cheap

Internet - of - Things

A Beginners Guide to the Internet and ESP8266

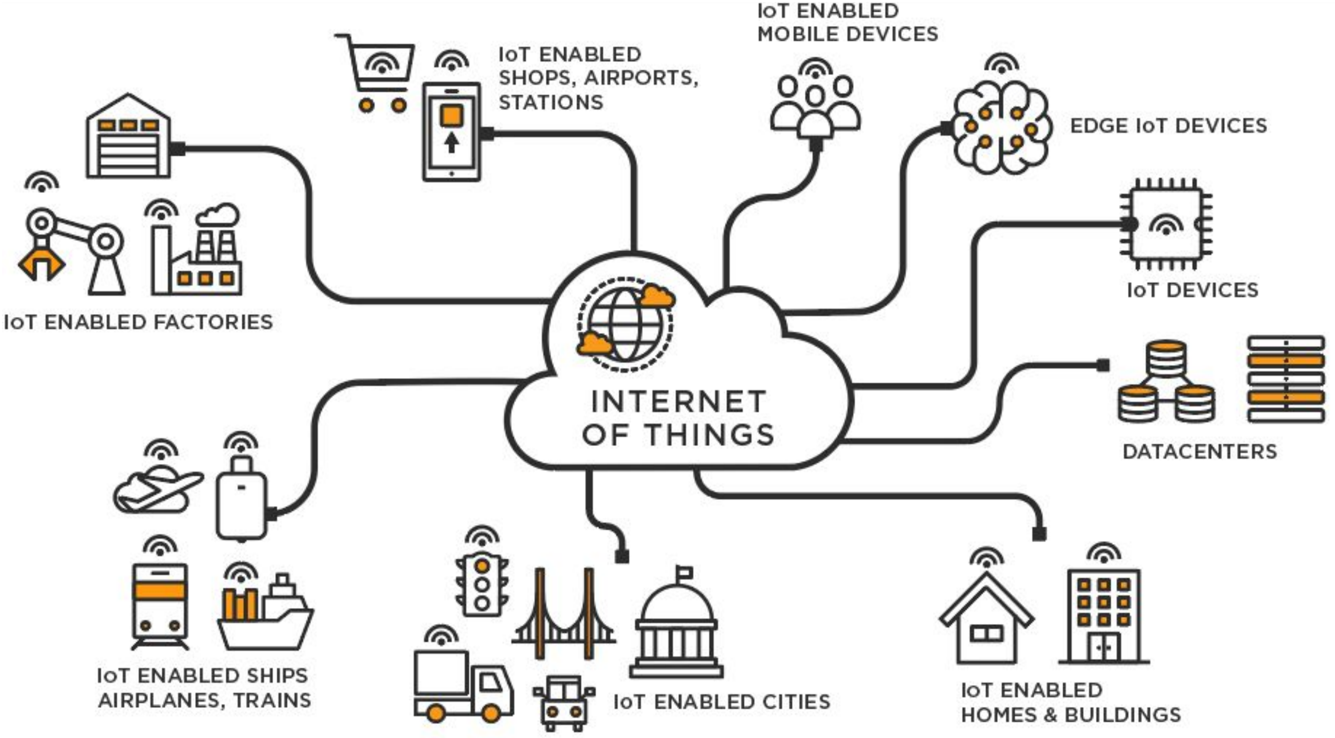
SINF - Semana de Informática, 2021
FEUP, Porto
João Pedro Dias & Bruno Lima

IoT, WoT, IoE, CPS, ...

“The network of devices that contain the hardware, software, firmware, and actuators which allow the devices to connect, interact, and freely exchange data and information.”

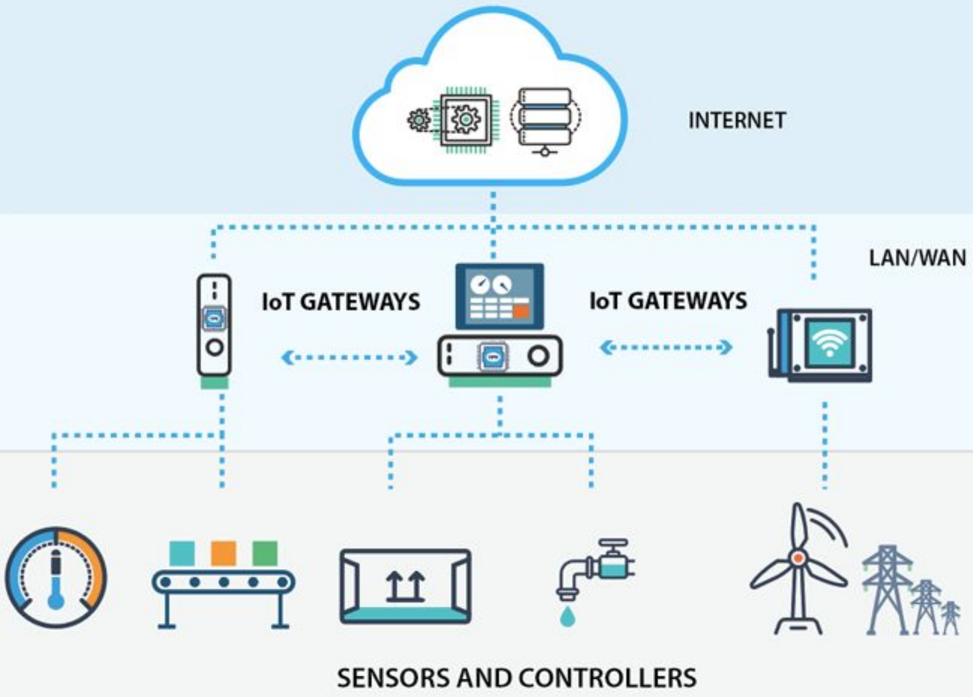
“(...) user or industrial devices that are connected to the internet. IoT devices include sensors, controllers, and household appliances.”

Internet of Things (IoT), NIST, https://csrc.nist.gov/glossary/term/internet_of_things_iot


IoT ENABLED SHOPS, AIRPORTS, STATIONS
IoT ENABLED MOBILE DEVICES
EDGE IoT DEVICES
IoT DEVICES
IoT ENABLED FACTORIES
INTERNET OF THINGS
DATACENTERS
IoT ENABLED SHIPS AIRPLANES, TRAINS
IoT ENABLED CITIES
IoT ENABLED HOMES & BUILDINGS

What is the Internet of Things (IoT)?, TIBCO Software, https://www.tibco.com/reference-center/what-is-the-internet-of-things-iot

