POLITECNICO
MILANO 1863

ANTLAB

Internet of Things Lab

Lab 1: Wokwi

Agenda

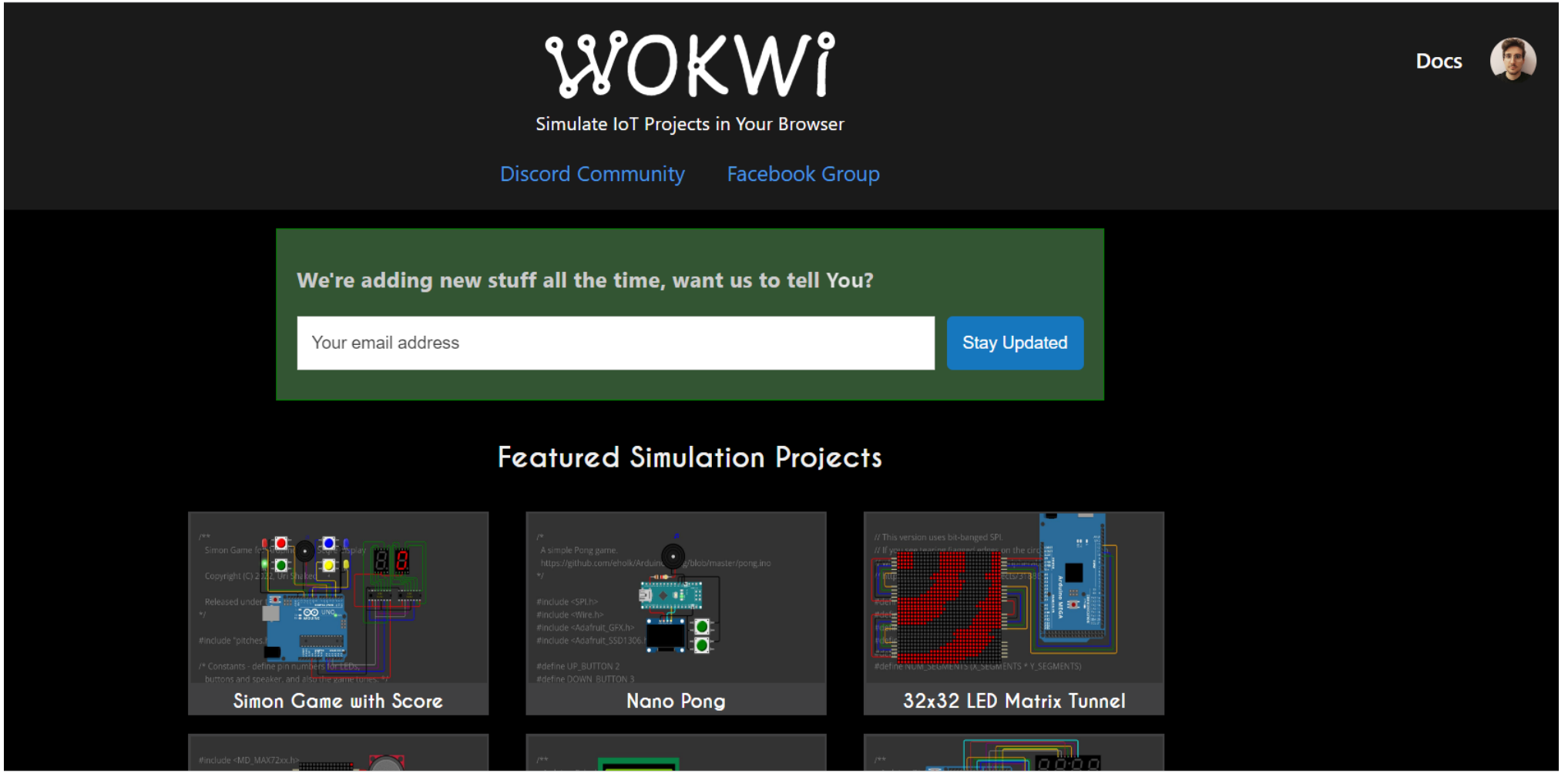
- **Wokwi**
  - Platform, devices etc…
- **EPS32 examples**
  - Leds, Interrupt, Sensors
- **Transmitting Data**
  - Simple TX/RX ESP32 application (ESP-NOW)

POLITECNICO
MILANO 1863

ANTLAB

Wokwi Platform

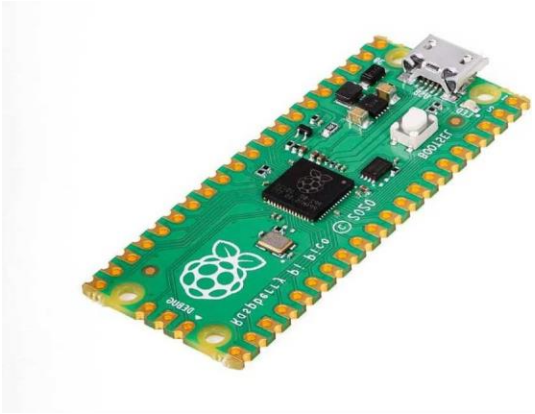
Wokwi platform



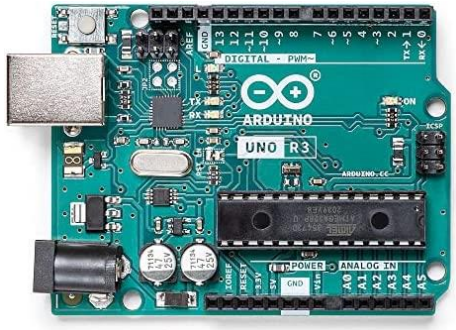
Available at: www.wokwi.com

ANTLAB

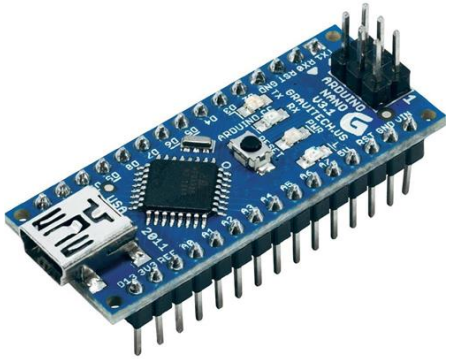
Supported Hardware



Raspberry
Pi Pico



Arduino Uno



Arduino Nano

ESP32

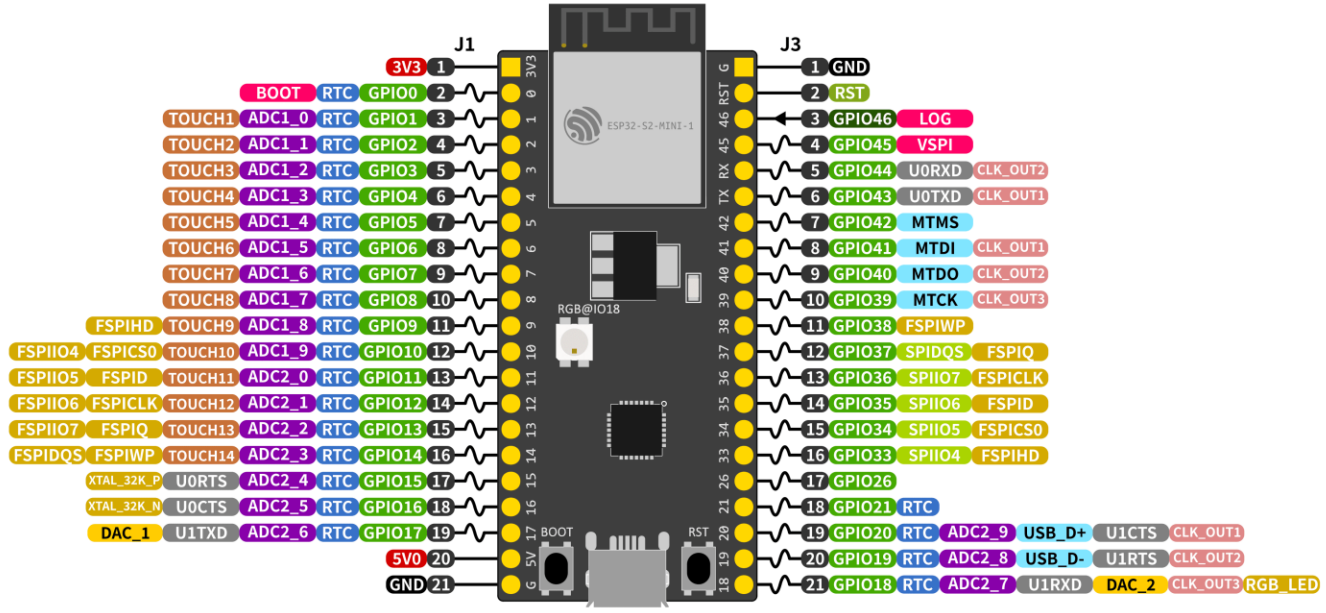
Arduino Mega

ANTLAB

ESP32 microcontroller

ESP32-S2-DevKitM-1

ESPRESSIF



ESP32-S2 Specs

32-bit Xtensa® single-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
320 KB SRAM (16 KB SRAM in RTC)
128 KB ROM
43 GPIOs, 4x SPI, 2x UART, 2x I2C,
Touch, I2S, RMT, LED PWM, USB-OTG,
TWAI®, 2x 8-bit DAC, 12-bit ADC

PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
JTAG/USB JTAG for Debugging and USB
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
DAC_X Digital-to-Analog Converter
STRAP Strapping Pin Functions
RTC RTC Power Domain (VDD3P3_RTC)
MISC Miscellaneous/Secondary functions
GND Ground
CLK_OUTx Clock Output
PWD Power Rails (3V3 and 5V)

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

ANTLAB

ESP32 Commercial Products

Wemo Smart Plugs by Belkin

SONOFF's POW Elite

Evapolar Personal Air Cooler

Shelly 1

ANTLAB

ESP32 microcontroller

20 ADC channels	20 channels of 12-bit SAR ADC with selectable ranges of 0-1V, 0-1.4V, 0-2V, or 0-4V
2 UART interfaces	2 UART interfaces with flow control and IrDA support
36 PWM outputs	25 PWM pins to control things like motor speed or LED brightness
2 DAC channels	Two 8-bit DACs to generate true analog voltages
SPI, I2C and I2S interface	Three SPI and one I2C interfaces for connecting various sensors and peripherals, as well as two I2S interfaces for adding sound to your project
14 Touch Pads	9 GPIOs with capacitive touch sensing

More here: https://lastminuteengineers.com/esp32-pinout-reference

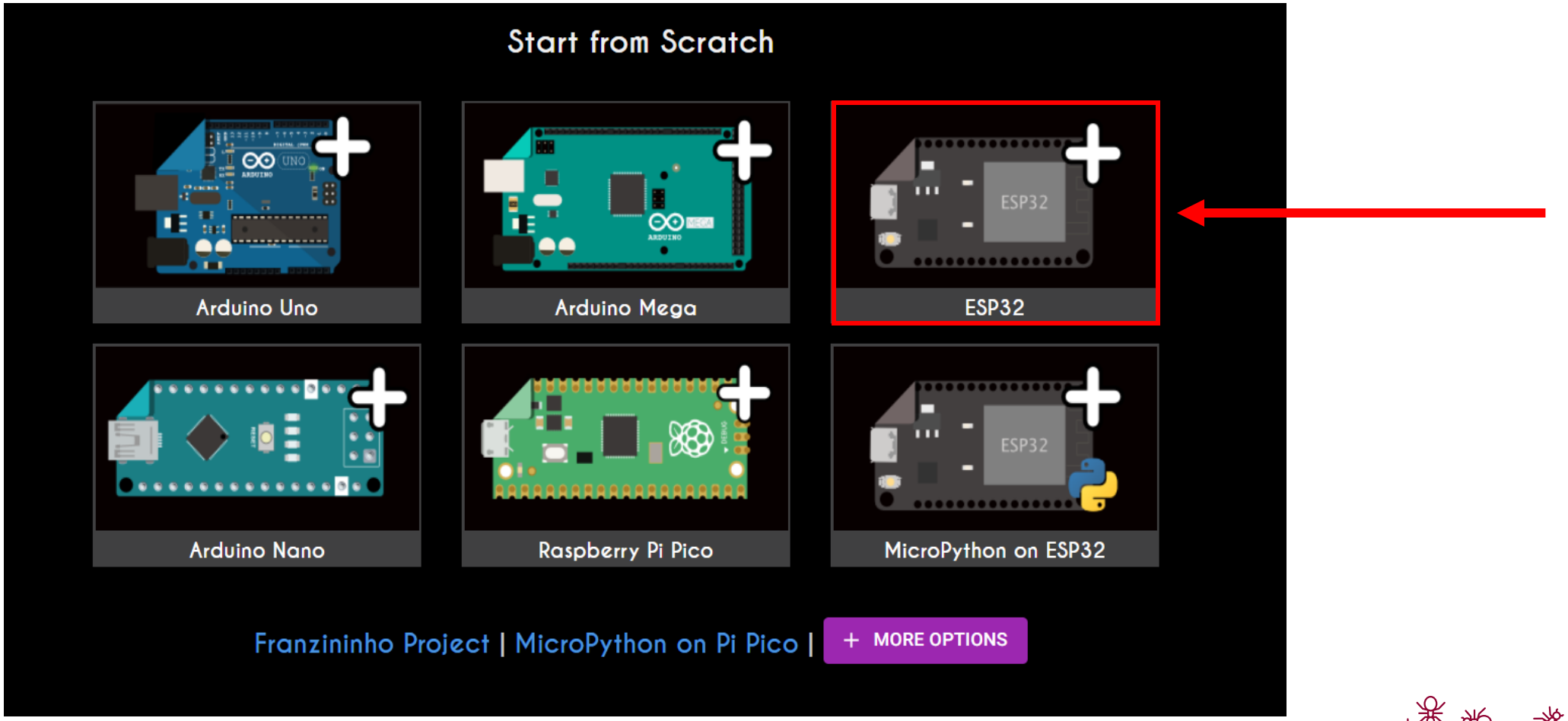
ANTLAB

POLITECNICO
MILANO 1863

Playing with ESP32

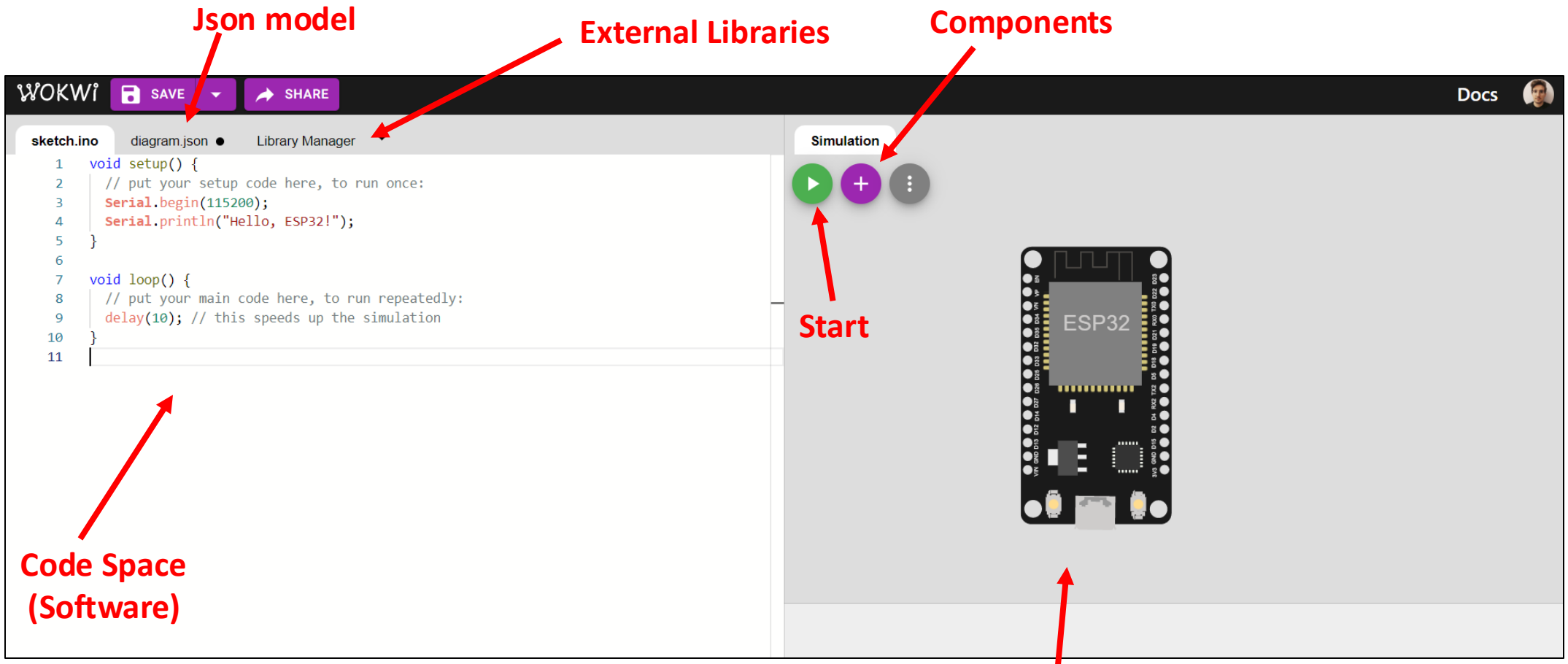
Start a new Project

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



ANTLAB

Start a new Project



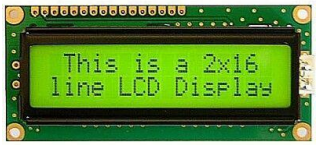
ANTLAB

Components

Resistors

Pushbutton

LCD Displays



Potentiometers

Leds

Temperature/Humidity sensor (DHT22)

