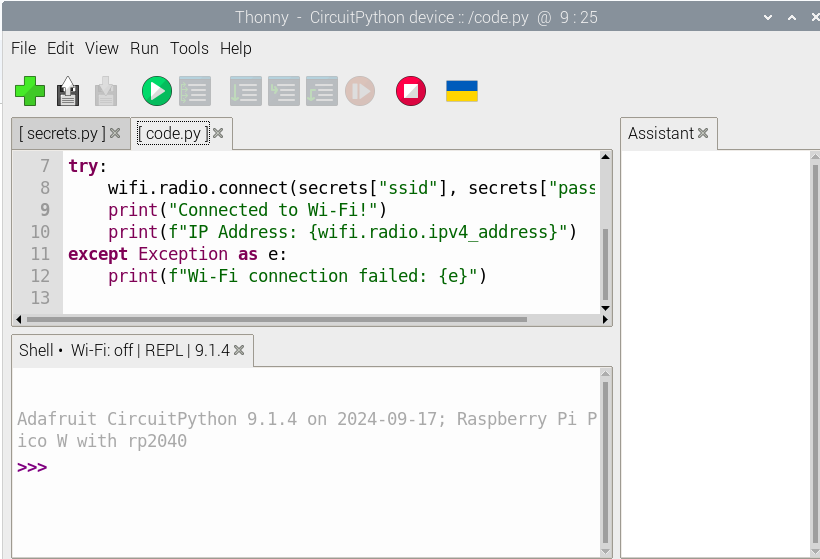


Figure 4

And, gasp… You have hooked up to your wifi as shown in Figure 5!

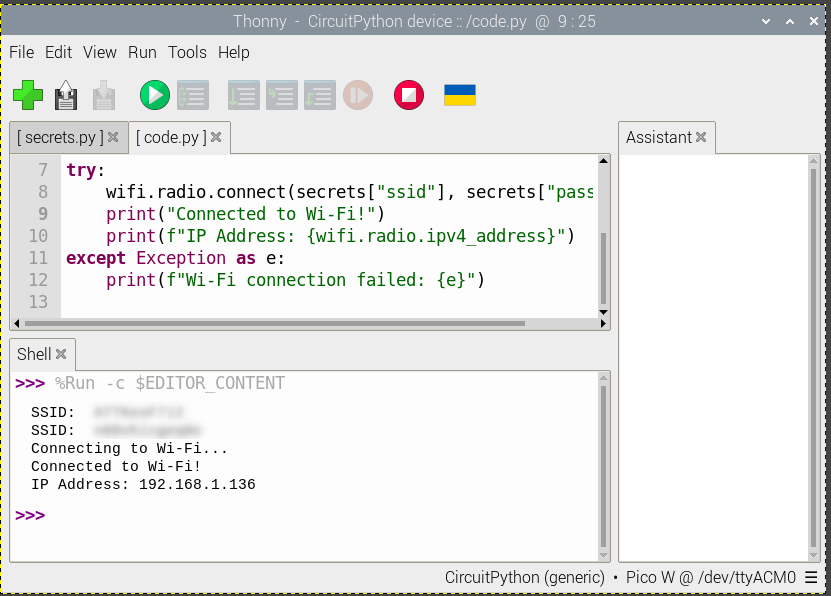


Figure 5: It’s alive!

If it isn’t alive, well poop. Now is the time to get very acquainted with ChatGPT, it will try its damnedest to help you get to this state. Good Luck.

**Picow Says ‘Hello, World!” on a Web Page**

| Copy this to the Thonny code.py file:   import wifi import socketpool from secrets import secrets # Import the secrets dictionary  # Connect to Wi-Fi print("Connecting to Wi-Fi...") try:  wifi.radio.connect(secrets["ssid"], secrets["password"])  print("Connected to Wi-Fi!")  print(f"IP Address: {wifi.radio.ipv4\_address}") except Exception as e:  print(f"Wi-Fi connection failed: {e}")  # Create a socket pool pool = socketpool.SocketPool(wifi.radio)  # Define the HTML content html = """ <!DOCTYPE html> <html> <head>  <title>Hello, World!</title> </head> <body>  <h1>Hello, World!</h1> </body> </html> """  # Start the web server print("Starting the web server...") try:  with pool.socket(pool.AF\_INET, pool.SOCK\_STREAM) as s:  s.bind(("0.0.0.0", 8080)) # Bind to port 8080  s.listen(1) # Listen for incoming connections  print("Listening on port 8080...")   while True:  conn, addr = s.accept()  print(f"Connection from {addr}")    # Read the request  try:  buffer = bytearray(1024)  size = conn.recv\_into(buffer)  request = str(buffer[:size], "utf-8")  print(f"Request: {request}")   # Send the HTTP response  response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n" + html  conn.sendall(response.encode("utf-8"))  except Exception as e:  print(f"Error handling request: {e}")  finally:  conn.close() except Exception as e:  print(f"Server error: {e}") |
| --- |

Again, download the code.py file, make sure to click the red button before the green. Cross your fingers and…

