



POLITECNICO
MILANO 1863



Internet of Things Lab

Lab 4: Wokwi

Agenda

- **Wokwi**
 - Platform, devices etc...
- **ESP 32 examples**
 - Leds, Interrupt, PWM, etc...
- **Playing with MQTT**
 - Simple Pub/Sub application on ESP32s

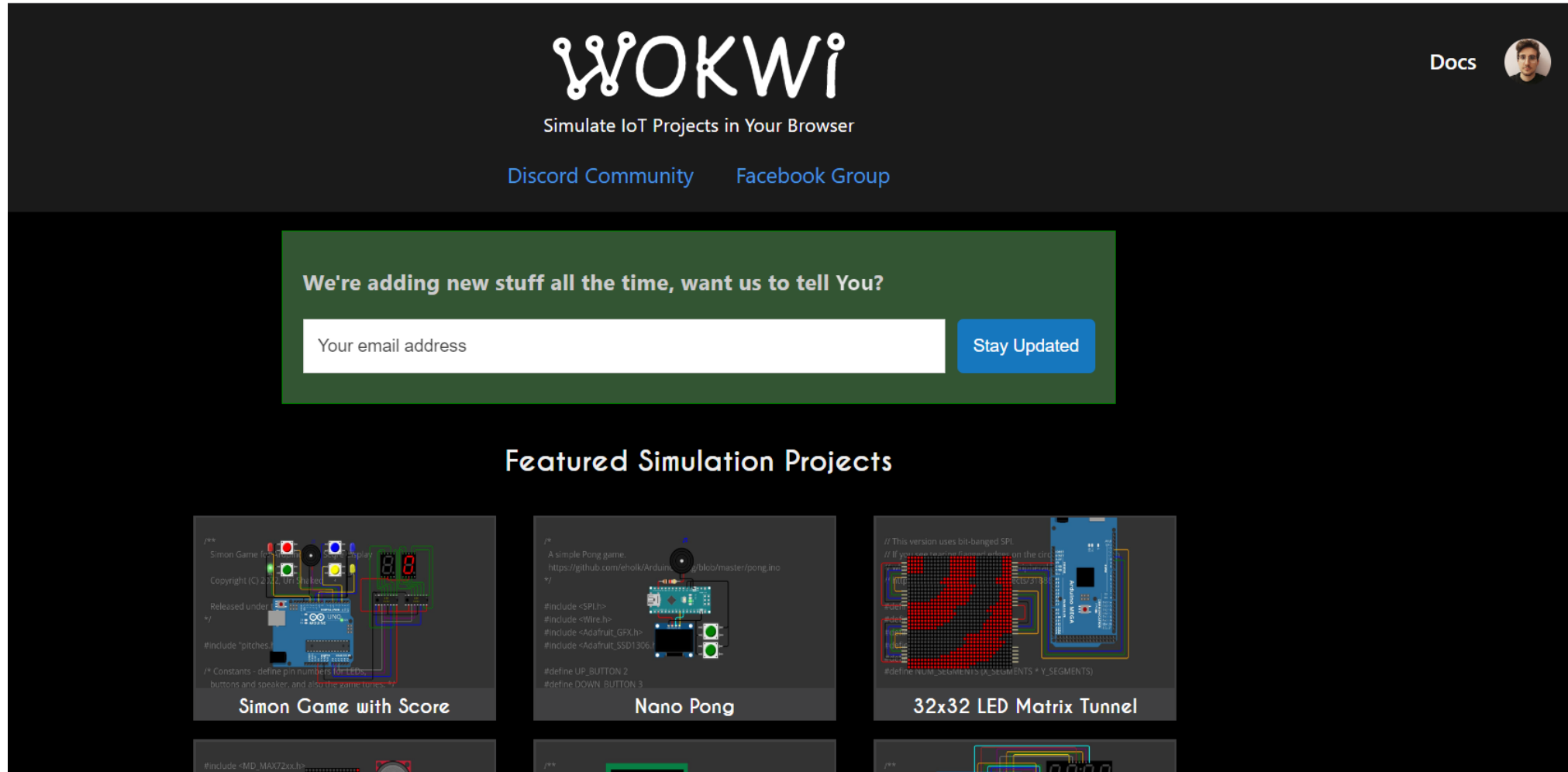


POLITECNICO
MILANO 1863




Wokwi Platform

Wokwi platform



The screenshot shows the Wokwi website homepage. At the top, the Wokwi logo is displayed in a stylized white font, with the tagline "Simulate IoT Projects in Your Browser" below it. To the right, there are links for "Docs" and a user profile picture. Below the header, there are links for "Discord Community" and "Facebook Group". A green banner in the center contains the text "We're adding new stuff all the time, want us to tell You?" followed by a text input field labeled "Your email address" and a blue "Stay Updated" button. Below the banner, the section "Featured Simulation Projects" displays three project thumbnails. The first thumbnail is titled "Simon Game with Score" and shows a circuit diagram with an Arduino Uno, a 16x2 LCD, and a 4x4 keypad. The second thumbnail is titled "Nano Pong" and shows a circuit diagram with an Arduino Nano, a 16x2 LCD, and a 4x4 keypad. The third thumbnail is titled "32x32 LED Matrix Tunnel" and shows a circuit diagram with an Arduino Uno and a 32x32 LED matrix. Each thumbnail includes a snippet of C++ code.

WOKWi
Simulate IoT Projects in Your Browser

[Docs](#) 

[Discord Community](#) [Facebook Group](#)

We're adding new stuff all the time, want us to tell You?

Your email address

[Stay Updated](#)

Featured Simulation Projects

Simon Game with Score

```
/**
 * Simon Game with Score
 * Copyright (C) 2015-2017 by Alex
 * Released under the MIT license
 */
#include "pitches.h"

// Constants - define pin numbers for LEDs,
// buttons and speaker, and also the game tones.
//
```

Nano Pong

```
/**
 * A simple Pong game.
 * https://github.com/eholk/Arduino/blob/master/pong.ino
 */
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306>

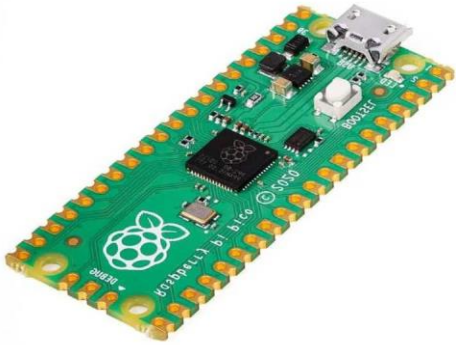
#define UP_BUTTON 2
#define DOWN_BUTTON 3
```

32x32 LED Matrix Tunnel

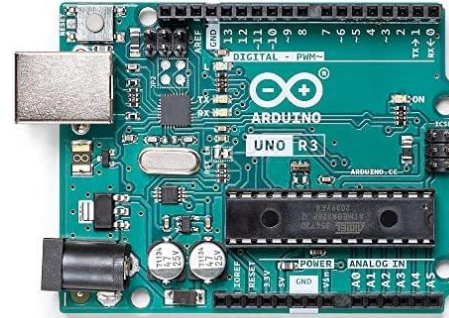
```
// This version uses bit-banged SPI
// If you have a hardware SPI port on the circuit board,
// you can use that instead.
#define NUM_SEGMENTS (X_SEGMENTS * Y_SEGMENTS)
```

