Bohunt School (Wokingham)

Connect Pico-W to best WiFi network

I've recently purchased some Pico-Ws. They are like normal Pico(es) but have WiFi and Bluetooth capability (although the BT drivers are not implemented at the moment), nevertheless they could be very useful in the IoT Club.

This simple guide documents my first steps in becoming familiar with the Pico-W's networking commands and shows in a step-by-step way how to get the Pico-W to find and connect to a network with the best signal strength.

As Python is one of the main programming languages you learn at school, I decided to install Micro-Python on the Pico-W. The steps to do this are well documented on the InterWeb (click here), so I won't repeat them here.

OK - let's get started

The Pico-W

Here's a photo of a Pico-W.

As you can see it looks very similar to a normal Pico, the main difference is the metal enclosure towards the left-hand end of the board - this houses the electronics for the WiFi and at the extreme edge of the board is the trapezoidal antenna.

IDE (Integrated Development Environment)

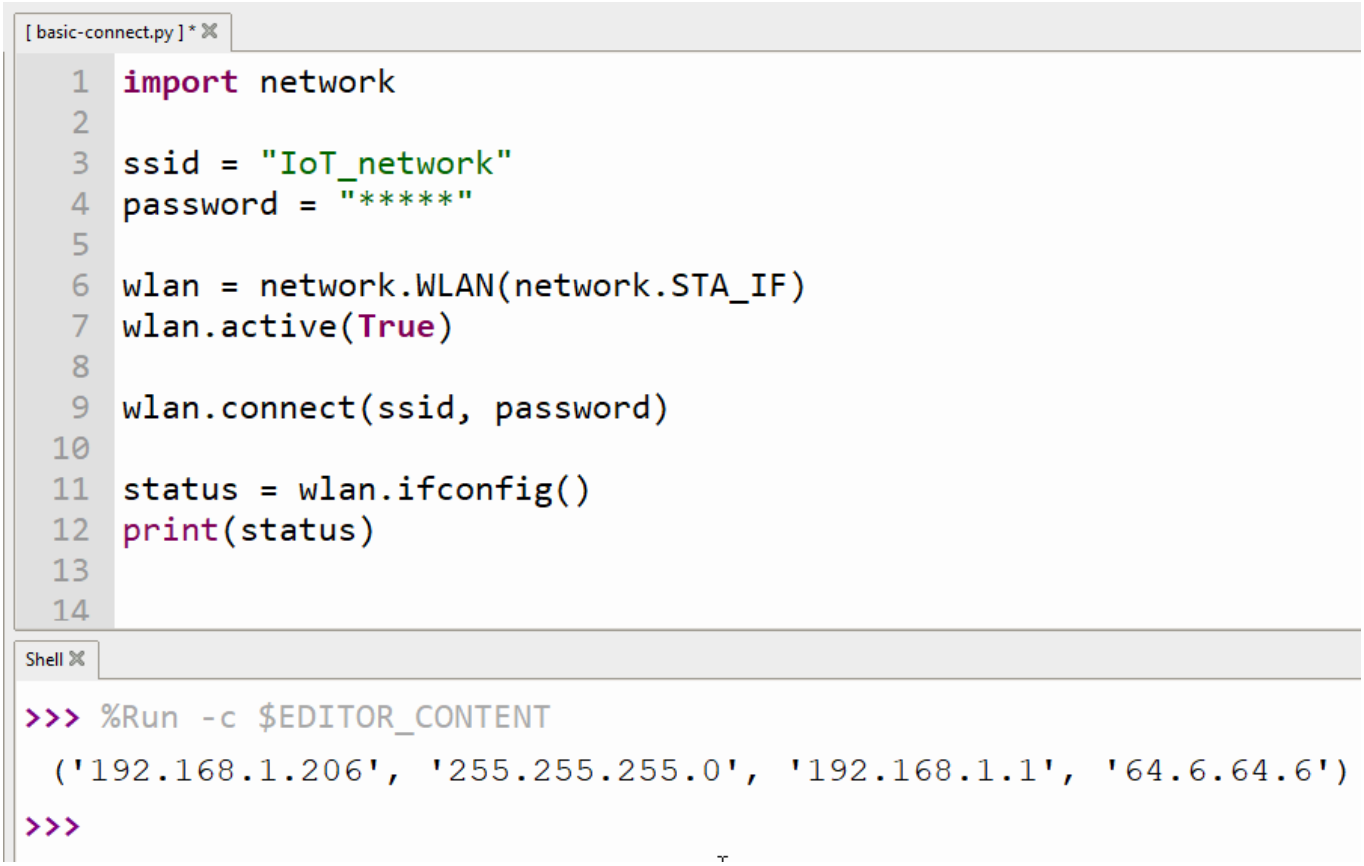
There are two brilliantly simple IDE packages to choose from, namely Thonny and the Mu Editor.

