

IoT Engineering

3: Sending Sensor Data to IoT Platforms

CC BY-SA, Thomas Amberg, FHNW
(unless noted otherwise)



Today

⅓ slides,

⅔ hands-on.

Slides, code & hands-on: tmb.gr/iot-3



Prerequisites

Install the Arduino IDE, set up ESP8266, get Wi-Fi:

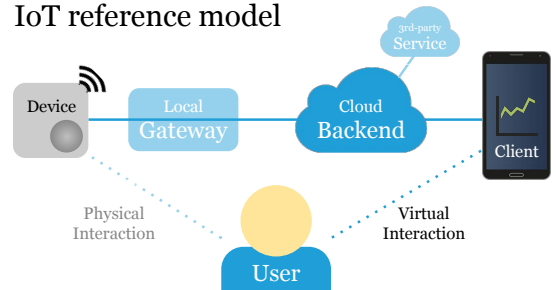
Check the Wiki entry on [Installing the Arduino IDE](#).

[Set up the Feather Huzzah ESP8266](#) for Arduino.

Get access to a Wi-Fi network without a portal.

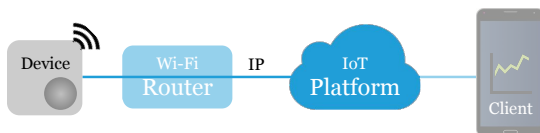
3

IoT reference model



4

Wi-Fi connectivity



5

Wi-Fi

Wi-Fi is based on [IEEE 802.11/a/b/g/n/...](#) standards.

Uses 2.4 GHz UHF and 5 GHz SHF ISM radio bands.

100 m line-of-sight, many materials absorb/reflect it¹.

[Throughput](#) depends on version, 11 Mbps up to Gbps.

Uses more energy than Bluetooth LE, less than 3/4G.

6

ESP8266 Wi-Fi setup

.ino

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(115200); // for debug output
  WiFi.mode(WIFI_STA); // _AP|_AP_STA|_OFF
  WiFi.begin("SSID", "PASSWORD"); // TODO
  while (WiFi.status() != WL_CONNECTED) {
    delay(100); // keeps watchdog happy
  }
  Serial.println(WiFi.localIP());
}
```

7

MAC address

The **MAC address**, e.g. 80:7d:3a:58:8a:ef, is a unique identifier assigned to the network interface controller (NIC) for data link layer communications.

Used as a network address for IEEE 802 technology including Ethernet, Wi-Fi and Bluetooth.

The first six digits **identify the vendor**, e.g. 80:7d:3a.

8

ESP8266 Wi-Fi MAC address

.ino

This code reads the ESP8266 Wi-Fi MAC address:

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(115200);
  Serial.print(WiFi.macAddress());
}
```

Some networks grant access based on MAC address. 9

HTTP Web request

A simple way to put or get data to/from a backend.

To debug HTTP, the **cURL** client is recommended.

```
$ curl -v tmb.gr/hello.json
```

```
> GET /hello.json HTTP/1.1\r\n
> Host: tmb.gr\r\n
> \r\n
```

10

HTTP Web response

```
< HTTP/1.1 200 OK\r\n
< Content-Type: application/json\r\n
< Content-Length: 32\r\n
< \r\n
{\n
  "message": "Hello, World!"\n
}
```

11

ESP8266 Wi-Fi client

.ino

```
WiFiClient client;
client.connect(host, port);
client.print(
  "GET /hello.html HTTP/1.1\r\n" \
  "Host: tmb.gr\r\n" \
  "\r\n");
while (client.connected() ||
  client.available()) {
  int ch = client.read(); ... }
```

12

Hands-on, 15': Wi-Fi

Build and run the previous Wi-Fi related examples.

Use the `.ino` link on each page to find the source.

The examples are in the course repository.

Make sure to use the ESP8266 board.

13

Sending sensor data

Here is a simple recipe for "remote sensing".

First, connect the device to the network.