Available at: www.wokwi.com

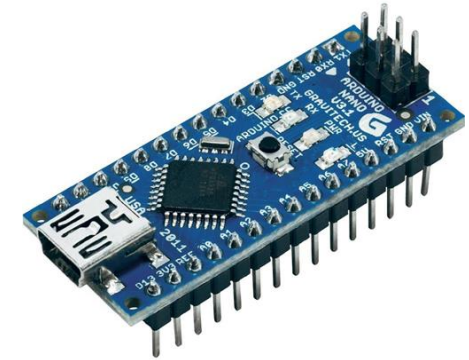
Supported Hardware



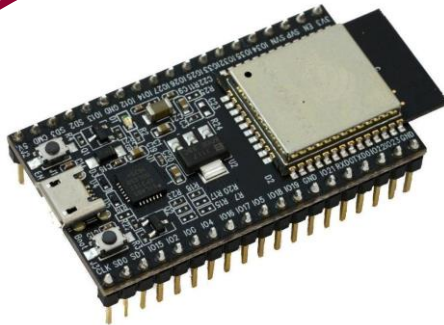
Raspberry
Pi Pico



Arduino Uno



Arduino Nano

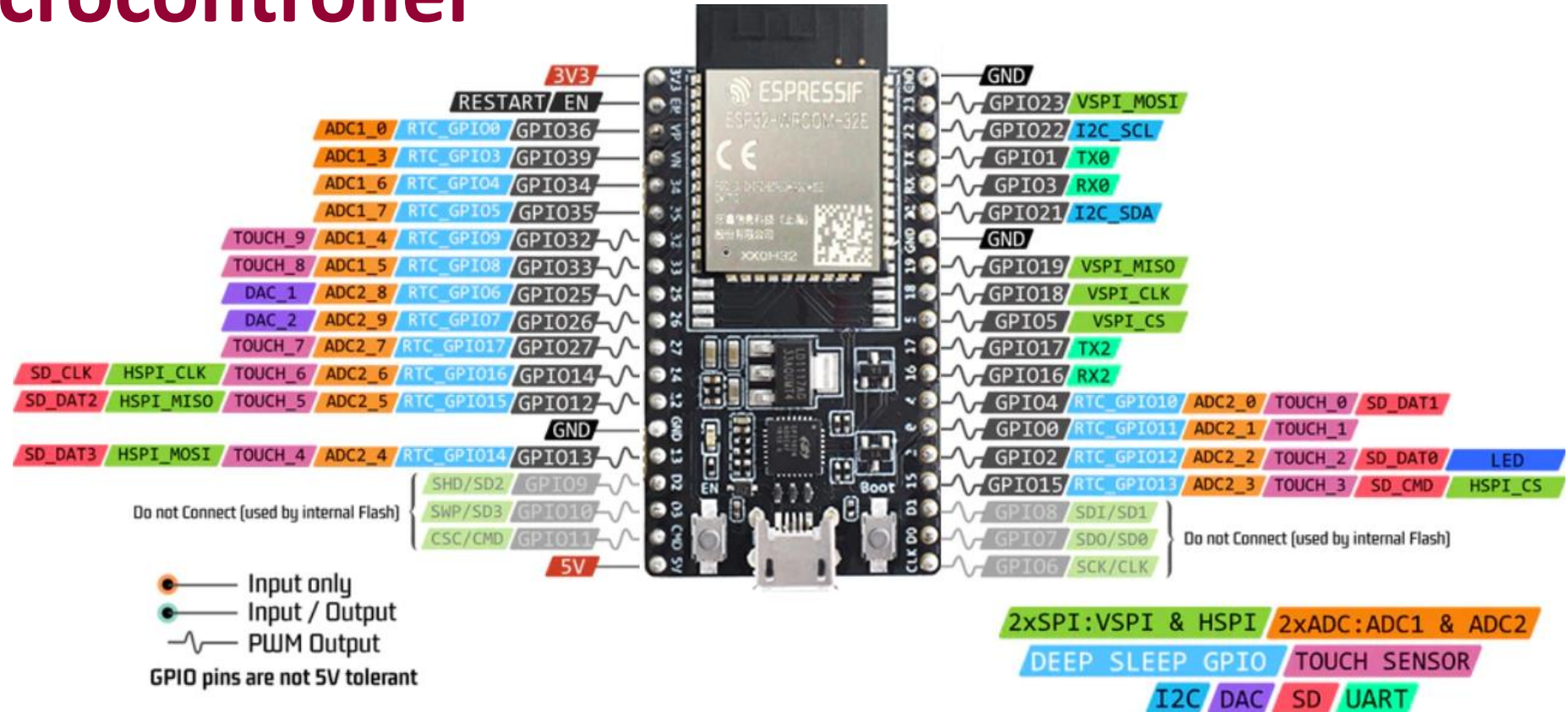


ESP32



Arduino Mega

ESP32 microcontroller



- 4MB RAM, Dual core CPU
- Built-in Wi-Fi and Bluetooth
- Low cost, low energy
- Open Source

