HACKER {school}

DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Microcontrollers for IoT

An OpenLab sprint presentation on MCUs, I/O devices and communication

João Barreiros C. Rodrigues Filipe Nobre Piçarra

DEEC-IST | HackerSchool

Foundations for Micro-controlled Systems

What is a Microcontroller?

- Small embedded computer
- Generically includes:
  - processor (CPU)
  - memory (RAM/Flash)
  - and I/O pins
- Normally the code runs on the baremetal
- Alternatively it can run a simple, deterministic OS
  - **FreeRTOS**

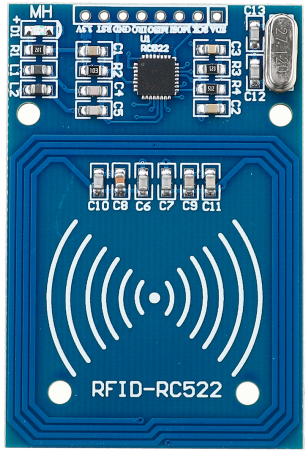
What are Sensors and Actuators?

- **Sensors**: collect data from the environment
- **Actuators:** perform actions over the environment

What are Sensors and Actuators?

These devices can communicate with the MCU through various protocols:

- General Purpose I/O (GPIO) and ADC I/O
- Inter-Integrated Circuit (I^2C)
- Serial Peripheral Interface (SPI)
- Universal Asynchronous Receiver-Transmitter (UART)
- etc. . .



