Creando una IoT

Un tutorial con Thingspeak, Arduino y ESP32

Taller de Iniciación a los Sistemas Ciber Físicos
MINA - Facultad de Ingeniería - Udelar

Temario

Una Nube: thingspeak

Un microcontrolador: ESP32

Un firmware: Arduino

Programemos nuestro microcontrolador

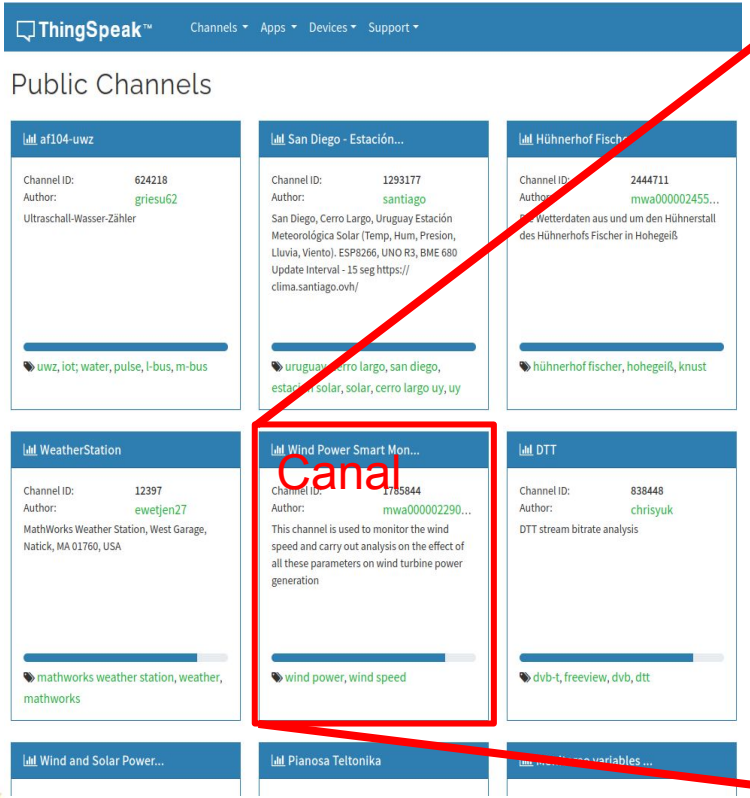
Conectémonos a la nube

Enviemos datos

Exploremos nuestros datos en la nube

Qué es una nube IoT

thingspeak.com



Qué es un microcontrolador (MCU)

Microcontrolador ESP32-PICO-KIT-1

Velocidad de reloj: 80MHz / 160MHz

RAM: 520KB

Flash: 4MB

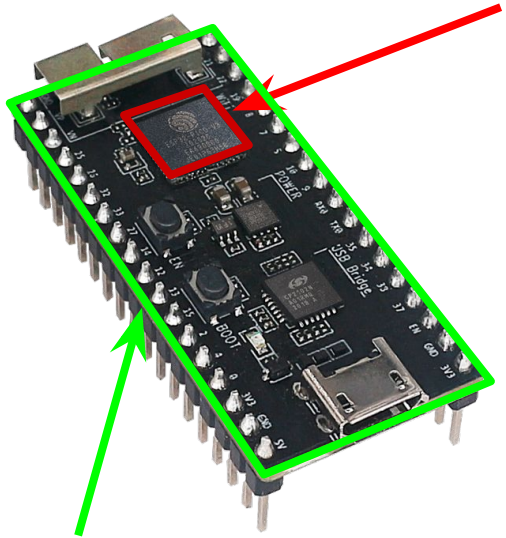
Alimentación: USB (5V)

Voltaje de funcionamiento: 3.3V

WiFi integrada (802.11b/g/n),
Bluetooth (v4, BLE)