More here: <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>



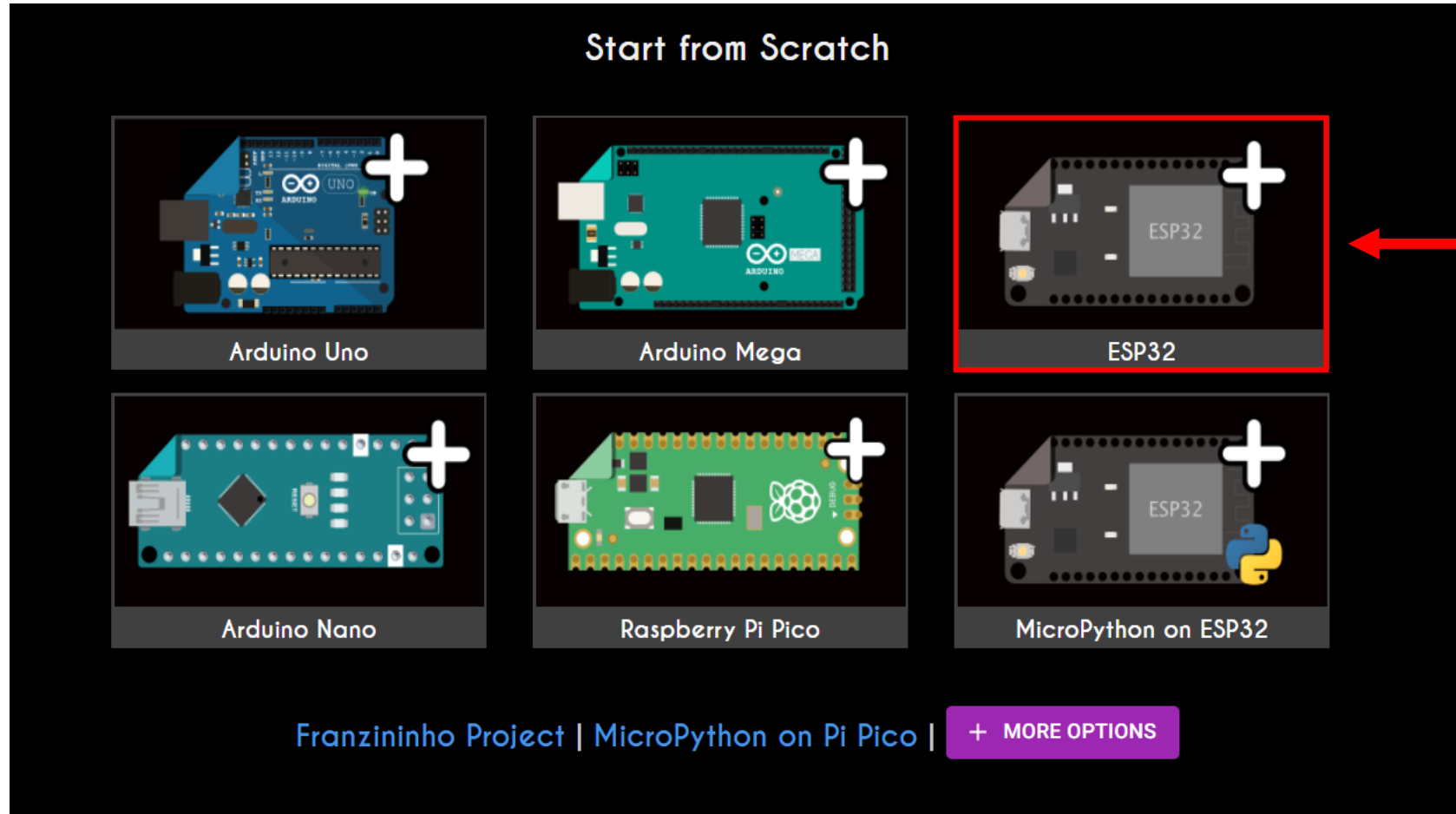
POLITECNICO
MILANO 1863



Playing with ESP32

Start a new Project

Scroll down the homepage and start a new ESP32 project from scratch



Start a new Project

The screenshot shows the WOKWI web IDE interface. The top bar includes the WOKWI logo, a 'SAVE' button, a 'SHARE' button, and a 'Docs' link with a user profile icon. The main workspace is divided into two panels. The left panel, labeled 'Code Space (Software)', contains a code editor with a file named 'sketch.ino' and a 'Library Manager' tab. The code in the editor is as follows:

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   Serial.begin(115200);  
4   Serial.println("Hello, ESP32!");  
5 }  
6  
7 void loop() {  
8   // put your main code here, to run repeatedly:  
9   delay(10); // this speeds up the simulation  
10 }  
11
```

The right panel, labeled 'Design space (Hardware)', features a 'Simulation' tab with three buttons: a green play button (labeled 'Start'), a purple plus button, and a grey three-dot menu button. Below these buttons is a detailed image of an ESP32 development board.

Annotations with red arrows point to the following elements:

- Json model**: Points to the 'diagram.json' tab in the top bar.
- External Libraries**: Points to the 'Library Manager' tab in the top bar.
- Components**: Points to the purple plus button in the 'Simulation' tab.
- Start**: Points to the green play button in the 'Simulation' tab.
- Code Space (Software)**: Points to the code editor area.
- Design space (Hardware)**: Points to the ESP32 board image.

Components

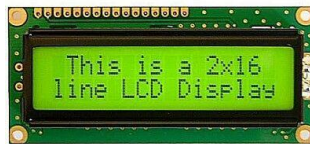
Resistors



Pushbutton



LCD Displays



Potentiometers



Leds



**Temperature/Humidity
sensor (DHT22)**



Speakers/Buzzers



First example: LEDs (1)

Part 1: Turn a LED on and off via software

```
1  #define LED 12 // The digital pin to which a led is connected.
2
3
4  void setup() {
5      //Pin Setup
6      pinMode(LED, OUTPUT);
7      //Serial Setup
8      Serial.begin(115200);
9
10 }
11 void loop() {
12     digitalWrite(LED, true);
13     delay(500);
14     digitalWrite(LED, false);
15     delay(500);
16
17 }
18
```

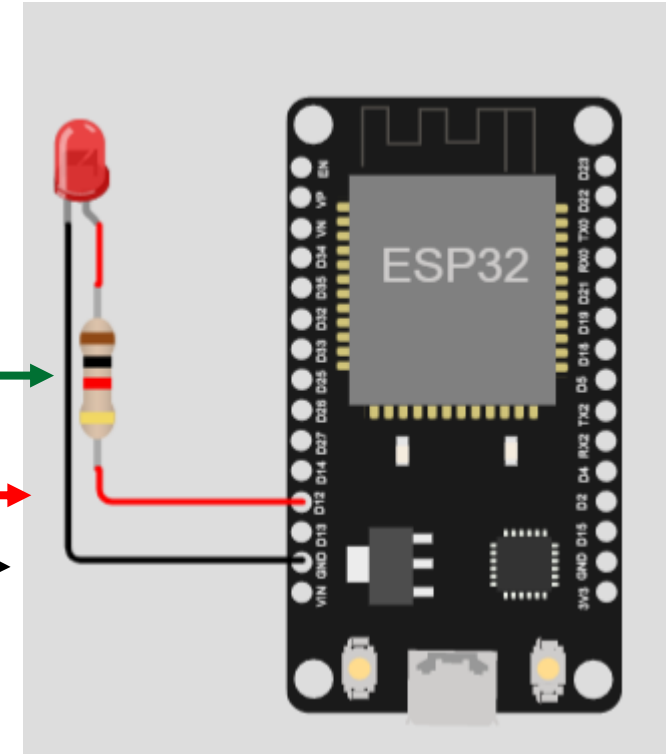
Code

100Ω

PIN D12

GROUND

Set LED ON/OFF



Design

First example: LEDs (2)

Part 2: Let's do it with a button now!

sketch.ino • diagram.json Library Manager ▾

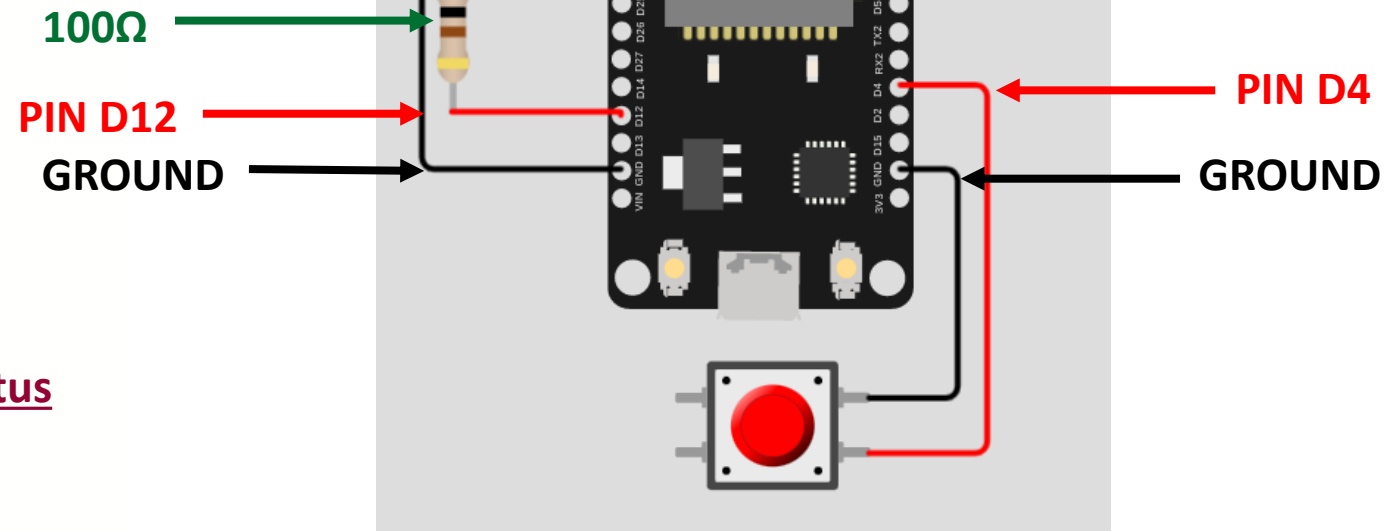
```