Speakers/Buzzers

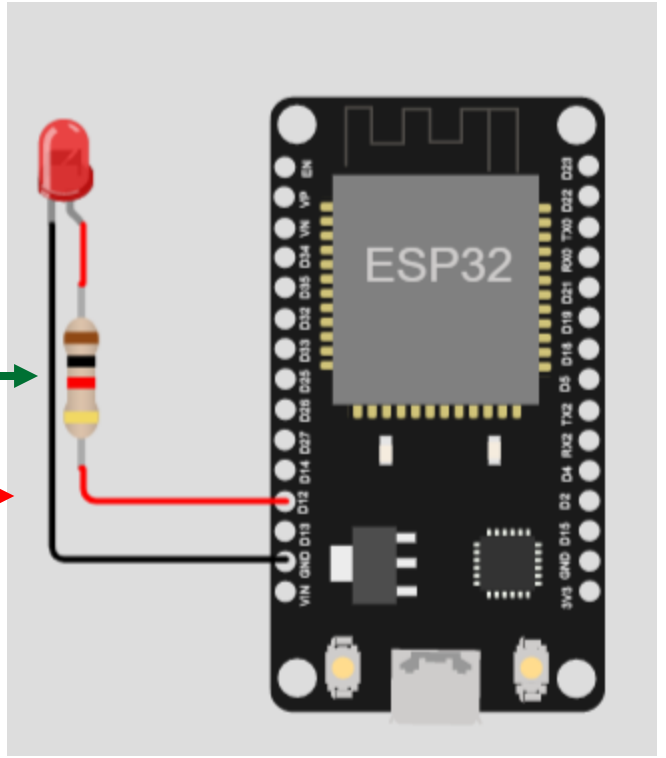
ANTLAB

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
```

Set LED ON/OFF



Code **Design**

ANTLAB

First example: LEDs (2)

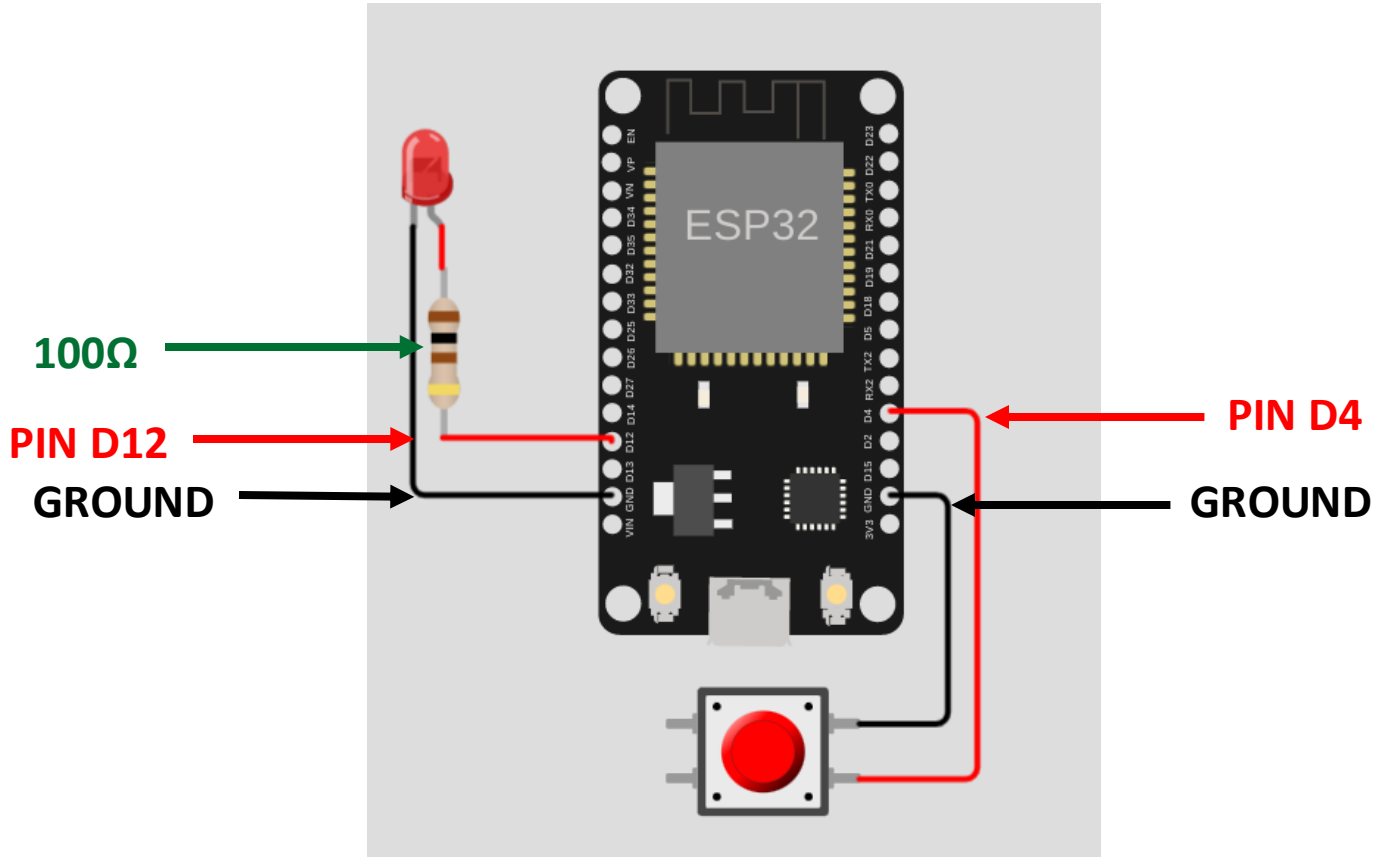
Part 2: Let's do it with a button now!

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

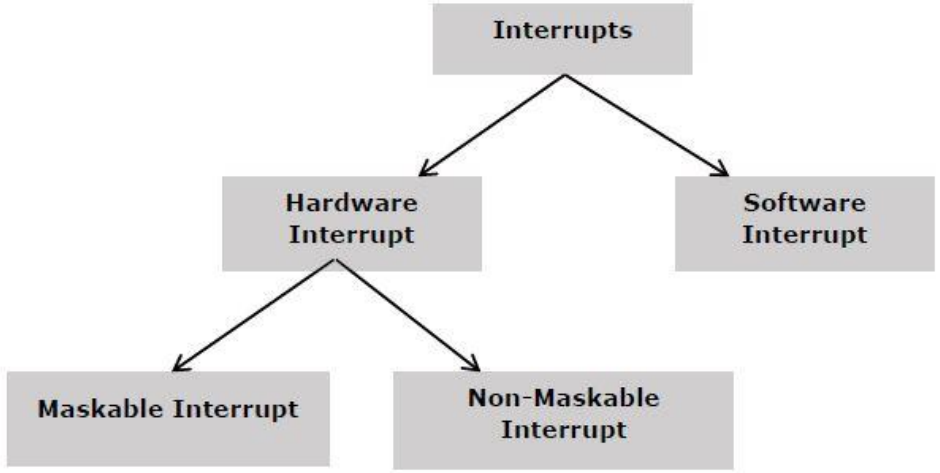
Set LED ON/OFF



Why INPUT_PULLUP? https://eepower.com/resistor-guide/resistor-applications/pull-up-resistor-pull-down-resistor/#

ANTLAB

Interrupts 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 it states 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

https://docs.espressif.com/projects/arduino-esp32/en/latest/api/gpio.html?highlight=isr#attachinterrupt

Interrupts in Action

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

sketch.ino ● diagram.json ● Library Manager

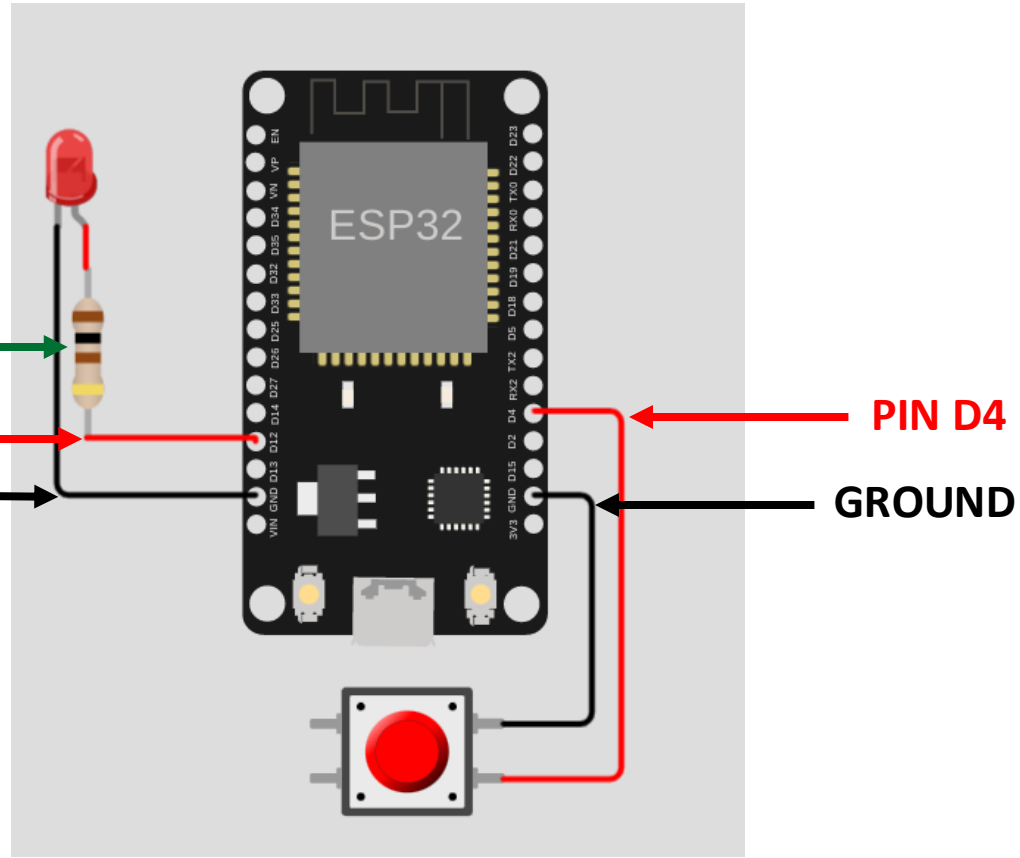
```
1   #define LED 12
2   #define BTN 4
3
4   void IRAM_ATTR trigger_led(){
5     digitalWrite(LED, !digitalRead(LED));
6   }
7   void setup() {
8     // put your setup code here, to run once:
9     Serial.begin(115200);
10    Serial.println("Hello, ESP32!");
11    pinMode(LED, OUTPUT);
12    pinMode(BTN,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

