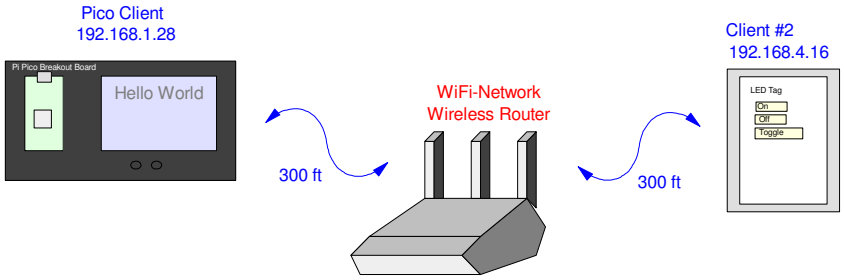
34. WiFi & Client Mode

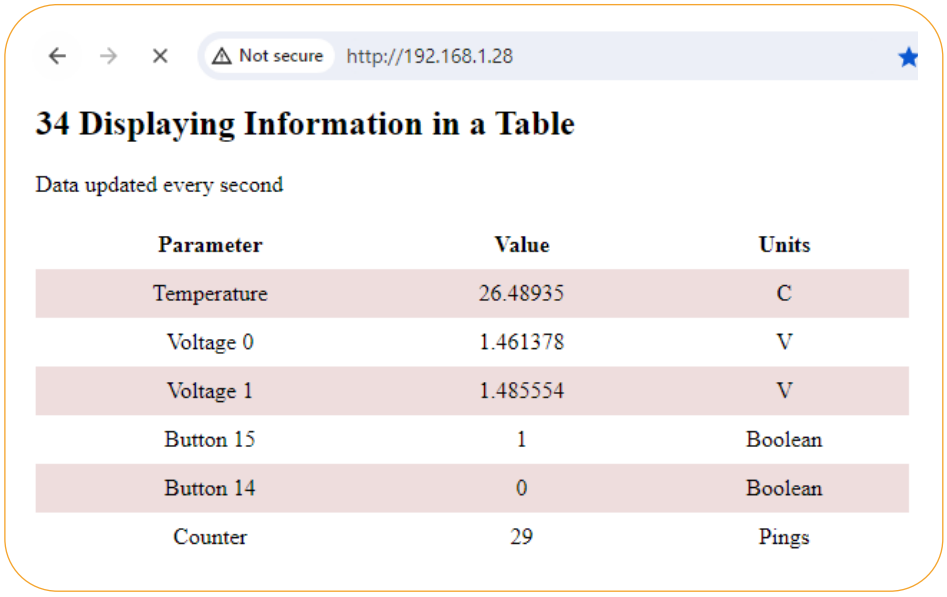
Introduction:

In the previous lectures, the Pico was set up in AP mode, making the Pico a WiFi server. In this lecture, the Pico is set up as a client connecting to a wireless network. This allows anyone on your server to access data from your Pico so long as they know the web address.



In Client Mode, the Pico connects to an existing WiFi network.
Other devices on this network can see and pass data to the Pico

At the end of this lecture, a web page will be created allowing you to see the status of your Pico board as follows:



Goal: Display the currant status of your Pico on the web

Connecting to a Router

The first step in connecting to a wireless network is to turn on the Pico's WiFi. There are two ways to do this.

- To set up the Pico as a client on an existing WiFi network (this lecture), use:

```
wlan = network.WLAN(network.STA_IF)
```

- To set up the Pico as the host for a stand-alone AP network (previous lectures), use:

```
wlan = network.AP_IF(network.STA_IF)
```

The basic code for connecting to a router is as follows:

```
import network, rp2, time

rp2.country('US')
wlan = network.WLAN(network.STA_IF))
wlan.active(True)

ssid = 'xxxx'
password = 'xxxx'

wlan.connect(ssid, password)

while( (wlan.isconnected() == 0) and (wlan.status() > 0)):
    print('Waiting to connect')
    time.sleep(1)

if(wlan.status() == 3):
    print('connection successful')
    print(wlan.ifconfig())
```

shell

```
Waiting to connect
Waiting to connect
Waiting to connect
Connection successful
('192.168.1.28', '255.255.255.0', '192.169.1.1', '192.168.1.1')
```