Our Microcontrolled System basics

ESP32 and ESP32 WROOM

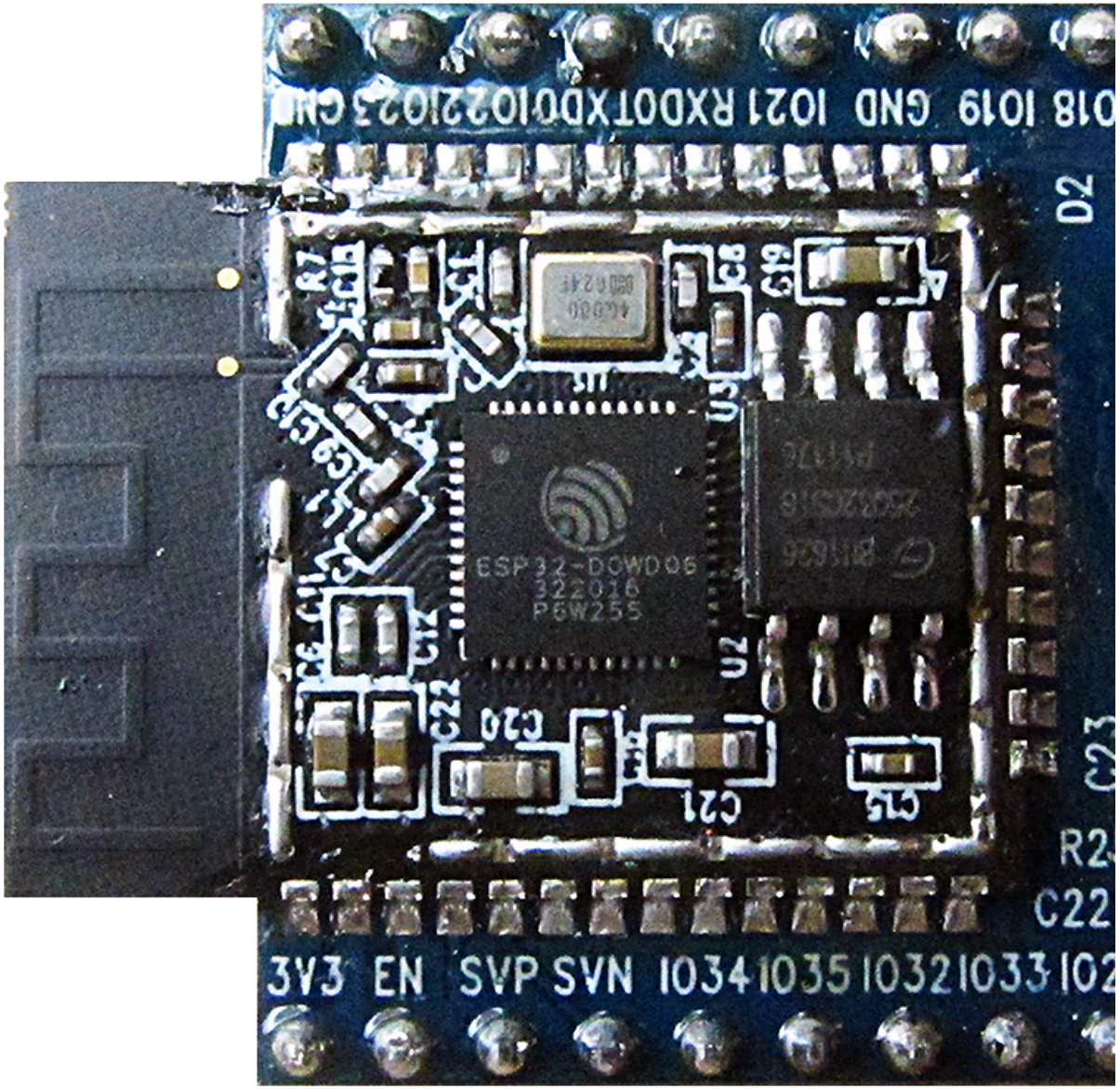
ESP32

- Dual-core 32-bit LX6 microprocessor
- Ultra-Low-Power Co-Processor
- 520 kB of on-chip SRAM for data and instructions
  - Out of which 8 kB can be acessed in Deep-Sleep Mode
- 34 uncommitted GPIO pins
  - ADC, DAC, etc. . .

ESP32 WROOM

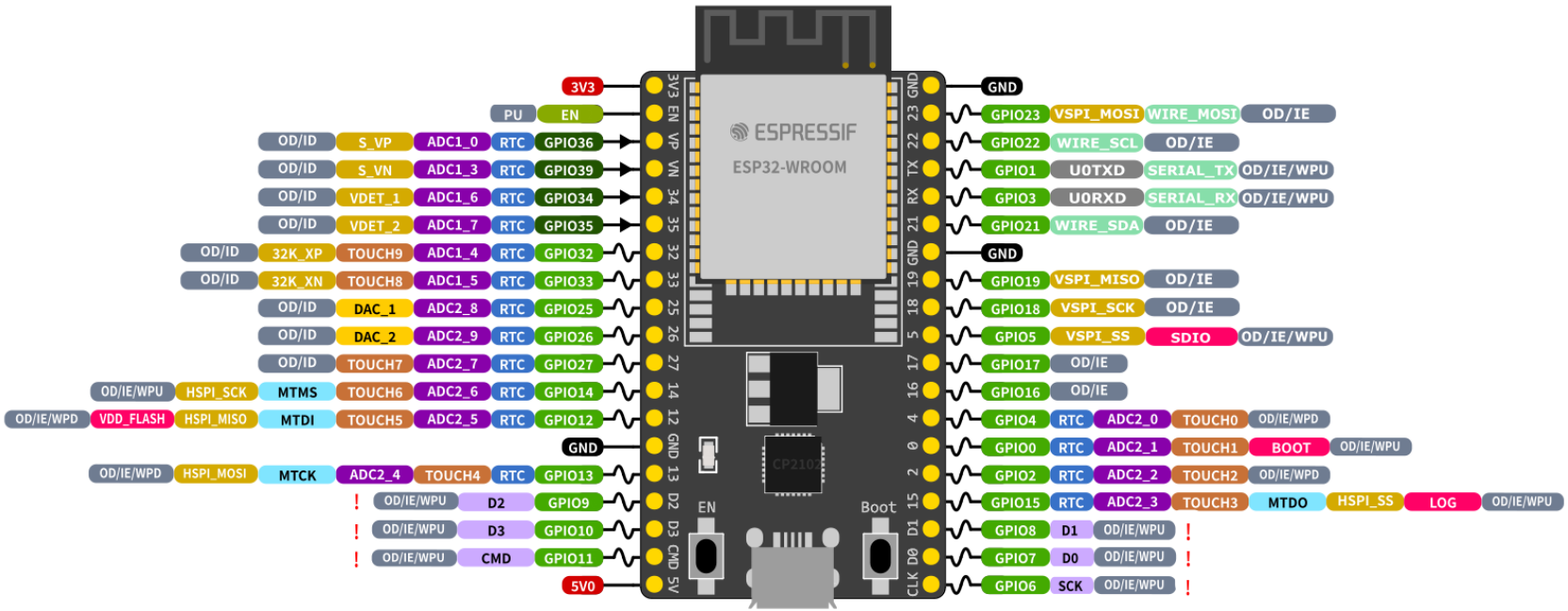
- Module that integrates a ESP32 chip
- Adds peripherals such as
  - Additional 4 MB of Flash Memory (PROGMEM)
  - Antennas
  - I/O shielding