Then, repeat the following steps:

- Read sensor values
- Send the data*

*Add a timestamp before or after sending, see below.

14

Transport Layer Security

Transport Layer Security (TLS) allows a device to:

- Encrypt a communication channel, for privacy.
- Verify that it talks to the right backend server.

Trust is based on certificates issued by [authorities](#).

HTTPS relies on TLS to secure HTTP connections.

See [this video](#) by @spiessa for an introduction.

15

ESP8266 secure Wi-Fi client

[.ino](#)

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

const char *host = "www.howssmyssl.com";
const char *path = "/a/check";
const int port = 443;

WiFiClientSecure client;
if (client.connect(host, port)) {
    // the connection is encrypted
}
```

16

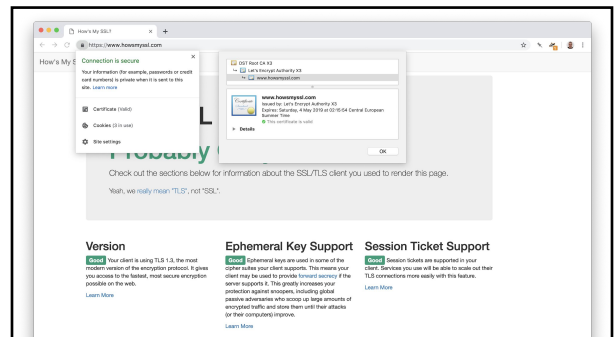
ESP8266 verify host fingerprint

[.ino](#)

```
// Browser > 🔒 > Certificate > Fingerprint
const char *fingerprint = "67 FF 3D BC D3
59 E2 C0 BD 04 3B 8A 57 CB 72 7C 0A 20 F5
E4"; // SHA-1, not really secure anymore

if (client.verify(fingerprint, host)) {
    Serial.println("certificate verified");
} else {
    Serial.println("certificate mismatch");
}
```

17



ESP8266 check CA certificate [.ino](#)

```
extern const unsigned char caCert[] PROGMEM;
extern const unsigned int caCertLen;

// make sure time() is set, see NTP client
WiFiClientSecure client; // use TLS
if (client.setCACert_P(caCert, caCertLen)) {
  if (client.connect(host, port)) {
    if (client.verifyCertChain(host)) {
      ...
    }
  }
}
```

19

Hands-on, 15': ESP8266 TLS clients

Build, run and compare the following TLS clients:

[Secure Wi-Fi client](#), with [fingerprint](#), with [CA check](#).

Locate/download the CA certificate in your browser.

Locate the SHA-1 fingerprint of the host certificate.

Bonus: Try to change the host to another Website.

20

IoT platforms

IoT platforms enable storing/displaying sensor data.

There are many examples, we start with these two:

[Dweet.io](#) stores name/value pairs in JSON format.

[ThingSpeak](#) stores sensor data and displays graphs.

Both receive data through HTTP POST requests.

21

Dweet.io

[Dweet.io](#) stores name/value pairs in JSON format.

Host: [dweet.io](#)

Port: 443

POST /dweet/for/THING_NAME?name=value

POST /dweet/for/THING_NAME?x=23&y=42&t=...

GET /get/dweets/for/THING_NAME

Or POST any valid JSON data in the request body.

22

Hands-on, 15': Dweet.io

[Dweet.io](#) works without an account, data is public.

Use your ESP8266 MAC address as THING_NAME.

On the ESP8266, read the analog pin Ao, then POST its value to /dweet/for/THING_NAME?a0=value

Use cURL or your browser to read stored data from https://dweet.io/get/dweets/for/THING_NAME

23

ThingSpeak

[ThingSpeak](#) stores sensor data and displays graphs.

Host: [api.thingspeak.com](#)

Port: 80 or 443

POST /update?key=WRITE_API_KEY&field1=42

GET /channels/CHANNEL_ID/feed.json?

key=READ_API_KEY

24

Hands-on, 15': ThingSpeak

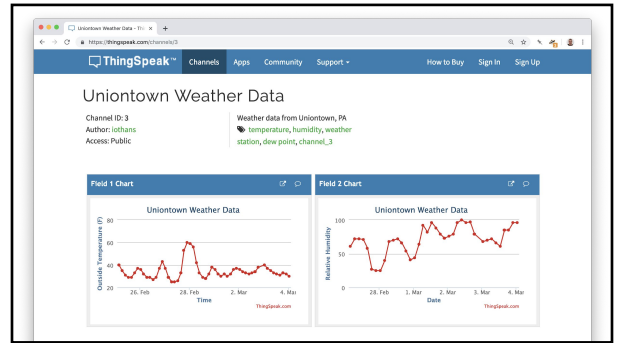
Get an [account](#) to create channels and get API keys.

Add the Arduino library with *Sketch > Include Library > Manage Libraries... > ThingSpeak > Install*

Try the example code *File > Examples > ThingSpeak > ESP8266 > WriteMultipleFields.ino*

Make sure values arrive in your ThingSpeak channel.

25



Timestamps

Adding a timestamp can happen in two places:

- On the backend, when a data packet just arrived.
- On the device, when a sensor value is measured.

The first requires sending immediately or discarding values, the second allows caching of measurements.

Trade-off: simplicity vs. accuracy & completeness.

27

Time

The time on a microcontroller is reset to 0 at startup.

Timestamps use Coordinated Universal Time ([UTC](#)).

There are different ways to get and keep UTC time:

- Get time from a standard Web server, using HTTP.
- Get time from a network time server, using NTP.
- Set and keep time with a real time clock (RTC).

28

ESP8266 Web-based time client [.ino](#)