Precio: 4-5usd / 10-15usd

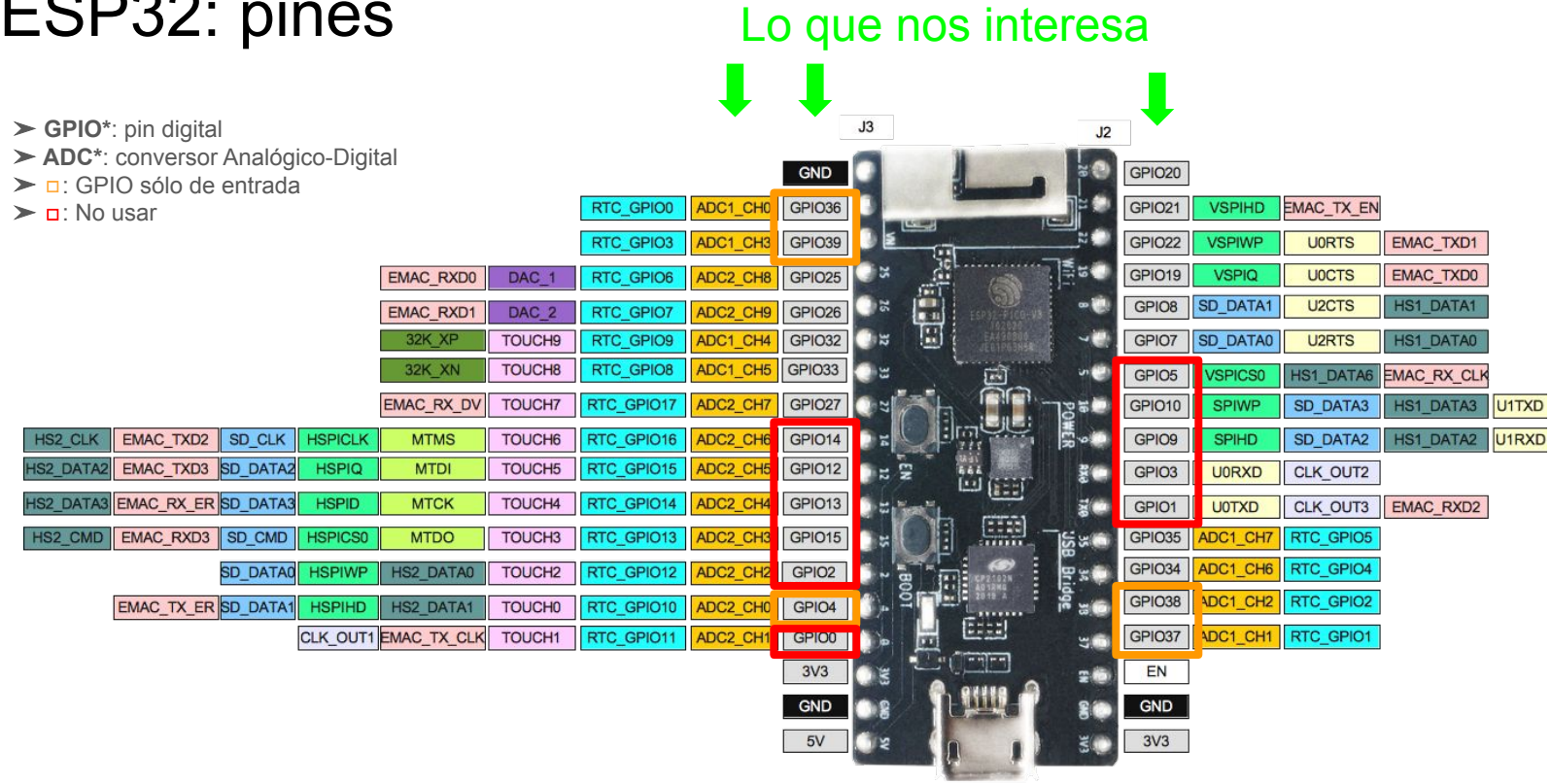
ESP32-PICO-V3



ESP32-PICO-KIT-1

ESP32: pines

- **GPIO***: pin digital
- **ADC***: conversor Analógico-Digital
- □: GPIO sólo de entrada
- □: No usar