1  #define LED 12 // The digital pin to which a led is connected.
2  #define BUTTON 4 // The digital pin to which the button is connected
3
4  bool led_value = 0; //Led status
5
6  void setup() {
7    //Pin Setup
8    pinMode(LED, OUTPUT);
9    pinMode(BUTTON, INPUT_PULLUP);
10   //Serial Setup
11   Serial.begin(115200);
12
13 }
14 void loop() {
15
16   Serial.println(digitalRead(BUTTON));
17   if (!digitalRead(BUTTON)){
18     led_value=!led_value;
19   }
20   delay(100);
21   digitalWrite(LED, led_value);
22   delay(100);
23 }
24

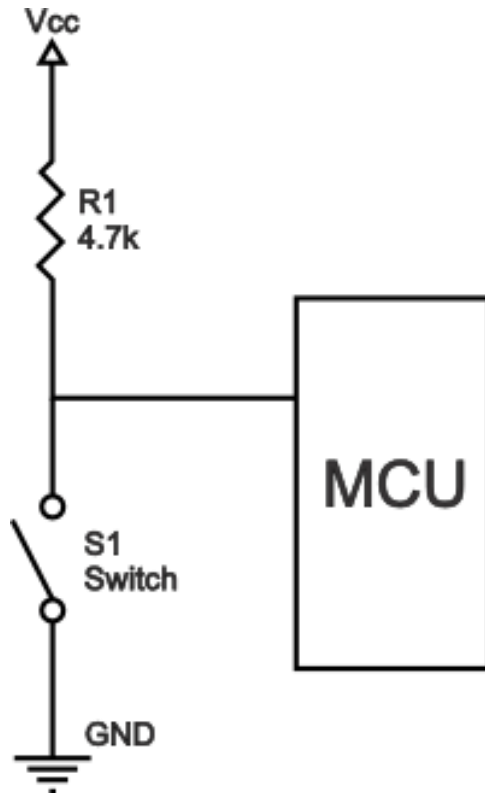
```

Check Button Status

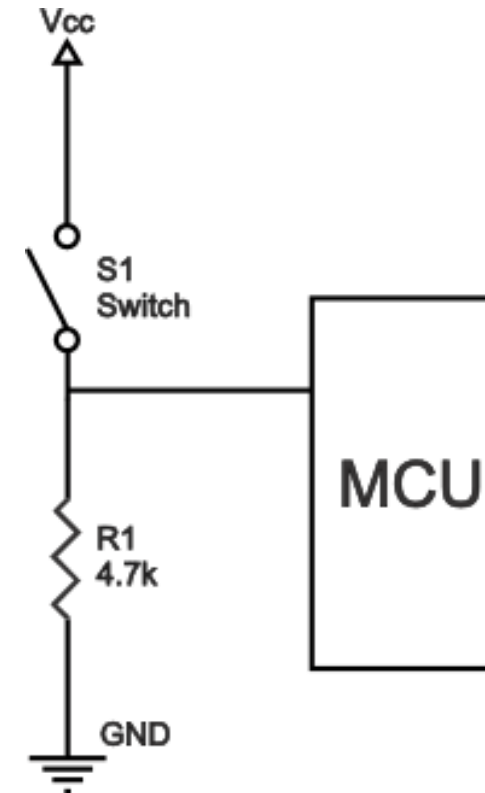
Set LED ON/OFF



PULLUP vs PULLDOWN



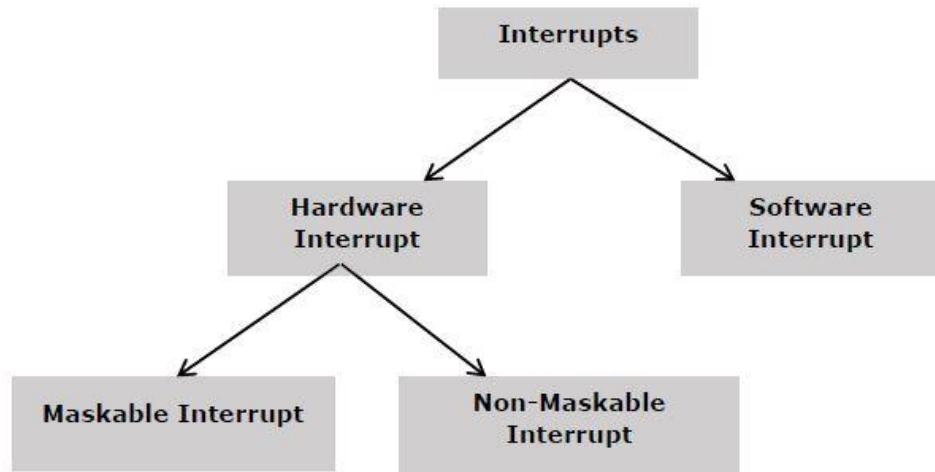
PULLUP RESISTOR



PULLDOWN RESISTOR

More here: <https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/#>

Interrupt and Interrupt Service Routine (ISR)



Interrupt is an event or signal that requests the CPU's attention.

- I. The processor completes the current instruction, save its state and starts the implementation of an Interrupt Service Routine (ISR)
- II. ISR is a program that tells the processor what to do when the interrupt occurs. After the ISR execution, control returns to the main routine where it was interrupted

Interrupts in Action

Goal: Turn a LED on and off with button using Interrupts

sketch.ino • diagram.json • Library Manager ▾