Basic code for connecting to a wireless network as a client

What this code does and some options are as follows:

active()

```
wlan.active()                   return True if the network is active
wlan.active('up')               activate the network interface
wlan.active('down')             deactivate the network interface
```

connect(ssid, password)

```
wlan.connect(ssid, password)    try to connect to a WiFi network
```

disconnect()

```
wlan.disconnect()               disconnect from the current network
```

scan()

```
wlan.scan()                     scan for networks
                                returns names of networks
```

status()

```
wlan.status()                   returns the status of the network
                                -3  failed to connect due to password
                                -2  failed to connect - no such ssid
                                -1  failed to connect for other reasons
                                0   idle, no connection, no activity
                                1   trying to connect
                                3   connection successful
```

isconnected()

```
wlan.isconnected()              true if connected
                                false if not connected
```

ifconfig()

```
x = ifconfig()                  returns four parameters
                                IP address
                                subnet mask
                                gateway server
                                DNS server
```

config()

```
wlan.config(channel=11)
wlan.config('ssid')
wlan.config('channel')
```

config(pm)

```
wlan.config(pm = 16)            disable power management
wlan.config(pm = 10555714)      maximum performance
wlan.config(pm = 17)            balanced performance vs. power
```

UR Request

Once connected to the web, you can open and read web pages. For example, to read in the contents of BisonAcademy.com, use the following code

```
import network
import urequests

rp2.country('US')
wlan = network.WLAN(network.STA_IF)
wlan.active(True)

ssid = 'xxxx'
password = 'xxxx'
wlan.connect(ssid, password)

r = urequests.get("https://BisonAcademy.com")
print(r.content)
r.close()
```

The results of the web page will be shown in the shell window (not sure how this actually helps, but it's something you can do)

```
>>> %Run -c $EDITOR_CONTENT
MPY: soft reboot
<!DOCTYPE html><html lang="en"><head><script
src="/gdpr/gdprscript.js?buildTime=1722004568&hasRemindMe=true&ste
alth=false"></script>
:
:
:
```

Creating a Web Page: Hello World

Starting with everyone's favorite example, let's create a web page that says *Hello World*

Goal: Create a web page that says *Hello World*

Start with the html code. This can be the same we used before:

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

HelloWorld.html Code for displaying *Hello World*

The main routine starts the same as before along with a subroutine *web_page()* which reads in the html file (again, same as before)::

```
import netman
import socket
from machine import Pin

ssid = 'xxxxx'
password = 'xxxxx'
country = 'US'

wifi_connection = netman.connectWiFi(ssid,password,country)

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

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

print('listening on', addr)
```

Start of main routine for creating a web page

The main routine then simply

- Waits for a ping in *wlan.accept()*
- Then echos back the html code with *cl.send(response)*
- Then closes the current ping

```
while(1):
    cl, addr = wlan.accept()
    print('client connected from', addr)
    request = cl.recv(1024)

    response = web_page()

    cl.send(response)
    cl.close()
```

Main loop: reply with the web page every time you're pinged

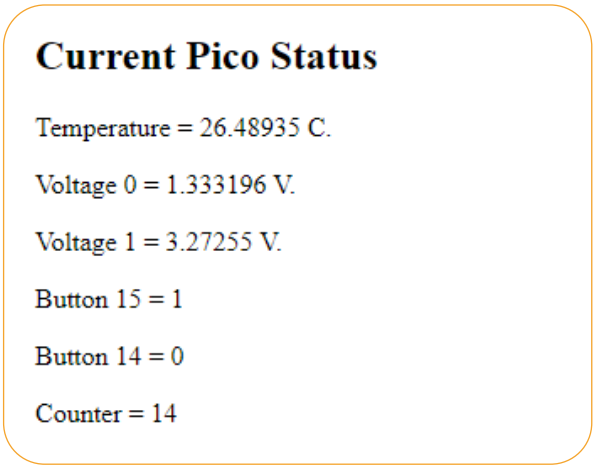
The shell window tells you each time the client was pinged (such as hitting F5 to refresh). This doesn't tell you much other than you're connected

```
MPY: soft reboot
waiting for connection...
waiting for connection...
waiting for connection...
connected
ip = 192.168.1.28
listening on ('0.0.0.0', 80)
client connected from ('192.168.1.3', 52527)
client connected from ('192.168.1.3', 52528)
client connected from ('192.168.1.3', 52529)
:
:
```

Shell window: Reports the status of the connection

Displaying Information in Text Format

By modifying the html code, you can display information. For fun, let's create a display like the following:



Goal: Display the status of the Pico in text format

Using a trick from before, the html code uses dummy variables for the data to be displayed

```
<!DOCTYPE html>
<html>

<head>
    <title>34 Text</title>
    <meta http-equiv="refresh" content="1">
</head>

<body>
    <h2>Current Pico Status</h2>
    <p>Temperature = aaaaa C.</p>
    <p>Voltage 0   = bbbbb V.</p>
    <p>Voltage 1   = ccccc V.</p>
    <p>Button 15   = ddddd </p>
    <p>Button 14   = eeeee </p>
    <p>Counter     = fffff </p>
</body>

</html>
```

34 Text html Code for displaying data with dummy variables for the data

One addition is the meta command shown in bold. This command causes the web page to update every one second - similar to hitting F5 refresh every second.