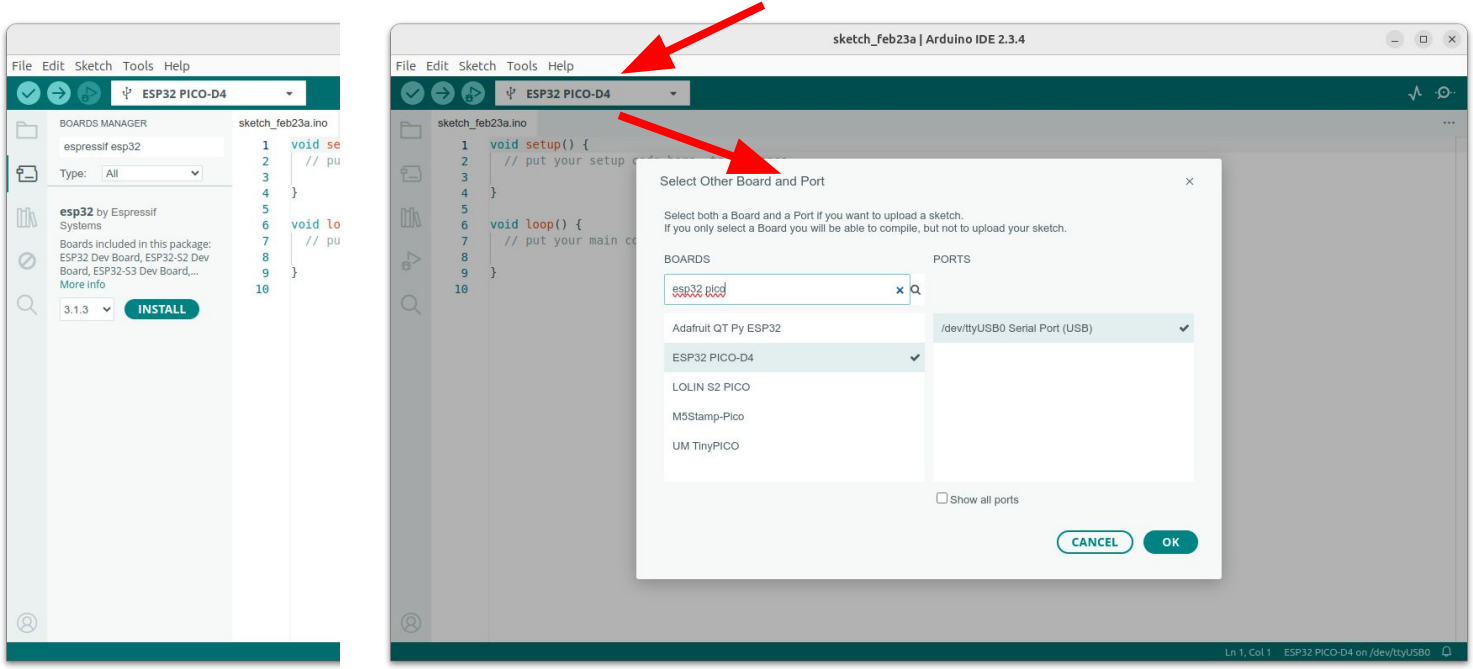


https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/hw-reference/esp32/get-started-pico-kit-1.html