```

1  #define LED 12
2  #define BTN 4
3
4  void IRAM_ATTR trigger_led(){           Interrupt
5      digitalWrite(LED, !digitalRead(LED)); Function
6  }
7  void setup() {
8      // put your setup code here, to run once:
9      Serial.begin(115200);
10     Serial.println("Hello, ESP32!");
11     pinMode(LED, OUTPUT);
12     pinMode(BTN, INPUT);
13     attachInterrupt(BTN, trigger_led, RISING);
14 }                                         Attach Interrupt
15
16 void loop() {
17     // put your main code here, to run repeatedly:
18     delay(10); // this speeds up the simulation
19 }

```

Code

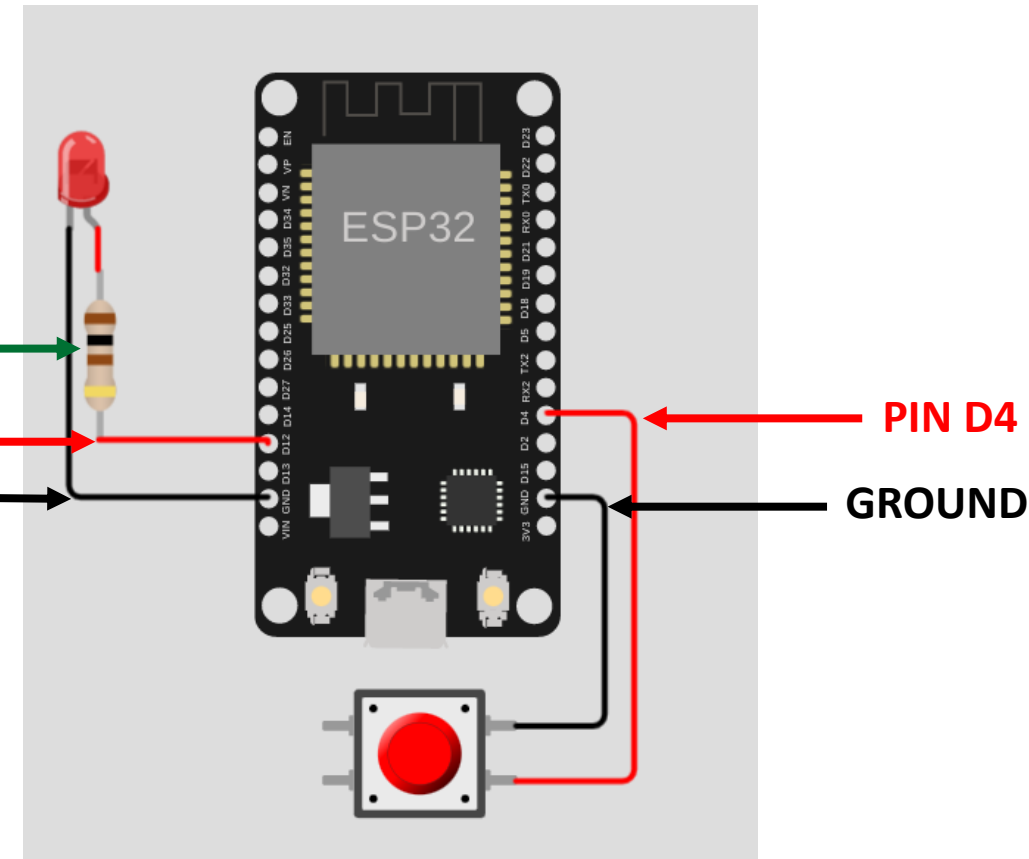
100Ω

PIN D12

GROUND

PIN D4

GROUND



Design

Interrupts in Action (with BOUNCE)

Goal: Turn a LED on and off with button using Interrupts

sketch.ino

diagram.json

Library Manager

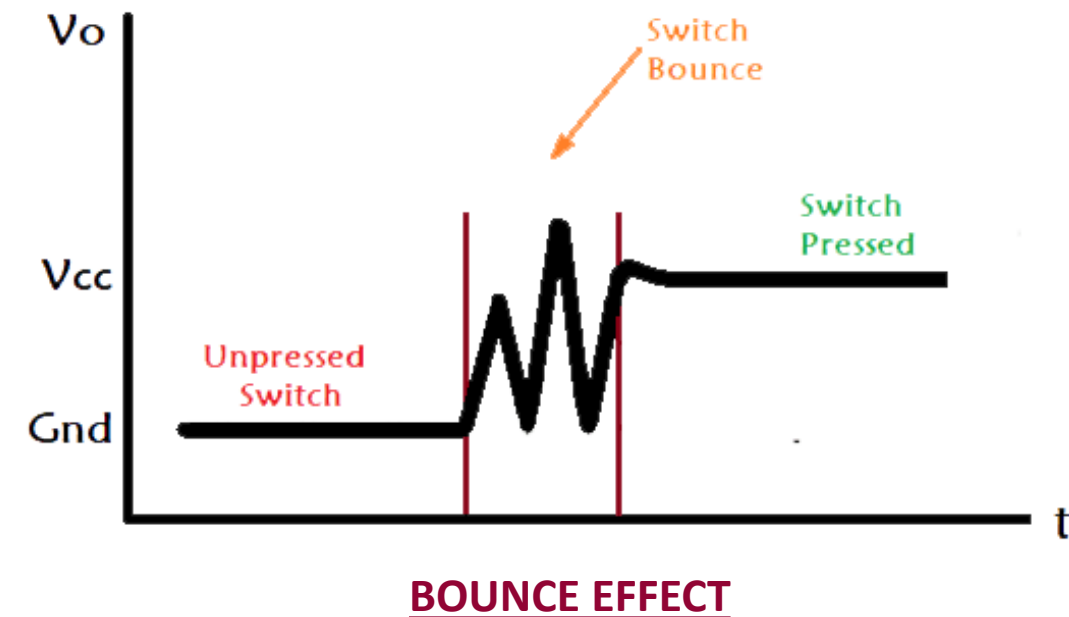
```

1  #define LED 12 // The digital pin to which the led is connected.
2  #define BUTTON 4 // The digital pin to which the button is connected.
3
4  #define DEBOUNCE_DELAY 50 //ms to wait to solve the bouncing problem
5
6  //Volatile: Tells to the compiler that this variable
7  // may change at any time (Used in ISR)
8  volatile int last_mills=0;
9
10 void IRAM_ATTR trigger_led(){
11     if (millis()-last_mills > DEBOUNCE_DELAY){
12         digitalWrite(LED, !digitalRead(LED));
13     }
14     last_mills=millis();
15 }
16
17 void setup() {
18     //Pin Setup
19     pinMode(LED, OUTPUT);
20     pinMode(BUTTON, INPUT);
21
22     //Interrupt Setup
23     attachInterrupt(BUTTON, trigger_led, RISING);
24 }
25 void loop() {
26 }
27