Interrupt Function

Attach Interrupt

Code

https://wokwi.com/projects/355358262276403201



Design

ANTLAB

Interrupts in Action (with BOUNCE)

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

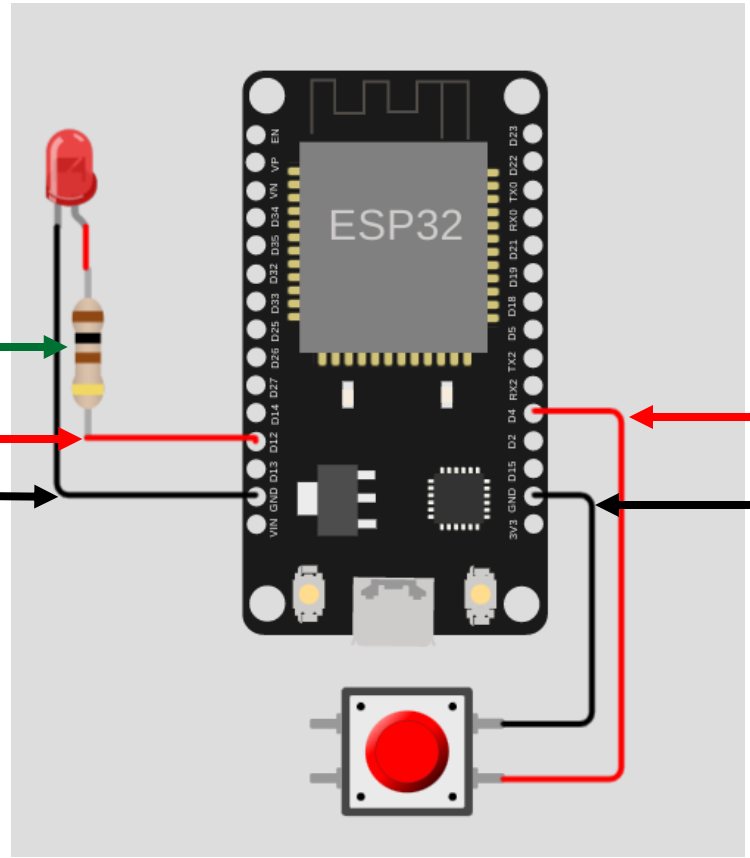
sketch.ino | diagram.json | Library Manager

```
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 chage 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

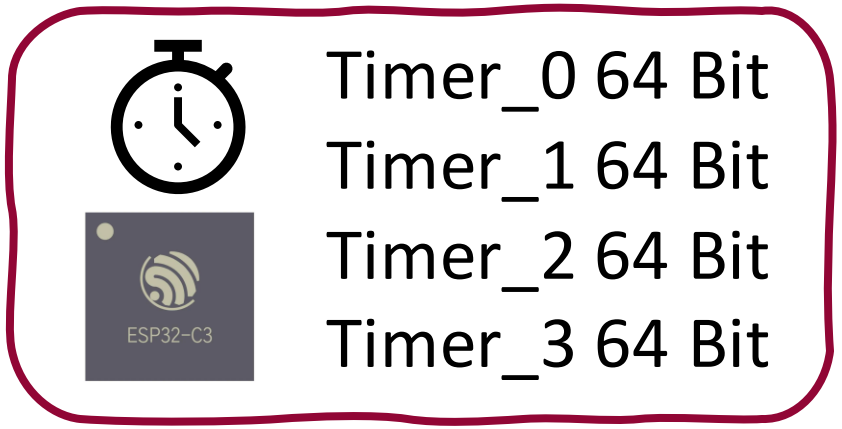
Code

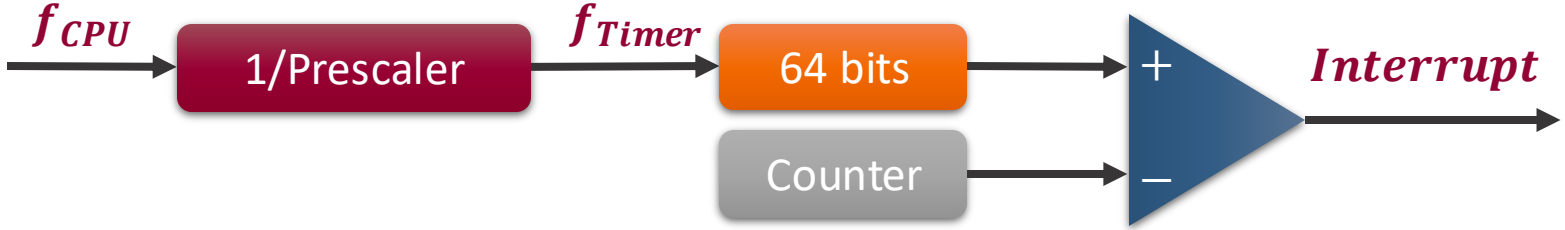


Design

ANTLAB

Timers





I. ESP32 Comes with 4 64 bits Timers
II. ESP32 Clock Runs at 80MHz
III. Prescaler value is divided to form a "tick" of the timer
IV. Each “tick” correspond to an increment of its counter
V. When a specific numbers of ticks are reached an ISR is triggered

https://docs.espressif.com/projects/arduino-esp32/en/latest/api/timer.html

ANTLAB

Timers (2.x Firmware version)

Goal: Turn a LED on and off every 1 second with Timer Interrupt

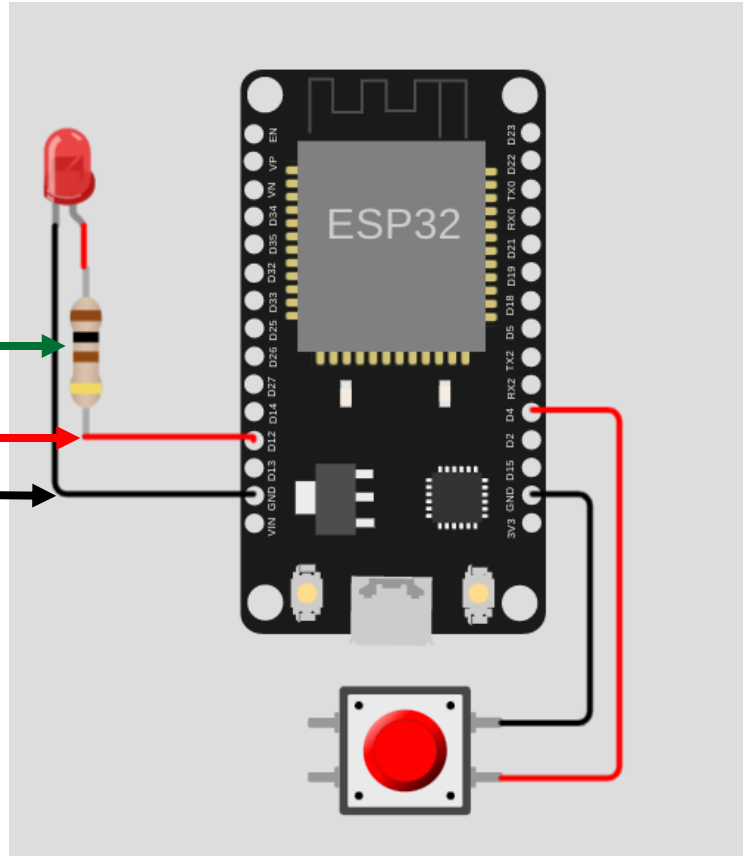
```
#define LED  12  // The digital pin to which a led is connected.
#define C_TIME 1000000 //Time in us
hw_timer_t *My_timer = NULL;

void IRAM_ATTR onTimer(){
  digitalWrite(LED, !digitalRead(LED));
}
void setup() {
  //Pin Setup
  pinMode(LED, OUTPUT);
  Serial.begin(115200);