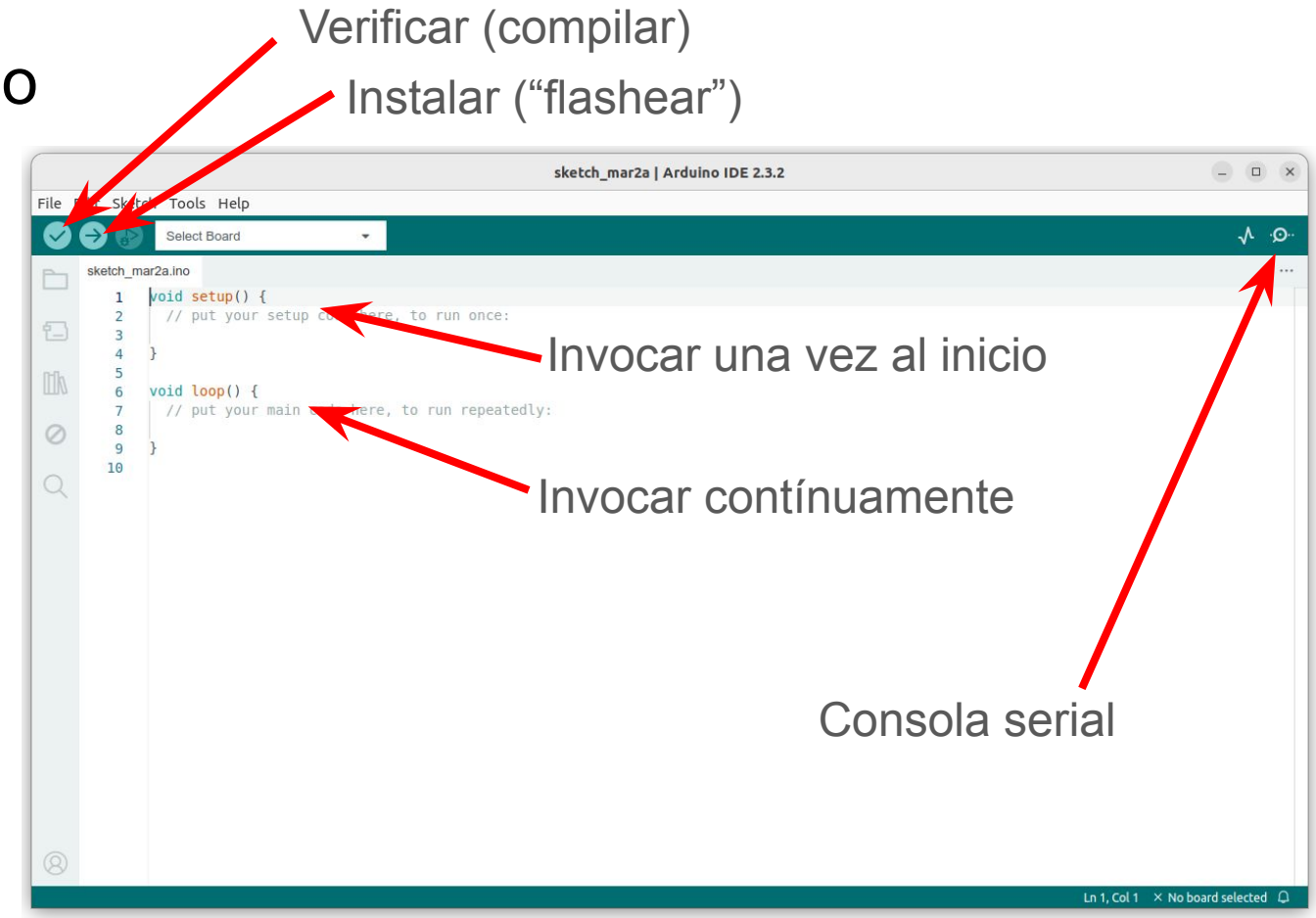
Cómo se programa un MCU

Arduino: agregar soporte para nuestro MCU

Seleccionar **ESP32 PICO-D4**



Arduino


Verificar (compilar)
Instalar (“flashear”)
sketch_mar2a | Arduino IDE 2.3.2
File
Tools
Help
Select Board
sketch_mar2a.ino
1 void setup() {
2 // put your setup
to run once:
3
4 }
5
6 void loop() {
7 // put your main
here, to run repeatedly:
8
9 }
10
Invocar una vez al inicio
Invocar contínuamente
Consola serial
Ln 1, Col 1
No board selected