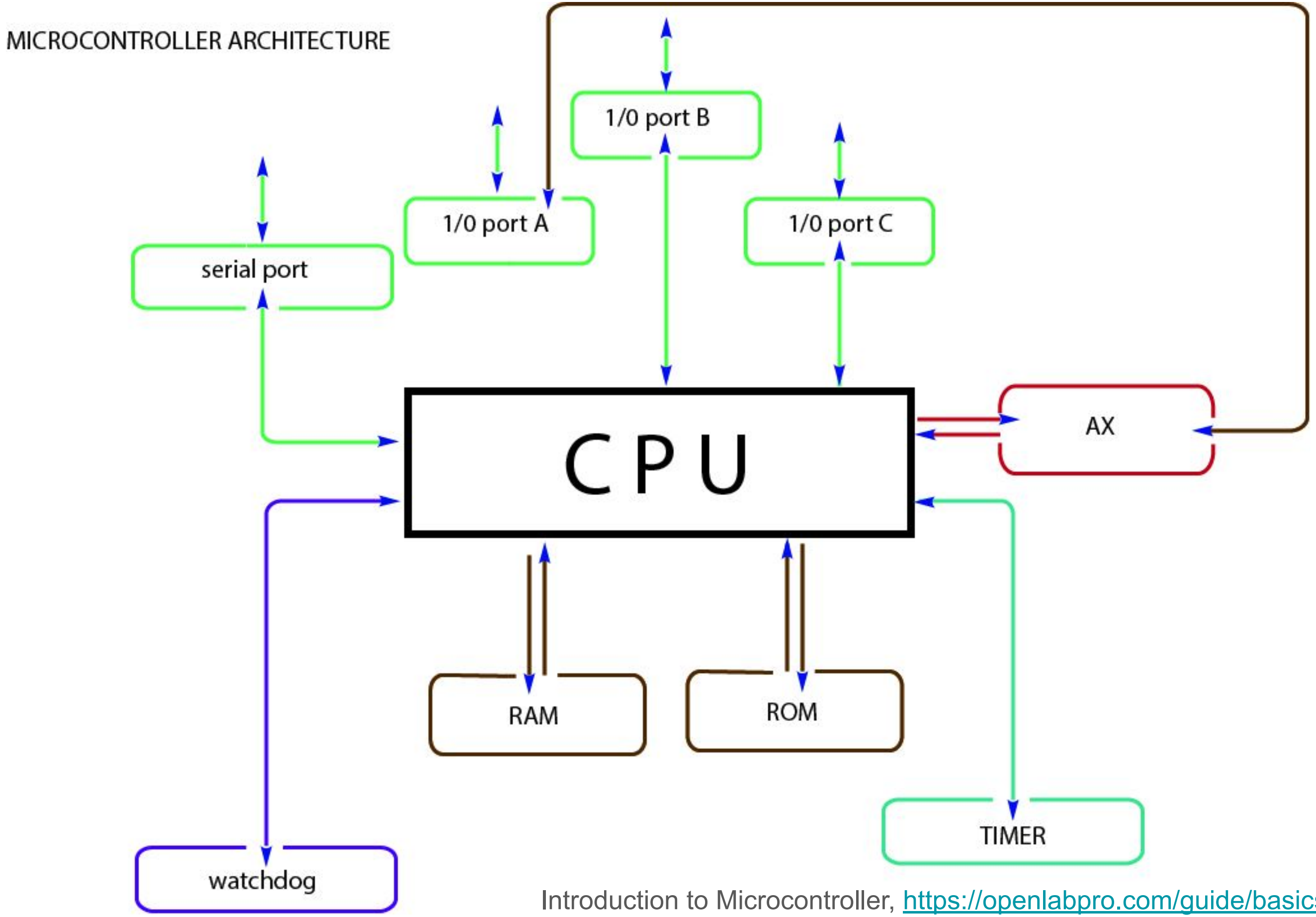
The IoT Three Tiers



Cloud Tier: (Virtualized) High-power Servers and Services

Fog Tier: Gateways, Data Aggregators, Pre-Processing, etc.

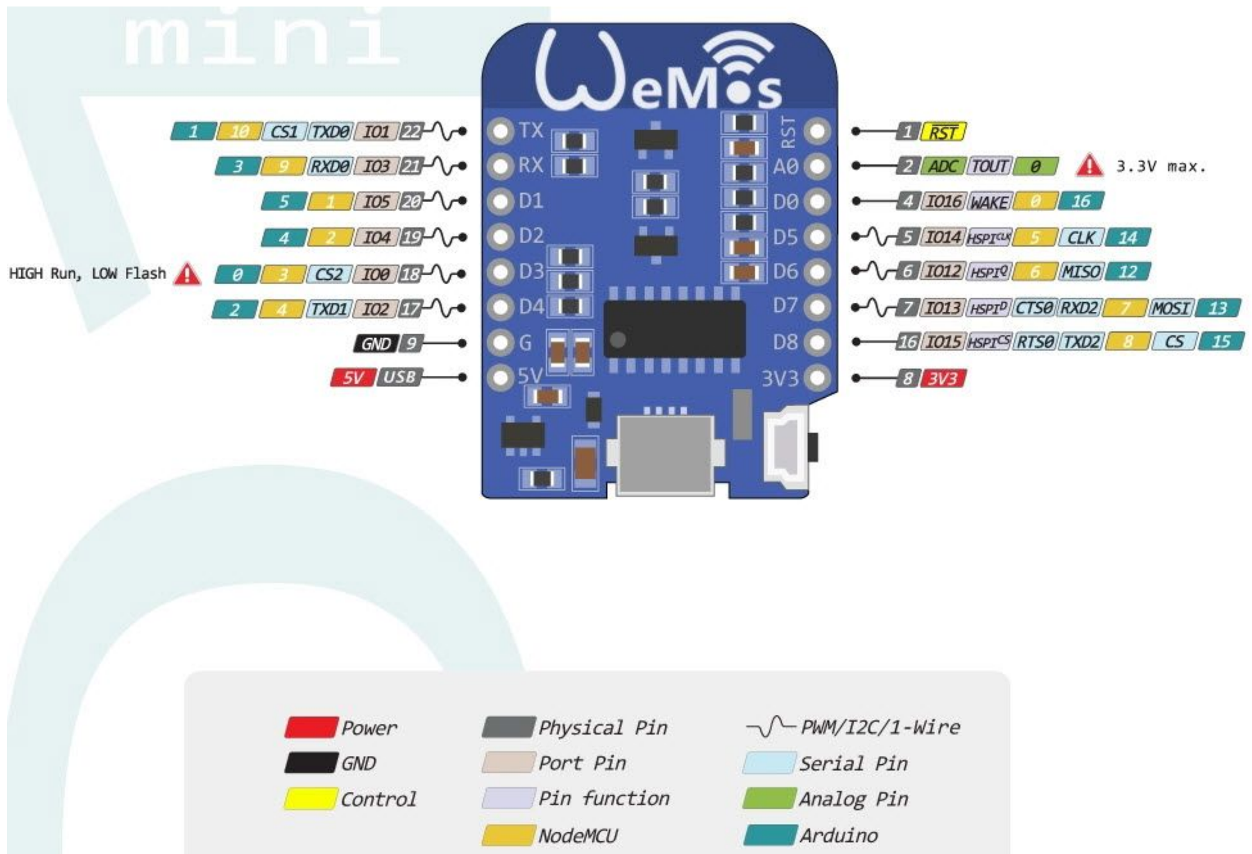
Edge Tier: Sensors, Actuators, and other Low-computational Tasks



Introduction to Microcontroller, https://openlabpro.com/guide/basics-of-microcontroller/

ESP8266

(Wemos D1 mini)



* Built-in LED is connected to D4

Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins	1(3.2V Max)
Clock Speed	80/160MHz
Flash	4Mb
RAM	80Kb
Connectivity	Wi-Fi
Cost	2-3$

LOLIN D1 mini, https://www.wemos.cc/en/latest/d1/d1_mini.html

Actuator (LED)

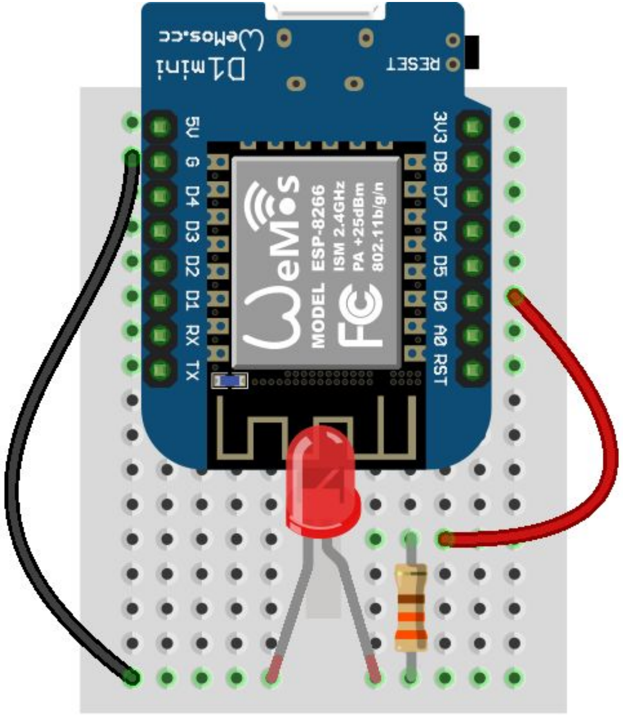
Actuators can be turned on/off by toggling a pin (e.g., **D0**). Other control modes exist, e.g., controlling a motor or the brightness of a LED can require pulse-width modulation (PWM).