ESP32 and ESP32 WROOM



ESP32-DevKitC4 / NodeMCU

ESP32-DevKitC



ESP32 Specs

32-bit Xtensa® dual-core @240 MHz
Wi-Fi IEEE 802.11b/g/n 2.4 GHz
Bluetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

PWM Capable Pin
GPIOX — GPIO Input Only
GPIOX — GPIO Input and Output
DAC_X — Digital-to-Analog Converter
DEBUG — JTAG for Debugging
FLASH — External Flash Memory (SPI)
ADCX_CH — Analog-to-Digital Converter
TOUCHX — Touch Sensor Input Channel
OTHER — Other Related Functions
SERIAL — Serial for Debug/Programming
ARDUINO — Arduino Related Functions
STRAP — Strapping Pin Functions

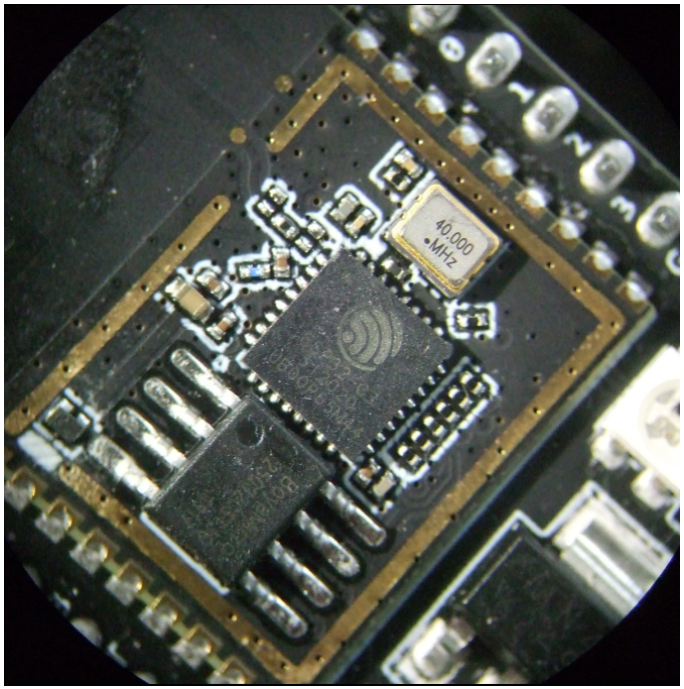
RTC — RTC Power Domain (VDD3P3_RTC)
GND — Ground
PWD — Power Rails (3V3 and 5V)
! — Pin Shared with the Flash Memory Can't be used as regular GPIO

