

5.1: UNDERSTANDING THE WIFI – HTTP PROTOCOLS

First, you need to know that the ESP8266 can operate as a Wi-Fi client (like a smartphone or laptop) and/or as an access point (like a Wi-Fi router or extender).

You can set this mode with:

```
WiFi.mode(m);
```

where m must be one of the following modes:

```
WIFI_AP (access point),
```

```
WIFI_STA (client),
```

```
WIFI_AP_STA
```

```
(AP and client) or WIFI_OFF.
```

Now let's connect to your access point:

```
WiFi.begin(WIFI_SSID, WIFI_PWD);
```

This will connect you to an access point given its SSID and the password. Please note that this call is not blocking. This means that the code will immediately proceed to the next instruction whether the ESP successfully connects to the access point or not.

5.1 HTTP

By connecting to the internet, you can exchange data between your ESP8266 and the network. Let's look at how we can load content from a web server using the Hyper Text Transfer Protocol (HTTP). Find the Code in the file [http.ino](#) (on Moodle).

Compile and upload it to your NodeMCU. In what follows is complete explanation to the code. Read it carefully.

How does this work?

First, we define the SSID and password of the Wi-Fi access point we want to connect to. Please note that there are better ways to do that.

- The Wi-Fi Manager (<https://github.com/tzapu/WiFiManager>) for instance starts the ESP8266 as an access point if it cannot connect to any SSID.
- You then use your smartphone to configure the Wi-Fi credentials and there is no need to hard code these into your firmware. But for the sake of simplicity let's ignore this here.

On line 14 we start connecting to the defined access point and wait until the connection is established.

After all there is no point to send requests to a server if the network connection is not confirmed yet.

Line 49 sends the request to the server. The command `GET /guide/ HTTP/1.1\r\n` might look strange to you. This is how your browser talks to the web server. GET is the command for the webserver, /guide/ is the resource on the server we want to get and HTTP/1.1 is the protocol that we are using.

- If you are interested in learning more about the HTTP, please read the specification document found in <https://tools.ietf.org/html/rfc2616>.

On line 63 we print out the response line by line (in case there were any).

- This can be quite complicated if we want to add encryption in the form of SSL to the connection.

This protects your data and makes sure that you are talking to the right server. With the following command we can verify that the host matches the given SHA1 fingerprint.

```
if (client.verify(fingerprint, host)) {  
    Serial.println("certificate matches");  
} else {  
    Serial.println("certificate doesn't match");  
return; 6 }
```

How can you find this fingerprint? Your browser can help you with this.

- You can use Chrome. First open the page you need the fingerprint for, in my case www.google.com. Then click on the little lock symbol and Certificate:
 - It will bring the certificate details up.
 - This will verify that you are talking to the google.com.

Questions for you to answer:

- What is the HTTP version found?
- What is the meaning of the status code? can you find two more examples of other codes.
- Change the website in your code to www.squix.org
- Compare the output to your previous one, what was changed.