Féidearthachtaí as Cuimse
Infinite Possibilities

Week 3
Networking with the Pico W

Fundamentals of IoT
Dr. Eoin Rogers (eoin.rogers@tudublin.ie)

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

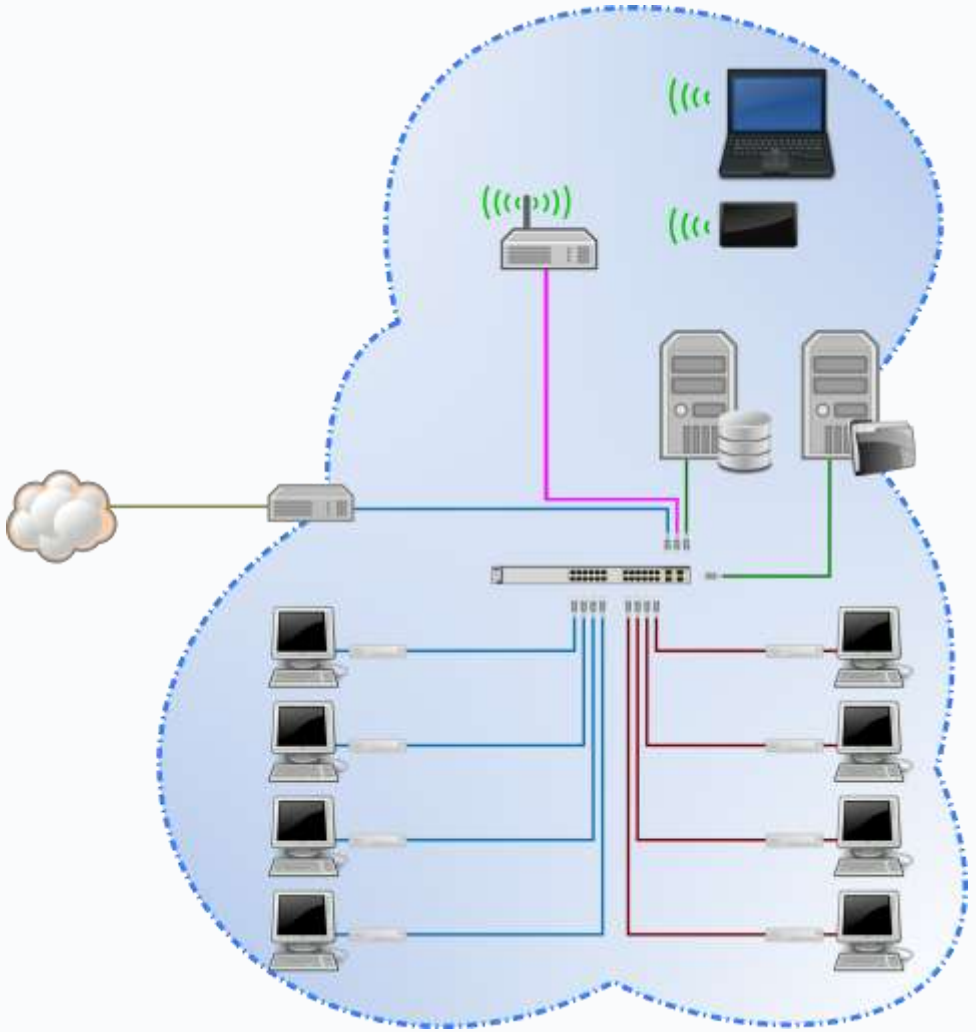
Lesson Outline

- Quick refresher on networking
- Connecting the Pico to the Internet via wi-fi
- Using the sockets API to send and receive data
- Basic processing of the data using string/bytes manipulation functions/methods