  My_timer = timerBegin(0, 80, true); //Timer initializer
  //0: hw timer number (ESP32 has 4 hw timers available
  //80: time divider. ESP32 clk 80MHz so we set evry tick to 1 us
  //true: counter shoud increment

  timerAttachInterrupt(My_timer, &onTimer, true);//Attach Interrupt

  timerAlarmWrite(My_timer, C_TIME, true);
  //C_TIME: number of microseconds after which the interrupt should occour
  //true: timer counter will reload after interrupt

  timerAlarmEnable(My_timer); //Just Enable
}
void loop() {
  delay(10);
}
```

Interrupt Function

Attach Interrupt

Code

https://wokwi.com/projects/389523700128678913



Design

ANTLAB

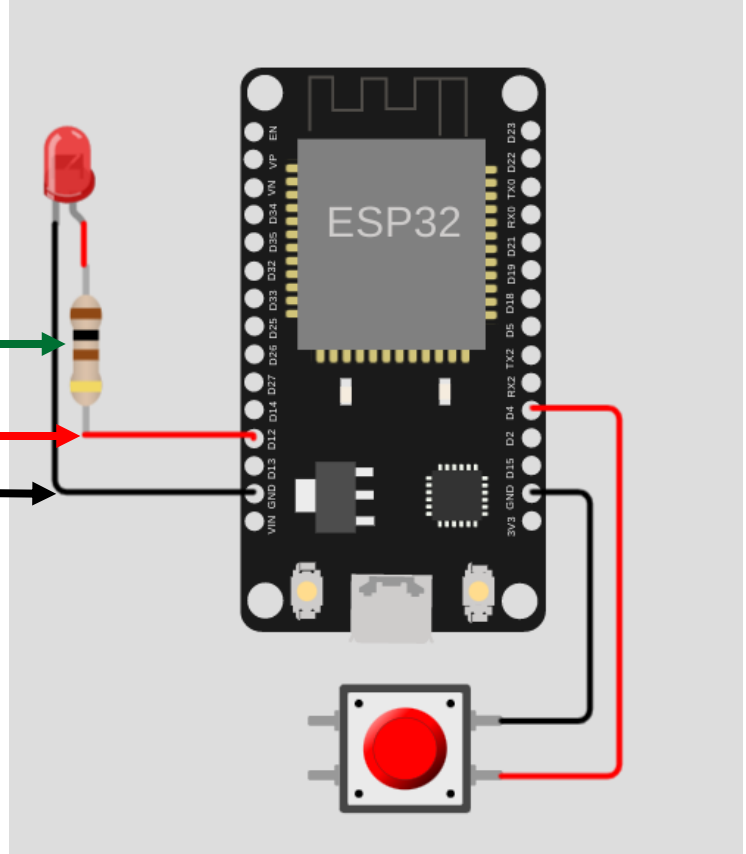
Timers (3.x Firmware version)

Goal: Turn a LED on and off every 1 second with Timer Interrupt

```
1  #define LED 15
2  #define C_TIME 1000000  // 1 second
3
4  void IRAM_ATTR trigger_led() {
5    digitalWrite(LED, !digitalRead(LED));
6  }
7
8  hw_timer_t* My_timer = NULL;
9
10 void setup() {
11   pinMode(LED, OUTPUT);
12
13   My_timer = timerBegin(1000000);  // Set timer frequency to 1 MHz
14
15   timerAttachInterrupt(My_timer, &trigger_led);
16
17   timerAlarm(My_timer, C_TIME, true, 0);
18   // Set alarm to trigger every C_TIME microseconds
19   //hw_timer_t * timer, alarm_value, autoreload, reload_count);
20 }
21
22 void loop() {
23   delay(100);
24 }
```

Interrupt Function

Attach Interrupt

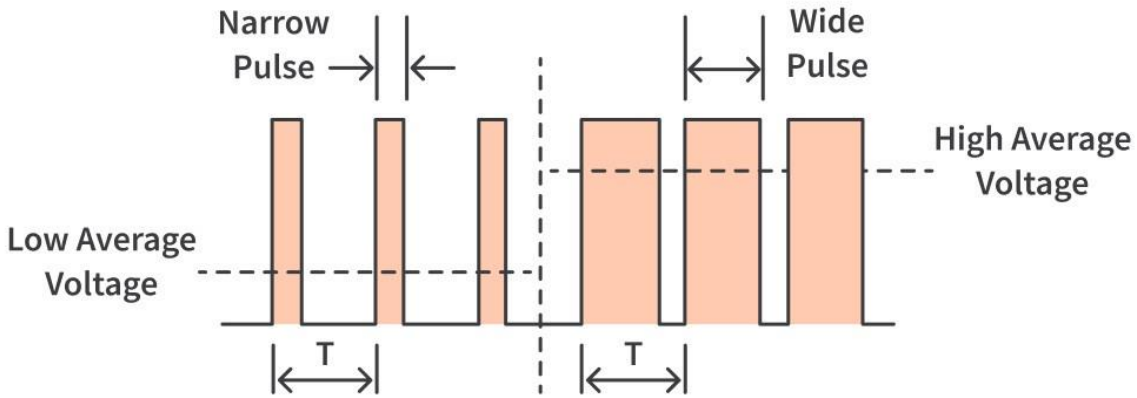


Code
https://wokwi.com/projects/389523700128678913

Design

ANTLAB

Pulse-Width Modulation (PWM)

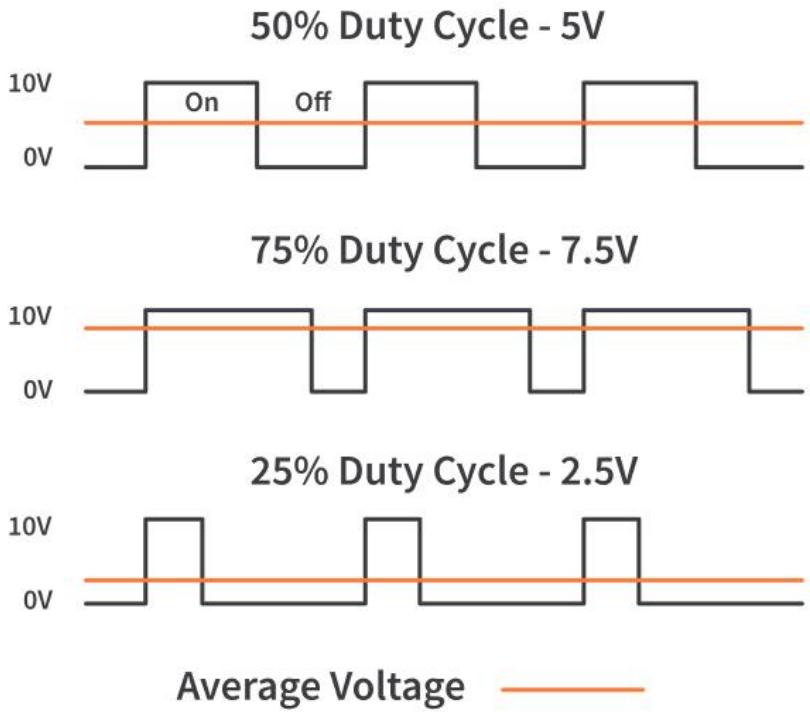


PWM is a technique to control analog devices, using a digital signal, to output an analog-like signal from a digital device

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

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

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



https://lastminuteengineers.com/esp32-pwm-tutorial/

PWM Implementation (1) - OPTIONAL

Goal: Control LED brightness with PWM and a slider

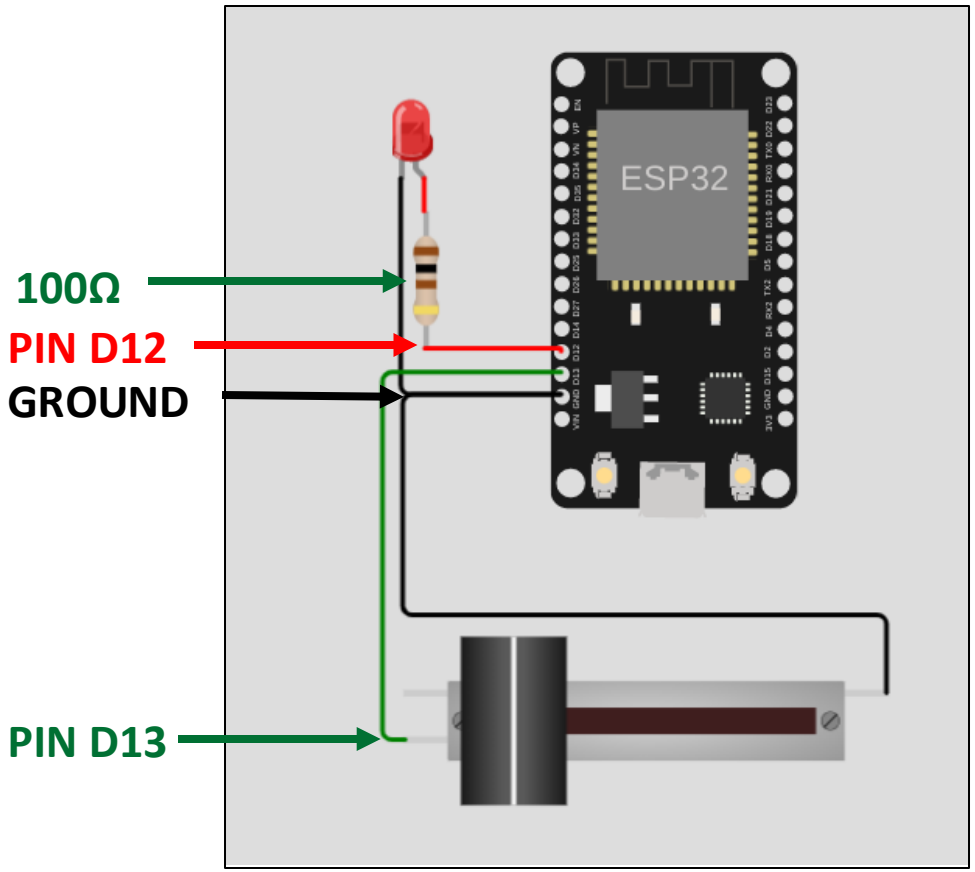
sketch.ino | diagram.json | Library Manager

