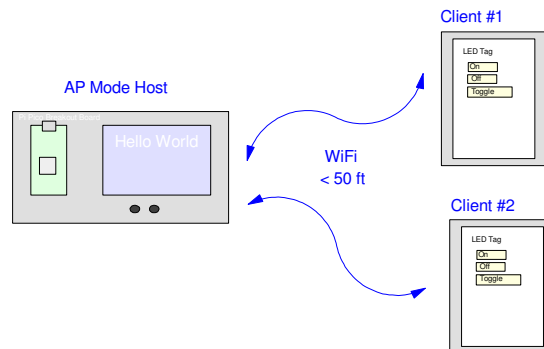


32. WiFi & AP Mode

Introduction:

The Raspberry Pi-Pico W has WiFi capabilities. This allows you to create your own WiFi network independent of having access to the internet, as well as the ability to connect to the internet. This also gives you a range of about 50 feet.



In AP mode, the Pico acts as a server, hosting a WiFi network for clients

In this lecture, we look at setting up a stand-alone network in AP mode which can have up to five devices attached to your Pi-Pico. At the end of this lecture, we'll create a web-page which has data that is updated every time you refresh the screen.

The screenshot shows a web browser interface with a URL bar displaying 'Not secure http://192.168.4.1'. The page title is 'Weather Data'. Below the title, it says 'Random Numbers for Demonstration Purposes'. There is a table with three columns: 'Parameter', 'Value', and 'Units'. The table contains four rows of data.

Parameter	Value	Units
Temperature	0.9815849	F
Humidity	0.5360076	%
Pressure	0.09588552	hPa
Light	0.308732	Lux

Goal of lecture #32: Create a web page in AP mode with live data

Note: Much of this information in this lecture comes from

- <https://www.youtube.com/watch?v=cZNoXXIEPbg>
- <https://medium.com/@shilleh/creating-a-wireless-network-with-raspberry-pi-pico-w-part-1-c896211f2bd6>
- <https://www.w3schools/html/>

Creating a Local Network

Let's start with creating a local network on your Pi-Pico that contains a single web page that says *Hello World*. The subroutine `web_page()` creates a text file using html coding to do this (more on this later).

```
import network
import time
import socket

def web_page():
    x = "<html><body><h1>Hello World</h1></body></html>"
    return(x)
```

Python code for creating a web page which says *Hello World*

While this works, it gets rather clunky for larger html files. An equivalent way to do this is to create a file on your Pi-Pico, such as *HelloWorld.html*

```
<html>
<body>
<h1>Hello World</h1>
</body>
</html>
```

html code for creating a web page which says *Hello World*

then read this file as a text string and return it. Note that

- `read()` returns the text file as a single string (good in this case).
- The carriage-return line-feeds need to be removed.

```
def web_page():
    f = open("HelloWorld.html", "rt")
    x = f.read()
    x = x.replace('\r\n', ' ')
    return(x)
```

Python code for reading in an html text file and returning it as a string

To create a local network, define the network's name and password.

- `network.WLAN` creates a local area network
- `config()` sets the network name and password
- `active(True)` starts the process of activating the LAN

```
ssid = 'Pico-Network'
password = 'PASSWORD'

ap = network.WLAN(network.AP_IF)
ap.config(essid=ssid, password=password)
ap.active(True)

while ap.active() == False:
    pass
print('AP Mode Is Active, You can Now Connect')
print('IP Address To Connect to:: ' + ap.ifconfig()[0])
```

Python code for creating a local area network in AP mode

Once the LAN is active,

- `socket.socket()` creates a new socket for this network
- `bind()` locks in the address for this web page

- *listen(5)* determines how many devices can connect to this LAN (five in this case)

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
```

Once active, lock the address and allow five clients

At this point, you can now receive and respond to pings from devices

s.accept() waits until you get a query from a device attached to the network (such as hitting refresh). This returns two parameters

- *conn* The status of the connection
- *addr* The address of the device who sent the message. Not the second byte is a counter.

```
conn, addr = s.accept()
print('conn = ', conn)
print('addr = ', addr)
print('Got a connection from %s' % str(addr))
```

shell

```
conn = <socket state = 3 timeout=-1 incoming=2000d1d8 off=0>
addr = ('192.168.4.16', 57986)
Got a connection from 192.168.4.16
```

Once set up, the address of the LAN is displayed in the shell window

To reply, a valid html page is sent back with *send()* then the communication is closed.