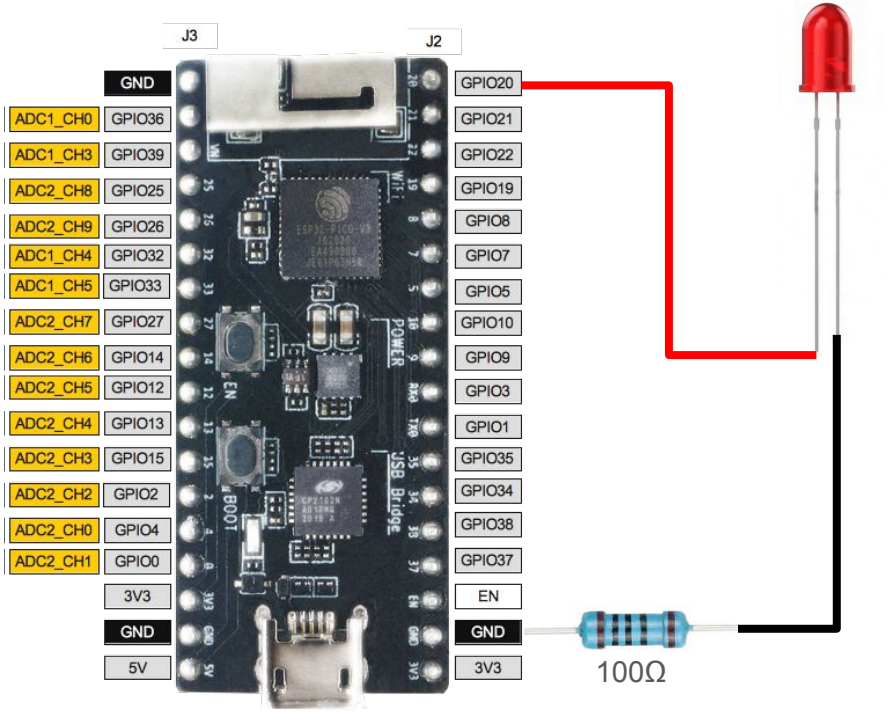
Arduino: estructura de un programa

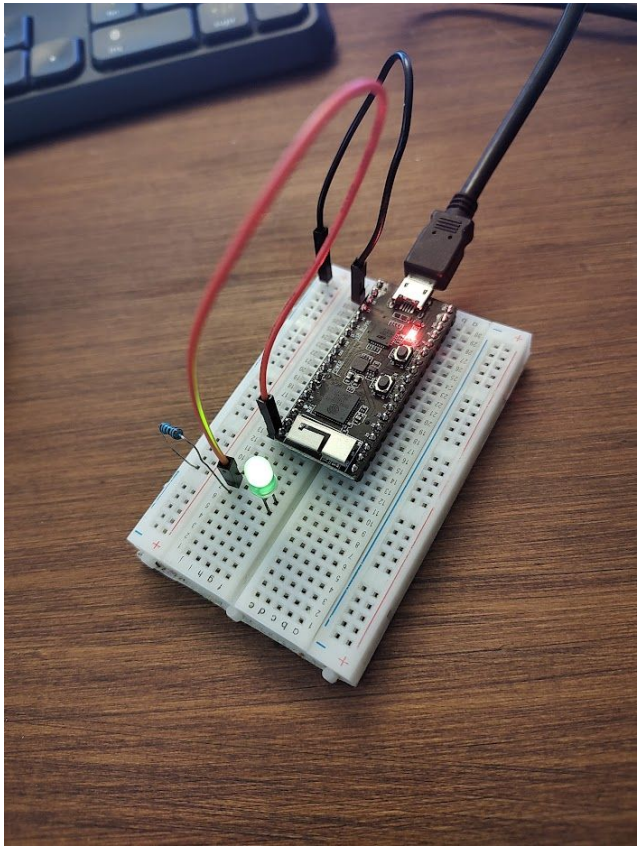
```
void setup() {
  Serial.begin(115200);
  Serial.println("Hello World!");
}

void loop() {
}
```