```
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(){
9     bool led_status = digitalRead(LED);
10
11    if (led_status && pwm_value!=C_TIME){
12      digitalWrite(LED, !led_status);
13      timerWrite(My_timer, pwm_value);
14
15    }else if (!led_status && pwm_value!=0){
16      digitalWrite(LED, !led_status);
17      timerWrite(My_timer, C_TIME-pwm_value);
18    }
19  }
20
```

Interrupt Function

Toggle LED

Set Timer Value



Wokwi project: https://wokwi.com/projects/355361225420427265

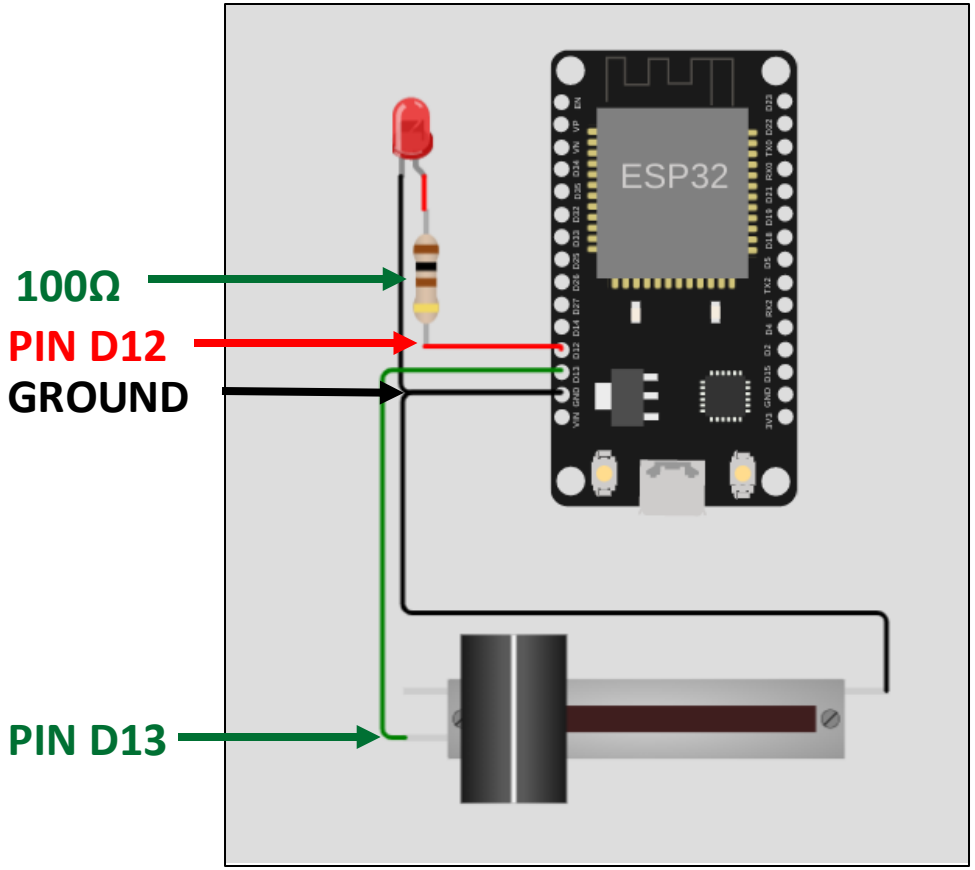
Usage of Integrated PWM here: https://lastminuteengineers.com/esp32-pwm-tutorial/

PWM Implementation (2) - OPTIONAL

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  }
```



Code

Design

ANTLAB

Sensors - Analog and Digital Read

GPIOs 12-39 Safe to use for Digital Read

2 12-bit SAR ADCs with 10 channels

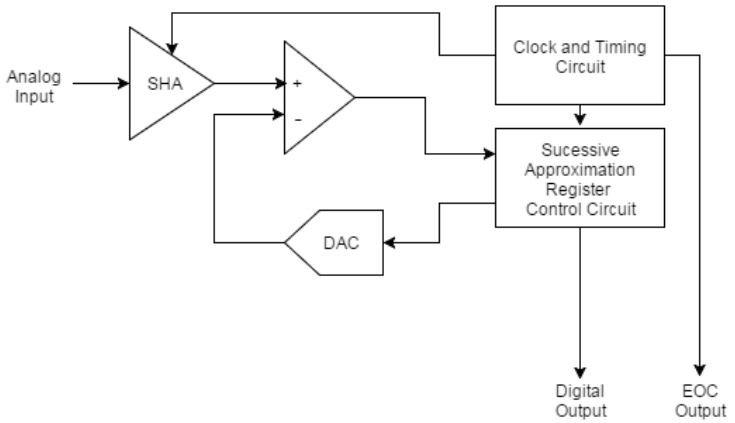
12bit ADC == 4096 (2^{12}) discrete analog levels on 20 pins

- GPIOs Pins in ESP32 can be used as Digital Input
- Used to communicate with sensors which include in their hardware a System on Chip (SoC)
- DHT22 sensor implement a custom protocol where data is transmitted on a single SDA line
- MPU6050 Acc & Gyro implements the Inter Integrated Circuit (I2C) protocol with a clock line (SCL) and a transmission line (SDA) to perform the communication between the ESP32 and the sensor SoC

I2C Guide: https://www.ti.com/lit/an/sbaa565/sbaa565.pdf

Sensors - Analog and Digital Read

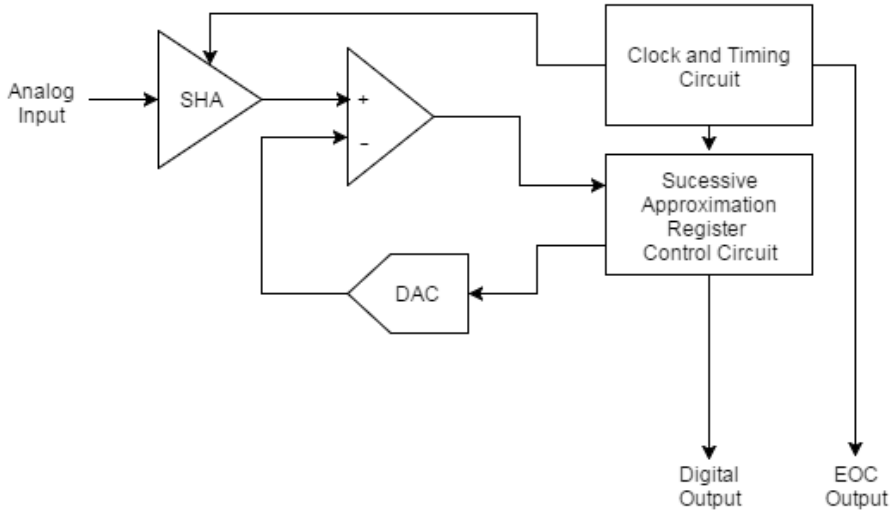
ESP32 ADC Successive Approximation Register (SAR):



- $f_{SAMPLING} = \frac{f_{Clock}}{ST+12}$ ST = sampling time (ADC clock cycles to charge the SHA capacitor + 12 clock cycles to convert to 12 bits accuracy)
- Up to 2MHz with Wi-Fi Disabled &Direct Memory Access
- One-shot ADC conversion result, or continuous ADC conversion results
- RTC Mode controlled by the Real Time Counter module and is suitable for low-frequency sampling operations.
- DMA mode: ADC-DMA (Direct Memory Access) efficient and CPU-independent transfer of ADC results to memory (for high-speed data acquisition)

ANTLAB

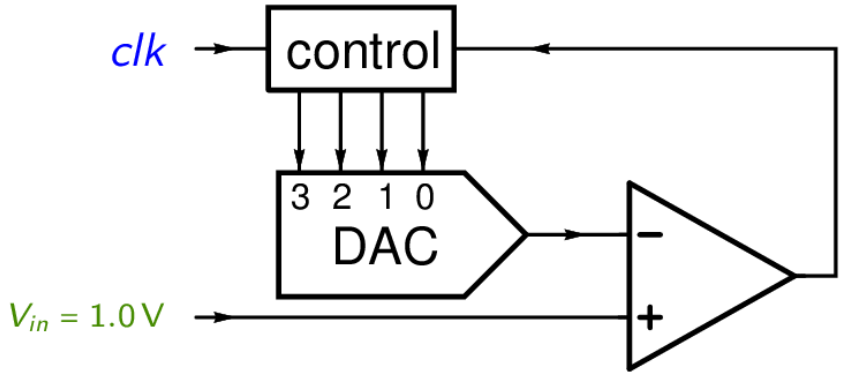
Sensors – How ADC SAR works (OPTIONAL)