GPIO STATE

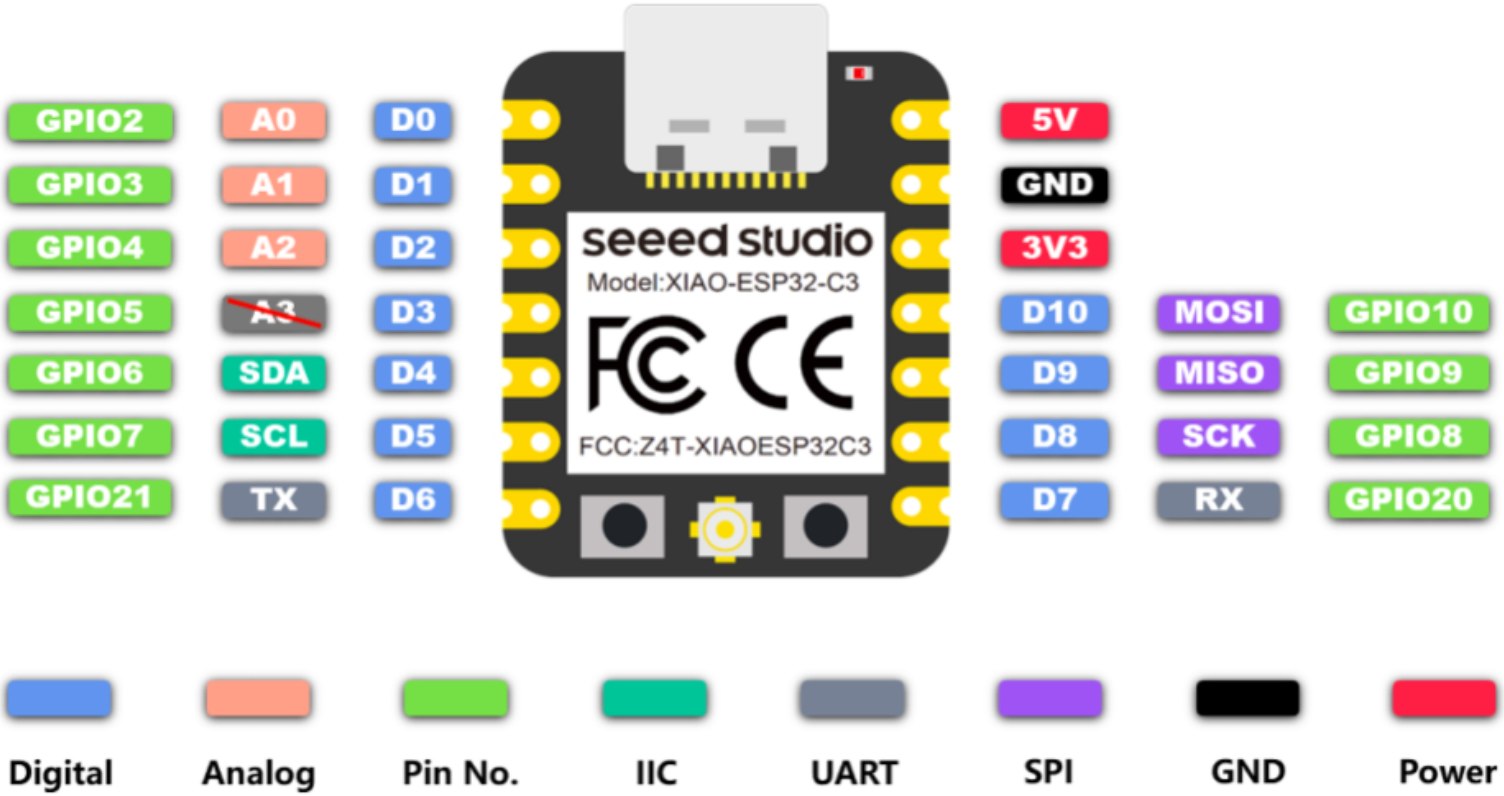
WPU: Weak Pull-up (Internal)
WPD: Weak Pull-down (Internal)
PU: Pull-up (External)
IE: Input Enable (After Reset)
ID: Input Disabled (After Reset)
OE: Output Enable (After Reset)
OD: Output Disabled (After Reset)