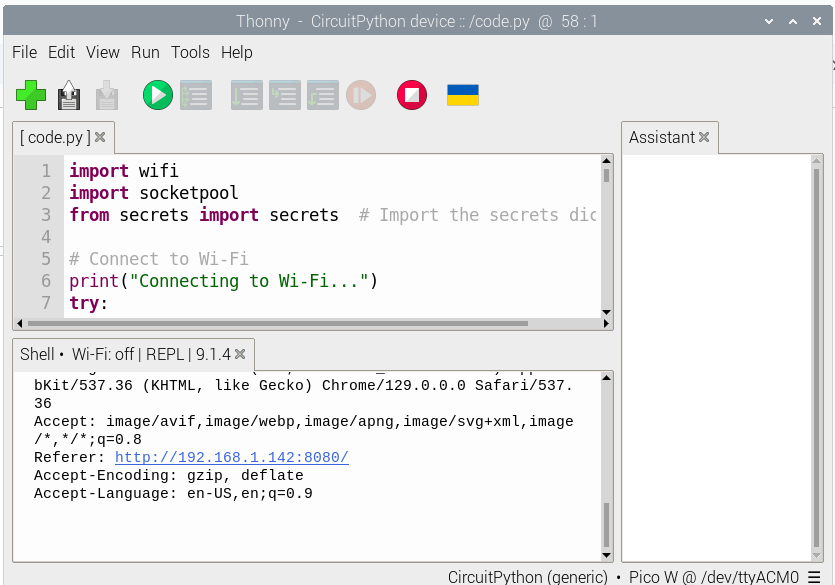


Figure 6: AND its says ‘Hello, World!’

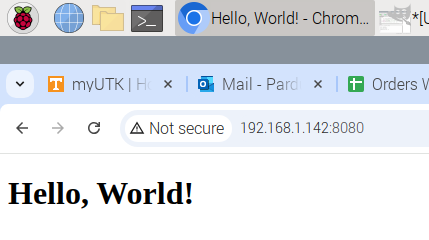


Figure 7: Whew, yes Hello, World!

In case you wonder why I’m being so overly emotive on this, it took me the better part of the afternoon ‘collaborating’ with ChatGPT to get this very simple, but critical piece of code running. Now let’s control an LED!’

**Controlling the Picow LED**

You know the drill. Here is the code:

| import wifi import socketpool import board import digitalio from secrets import secrets # Import the secrets dictionary  # Initialize the onboard LED led = digitalio.DigitalInOut(board.LED) led.direction = digitalio.Direction.OUTPUT  # Connect to Wi-Fi print("Connecting to Wi-Fi...") try:  wifi.radio.connect(secrets["ssid"], secrets["password"])  print("Connected to Wi-Fi!")  print(f"IP Address: {wifi.radio.ipv4\_address}") except Exception as e:  print(f"Wi-Fi connection failed: {e}")  # Create a socket pool pool = socketpool.SocketPool(wifi.radio)  # Define the HTML content with a button to toggle the LED def **generate\_html**(led\_state):  led\_status = "ON" if led\_state else "OFF"  html = f"""  <!DOCTYPE html>  <html>  <head>  <title>Pico W LED Control</title>  </head>  <body>  <h1>Pico W LED Control</h1>  <p>The LED is currently <strong>{led\_status}</strong>.</p>  <form action="/toggle" method="post">  <button type="submit">Toggle LED</button>  </form>  </body>  </html>  """  return html  # Start the web server print("Starting the web server...") try:  with pool.socket(pool.AF\_INET, pool.SOCK\_STREAM) as s:  s.bind(("0.0.0.0", 8080)) # Bind to port 8080  s.listen(1) # Listen for incoming connections  print("Listening on port 8080...")   while True:  conn, addr = s.accept()  print(f"Connection from {addr}")    # Read the request  try:  buffer = bytearray(1024)  size = conn.recv\_into(buffer)  request = str(buffer[:size], "utf-8")  print(f"Request: {request}")   # Check if the request is to toggle the LED  if "POST /toggle" in request:  led.value = not led.value # Toggle the LED state   # Send the HTML response  response = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n" + generate\_html(led.value)  conn.sendall(response.encode("utf-8"))  except Exception as e:  print(f"Error handling request: {e}")  finally:  conn.close() except Exception as e:  print(f"Server error: {e}") |
| --- |

Click on the blue line: <http://192.168.1.142:8080/toggle> and it might take you to the web page.

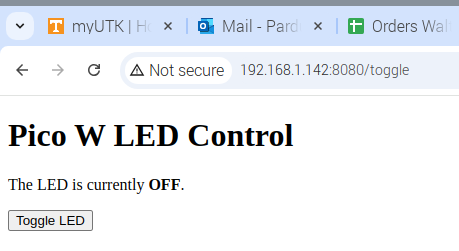


Figure 8: Let there be light, and there was light.

Let me follow up here with the admission at this point I was having trouble with Google Chrome on both my RPi5 and my Android phone. Both wanted to search for the <http://192.168.1.142:8080/toggle> as a search term rather than as an IP address. I had to fiddle with it a bit, finally just typing in: <http://192.168.1.142:8080> without the \toggle to get it to work. But we are certainly making progress.

And now with our new God-like power, we turn the picow LED on and off demonstrating that, yes, we can talk to a picow with a browser on our computer or phone and actually control real world things at a distance.