In Arduino language, **`digitalWrite()`**:

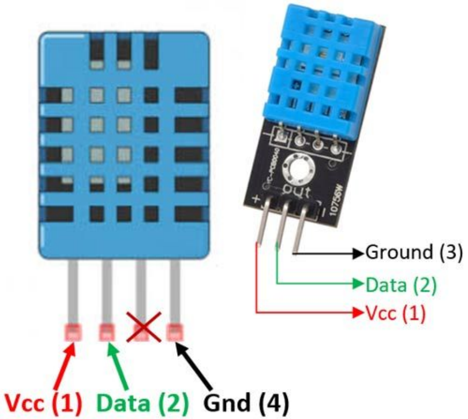
If the pin has been configured as an **`OUTPUT` with `pinMode()`**, its voltage will be set to the corresponding value:

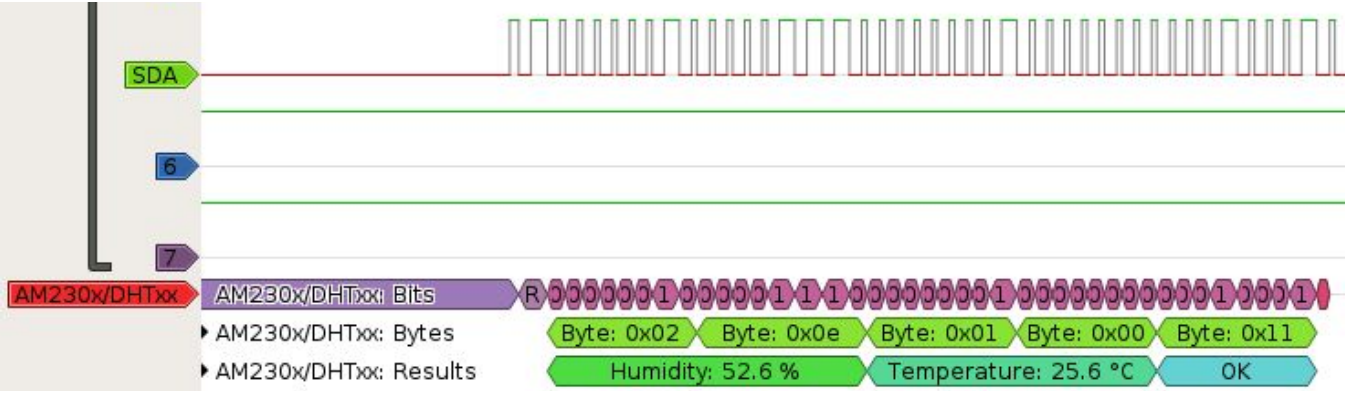
- **5V (or 3.3V on 3.3V boards) for `HIGH`**
- **0V (ground) for `LOW`**

`Wemos D1 mini has a built-in LED (part of the ESP8266 MCU), used for signalling RX/TX activity, but can be used for other purposes.`



Sensors (DHT11)



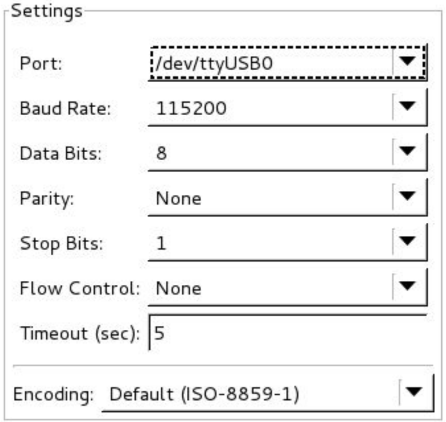


DHT11 is a **single wire digital humidity and temperature sensor**, which provides humidity and temperature values serially with **one-wire protocol**. DHT11 sensor provides relative **humidity value in percentage (20 to 90% RH)** and **temperature values in degree Celsius (0 to 50 °C)**.

DHT11, https://www.electronicwings.com/sensors-modules/dht11

MCU <-> UART <-> USB <-> Terminal

Serial Port Configuration



Port: /dev/ttyUSB0

Baud Rate: 115200

Data Bits: 8

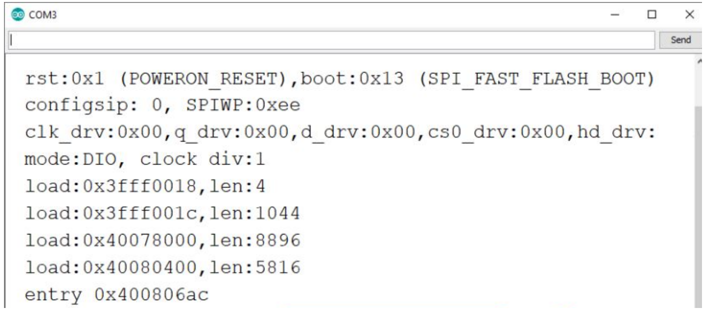
Parity: None

Stop Bits: 1

Flow Control: None

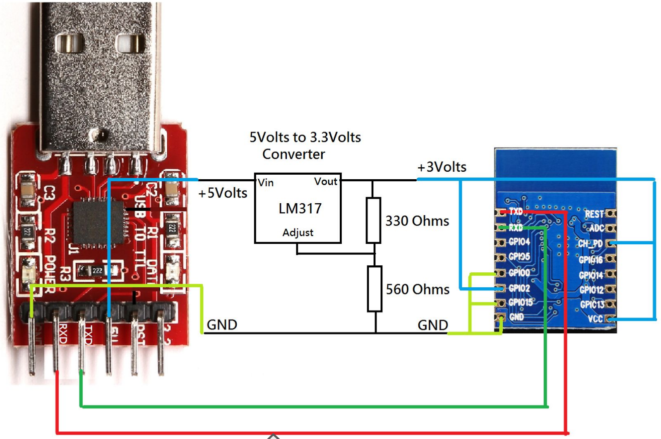
Timeout (sec): 5

Encoding: Default (ISO-8859-1)



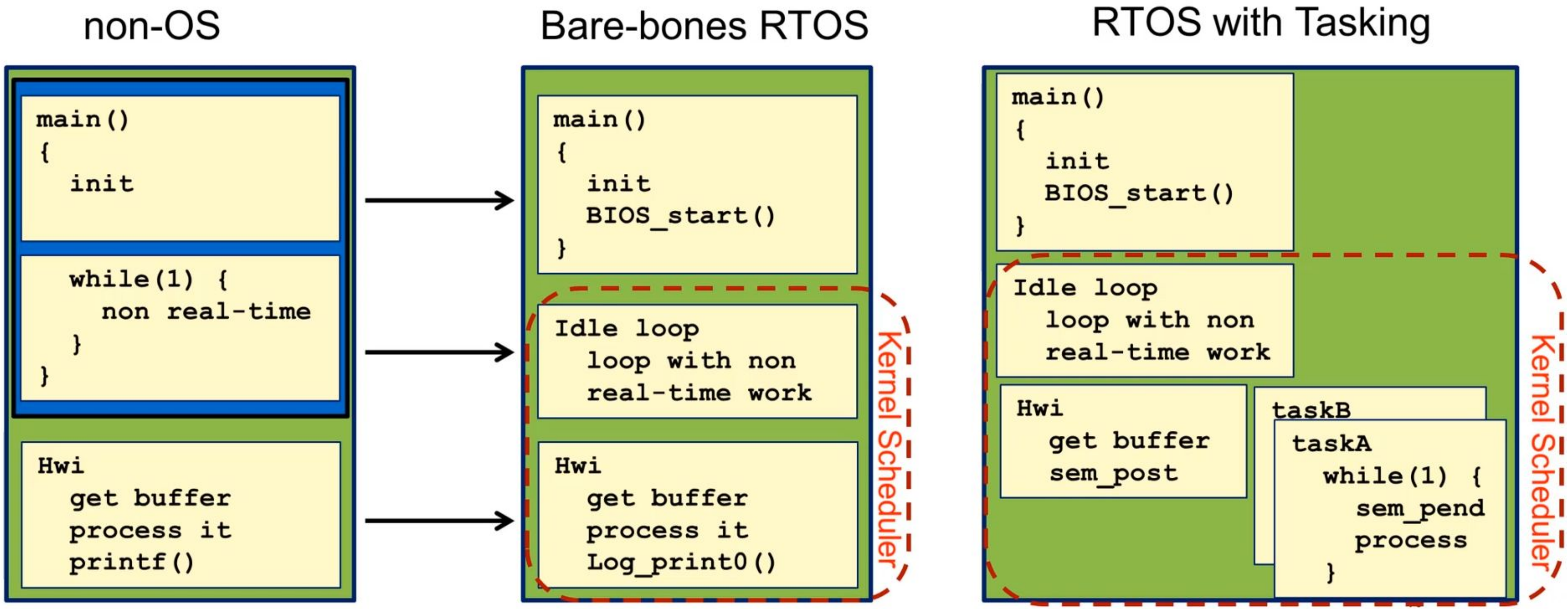
```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
```

Serial Terminal



UART to USB Adapter (built-in in Wemos D1 and most dev boards)

The many faces of Programming Embedded Devices



non-OS

```
main()
{
  init

  while(1)  {
    non real-time
  }
}

Hwi
  get buffer
  process it
  printf()
```

Bare-bones RTOS

```
main()
{
  init
  BIOS_start()
}

Idle loop
  loop with non
  real-time work

Hwi
  get buffer
  process it
  Log_print0()
```

Kernel Scheduler

RTOS with Tasking

```
main()
{
  init
  BIOS_start()
}

Idle loop
  loop with non
  real-time work

Hwi
  get buffer
  sem_post

taskB

taskA
  while(1)  {
    sem_pend
    process
  }
```