ESP32-C3

- RISC-V core
- Ultra-Low-Power
- 400kB of on-chip SRAM.
- Although with less pins, now supports BLE and on-chip encryption



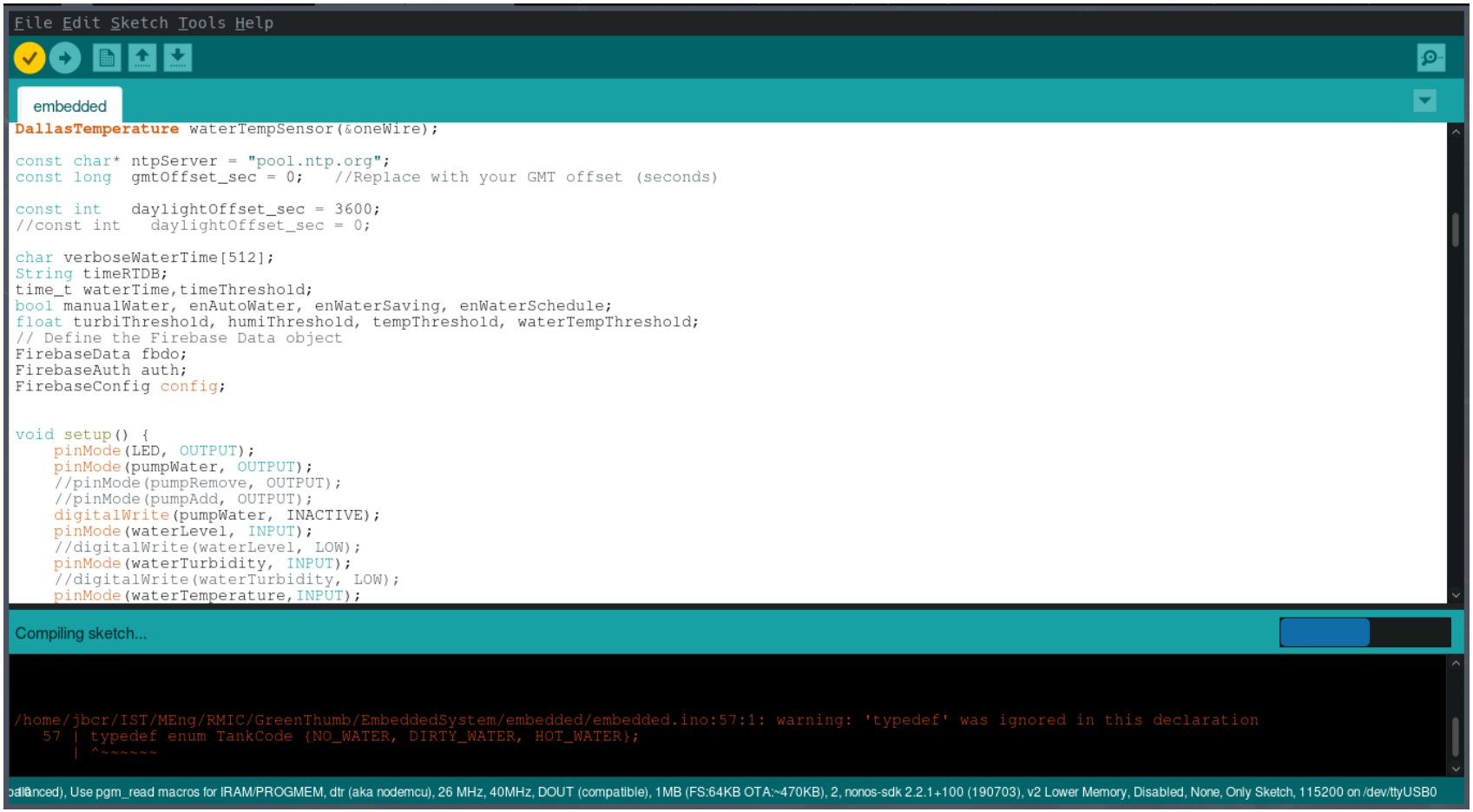
XIAO ESP32-C3



A3(GPIO5) - Uses ADC2, which may become inoperative due to false sampling signals. For reliable analog reads, use ADC1 instead. Refer to the ESP32-C3 datasheet.