```
response = web_page()
conn.send(response)
conn.close()
```

The whole program looks like the following:

```
import network
import time
import socket

def web_page():
    f = open("HelloWorld.html", "rt")
    x = f.read()
    x = x.replace('\r\n', ' ')
    return(x)

ssid = 'Pico-Network'
password = 'PASSWORD'

ap = network.WLAN(network.AP_IF)
ap.config(essid=ssid, password=password)
ap.active(True)

while ap.active() == False:
    pass
print('AP Mode Is Active, You can Now Connect')
print('IP Address To Connect to:: ' + ap.ifconfig()[0])

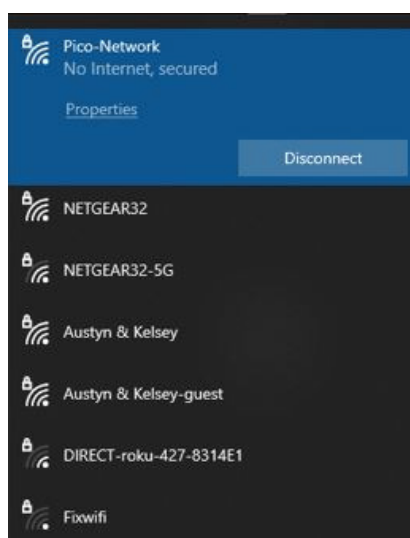
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while(1):
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    print('Content = %s' % str(request))
    response = web_page()
    conn.send(response)
    conn.close()
```

shell

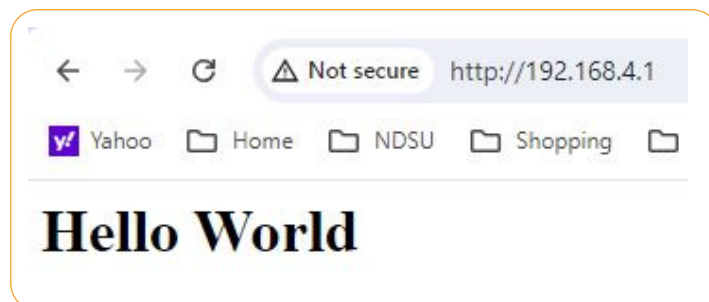
```
AP Mode Is Active, You can Now Connect
IP Address To Connect to:: 192.168.4.1
```

If you look for WiFi network, you should see Pico-Network



Connect to the name of your LAN (Pico-Network in this example)

If you connect to web page 192.168.4.1, you will see the html image



If all goes well, you'll see the html page

You will also see the reply from the connection in the shell window. This doesn't mean much here, but later on you can use this to pass data back and forth.

```
AP Mode Is Active, You can Now Connect
IP Address To Connect to:: 192.168.4.1
Got a connection from ('192.168.4.16', 41178)

Content = b'GET / HTTP/1.1\r\nHost: 192.168.4.1\r\nConnection:
keep-alive\r\nCache-Control:
max-age=0\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent:
Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/127.0.0.0 Mobile Safari/537.36\r\nAccept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.7\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language:
en-US,en;q=0.9\r\n\r\n'
```

Shell window if everything goes well

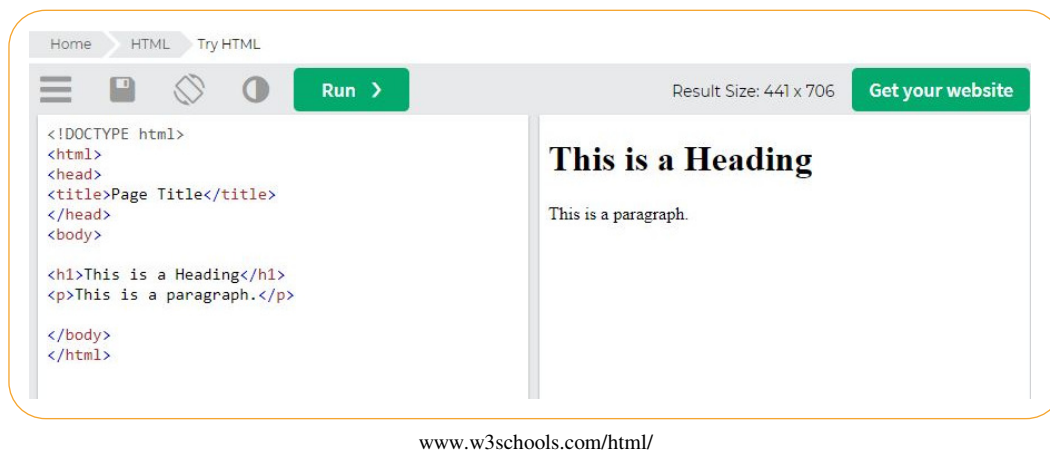
HTML Coding

Backing up a bit, in the previous example, html code was used to display *Hello World*:

```
<html>
<body>
<h1>Hello World</h1>
</body>
</html>
```

You can do a lot more than this with html coding. You can even take several courses on html programming.

Note: A good place to go for learning html coding is w3schools. This has several lessons on html programming as well as interactive windows where you can see what your code produces:



Here, we're just going to go over some basics of html programming, eventually allowing you to generate a web page with

- headings,
- paragraphs, and
- table.

A couple of things to note:

- html is not case sensitive
- Single quotes and double quotes are interchangeable

For example, to create a string which contains quote symbols, you could use

```
x = "To quote Charlie Brown, 'Rats.'"
```

is the same as

```
x = 'To quote Charlie Brown, "Rats."'
```

The basic format for a html page is as follows:

```
<!DOCTYPE html>
<html>
<body>

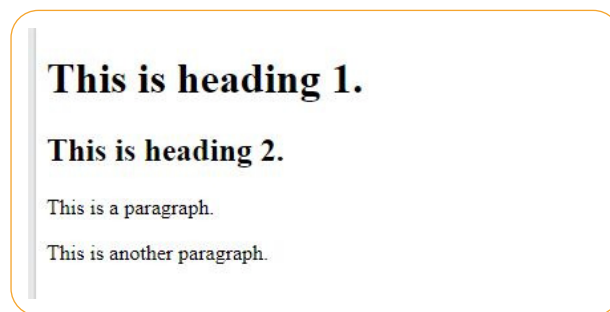
<h1>This is heading 1.</h1>
<h2>This is heading 2.</h2>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

Basic html page

which displays as



result of previous html code

Some of the things you can add to his file are as follows:

Adding a link

```
<a href="http2://www.w3schools.com">This is a link</a>
```

Adding a carriage return

```
<br>
```

Hyperlink <a>

Adding an image <src>

```

```

The src Attribute

HTML images are defined with the `img` tag, and the filename of the image source is specified in the `src` attribute:



adding an image to a web page

Alternate text <alt>

If the image can't be displayed, the text to display instead

```
alt="Glacier NP"
```

Style: Set the color

```
<p style="color:red;">
```

Style: Set the font size

```
<p style="font-size:20px;">Paragraph in 20 point font.<\p>
<p style="font-size:300%;">Paragraph 300% font.<\p>
```

Style: Set background color

```
<body style="background-color:powderblue;">
<h1 style="background-color:tomato;">Heading</h1>
```

Style: Font

```
<h1 style="font-family:ariel;">This is a heading</h1>
```

- Some fonts available include

- Arial
- **Arial Black**
- Comic Sans
- Courier
- Georgia
- Helvetica
- **Imact**
- Palatino
- Tahoma
- Trebuchet MS
- Times New Roman
- Verdana

'Twas brillig and the slighy toves
Did gyre in the gimple in the wabe
 All mimsy were the borogoves
 And the mome rathe outgrabe.,
 "Beware the jabberwock, my son!
 The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun,
 The frumious Bandersnatch!"
 He took his vorpal sword in hand;
 Long time the manxome foe he sought--
 So rested be by the Tumtum tree
 And stood a while in thought