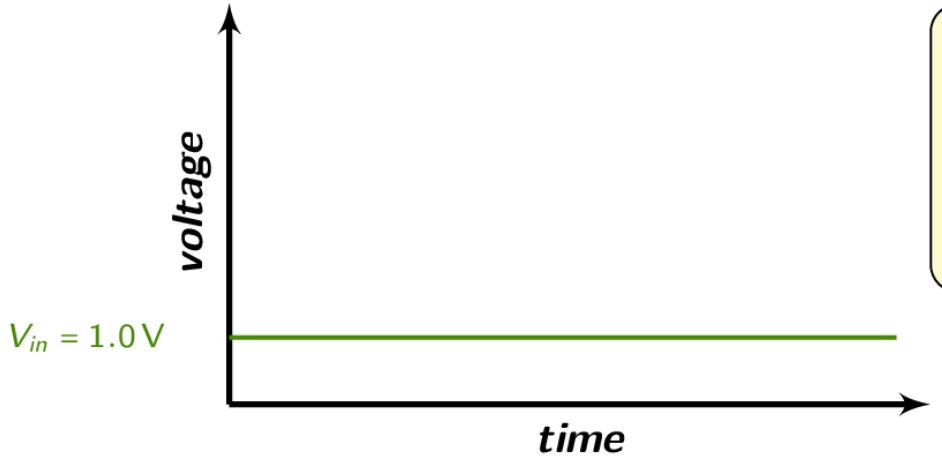


1. Analog signal is sampled and held.
2. For each bit, the SAR logic outputs a binary code to the DAC to determine the state of the current bit
3. Once all bits have been approximated, the digital approximation is output (EOC)

ANTLAB

Sensors – How 4bit SAR works (OPTIONAL)





Resolution:	
5 V × 1/2	2.5000 V
5 V × 1/4	1.2500 V
5 V × 1/8	0.6250 V
5 V × 1/16	0.3125 V
...	
5 V × 1/1024	0.0049 V

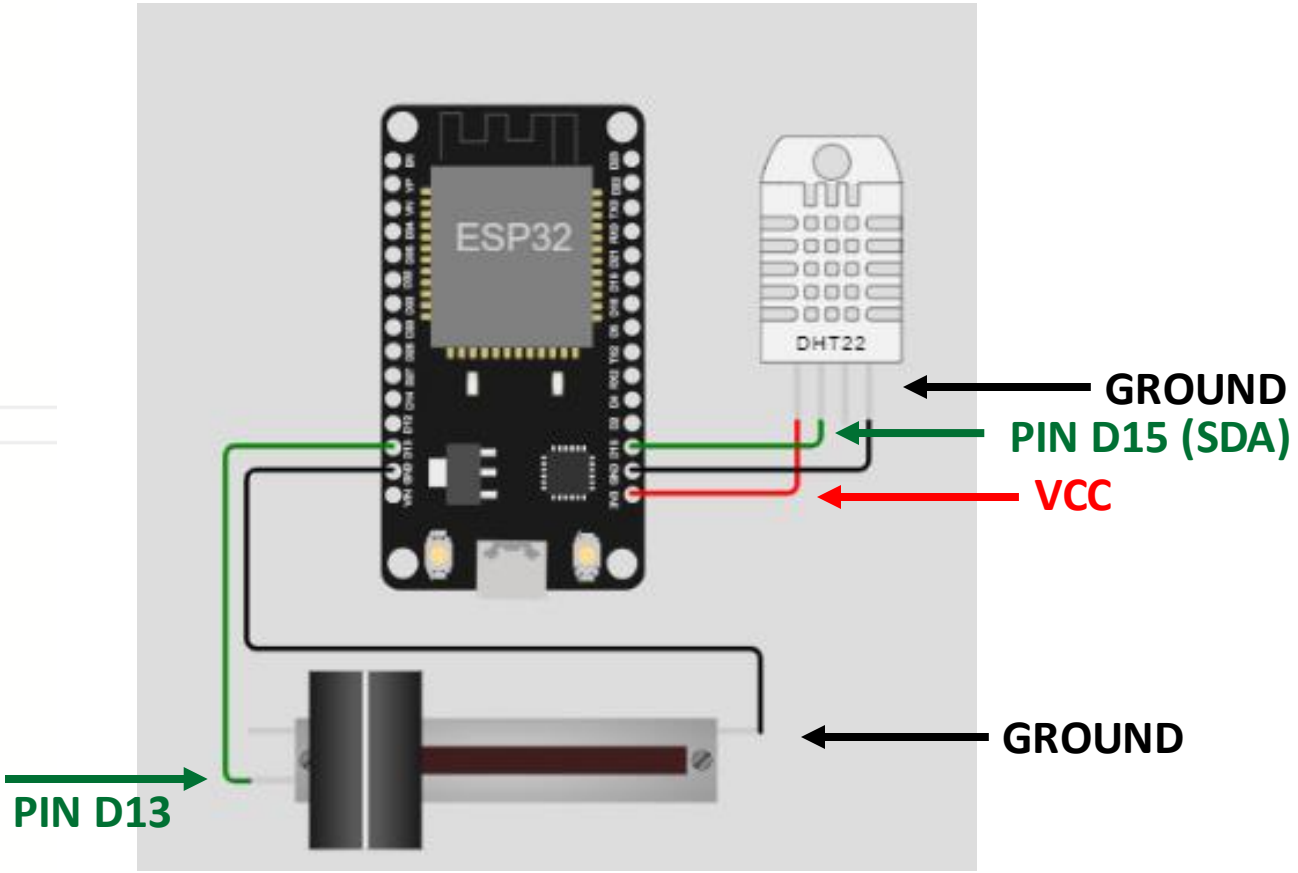
Ref: https://en.m.wikipedia.org/wiki/File:4-bit_Successive_Approximation_DAC.gif

ANTLAB

Digital and Analog Read

Goal: Read the values of a Slide and a DHT22 Temperature and Humidity Sensor

```
9   #include "DHTesp.h" //https://github.com/beegee-tokyo/DHTesp/tree/master
10
11  #define LED  12  // LED Digital pin
12  #define SLIDE 13 // Potentiometer Digital pin
13  #define DHT_PIN 15
14
15  DHTesp dhtSensor;
16
17  void setup() {
18  pinMode(SLIDE, INPUT);
19  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
20  Serial.begin(115200);
21  }
22
23  void loop() {
24    float temperature = dhtSensor.getTemperature();
25    float humidity = dhtSensor.getHumidity();
26    int slide = analogRead(SLIDE);
27    int mapped_value = map(slide, 0, 4095, 0, 100);
28
29    Serial.print(slide);
30    Serial.println("Temp: " + String(temperature, 2) + "°C");
31    Serial.println("Humidity: " + String(humidity, 1) + "%");
32    Serial.println("----");
33    delay(1000);
```