In this guide I'll be using Thonny.

Bohunt School *(Wokingham)*

Connect Pico-W to best WiFi network

Using this picture as a guide, type this code into Thonny's editor window.



Line-1 imports the network library followed by a definition of the IoT Club's network and password. Not real. See Mr D for the real settings.

Line-6 creates a station interface using the WLAN class so we can join a network. You can also configure the Pico-W to act as an access point, which will be covered in a later talk and tutorial-guide.

Line-7 activates the WiFi electronics inside the Pico-W.

Line-9 starts a connection to the network specified in Line-3.

For this simple program it is assumed the connection is successful, which means Line-11 and Line-12 can be used to show the connection information.

The lower part of the picture shows the results when the program is executed.

The Pico-W has been assigned an IP address of 192.168.1.206 by the IoT Club's router which acts as a gateway at address 192.168.1.1 The last address (64.6.64.6) is the DNS server the Club uses. If you missed the talk where these items were discussed, have a quick word with Mr D for a brief explanation.

As you can see the connection was made successfully to the WiFi. The next page shows how to handle and report a failed connection.

Handling a failed connection

Using this picture as a guide, type this code into Thonny's editor window.

```
1  import network
2  import time
3
4  ssid     = "IoT_Network"
5  password = "*****"
6
7  wlan = network.WLAN(network.STA_IF)
8  wlan.active(True)
9  wlan.connect(ssid, password)
10
11 # Wait for connect or fail
12 max_wait = 10
13 while max_wait > 0:
14     if wlan.status() < 0 or wlan.status() >= 3:
15         break
16     max_wait -= 1
17     print('waiting for connection...')
18     time.sleep(1)
19
20 # Handle connection error
21 if wlan.status() != 3:
22     raise RuntimeError('network connection failed')
23 else:
24     print('connected')
25     status = wlan.ifconfig()
26     print( 'ip = ' + status[0] )
27
28 while (True):
29     print("WiFi status " + str(wlan.status()) )
30     time.sleep(5)
31
```

Every thing is the same up to Line-12 where a loop is created that checks every second to see if there has been a successful connection (code 3) or some sort of failed connection (indicated by a negative value).

Lines-21 onwards report on the type of connection that has been made.