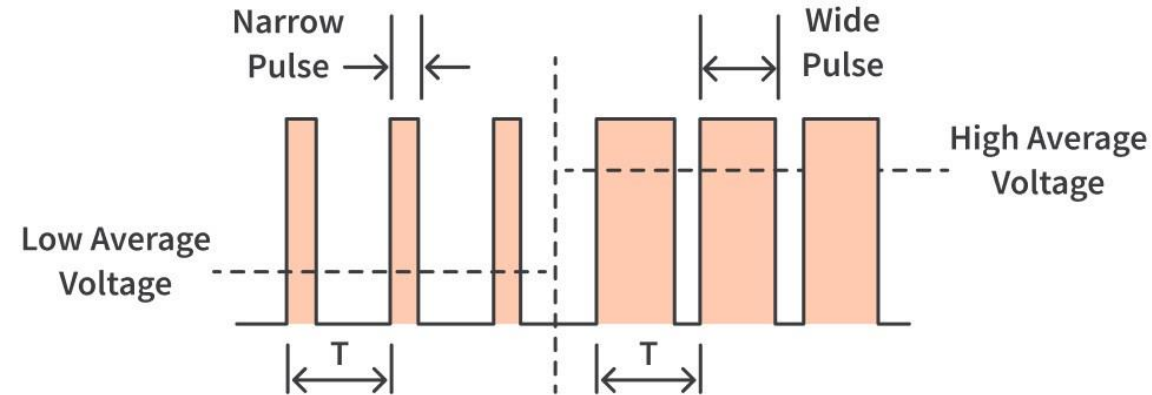
```

Interrupt
Function

Attach Interrupt



Pulse-Width Modulation (PWM)

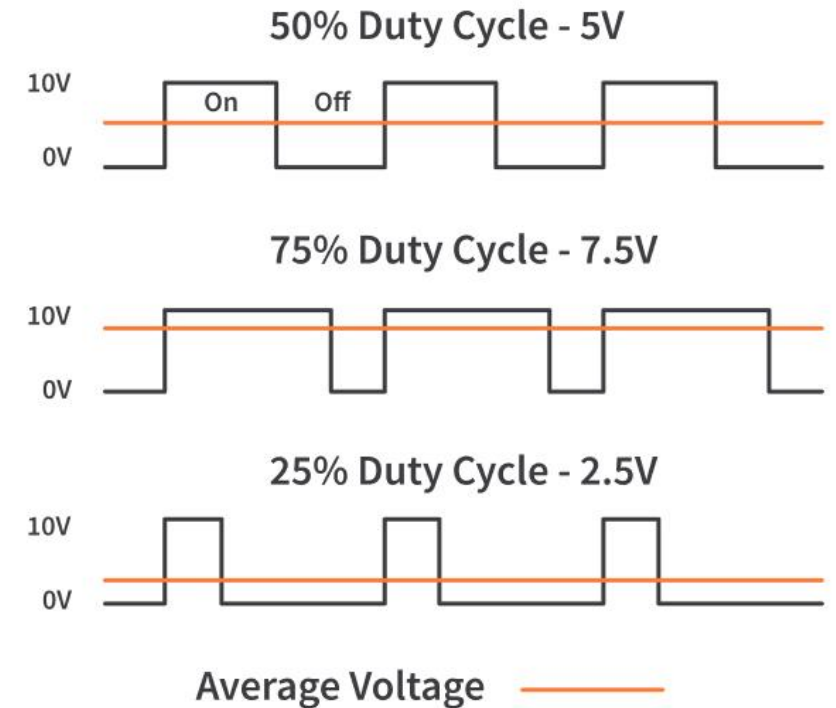


PWM is a technique to control analog devices, using a digital signal, to output an analog-like signal from a digital device

Duty cycle: *percentage of time a digital signal is “on” over an interval or period*

$$D = \frac{T_{on}}{Period} * 100$$

$$V_{avg} = \frac{D}{100} * V_{max}$$



PWM in Action (1)

Goal: Control LED brightness with PWM and a slider

sketch.ino

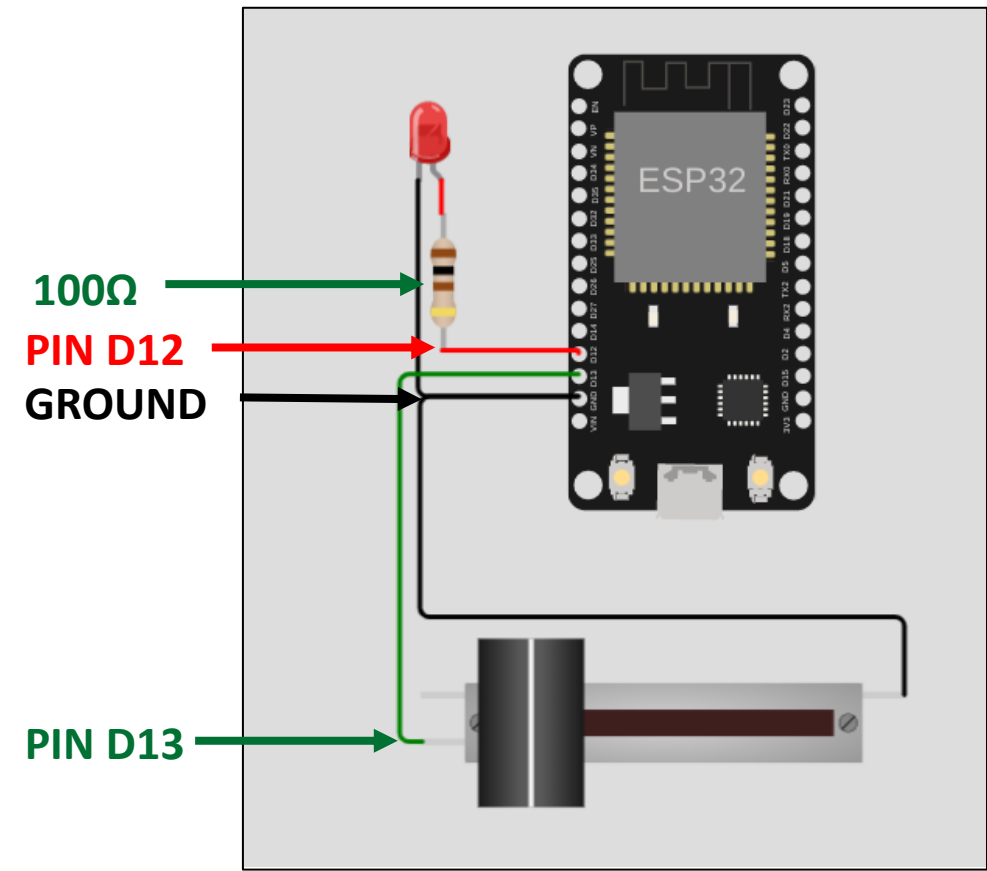
diagram.json

Library Manager

```

1  #define LED 12 // LED Digital pin
2  #define SLIDE 13 // Potentiometer Digital pin
3  #define C_TIME 100 //PWM Period in us
4
5  volatile int pwm_value = 0; //Duty Cycle Time
6  hw_timer_t *My_timer = NULL;
7
8  void IRAM_ATTR onTimer(){           Interrupt Function
9      bool led_status = digitalRead(LED);
10
11      if (led_status && pwm_value!=C_TIME){
12          digitalWrite(LED, !led_status); Toggle LED
13          timerWrite(My_timer, pwm_value);
14      }else if (!led_status && pwm_value!=0){
15          digitalWrite(LED, !led_status);
16          timerWrite(My_timer, C_TIME-pwm_value); Set Timer Value
17      }
18  }
19  }
20

```



PWM in Action (2)

Goal: Control LED brightness with PWM and a slider

sketch.ino

diagram.json

Library Manager