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

void loop() {
  Serial.println("Hello World!");
}
```

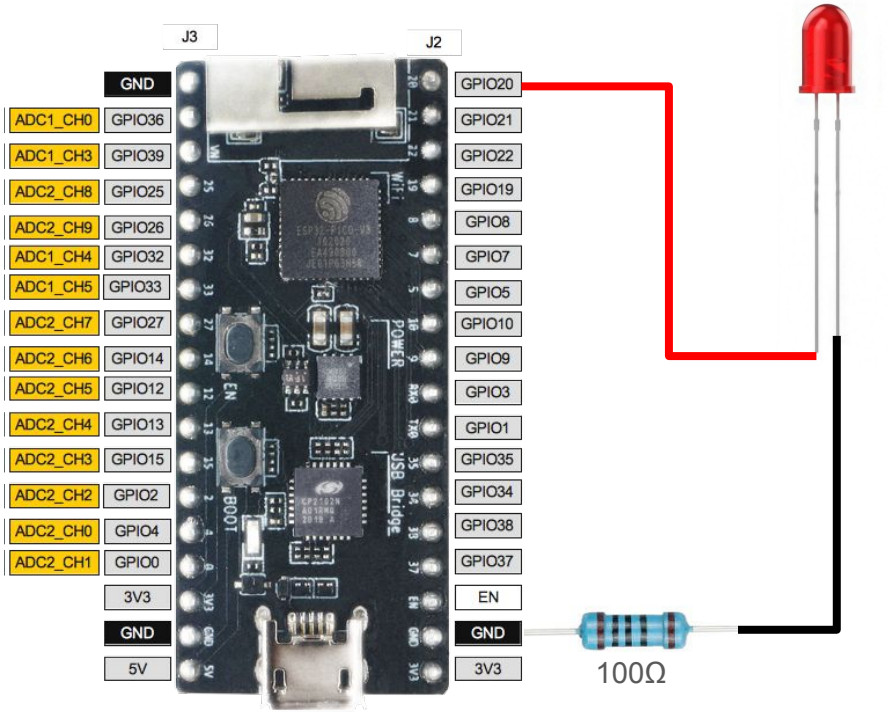
Hola mundo en MCU

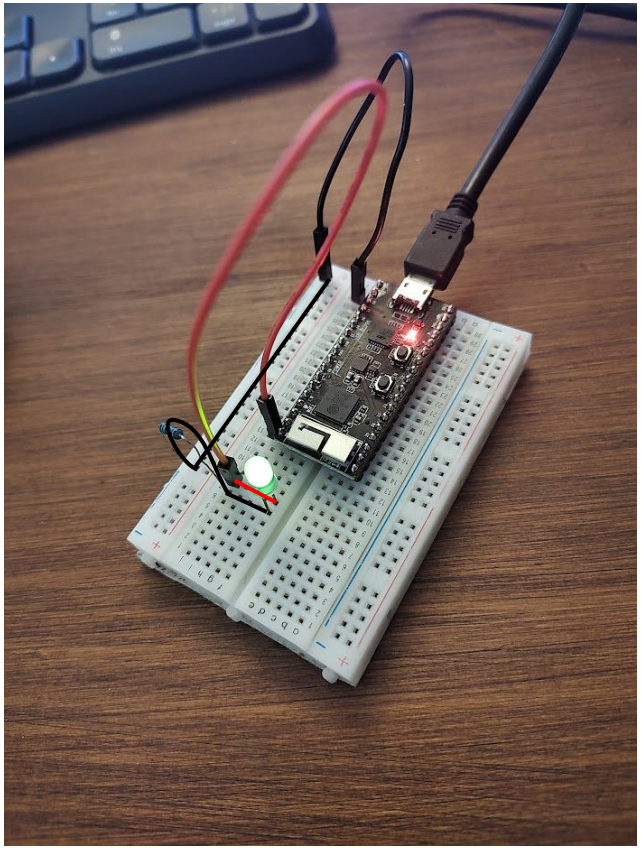
Arduino: escribir en pin digital





Arduino: escribir en pin digital



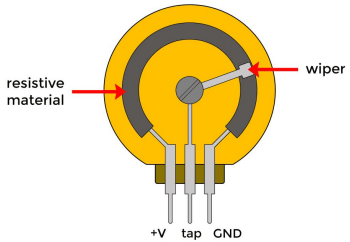


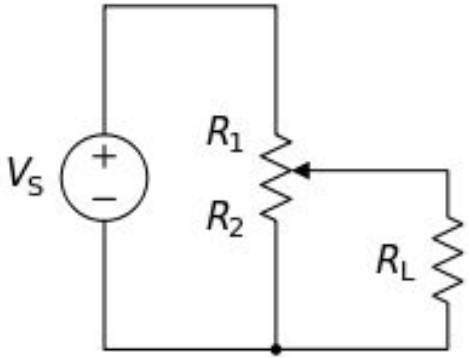
Arduino: escribir en pin digital

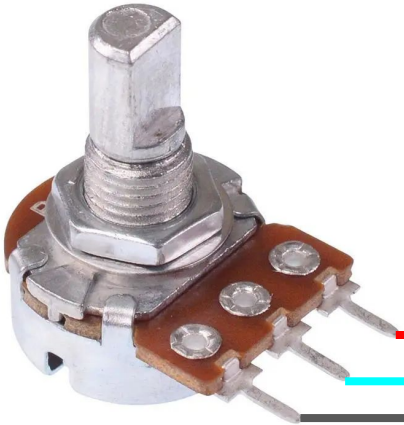
```
#define LED_PIN 20