```
> HEAD / HTTP/1.1\r\n
> Host: google.com\r\n
> \r\n
< HTTP/1.1 301 Moved Permanently\r\n
< Location: http://www.google.com/\r\n
< Content-Type: text/html\r\n
< Date: Sat, 02 Mar 2019 17:10:20 GMT\r\n
< \r\n
```

29

Network Time Protocol

Network Time Protocol ([NTP](#)) is a network protocol for clock synchronization between computer systems¹.

Synchronizes participating computers to within a few milliseconds of Coordinated Universal Time ([UTC](#)).

Implementations send and receive timestamps using the User Datagram Protocol ([UDP](#)) on [port 123](#).

30

ESP8266 built-in NTP client [.ino](#)

```
configTime(timezone * 3600, dst_offset,
  "pool.ntp.org", "time.nist.gov");
// wait for time() being adjusted
while (time(NULL) < 28800 * 2) {
  delay(500);
}
// time() is set
time_t now = time(NULL);
```

31

Hands-on, 15': ESP8266 NTP clients

Build, run and compare the following NTP clients:

The [Web-based time client](#) and [built-in NTP client](#).

Arduino > Examples > ESP8266WiFi > [NTPClient](#).

Bonus: Read the code of this [low memory version](#).

Which one would you use, and why?

32

Hands-on, 15': Temperature sensor

Design a connected temperature sensor as specified:

Gets current time and date in [ISO 8601](#) UTC format.

Gets temperature & humidity from a [DHT11 sensor](#).

Connects* to [api.thingspeak.com](#) port 443 with TLS.

Posts sensor values, timestamp every 30 seconds.

*) And robustly reconnects, if disconnected.

33

Summary

We learned to connect a device to a Wi-Fi network.

We sent sensor measurements to an IoT platform.

We looked at ways to get UTC time on a controller.

These are the basics of remote sensing.

Next: Internet Protocols, HTTP and CoAP.

34

Homework, max. 3h

Implement or finish the temp. sensor you designed.

Post the IoT platform data feed URL* to the Slack.

Commit the Arduino code to the hands-on repo.

Measure the temperature for at least 24 hours.

*) Ideally public, we'll take a look together.

35

Feedback?

Find me on <https://fhnw-iot.slack.com/>

Or email thomas.amberg@fhnw.ch

Slides, code & hands-on: tmb.gr/iot-3