Networking

A quick refresher on modern computer networking

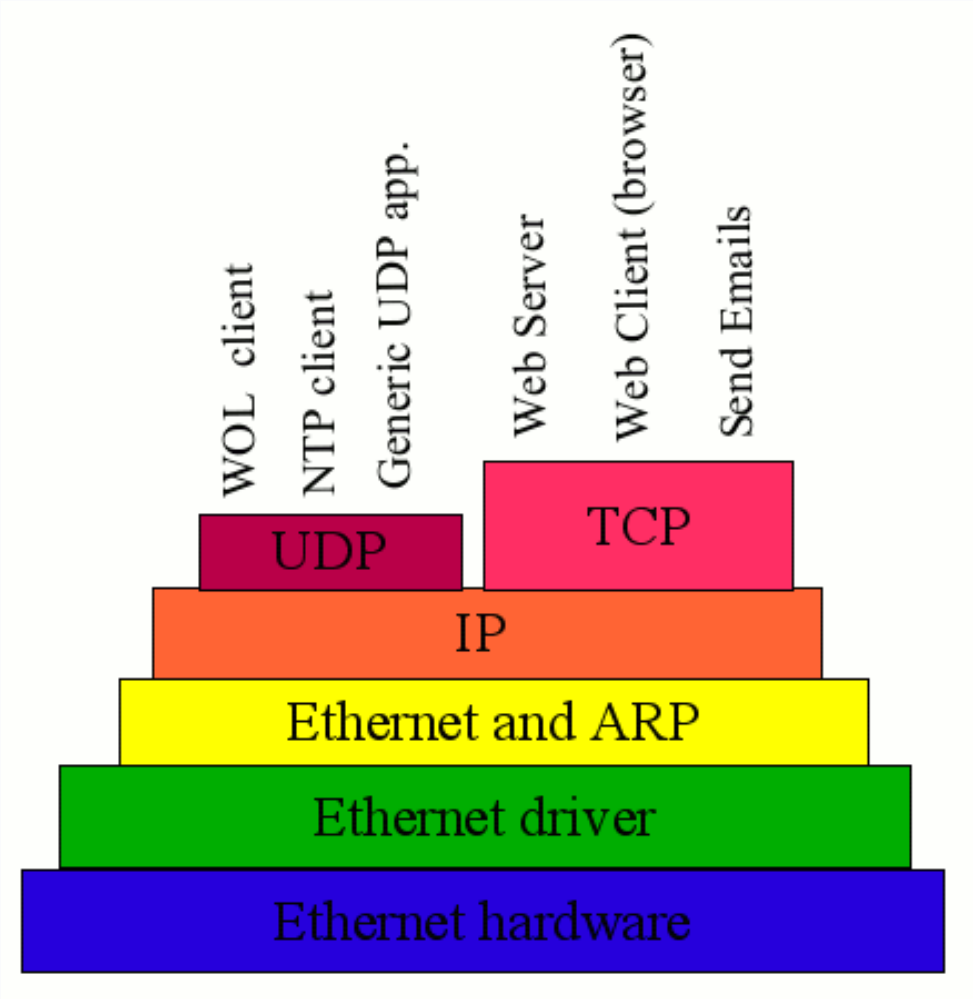
Networking



OSI networking model

- Networks are structured as a series of **layers**
- Each layer **takes data from the layer above**, packages it into a datagram format, and **passes this to the layer below**
- There are **two exceptions** to this: the very topmost and bottommost layers:
  - The **physical layer** represents hardware and passes raw bits on the physical network medium
  - The **application layer** is expected to consume the data in some way

TCP/IP and UDP/IP stacks



Addressing

- Each of these protocols have there own addressing schemes
- Can anyone name the form of “address” used by:
  - Ethernet
  - WiFi
  - IP
  - TCP

What happens when you type google.ie into your browser?

What happens when you type google.ie into your browser?

- A **DNS lookup** is performed to find the IP address
- A **HTTP request** is created
- The request is packaged into a **TCP segment**
- The segment is packaged into an **IP packet**
- The packet is packaged into a **WiFi frame**
- The frame is transmitted over the network

Connecting the Pico

The Pico W has a built-in WiFi chip

Scanning for networks

```
from network import WLAN

wifi = WLAN(WLAN.IF_STA)
wifi.active(True)

security_type = {
    0: 'None',
    1: 'WEP',
    2: 'WPA-PSK',
    3: 'WPA2-PSK',
    4: 'WPA/WPA2-PSK',
    5: 'WPA2 Enterprise',
    6: 'WPA3-PSK',
    7: 'WPA2/3-PSK',
    8: 'WAPI-PSK',
    9: 'OWE',
}
```

```
for (
        ssid,
        bssid,
        channel,
        rssi,
        security,
        hidden
) in wifi.scan():
    ssid = ssid.decode('utf-8')
    security = security_type[security]
    print(f'Found network "{ssid}" using ' +
        'channel {channel} with security ' +
        '{security}')
```

WiFi channels

WiFi uses radio configurations called **channels**.
Unfortunately, the Pico only supports a subset of the channels that a typical laptop or phone would work with. In particular, it doesn't work with 5GHz channels, only the older 2.4GHz

Connecting to a network

```
from network import WLAN
import time, socket

wifi = WLAN(WLAN.IF_STA)
wifi.active(True)

ssid = '...'
password = '...'

wifi.connect(ssid, password)

time.sleep(5)
```

```
if wifi.status() != 3:
    print('Wifi couldn\'t connect')
else:
    tudublin_dns = socket.getaddrinfo(
        'tudublin.ie',
        443
    )
    tudublin_ip = tudublin_dns[0][-1][0]
    print(
        'The IP address for the ' +
        f'TU Dublin website is {tudublin_ip}'
    )
```

Function to connect to WiFi

```
def connect(
        wifi_obj,
        ssid,
        password,
        timeout=10
):
    wifi_obj.connect(ssid, password)

    while timeout > 0:
        if wifi_obj.status() != 3:
            time.sleep(1)
            timeout -= 1
        else:
            return True

    return False
```

A very basic “server”

```
port_number = 80

if not connect(wifi, ssid, password):
    print('Wifi couldn\'t connect')
else:
    s = socket.socket()
    s.bind(('0.0.0.0', port_number))
    s.listen()
    ip = wifi.ifconfig()[0]
    print(f'Listening on IP {ip}')
    while True:
        cxn, addr = s.accept()
        print(f'Connected to {addr}')
        data = cxn.recvfrom(200)
        print(data, len(data))
        cxn.close()
```

BSD sockets in Python

The only thing missing from the previous code snippet is using **`cxn.sendall()`** to send some raw Python bytes back to the client!

Link to tutorial

Dealing with Python bytes

Python distinguishes between the str (string) and bytes datatype

Dealing with bytes

- **Bytes-to-string**: `my_bytes.decode()` (assumes UTF-8)
- **String-to-bytes**: `my_string.encode()`
- **Bytes-to-int**: each byte can be indexed into to get an integer, but are you sure you don't want to convert to a string first?
- **Int-to-bytes**: Create a list of integers in the range [0, 255], then pass it to `bytes()`

Useful methods on strings and bytes

- **Slicing `[x:y]`** (inclusive of **`x`**, exclusive of **`y`**)
- **`.index(query)`** and **`.find(query)`** Find the index of the first instance of the **`query`**
- **`.startswith(query)`** Returns true if the object starts with **`query`**
- **`.endswith(query)`** Returns true if the object ends with **`query`**

Summary

- Went back over networking basics
- How to scan for networks and connect to them on the Pico W using MicroPython
- We can then use Python's socket API (almost) the same way as in CPython
- It's important to understand how to use strings and bytes objects in Python effectively to work with these APIs

That’s all for this week

Thanks for your attention!