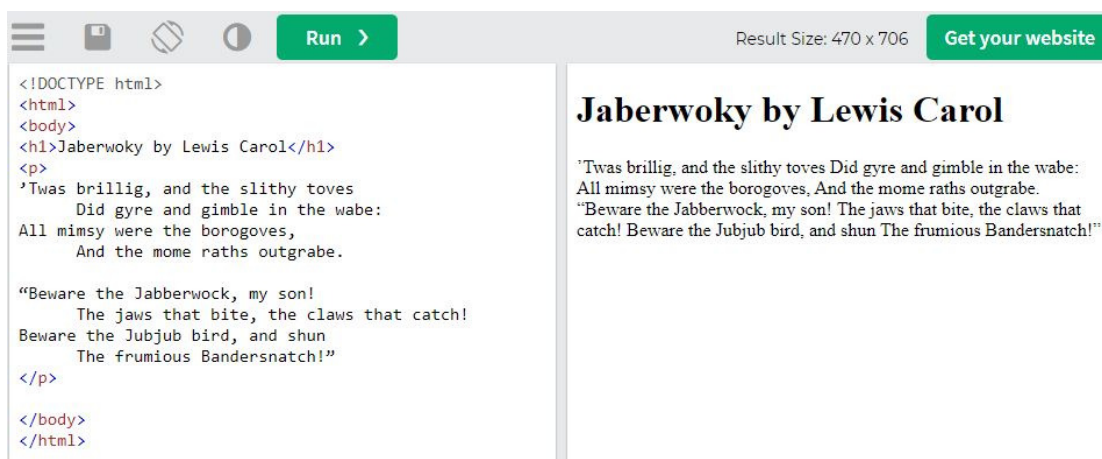
Jaberworky by Lewis Carol

Style: Text Align

`<p style="text-align:center;">Centered paragraph.</p>`
options: left, center, right

Paragraphs `<p>`

- double spaces, carriage returns are ignored - have no effect on the resulting display



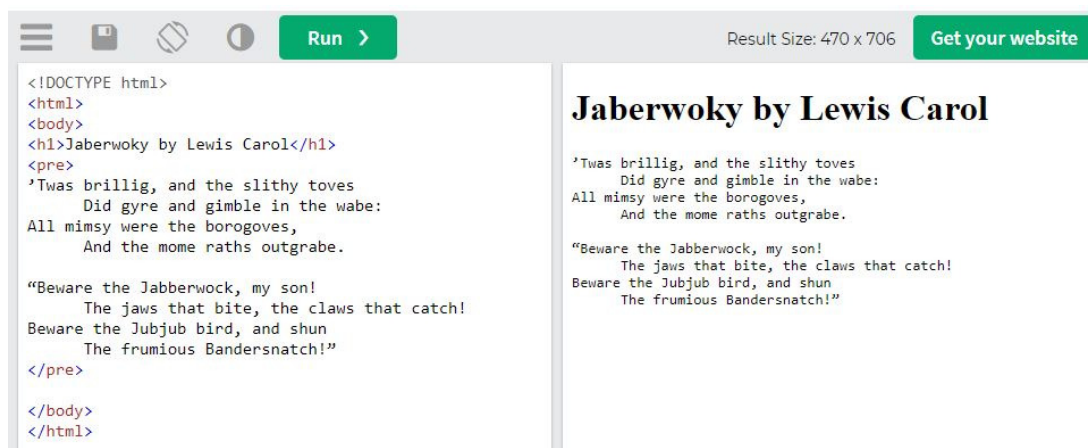
with `<p>`, double spaces, indentation, and carriage returns are ignored
www.w3schools.com/html/

Horizontal Rule `<hr>`

- draw a horizontal line

Preformatted Text `<pre>` `<\pre>`

- `<p>` ignores carriage returns and spaces. If you want to preserve the spaces and carriage returns in the html files, use the `<pre>` command instead.



`<pre>` preserves double spaces, indentation, and carriage returns.
www.w3schools.com/html/

Formatting Text. Each of these are terminated with a back-slash (----)

- bold face
- also bold face
- <i> italic
- <mark> marked text
- <small> smaller text
- deleted text
- <sub> subscript
- <sup> superscript

Tables

Tables are a nice way to present information, such as weather data. The format for a table in html is:

```
<table>                                start of table
  <tr>                                start of row
    <th>Sensor</th>                  table heading, column #1
    <th>Reading</th>
    <th>Units</th>
  </tr>                                end of first row
  <tr>                                start of second row
    <td>Temp</td>                  table data
    <td>74.35</td>
    <td>F</td>
  </tr>                                end of second row
</table>                              end of table

<p>Example of html table</p>
```

which displays as

HTML Tables		
Sensor	Reading	Units
Temp	74.35	F
Example of html tables.		

result of previous html code

To add borders to a table, use the *border* statement. The following code adds a 1 pixel solid black border to the table, all rows, and all data cells

```
table, th, td {
  border: 1px solid black
}
```

HTML Tables

Sensor	Reading	Units
Temp	74.35	F

Example of html tables.