Code

https://wokwi.com/projects/389525221888344065

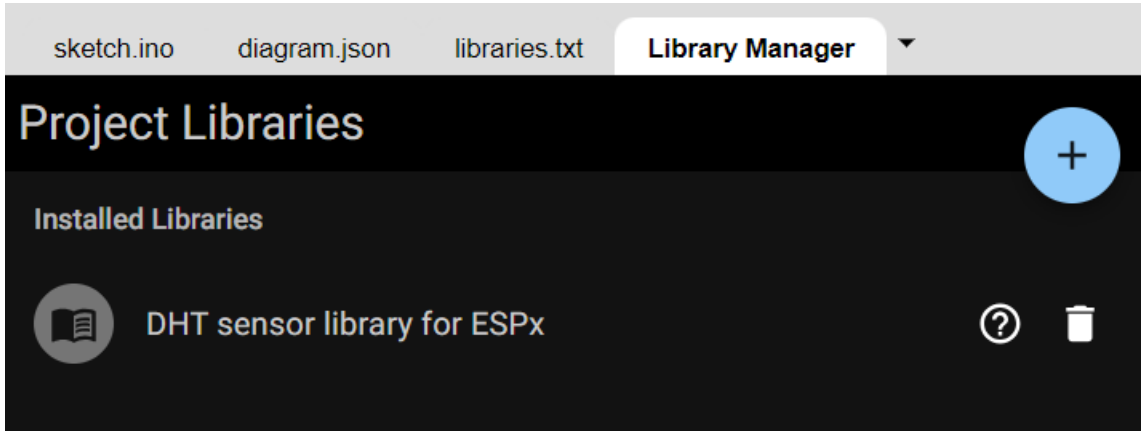
Design

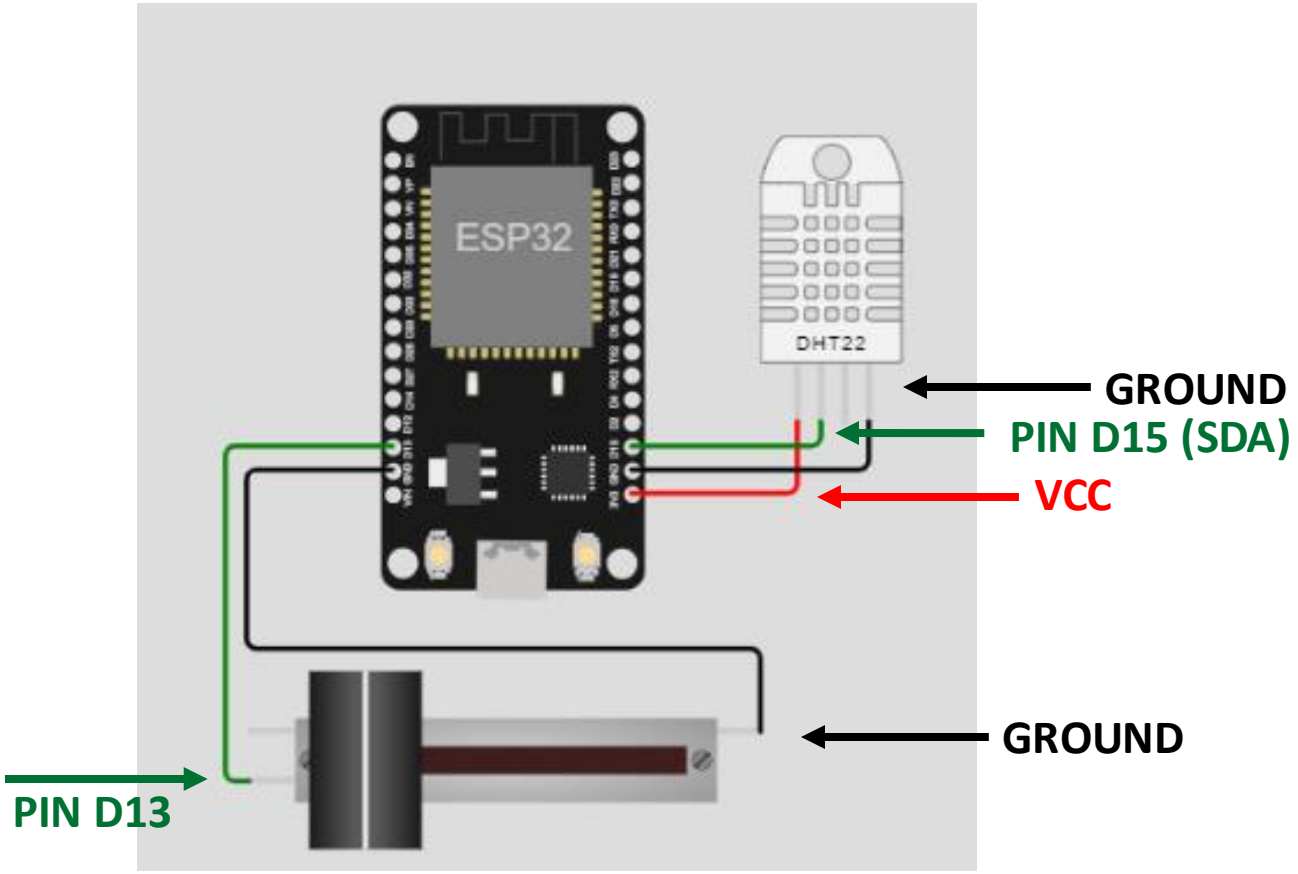
ANTLAB

Digital and Analog Read

Goal: Read the values of a Slide and a DHT22 Temperature and Humidity Sensor

Adding a Library !



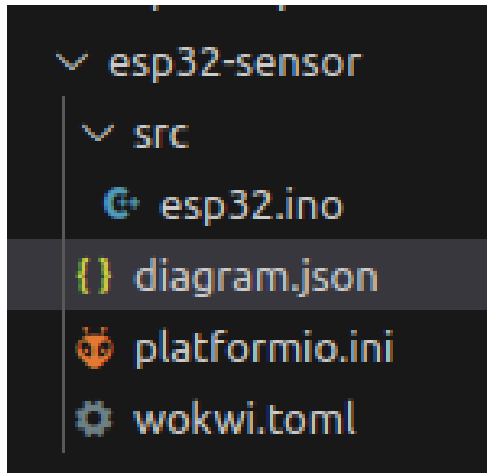


Design

ANTLAB

Working with VSCode

Install the VM (suggested) and open VScode, or follow this guide (pro users):
https://github.com/antonio-boiano/Wokwi



ESP32 Code

Wokwi UI, and execute emulation. (Open With Text editor to edit the file)

PlatformUI compiler configuration – here you can set-up the libraries

Wokwi configuration – i.e. setup of serial monitor, firmware file path

Project structure

Open VSCode in the VM to start to play with Wokwi. Open the diagram.json of any project you like and press play

ANTLAB

Building with VSCode

If you want to create a new project:

1. Copy the esp32-empty folder and rename it (i.e. esp32-test)
2. Edit the esp32.ino in the src folder and the diagram.json as you like
   - Diagram.json can be edited only through code. If you wish to use the visual editor copy and paste the json configuration from the WebApp
3. Compile the folder:
   - Open a terminal in the main folder Wokwi
   - Run: bash compile.sh
   - When asked, put the name of the folder you wish to compile (i.e. esp32-test)
4. Wait for the compile to end and run the simulation through UI by opening the diagram.json

Building with VSCode

If you want to create a new project:

5. If you need a specific library refer to the following link to find them, and edit the platform.ini lib_deps field accordingly (see esp32-sensor project)

```
[env:esp32]
platform = espressif32
framework = arduino
board = esp32dev
lib_deps = beegee-tokyo/DHT sensor library for ESPx@^1.19
```

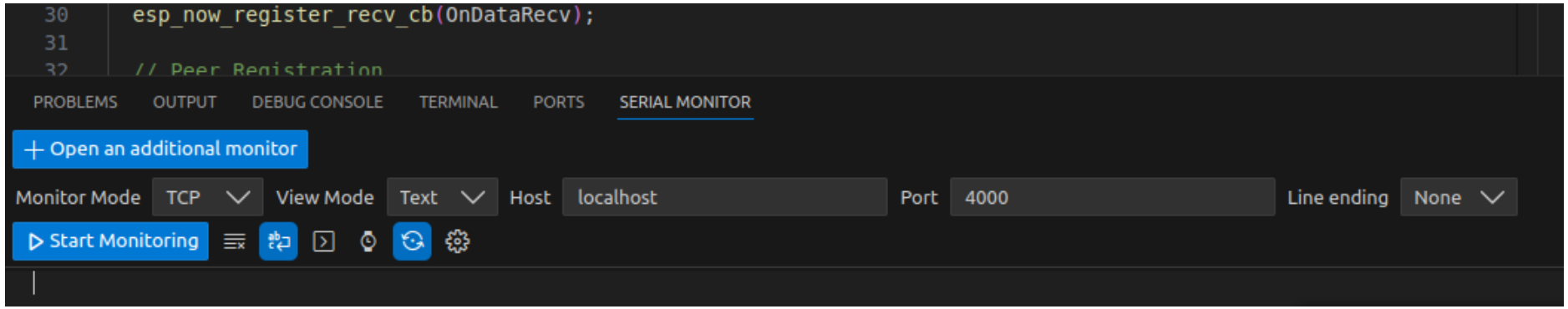
Building with VSCode

If you want to create a new project:

6. To interact with the serial add to the wokwi.toml rfc2217ServerPort = 4000 (see esp32-esp_now) as an example.

```
[wokwi]
version = 1
elf = ".pio/build/esp32/firmware.elf"
firmware = ".pio/build/esp32/firmware.bin"
rfc2217ServerPort = 4000
```

7. Open the serial monitor configured as:



ANTLAB

POLITECNICO
MILANO 1863

Playing with WiFi

Make Two ESP-32 talk (ESP-NOW)

"ESP-NOW is a wireless communication protocol defined by Espressif, which enables the direct, quick and low-power control of smart devices, without the need of a router. ESP-NOW can work with Wi-Fi and Bluetooth LE"

- 10 Encrypted peers and 20 Unencrypted peers supported
- 250-byte payload length
- Low Power Consumption
- Support for one to many, many to one and bi-directional communication

ANTLAB

Make Two ESP-32 talk (ESP-NOW)

Goal: Create a messaging app with two ESP32 communicating among each other

Make Two ESP-32 talk (ESP-NOW)

Goal: Create a messaging app with two ESP32 communicating among each other

Get Board MAC Address

```
#include <esp_now.h>
#include <WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  Serial.print("ESP Board MAC Address:  ");
  Serial.println(WiFi.macAddress());
}

void loop() {
  delay(1000);
}
```

Make Two ESP-32 talk (ESP-NOW)

Goal: Create a messaging app with two ESP32 communicating among each other

Get Board MAC Address

```
#include <esp_now.h>
#include <WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  Serial.print("ESP Board MAC Address:  ");
  Serial.println(WiFi.macAddress());
}

void loop() {
  delay(1000);
}
```

To make the ESP-NOW protocol work with the Wokwi emulator, a Broadcast address (8C:AA:B5:84:FB:90) is used and the Transmission and Reception of messages happens on the Same Board

ANTLAB

Make Two ESP-32 talk (ESP-NOW)

Goal: Create a messaging app with two ESP32 communicating among each other

Send and Receive Callback

```
#include <WiFi.h>
#include <esp_now.h>

// MAC receiver
uint8_t broadcastAddress[] = {0x8C, 0xAA, 0xB5, 0x84, 0xFB, 0x90};

esp_now_peer_info_t peerInfo;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("Send Status: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Ok" : "Error");
}

//Receiving Callback
void OnDataRecv(const uint8_t * mac, const uint8_t *data, int len) {
  Serial.print("Message received: ");
  char receivedString[len];
  memcpy(receivedString, data, len);
  Serial.println(String(receivedString));
}
```

Use the Broadcast Address

Setup and Loop

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  esp_now_init();
  //send callback
  esp_now_register_send_cb(OnDataSent);
  //receive callback
  esp_now_register_recv_cb(OnDataRecv);
  // Peer Registration
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  esp_now_add_peer(&peerInfo);
}

void loop() {
  while (!Serial.available()); // Wait for input
  String message = Serial.readStringUntil('\n');
  esp_now_send(broadcastAddress, (uint8_t*)message.c_str(), message.length() + 1);
}
```

Wait User Input and send the message

https://wokwi.com/projects/389617682197482497

ANTLAB

More on Wokwi…

- ESP-NOW Library API: https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/espnow.html/
- Use featured project as guidelines
- Check for trending projects for nice ideas

Discord group

Facebook group