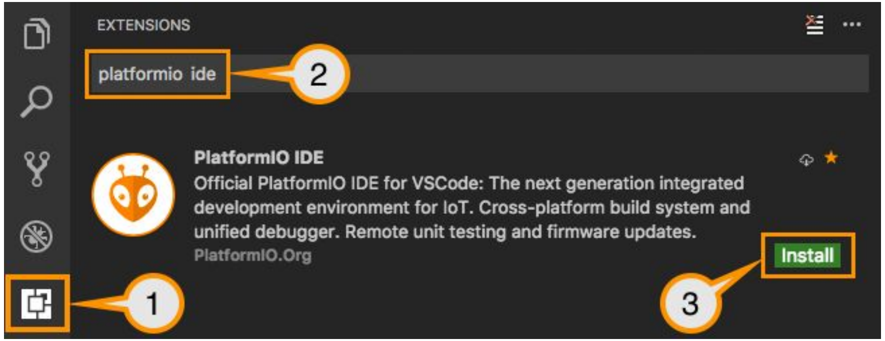
Kernel Scheduler

RTOS Concepts overview, https://training.ti.com/rtos-concepts-overview?context=1128562-1128560

PlatformIO

“**PlatformIO** is a **cross-platform, cross-architecture, multiple framework**, professional tool for embedded systems.”

- PlatformIO IDE, as a VS Code or Atom extension
- PlatformIO Core (CLI), standalone or as part of the extension
- Comes with:
    - Unit Testing
    - Static Code Analysis
    - Remote Development

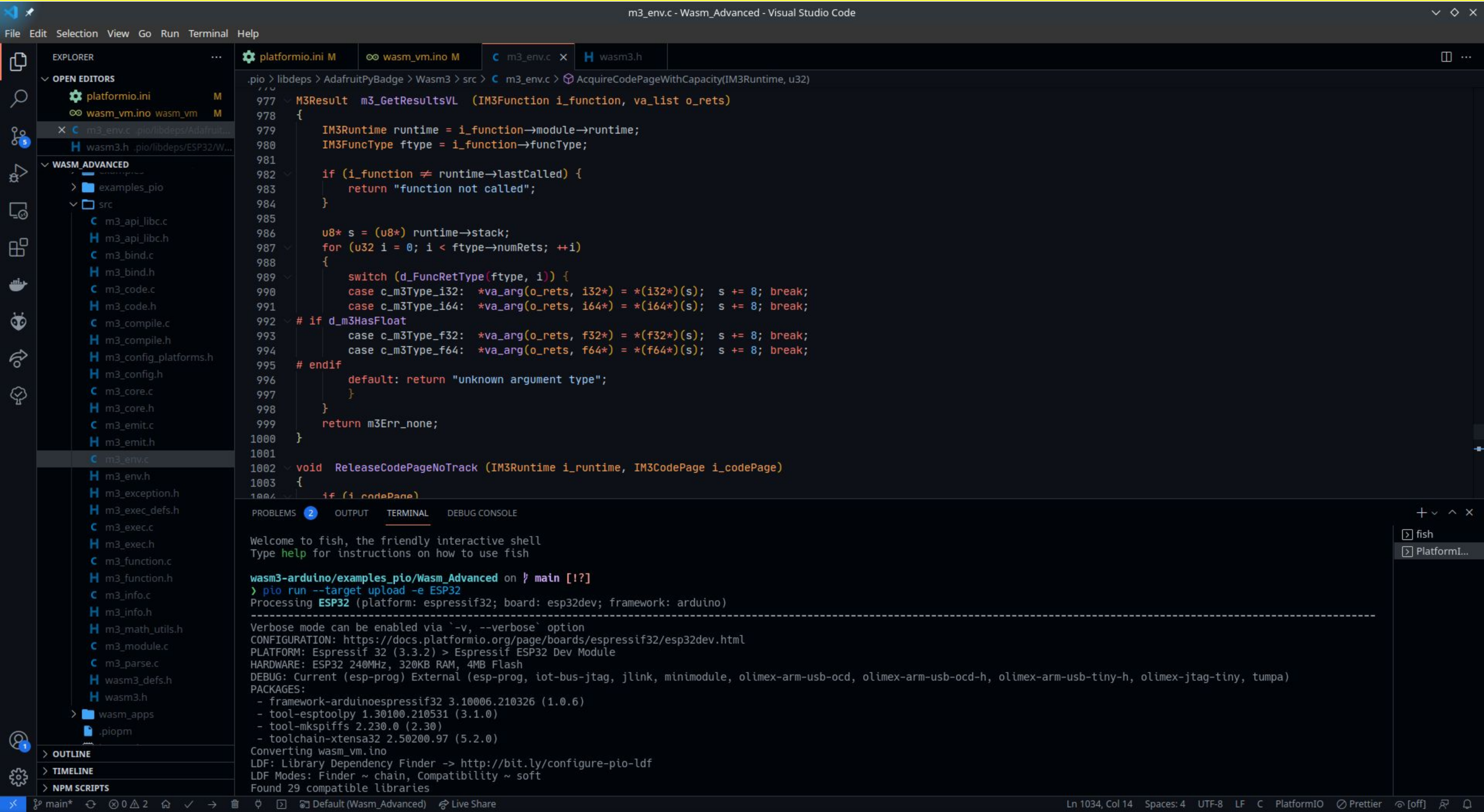


PlatformIO IDE

Official PlatformIO IDE for VSCode: The next generation integrated development environment for IoT. Cross-platform build system and unified debugger. Remote unit testing and firmware updates.