Bohunt School *(Wokingham)*

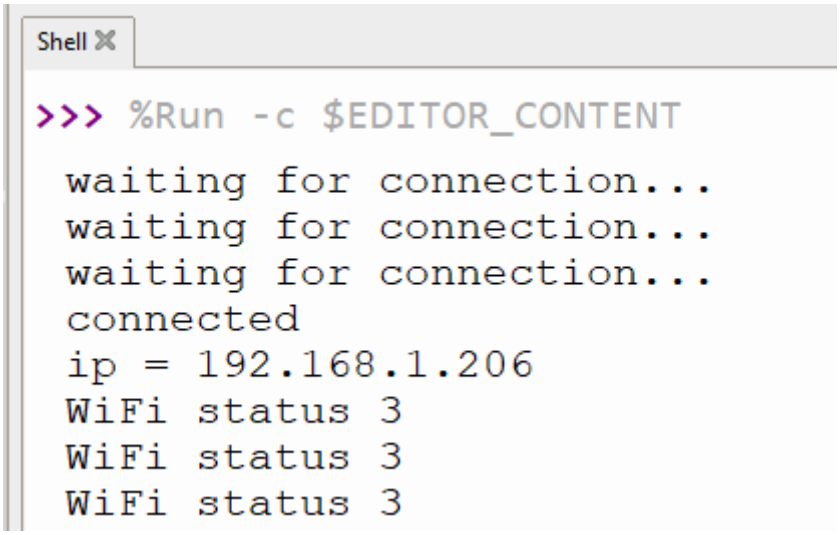
Connect Pico-W to best WiFi network

Connection status codes

Code 0 Link Down
Code 1 Link Join
Code 2 No IP
Code 3 Link Up <<< This is the one we are hoping for

Code -1 Link Failed
Code -2 Link No Net
Code -3 Bad authorisation <<< Probably means an incorrect password

Here a picture of the connection sequence with this Python script.



Scanning for a network

Up to this point you knew the name of the network you wanted to connect to. But what would you do if you didn't know the name of the WiFi network?

Luckily there is a command wlan.scan() that will scan for available networks.

Here's how it can be used in a Micro-Python script.

```
1  import network
2  from time import sleep
3
4  wlan = network.WLAN(network.STA_IF)
5  wlan.active(True)
6
7  wifilist = wlan.scan()
8  for i in wifilist:
9      print(i[0].decode() + "    " + str(i[3]))
10
```

The scan command returns an array of tuples where column 0 is the name of a WiFi network while column 3 is the signal strength. More details about the scan command are given at the very end of this guide.

Here's what the report looks like when it was run at Mr D's house.

```
Teamwork    -68
AProof    -48
APblue    -64
APyellow    -67
Spencer    -91
APstudy    -90
APloft    -81
VM6425637    -74
Virgin Media    -75
```

Running the script at the IoT Club would be a bit boring as it would only show two networks - the IoT Club's network and the school's network.

Connecting to the network with the strongest signal

The above picture shows the network with the strongest signal (-48 dB) is from the AProof network. Some of the networks, like Spencer, VM6425637 and Virgin Media, are foreign and not part of Mr D's home network.

This means when a scan is performed, the results need to be filtered so foreign networks are removed (as a connection to these networks is not possible).

As shown below, a Micro-Python dictionary is used to specify the networks that are available and can be connected to by the Pico-W.

```
12 Details of the networks is held in a Python dictionary called myNetworks
13 inside the file called 'secret_networks.py'
14
15 This is the format...
16 myNetworks = {
17     'networkA': 'passwdA',
18     'networkB': 'passwdB',
19
20     'networkZ': 'passwdZ'
21 }
```

Note:
This is just a pro forma to show the layout. You will need to enter the names for your 'real' networks and 'real' passwords. This Micro-Python dictionary is held in a file called 'secret_networks.py' on your filing system.

Bohunt School *(Wokingham)*

Connect Pico-W to best WiFi network

As this Python script will get quite lengthy, it has been broken into a series of 'functions'. See Mr D if you missed the talk about using functions.

Here's the first part of the program that imports various libraries. It also reads in the Micro-Python dictionary that was discussed on the previous page.

```
25 import network
26 from time import sleep
27 from secret_networks import myNetworks
28
```

The first function called 'find_best_network' is basically an expanded copy of what was written on page-4

```
52 def find_best_network():
53     global bestNetwork, bestSignal
54     wifilist = wlan.scan()
55
56     best_rssi_so_far = -100
57
58     for i in wifilist:
59         if i[3] > best_rssi_so_far and i[0].decode() in myNetworks:
60             best_rssi_so_far = i[3]
61             bestNetwork = i[0].decode()
62             bestSignal  = str(i[3])
63
64         print(i[0].decode() + "    " + str(i[3]))
65
66     print("\nBest network to join is "+ bestNetwork + " with a signal strength of " + bestSignal )
```

Line-56 initially sets the variable 'best_rssi_so_far' to -100 as this is the worst signal strength that could be obtained. As the program goes round the 'for' loop it checks the WiFi strength of each network against this value. If the WiFi strength is better than -100 (which it should be) then the variable is overwritten on Line-60 and the 'bestNetwork' and 'bestSignal' details are recorded on lines 61 and 62. When the program leaves after Line-66 these variables are holding the details of the "best" network.