Setup and Microcontroller programming

For ease of use we will be using **Arduino IDE**



I/O and Wireless fundamentals

Digital GPIO

```
digitalRead(PIN);
digitalWrite(PIN, value);
```

Analog GPIO

```
analogRead(PIN);
analogWrite(PIN, value);
```

More I/O

I²C

```
#include <Wire.h>

Wire.begin(SDA_PIN, SCL_PIN);
Wire.write(data);
```

UART

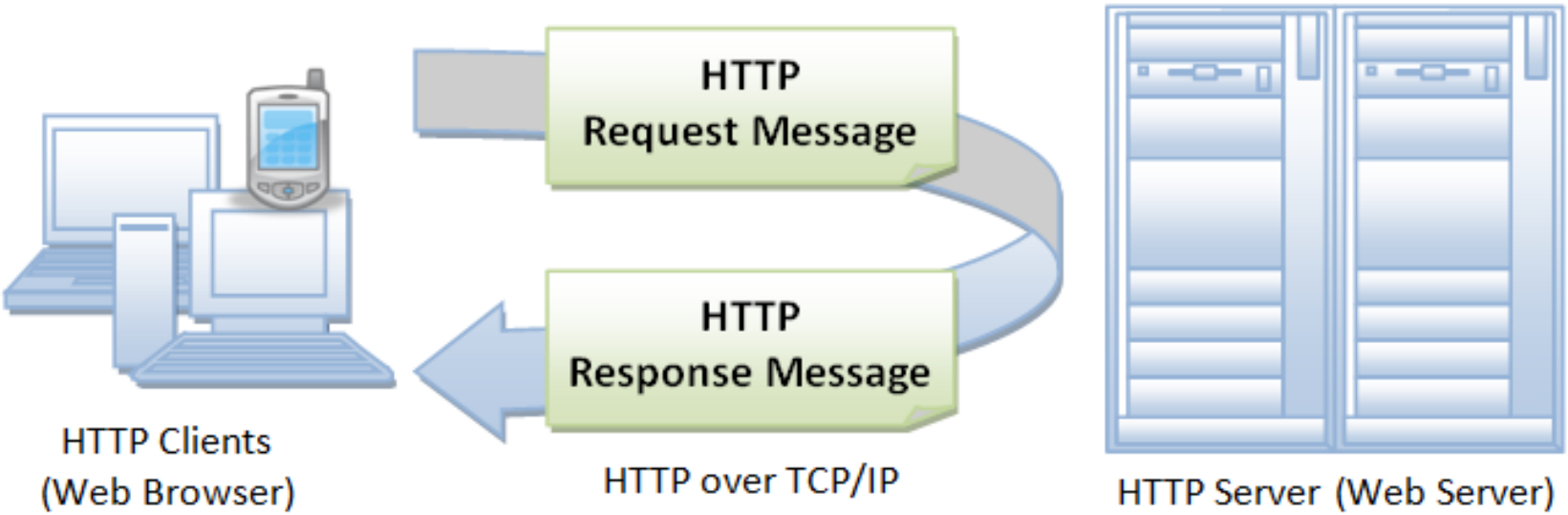
```
HardwareSerial mySerial(2);  // Use UART2
mySerial.begin(BAUD_RATE, SERIAL_8N1, RXD_PIN, TXD_PIN);
```

Integrating our sys. with the Internet

Wi-Fi

```
#include <WiFi.h>
/*WiFi credentials*/
#define WIFI_SSID "Labs-LSD"
#define WIFI_PASSWORD "aulaslsd"
void setup(){
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to WiFi");
    while(WiFi.status() != WL_CONNECTED){
        Serial.print(".");
        delay(500);
    }
    Serial.println("");
    Serial.println("Connected to WiFi!");
}
```