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);  // LED pin as output.
  Serial.println("\n\nLED pin: " + String(LED_PIN));
}

void loop() {
  Serial.print("1 ");
  digitalWrite(LED_PIN, HIGH);
  delay(1000);
  Serial.print("0 ");
  digitalWrite(LED_PIN, LOW);
  delay(1000);
}
```

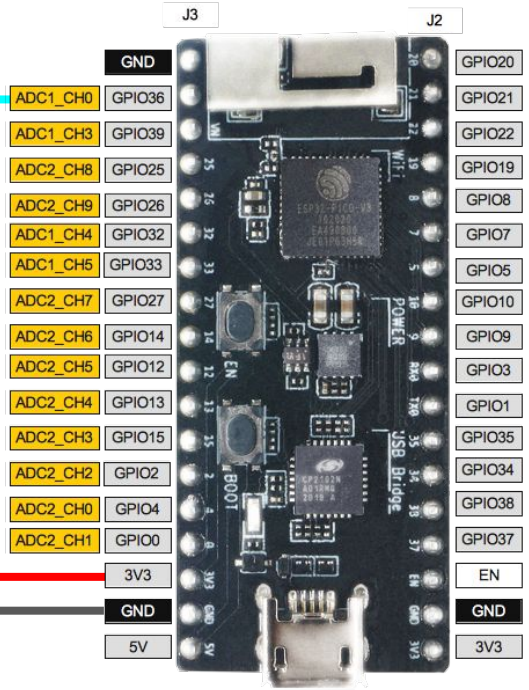
Cómo se lee un sensor analógico

Arduino: cablear una resistencia variable

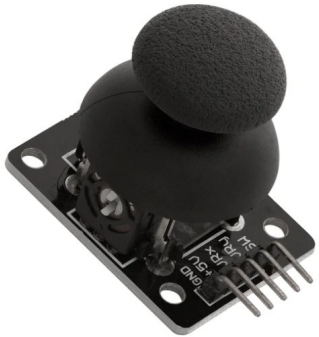


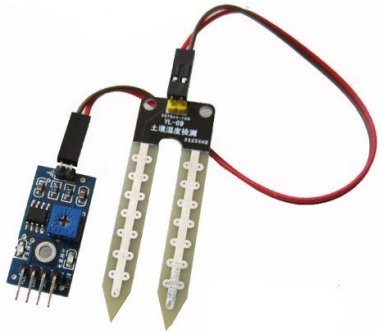


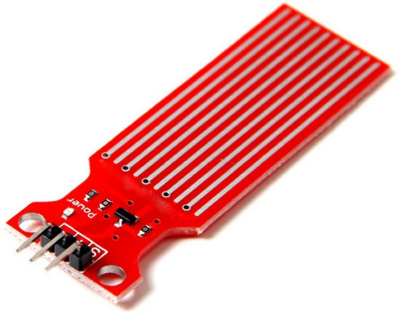