Going back to look at Line-59 you can see the second part of the 'if' test (just after the 'and' operator) is checking if the current network (with the best signal strength) is actually in the list of available networks (in the Micro-Python dictionary).

If you are lost at this point or need further explanation of any piece of code, have a quick word with Mr D or the Club's tutor.

The next function, called 'connect_to_best_network' is basically a cut-down version of what appears on page-3. This function expects two parameters to be passed to it at the place holders… *best_ssid* and *best_passwd*

```
29  def connect_to_best_network(best_ssid, best_passwd):
30      wlan.connect(best_ssid, best_passwd)
31
32      # Wait for connect or fail
33      max_wait = 10
34      while max_wait > 0:
35          if wlan.status() < 0 or wlan.status() >= 3:
36              break
37          max_wait -= 1
38          print('waiting for connection...')
39          sleep(1)
40
41      if wlan.status() == -3:
42          raise RuntimeError('Password is incorrect')
43      elif wlan.status() != 3:
44          print(wlan.status() )
45          raise RuntimeError('network connection failed')
46      else:
47          print('connected')
48          status = wlan.ifconfig()
49          print( 'ip = ' + status[0] )
50
```

The final piece of code is the Main program that calls the above functions.

```
68  # Main program
69  wlan = network.WLAN(network.STA_IF)
70  wlan.active(True)
71  current_network = "never_connected"
72
73  while (True):
74      find_best_network()
75      if bestNetwork == current_network:
76          print('You are already connected to the best network')
77      else:
78          wlan.disconnect() # Disconnect from current network
79          current_network = bestNetwork
80          connect_to_best_network(bestNetwork, myNetworks[bestNetwork])
81
82      sleep(5)
83
84  # End of main program
85
```

The Main program is fairly straightforward. It starts-off by setting the variable 'current_network' to the string "never_connected".

Then on Line-74 the function 'find_best_network' is called to find the best network. This is then compared on Line-75. If the network names are the same (not very likely on the first run through the loop) then nothing is done.

If however the network names are different (VERY likely) then lines 78 and 79 will disconnect from the current network, overwrite the variable 'current_network' with the value of 'best_network' that has just been found.

Line-80 will call the function 'connect_to_best_network' and pass the appropriate named variables to it.

Please note this is a simple test program where the sequence is repeated every 5-seconds.

A possible real-life application could be in a device that moves around a building like a robotic vacuum cleaner. Periodically, as the vacuum cleaner moves around the building it could check whether it is connected to best WiFi network.

Another example is the set of monitors Mr D uses in his greenhouse and growhouse (also known as a cold frame) to sense temperature and humidity. These monitors are from time to time moved, so using this program would guarantee that they were able to connect to the best network.

I'm sure you can come up with other examples of where you believe this program would could be used (and discussed at one of the Club's meetings).

Links to the Micro-Python code used in this guide

Here's a link to the complete Micro-Python code.

Here's a link to the Micro-Python dictionary of available networks.

Summary

This tutorial guide has introduced you to some of the networking commands associated with the Raspberry Pi Pico-W. A number of Micro-Python programs have been described to connect to a network, deal with connection problems and finally organise how to scan for a network with the best signal strength on a regular basis.

End of the tutorial guide

Bohunt School (Wokingham)

Connect Pico-W to best WiFi network

Additional Useful Information

scan command

As was mentioned previously, the scan() command returns an array of tuples.

Each tuple is made-up of the following six parts.

Column 0 - ssid	**Name of the WiFi network**
Column 1 - bssid	**MAC address**
Column 2 - channel	
Column 3 - RSSI	**Signal strength expressed as a negative value**
Column 4 - security	**???**
Column 5 - hidden	**????**

Further details about scan() can be found in this document.