The routine *web_page()* is similar to before, except that data is passed which replaces the dummy variables.

```
def web_page(Temp, V0, V1, B15, B14, N):
    f = open("34 Text.html","rt")
    x = f.read()
    x = x.replace('\r\n',' ')
    x = x.replace('aaaaa', str(Temp))
    x = x.replace('bbbbb', str(V0))
    x = x.replace('ccccc', str(V1))
    x = x.replace('ddddd', str(B15))
    x = x.replace('eeeee', str(B14))
    x = x.replace('fffff', str(N))
    return(x)
```

Modified *web_page()* routine which inserts data into the previous web page

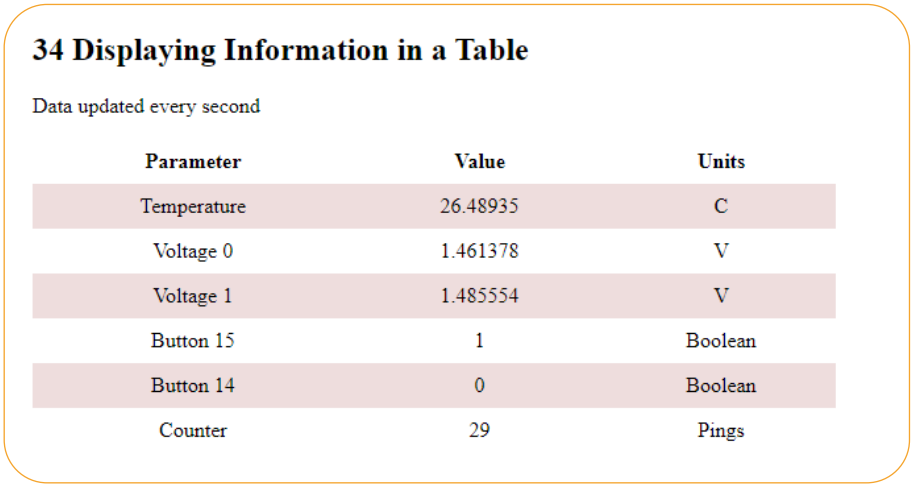
The main loop then

- Waits for a ping (which will happen every second),
- Collects data, then
- Replies with the updated web page

```
a2d0 = ADC(0)
a2d1 = ADC(1)
a2d4 = ADC(4)
k = 3.3 / 65520
B15 = Pin(15, Pin.IN)
B14 = Pin(14, Pin.IN)

N = 0
while(1):
    cl, addr = wlan.accept()
    print('client connected from', addr)
    request = cl.recv(1024)

    N += 1
    V0 = a2d0.read_u16()*k
    V1 = a2d1.read_u16()*k
    a4 = a2d4.read_u16()
    Temp = 0.02927*(14940 - a4)

    response = web_page(Temp, V0, V1, B15.value(), B14.value(), N)

    cl.send(response)
    cl.close()
```

Displaying Data in Table Format

Finally, display the same information as before, only in a table which is updated every second.



34 Displaying Information in a Table

Data updated every second

Parameter	Value	Units
Temperature	26.48935	C
Voltage 0	1.461378	V
Voltage 1	1.485554	V
Button 15	1	Boolean
Button 14	0	Boolean
Counter	29	Pings

Goal: Display data in a table which is updated every second

Since the same data is being displayed, the main routine remains unchanged. All that changes is the html code. Once again, dummy variables are used for the data

```
<!DOCTYPE html><html>
<head>
  <title>34 Table</title>
  <meta http-equiv="refresh" content="1">
  <style>
    table {  border-collapse: collapse;   width: 80%; }
    th, td {  text-align: center;  padding: 8px; }
    tr:nth-child(even) { background-color: #EEDDDD; }
    </style>
  </head>
<body>
  <h2>34 Displaying Information in a Table</h2>
  <p>Data updated every second</p>
  <table>
    <tr> <th>Parameter</th>   <th>Value</th>   <th>Units</th> </tr>
    <tr> <td>Temperature</td> <td> aaaaa </td> <td>C</td> </tr>
    <tr> <td>Voltage 0</td>   <td> bbbbb </td> <td>V</td> </tr>
    <tr> <td>Voltage 1</td>   <td> ccccc </td> <td>V</td> </tr>
    <tr> <td>Button 15</td>   <td> ddddd </td> <td>Boolean</td> </tr>
    <tr> <td>Button 14</td>   <td> eeeee </td> <td>Boolean</td> </tr>
    <tr> <td>Counter</td>   <td> fffff </td>   <td>Pings</td>   </tr>
    </table>
  </body>
</html>
```

Summary

In this lecture, the procedure to set up your Pico as a client on an existing WiFi network was presented. This allows anyone who knows the Pico's URL address to see the status of the Pico.

References

- pepe80.com
- https://www.w3schools/tags/att_input_type