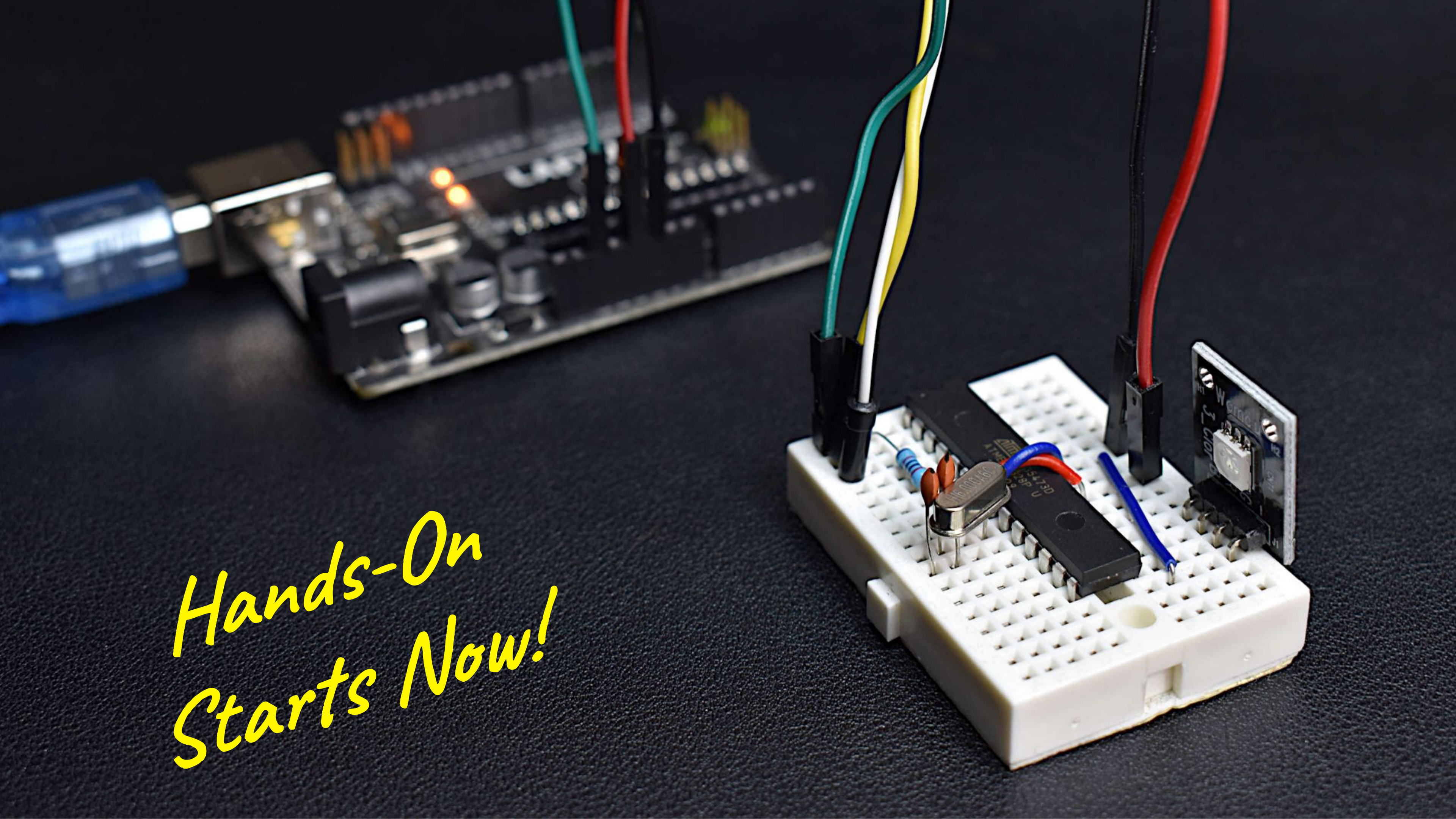
PlatformIO.Org

PlatformIO, https://platformio.org/



```c
M3Result  m3_GetResultsVL  (IM3Function i_function, va_list o_rets)
{
    IM3Runtime runtime = i_function->module->runtime;
    IM3FuncType ftype = i_function->funcType;

    if (i_function != runtime->lastCalled) {
        return "function not called";
    }

    u8* s = (u8*) runtime->stack;
    for (u32 i = 0; i < ftype->numRets; ++i)
    {
        switch (d_FuncRetType(ftype, i)) {
        case c_m3Type_i32:  *va_arg(o_rets, i32*) = *(i32*)(s);  s += 8; break;
        case c_m3Type_i64:  *va_arg(o_rets, i64*) = *(i64*)(s);  s += 8; break;
# if d_m3HasFloat
        case c_m3Type_f32:  *va_arg(o_rets, f32*) = *(f32*)(s);  s += 8; break;
        case c_m3Type_f64:  *va_arg(o_rets, f64*) = *(f64*)(s);  s += 8; break;
# endif
        default: return "unknown argument type";
        }
    }
    return m3Err_none;
}

void  ReleaseCodePageNoTrack (IM3Runtime i_runtime, IM3CodePage i_codePage)
{
    if (i_codePage)
```

```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish

wasm3-arduino/examples_pio/Wasm_Advanced on  main [!?]
❯ pio run --target upload -e ESP32
Processing ESP32 (platform: espressif32; board: esp32dev; framework: arduino)
--------------------------------------------------------------------------------------------------------------------------
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif32/esp32dev.html
PLATFORM: Espressif 32 (3.3.2) > Espressif ESP32 Dev Module
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
DEBUG: Current (esp-prog) External (esp-prog, iot-bus-jtag, jlink, minimodule, olimex-arm-usb-ocd, olimex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa)
PACKAGES:
 - framework-arduinoespressif32 3.10006.210326 (1.0.6)
 - tool-esptoolpy 1.30100.210531 (3.1.0)
 - tool-mkspiffs 2.230.0 (2.30)
 - toolchain-xtensa32 2.50200.97 (5.2.0)
Converting wasm_vm.ino
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 29 compatible libraries
```



Hands-On Starts Now!

platformio.ini - Project Configuration File

```
[platformio]
default_envs = d1_mini


[env:d1_mini]
platform = espressif8266
board = d1_mini
framework = arduino
monitor_speed = 115200
```

Useful for more than one target

Target microcontroller

Target board

Target framework / OS

Serial Port baudrate

https://docs.platformio.org/en/latest/projectconf/index.html

~~Hello World~~ Blink (src/main.ino)

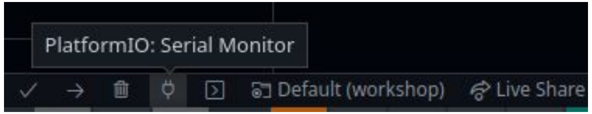
```
// the setup function runs once when you press reset or power the board

#include <Arduino.h>

#define LED D4

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever

void loop() {
  digitalWrite(LED, HIGH);             // Arduino: turn the LED on (HIGH)
                                       // D1 Mini: turns the LED *off*
  delay(1000);                         // wait for a second
  digitalWrite(LED, LOW);              // Arduino: turn the LED off (LOW)
                                       // D1 Mini: turns the LED *on*
  delay(1000);                         // wait for a second
}
```

Pull up vs pull down resistors; https://www.seeedstudio.com/blog/2020/02/21/pull-up-resistor-vs-pull-down-differences-arduino-guide/

https://www.arduino.cc/reference/en/#structure

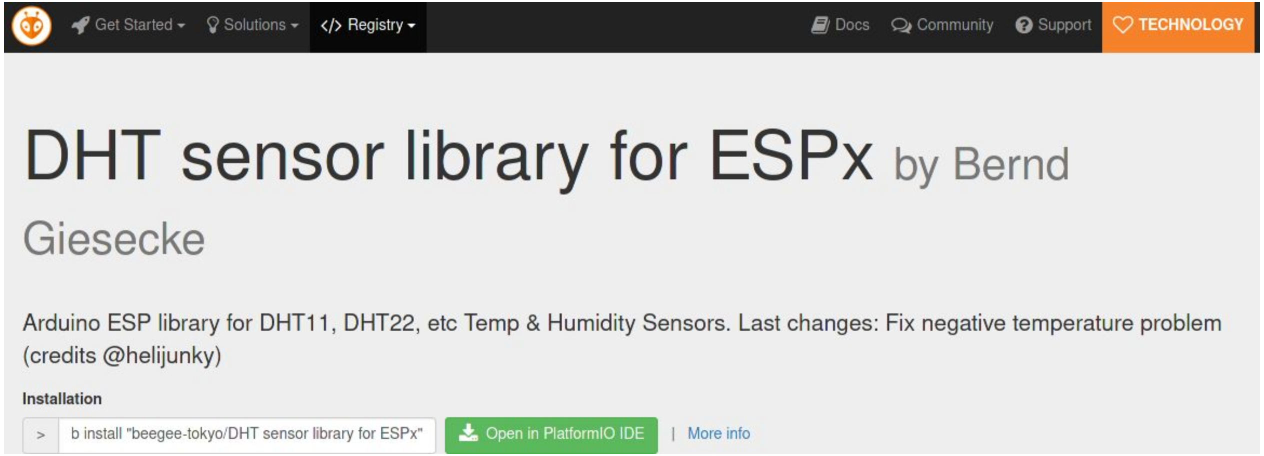
Blink & Hello from Serial World (src/main.ino)

```
#include <Arduino.h>

#define LED D4

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
  // initialize serial output
  Serial.begin(115200);
}

void loop() {
  digitalWrite(LED, HIGH);          // Arduino: turn the LED on (HIGH)
                                    // D1 Mini: turns the LED *off*
  Serial.println("Hello ...");      // Prints Hello to Serial
  delay(1000);                      // wait for a second
  digitalWrite(LED, LOW);           // Arduino: turn the LED off (LOW)
                                    // D1 Mini: turns the LED *on*
  Serial.println(" ... World!");    // Prints World! to Serial
  delay(1000);                      // wait for a second
}
```



https://www.arduino.cc/reference/en/#functions

The easy way to Interact with a Sensor



DHT sensor library for ESPx by Bernd Giesecke

Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors. Last changes: Fix negative temperature problem (credits @helijunky)