HTTP


HTTP
Request Message
HTTP
Response Message
HTTP Clients
(Web Browser)
HTTP over TCP/IP
HTTP Server (Web Server)

HTTP Methods

HTTP Methods and Their Meaning

Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data

HTTP POST 1

```
#include <HTTPClient.h>
/* HTTP endpoint*/
#define serverURL "http://chat.fpicarras.dev/send"
#define ID "<yourID>"

void loop(){
    if(Serial.available()){
        String msg = Serial.readStringUntil('\n');
        msg.trim();
        if (msg.length() == 0) return;
```

HTTP POST 2

```
        HTTPClient http;
        http.begin(serverURL);
        http.addHeader("Content-Type",
                       "application/json");
        String body = "{\"message\":\"" + msg
                       + "\", \"id\":\"" +
                       ID + "\"}";
        int code = http.POST(body);
        Serial.println("Sent: " + msg + " (" + code + ")");
        http.end();
        delay(1000); // prevent spamming
    }/*end if*/
} /*end loop*/
```

Power Management

Active, Inactive and Sleep-Modes

Power mode	Description			Power Consumption
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 5-4 for details.
	Wi-Fi/BT Tx packet			
	Wi-Fi/BT Rx and listening			
Modem-sleep	The CPU is powered up.	240 MHz *	Dual-core chip(s)	30 mA ~ 68 mA
			Single-core chip(s)	N/A
		160 MHz *	Dual-core chip(s)	27 mA ~ 44 mA
			Single-core chip(s)	27 mA ~ 34 mA
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA
			Single-core chip(s)	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	The ULP coprocessor is powered up.			150 μA
	ULP sensor-monitored pattern			100 μA @1% duty
	RTC timer + RTC memory			10 μA
Hibernation	RTC timer only			5 μA
Power off	CHIP_PU is set to low level, the chip is powered down.			1 μA

Wi-Fi “sleep”

```
WiFi.setSleep(true);
```

This is the default behaviour! For intermitent connection to Wi-Fi we can set:

```
WiFi.setSleep(false);
```

which has an increased power consumption

Wi-Fi disable (Modem-sleep)

```
WiFi.disconnect(true); // Turn off Wi-Fi
WiFi.mode(WIFI_OFF);   // Disable Wi-Fi
```

Requires the WiFi object to be restarted!!

Light-Sleep

```
#include "esp_sleep.h"
// GPIO for button wake-up
#define WAKEUP_PIN GPIO_NUM_33
// Change to your actual button GPIO

    // Configure RTC GPIO for wake-up
    esp_sleep_enable_ext0_wakeup(WAKEUP_PIN, LOW);
    // Wake on button press (LOW)
    Serial.println("Going to deep sleep...");
    delay(1000);
    // Enter light sleep mode
    esp_light_sleep_start();
```

Deep-Sleep

```
#include "esp_sleep.h"
// GPIO for button wake-up
#define WAKEUP_PIN GPIO_NUM_33
// Change to your actual button GPIO

    // Configure RTC GPIO for wake-up
    // Wake on button press (LOW)
    esp_sleep_enable_ext0_wakeup(WAKEUP_PIN, LOW);
    Serial.println("Going to deep sleep...");
    delay(1000);
    // Enter deep sleep mode
    esp_deep_sleep_start();
```

To Finish our code...

Colocar na secção `loop`:

```
if (msg == "sleep") {
    Serial.println("Sleeping for 10 seconds...");
    WiFi.disconnect(true);  // disconnect from WiFi and cl
    WiFi.mode(WIFI_OFF);    // turn off WiFi radio
    esp_sleep_enable_timer_wakeup(10 * 1000000);
    esp_deep_sleep_start();
}
```

Concluding Remarks

Additional topics of interest

- Communication Security
  - SSL/TLS Encryption
- AI on the Edge
  - TinyML