```
21 void setup() {
22   pinMode(LED, OUTPUT);
23   pinMode(SLIDE, INPUT);
24
25   My_timer = timerBegin(0, 80, true); //Timer initializer
26   //0: hw timer number (ESP32 has 3 hw timers available
27   //80: time divider. ESP32 clk 80MHz so we set evry tick to 1 us
28   //true: counter shoud increment
29
30   timerAttachInterrupt(My_timer, &onTimer, true); //Attach Interrupt
31
32   timerAlarmWrite(My_timer, C_TIME, true);
33   //C_TIME: number of microseconds after which the interrupt should occur
34   //true: timer counter will reload after interrupt
35
36   timerAlarmEnable(My_timer); //Just Enable
37 }
38 void loop() {
39   pwm_value = map(analogRead(SLIDE), 0, 4095, 0, C_TIME);
40 }
```



POLITECNICO
MILANO 1863



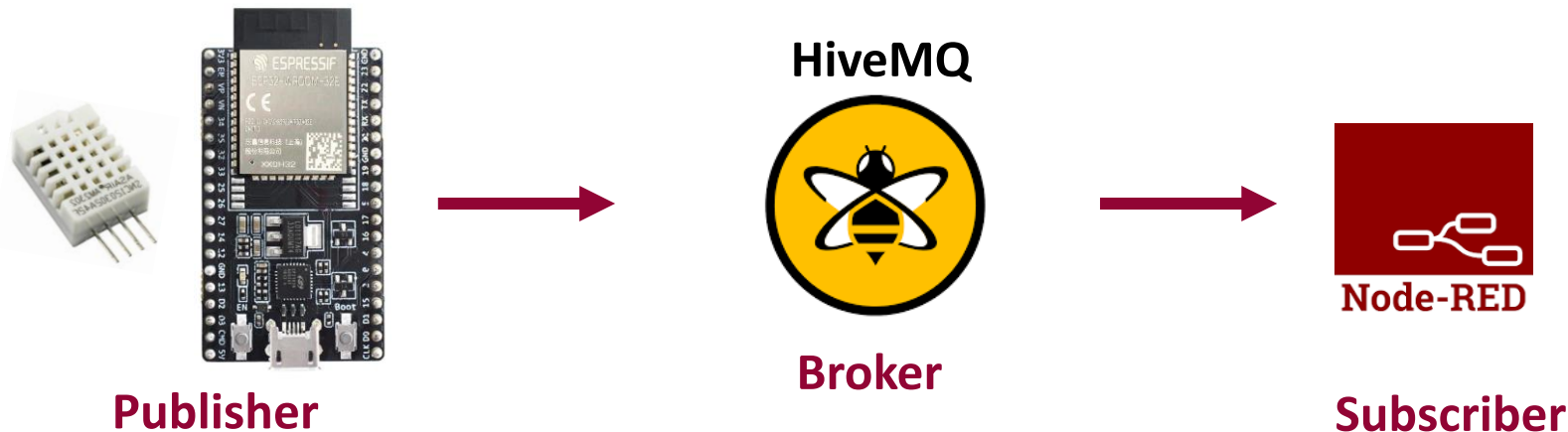
Playing with MQTT

Simple Pub/Sub Application

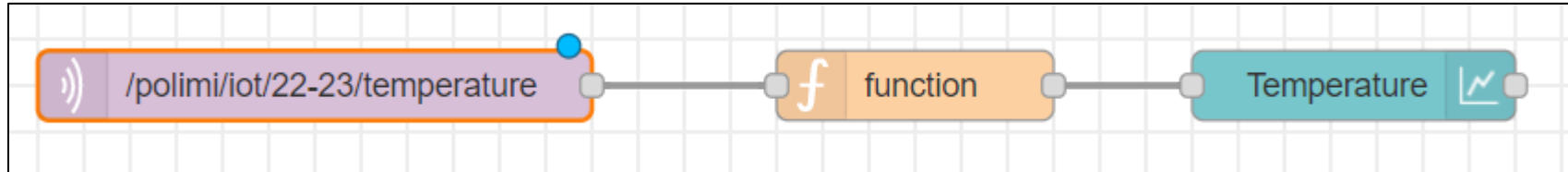
Goal: Read the temperature from a DHT22 sensor and update the status on a remote device using MQTT (device in node-red)

Two projects:

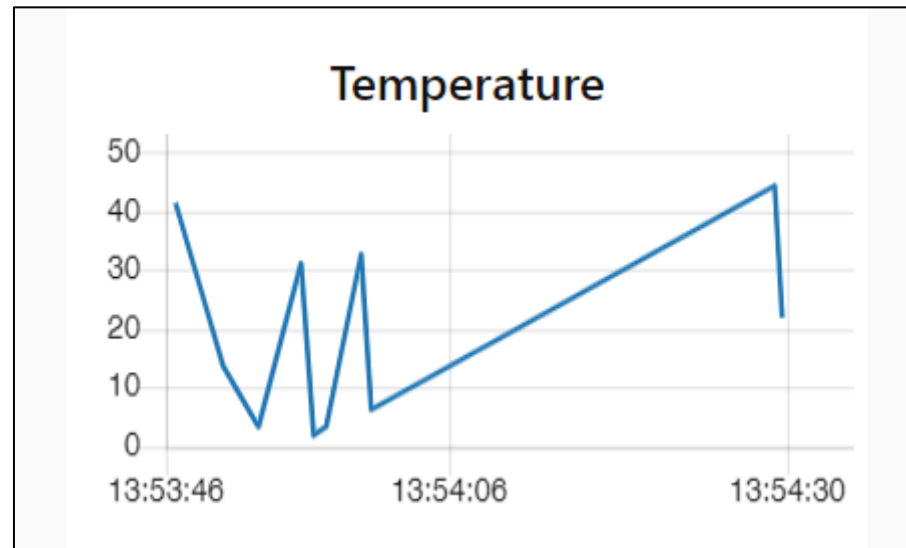
1. **Publisher:** The ESP32 reads the values from the DHT22 and publishes to the MQTT broker (**broker.hivemq.com**)
2. **Subscriber:** Node-Red node receiving the message and updating a plot



Simple Pub/Sub Application: Subscriber



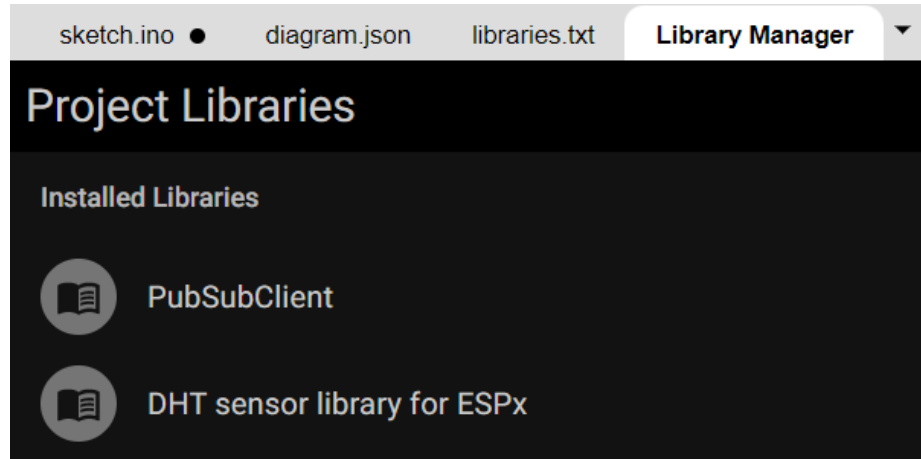
Node-Red flow



Node-Red chart

Simple Pub/Sub Application: Publisher (1)

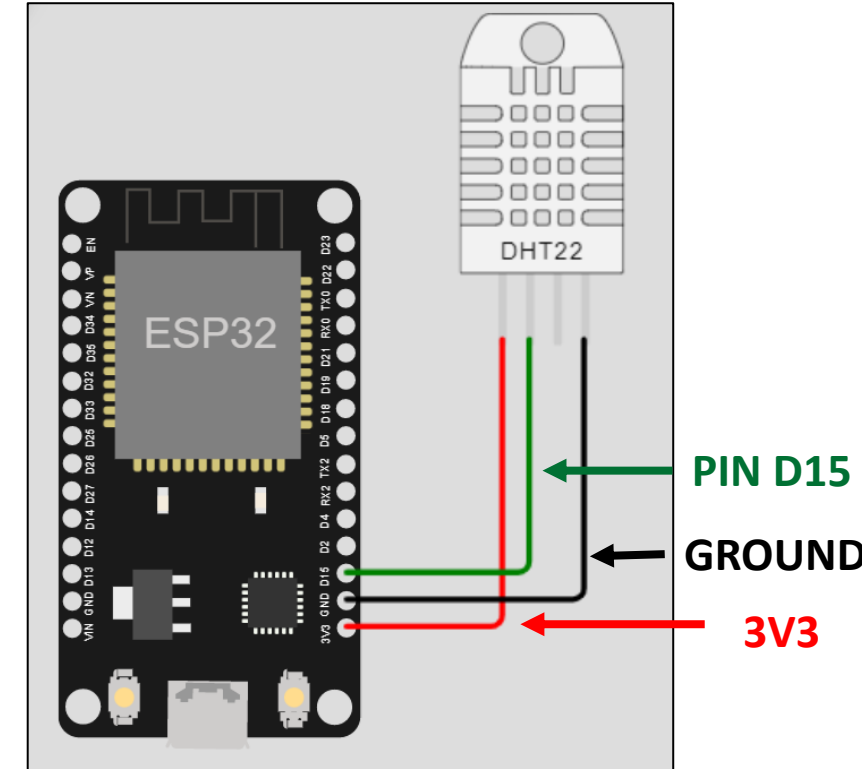
Add libraries first!