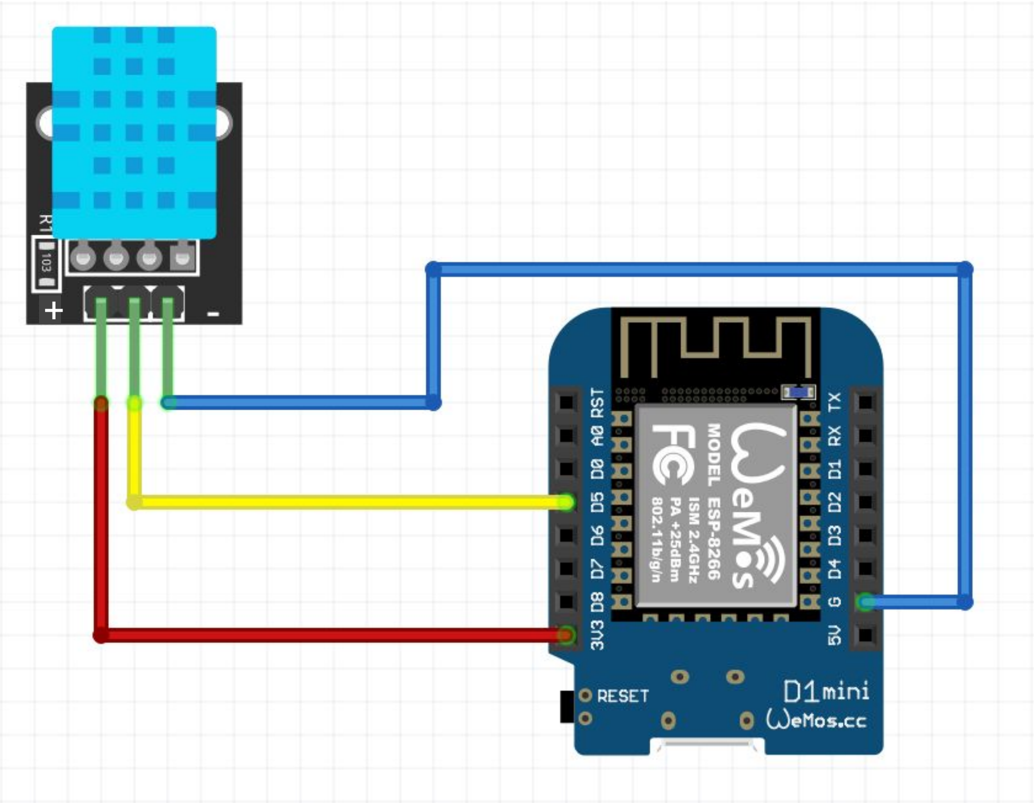
Installation

```
> b install "beegee-tokyo/DHT sensor library for ESPx"
```

```
$ pio lib install "beegee-tokyo/DHT sensor library for ESPx"

# platform.ini is automatically updated with lib_deps (libs can be added manually to the file)
# lib_deps = beegee-tokyo/DHT sensor library for ESPx@^1.18.0
```

PIO Registry, https://docs.platformio.org/en/latest/projectconf/index.html

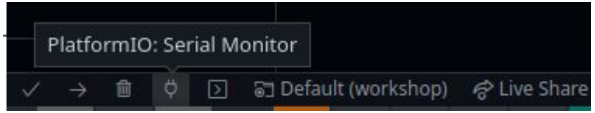
Wemos & DHT11 Circuit



https://www.etsy.com/shop/UnsafeWarnings

Reading the Data (src/main.ino)

```
#include "DHTesp.h"
⋮
#define DHTPIN D5
DHTesp dht;
⋮
void setup() {
  ⋮
  dht.setup(DHTPIN, DHTesp::DHT11); // connect DHT sensor to GPIO D5, and declare sensor type (DHT11)
}

void loop() {
  ⋮
  //delay(dht.getMinimumSamplingPeriod()); // this is not need if we main the 1000 delay

  float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();

  Serial.printf("Temperature: %f, Humidity: %f\%\n", temperature, humidi
}
```

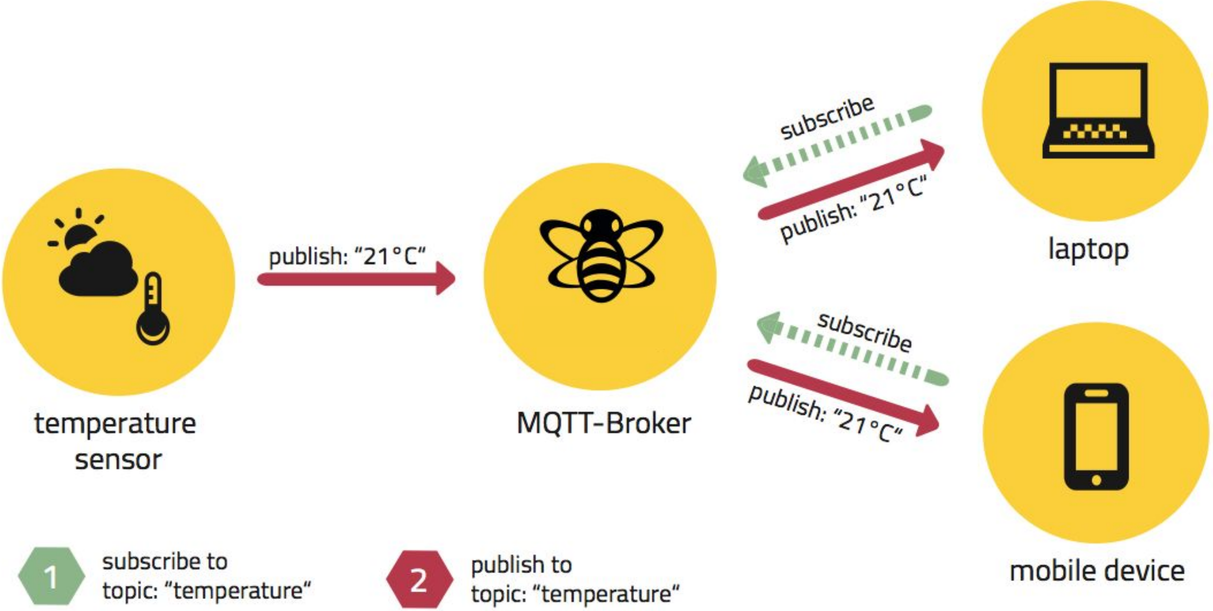


Break

The *I* in *I*oT stands for ~~Security~~ Internet…

but where is it?

MQTT and the world of Pub/Sub



QoS Levels:

- *At most once (0)*
- *At least once (1)*
- *Exactly once (2)*

Birth and Last Will and Testament (LWT) messages.

Birth *is used to send a message after the service has started, and the* **LWT** *is used to notify other clients about a disconnected client.*

TCP-based, can be used directly or with Web Sockets.

A little more on MQTT...

A MQTT broker is required, but there are several freely available, e.g.:

- Broker: **broker.emqx.io**
- TCP Port: **1883**
- Websocket Port: **8083**

To make it easy to experiment with, we will use MQTT over WebSockets.

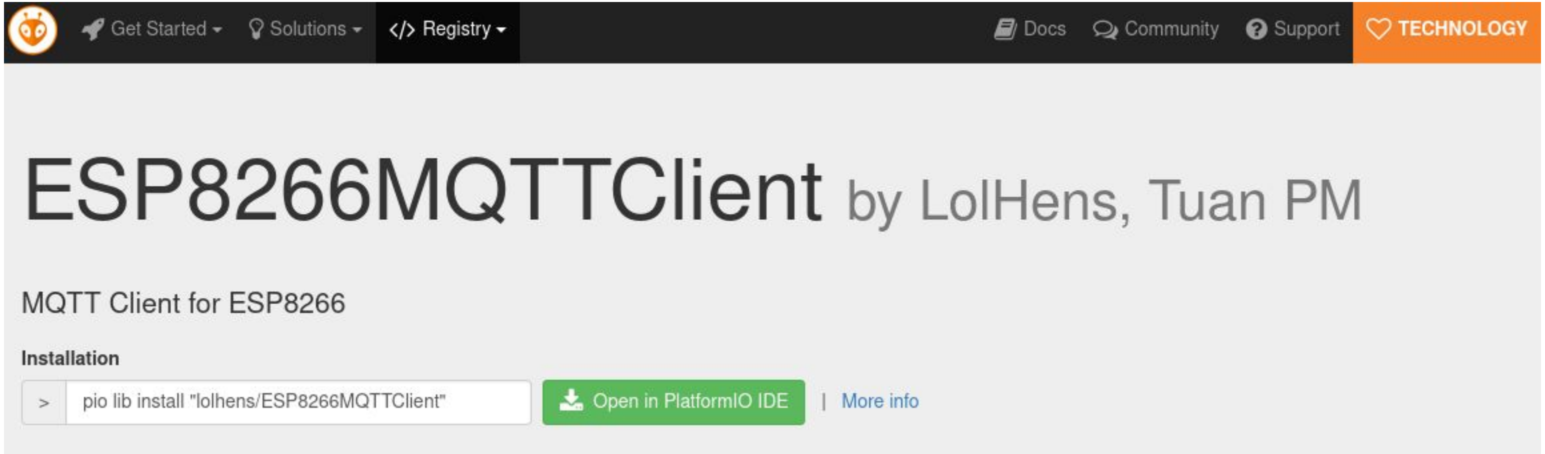
- We can use the browser to interact with the broker without additional stuff.
- http://tools.emqx.io/

Side-quest: Wi-Fi

```
⋮
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

const char *ssid = "........";
const char *password = "........";

void setup() {
  ⋮
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("Success!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  ⋮
}
```

https://arduino-esp8266.readthedocs.io/en/3.0.2/

Getting the Libs



ESP8266MQTTClient by LolHens, Tuan PM

MQTT Client for ESP8266

Installation

```
> pio lib install "lolhens/ESP8266MQTTClient"
```

```
$ pio lib install "lolhens/ESP8266MQTTClient"

# lib_deps =
        beegee-tokyo/DHT sensor library for ESPx@^1.18.0
        lolhens/ESP8266MQTTClient @ ^1.1.1
```

PIO Registry, https://docs.platformio.org/en/latest/projectconf/index.html

MQTT *to the Internet and beyond!* (1/2)

```
⋮
#include <Hash.h>
#include <ESP8266MQTTClient.h>

MQTTClient mqtt;
const char *mqttBroker = "ws://broker.emqx.io:8083/mqtt";

void setup() {
  ⋮
  configTime(3 * 3600, 0, "pool.ntp.org", "time.nist.gov");

  ⋮ more on the next slide

  mqtt.begin(mqttBroker);
}

void loop() {
  ⋮
  mqtt.handle();
  mqtt.publish("/workshop123/temperature", String(temperature, 2), 0, 0);
}
```

We could use the secure version, but let’s keep it unsafe for simplicity purposes.