If you want to remove the double lines, use the *collapse* statement

```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

HTML Tables

Sensor	Reading	Units
Temp	74.35	F

Example of html tables.

Color can be added to the table. Color is a 24-bit number in the order of red - green - blue:

```
#FF0000      red
#00FF00      green
#0000FF      blue
#FFFFFF      white
```

For example, make the

- border dark green (#008800)
- heading background color dark pink (#FFDDDD)
- data background light pink (#FFEEEE)

would be

```
table, th, td {
  border: 1px solid white;
  border-collapse: collapse;
}
th, td {
  border-color: #008800;
}
th {
  background-color: #FFDDDD;
}
td {
  background-color: #FFEEEE;
}
```

HTML Tables

Sensor	Reading	Units
Temp	74.35	F

Example of html tables.

Displaying Data in a Table

Suppose you want to generate a display where the values change based upon current readings:

← → ↻ ⚠ Not secure http://192.168.4.1

Weather Data

Random Numbers for Demonstration Purposes

Parameter	Value	Units
Temperature	0.9815849	F
Humidity	0.5360076	%
Pressure	0.09588552	hPa
Light	0.308732	Lux

Example of displaying live data in a table

There are probably better ways to do this, but one solution is to create a table using dummy values for the data (aaaaa to ddddd in this case):

```
<!DOCTYPE html><html>
<head>
  <style>
    table { border-collapse: collapse; width: 80%; }
    th, td { text-align: left; padding: 8px; }
    tr:nth-child(even) { background-color: #EEDDDD; }
    th, td, p, h2 { font-size: 200%; }
  </style>
</head>
<body>
  <h2>Weather Data</h2>
  <p>Random Numbers for Demonstration Purposes</p>
  <table>
    <tr> <th>Parameter</th> <th>Value</th> <th>Units</th> </tr>
    <tr> <td>Temperature</td> <td> aaaaa </td> <td>F</td> </tr>
    <tr> <td>Humidity</td> <td> bbbbbb </td> <td>%</td> </tr>
    <tr> <td>Pressure</td> <td> cccccc </td> <td>hPa</td> </tr>
    <tr> <td>Light</td> <td> ddddd </td> <td>Lux</td> </tr>
  </table>
</body>
</html>
```

html code to create a table. aaaaa to ddddd will be replaced with real data

Passed data can replace these dummy variables with a modified *web_page()* routine:

```
def web_page(x0, x1, x2, x3):
    f = open("Table.html", "rt")
    x = f.read()
    x = x.replace('\r\n', ' ')
    x = x.replace('aaaaa', str(x0))
    x = x.replace('bbbbb', str(x1))
    x = x.replace('ccccc', str(x2))
    x = x.replace('ddddd', str(x3))
    return(x)
```

web_page() subroutine which inserts data into the web page

The main routine can then insert data into the web page (random numbers used here for illustration purposes).

```
while(1):
    conn, addr = s.accept()
    print('Got a connection from %s' % str(addr))
    request = conn.recv(1024)
    print('Content = %s' % str(request))
    response = web_page(random(), random(), random(), random())
    conn.send(response)
    conn.close()
```

Main routine which updates data every request

The resulting display shows data which is updated every time you refresh the screen (hit F5)

Parameter	Value	Units
Temperature	0.9815849	F
Humidity	0.5360076	%
Pressure	0.09588552	hPa
Light	0.308732	Lux

Resulting web page. Values are updated every time you refresh the display (F5)

Summary:

In AP mode, the Pi-Pico W sets up a stand-alone WiFi network other devices can connect to. By setting up a web page with parameters which vary based upon current readings, real-time data can be passed on this network.

Two-way communications is also possible - but left for a future lecture.

References

<https://www.youtube.com/watch?v=cZNoXXIEPbg>

<https://medium.com/@shilleh/creating-a-wireless-network-with-raspberry-pi-pico-w-part-1-c896211f2bd6>

<https://www.w3schools.com/html/default.asp>