```

1 //Include needed libraries
2 #include <PubSubClient.h>
3 #include <WiFi.h>
4 #include "DHTesp.h"
5
6 #define PIN_DHT 15 //PIN USED FOR DATA IN DHT22 SENSOR
7
8 #define MQTT_TOPIC "/polimi/iot/22-23/temperature" //the MQTT topic
9 #define MQTT_CLIENT_ID "iot-polimi-palmese-pub1" // the MQTT client identifier
10
11 char strTemperature[10]; //string where we store the temperature to publish
12
13 DHTesp dht;
14 WiFiClient espClient;
15 PubSubClient mqtt_client(espClient);
  
```

This is the MQTT client



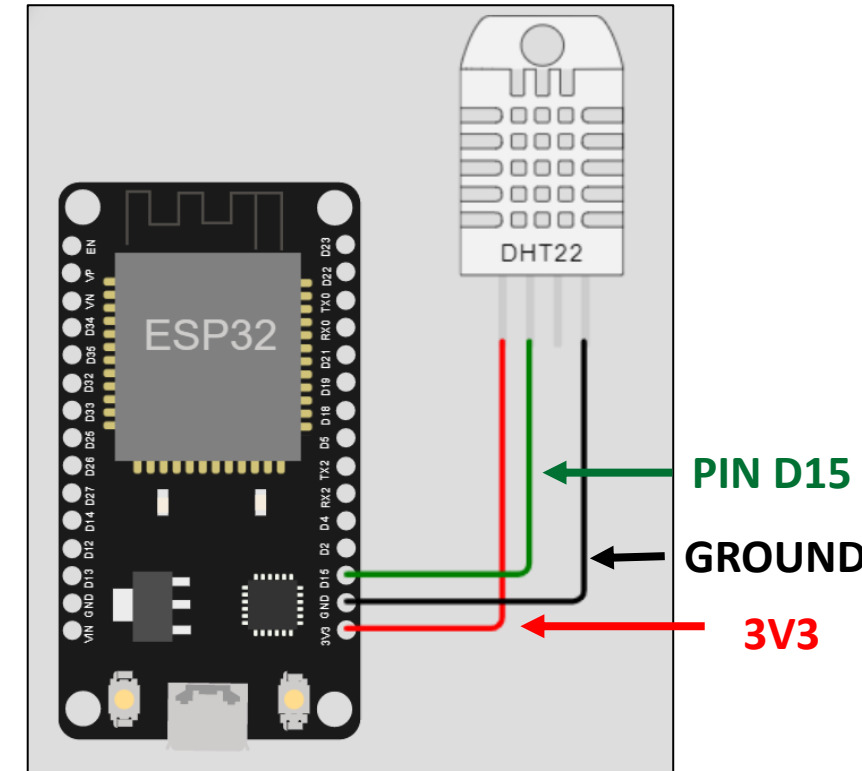
Simple Pub/Sub Application: Publisher (2)

Setup

```

17 void setup() {
18     //init pins
19     dht.setup(PIN_DHT, DHTesp::DHT22);
20
21     // Setup Wi-Fi
22     WiFi.mode(WIFI_STA);
23     WiFi.begin("Wokwi-GUEST", "");
24
25     delay(1000);           Give time to connect!
26
27     // Setup MQTT
28     mqtt_client.setServer("test.mosquitto.org", 1883);
29     mqtt_client.connect(MQTT_CLIENT_ID);
30 }

```



Main Loop

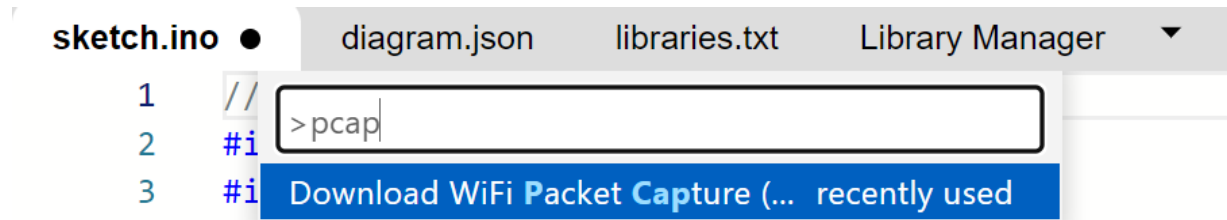
```

32 void loop() {
33     TempAndHumidity data = dht.getTempAndHumidity();
34     sprintf(strTemperature, "%.2fc", data.temperature);
35
36     mqtt_client.publish(MQTT_TOPIC, strTemperature); Publish!
37
38     delay(1000);
39 }

```


Download the PCAP in WOKWI

- In the code field press **F1**, a menu will open
- Search for 'PCAP' and click on "**Download Wi-Fi Packet Capture**"



- A PCAP file will be downloaded, with all the packets from/to the ESP32 device



More on Wokwi...

- PubSub Library API: <https://pubsubclient.knolleary.net/>
- Use **featured project** as guidelines
- Check for **trending projects** for nice ideas



Discord group



Facebook group