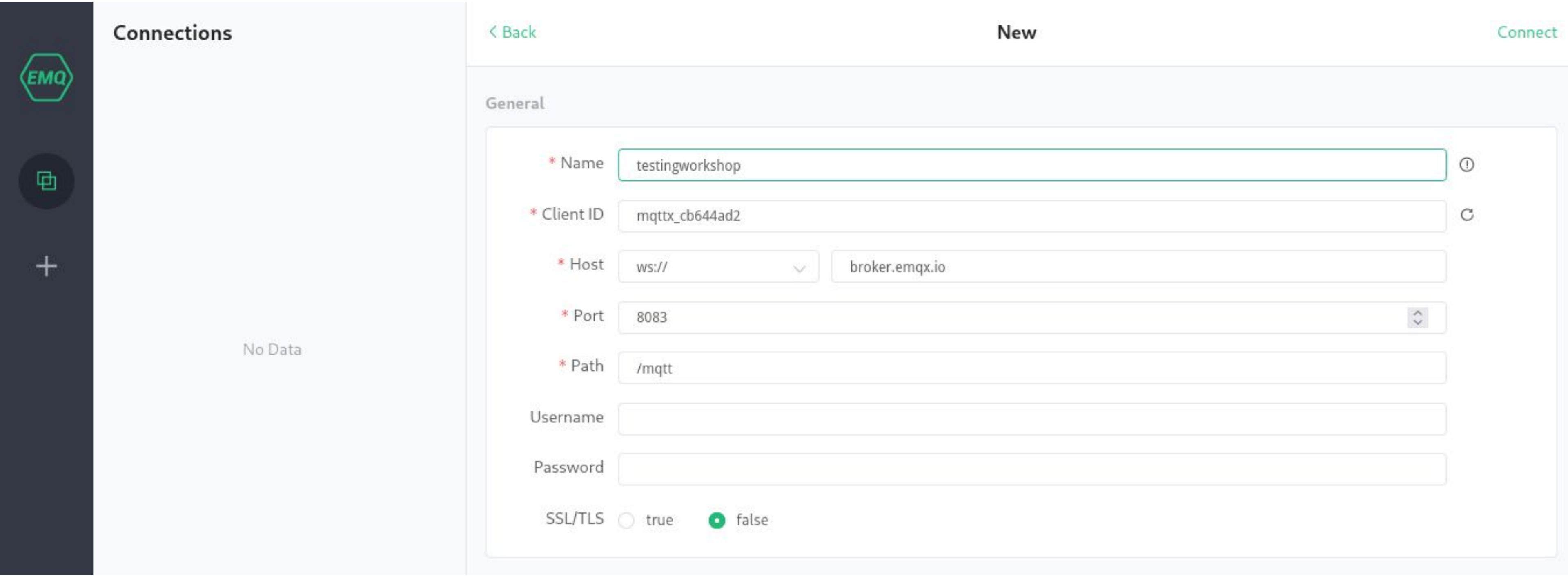
There is no RTC on Wemos, thus we need to sync time on every boot.

```
//mqtt.begin(mqttBroker, {
      .lwtTopic = "workshop123/lwt",
      .lwtMsg = "offline",
      .lwtQos = 0,
      .lwtRetain = 0});
```

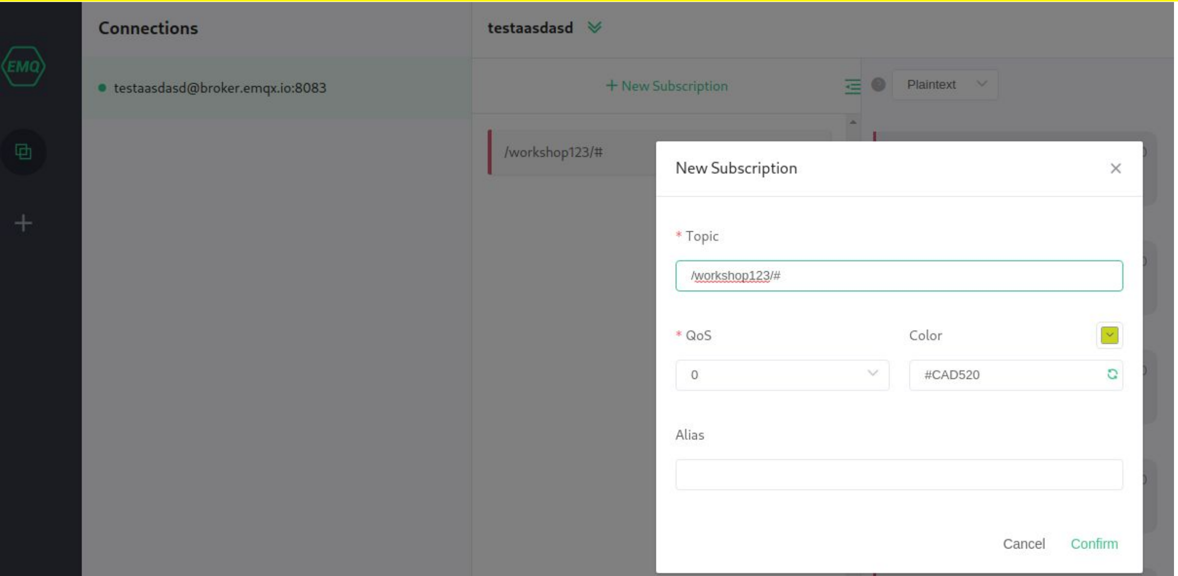
MQTT *to the Internet and beyond!* (2/2)

```
mqtt.onData([](String topic, String data, bool cont) {
  Serial.printf("Data rx, topic: %s, data: %s\n", topic.c_str(), data.c_str());
});

mqtt.onSubscribe([](int sub_id) {
  Serial.printf("Subscribe topic id: %d ok\n", sub_id);
});

mqtt.onConnect([]() {
  Serial.printf("MQTT: Connected\n");
  mqtt.subscribe("/workshop123/example", 1);
});
```

Interacting over Web (tools.emqx.io)



* Name	testingworkshop
* Client ID	mqttx_cb644ad2
* Host	ws:// broker.emqx.io
* Port	8083
* Path	/mqtt
Username	
Password	
SSL/TLS	○ true ● false



● testaasdasd@broker.emqx.io:8083

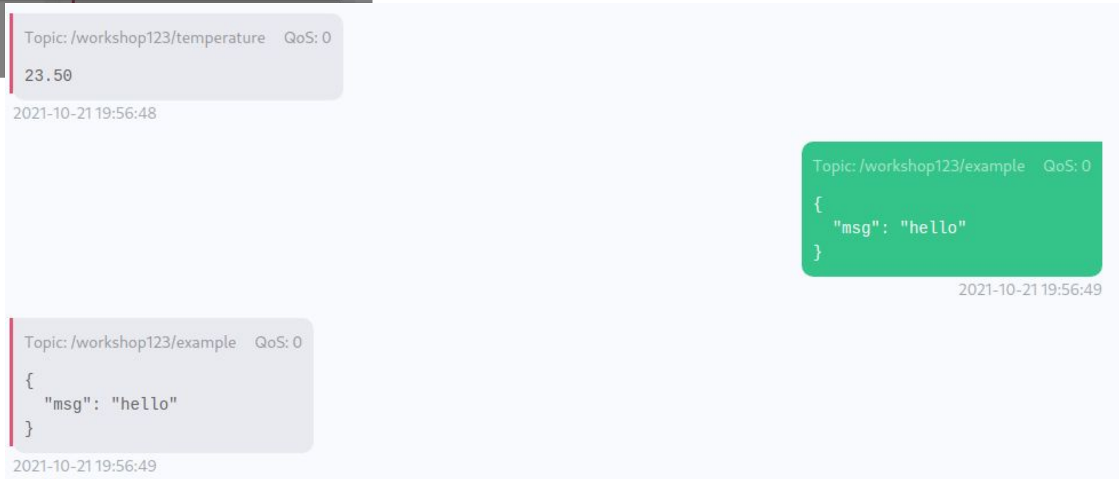
New Subscription

* Topic: /workshop123/#

* QoS: 0

Color: #CAD520

Alias:



Topic: /workshop123/temperature QoS: 0

```
23.50
```

2021-10-21 19:56:48

Topic: /workshop123/example QoS: 0

```
{
  "msg": "hello"
}
```

2021-10-21 19:56:49

Topic: /workshop123/example QoS: 0

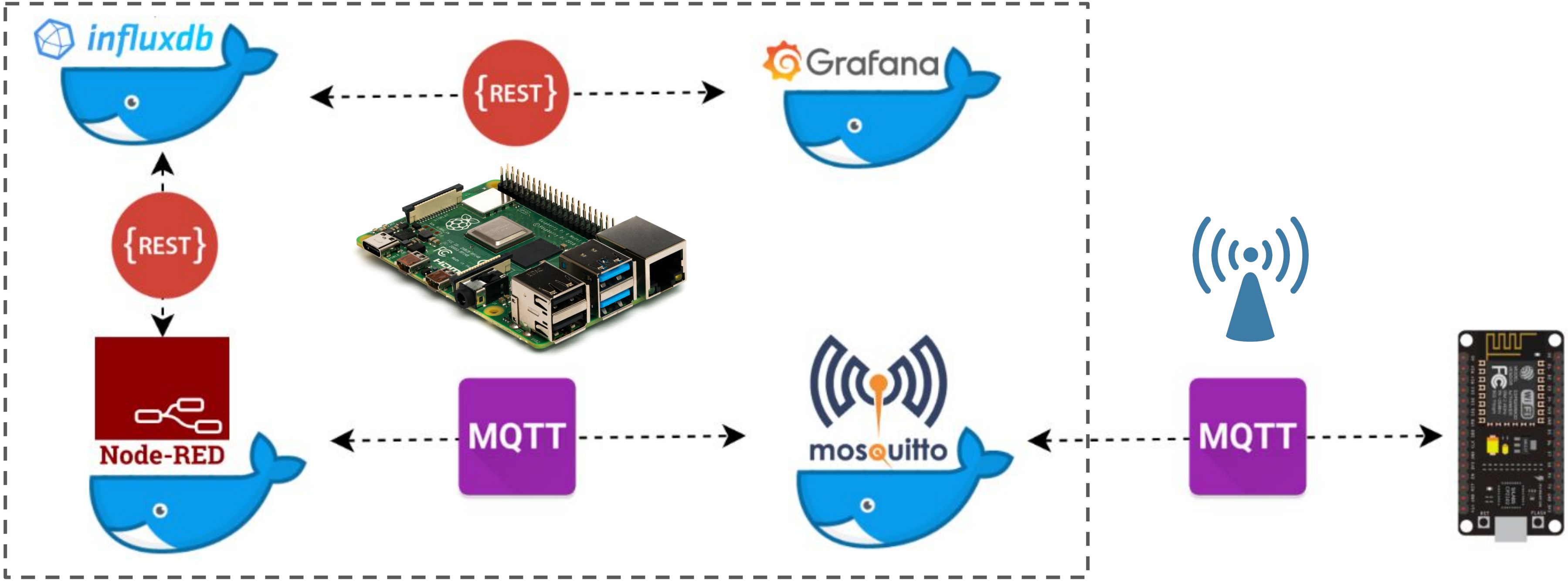
```
{
  "msg": "hello"
}
```

2021-10-21 19:56:49

Next steps (ideas)

- ➔ Publish JSON messages
  - ◆ Find a lib, install, read the example, ...
- ➔ Toggle the LED remotely
  - ◆ Subscribe and change state in accordance to the message.
- ➔ Program your system with Node-RED
  - ◆ Install it and make your first flow to periodically toggle the LED
- ➔ Make a Dashboard with Grafana
- ➔ Store historical data with InfluxDB or other Time-Series database
- ➔ Install and configure your own broker, dashboard and database
  - ◆ Mosquitto, InfluxDB, Grafana, and Node-RED in Docker

Motivational Example: *PiHeadquarters*



PiHeadquarters, https://github.com/jpdias/piheadquarters

Read More

- IoT for Beginners - A Curriculum, https://github.com/microsoft/IoT-For-Beginners
- OWASP Internet of Things (Top 10), https://owasp.org/www-project-internet-of-things/
- Build Computer from Scratch, https://eater.net/
- Adafruit Learning System, https://learn.adafruit.com/
- Pimoroni Learning, https://learn.pimoroni.com/
- Awesome IoT List, https://github.com/phodal/awesome-iot
- https://twitter.com/internetofshit
- Andreas Spiess, https://www.youtube.com/channel/UCu7_D0o48KbfhpEohoP7YSQ

Project ideas:

- https://hackster.io
- https://hackaday.com/
- https://create.arduino.cc/projecthub

I want to spend *some* money...

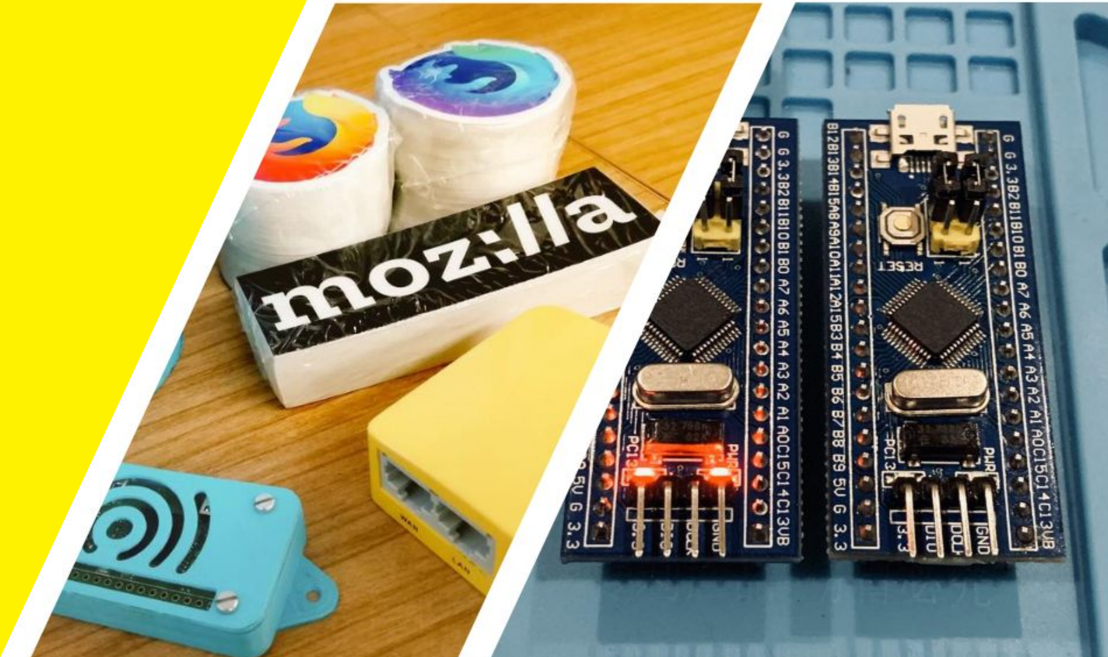
$ AliExpress, all the components, cheap (pick 10-day delivery to ensure delivery)
$ PCBWay, https://www.pcbway.com/ (making PCB, 5 for 5$ + ports)

$$ Mauser.pt, https://mauser.pt/
$$ PTRobotics, https://ptrobotics.com/
$$ Mouser.com, https://pt.mouser.com/ (all the things, free ports +50€)
$$ Farnell.com, https://pt.farnell.com/ (all the things)

$$$ Pimoroni, https://shop.pimoroni.com/
$$$ Adafruit, https://www.adafruit.com/

Call for Interest in IoT research:

- Software Engineering
- Visual programming & low-code
- Orchestration heterogeneous systems
- Autonomic Computing (self-healing)
- Fault-tolerance & Dependability
- Privacy & security
- Embedded and retro computing



It's a Wrap!

João Pedro Dias
https://jpdias.me
@jpd1as

Bruno Lima
https://brunolima.info