Sensores analógicos





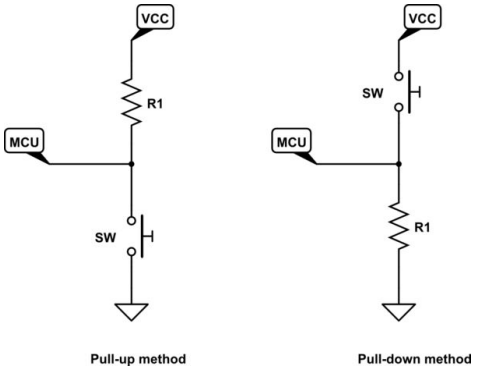


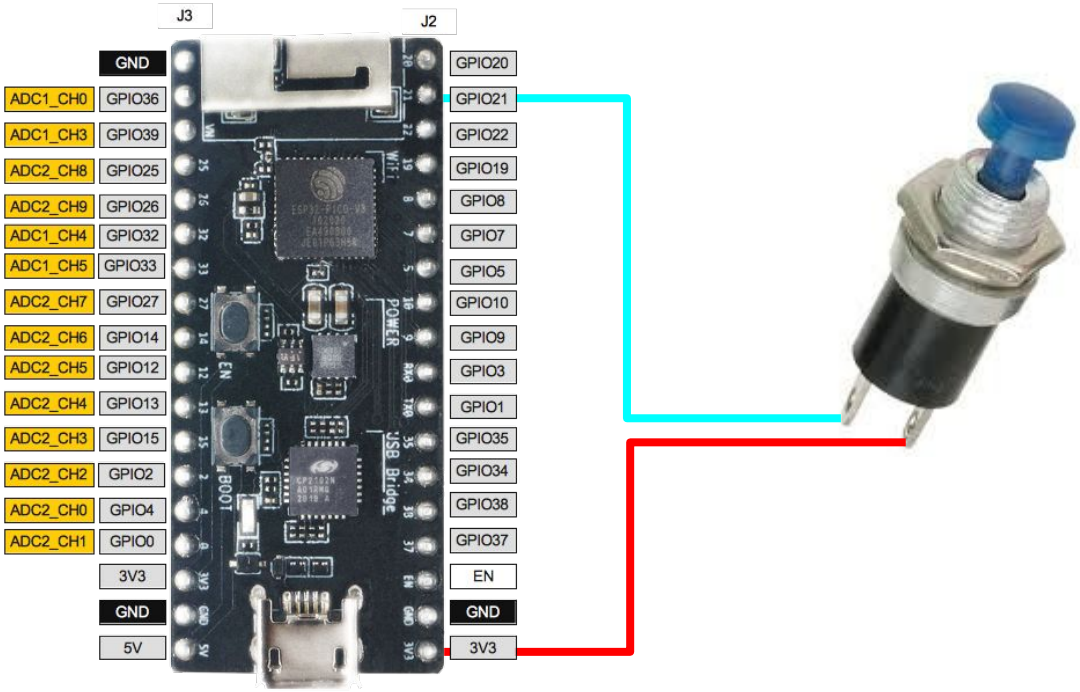
Arduino: conversor analógico-digital (ADC)

```
#define R1_PIN 36
#define LED_PIN 20

int last_change = 0;
int led_state = 0;

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  int now = millis();
  int delay = analogRead(R1_PIN);
  if (now - last_change > delay) {
    last_change = now;
    led_state = 1 - led_state;
    Serial.println(String(delay));
    digitalWrite(LED_PIN, led_state);
  }
}
```

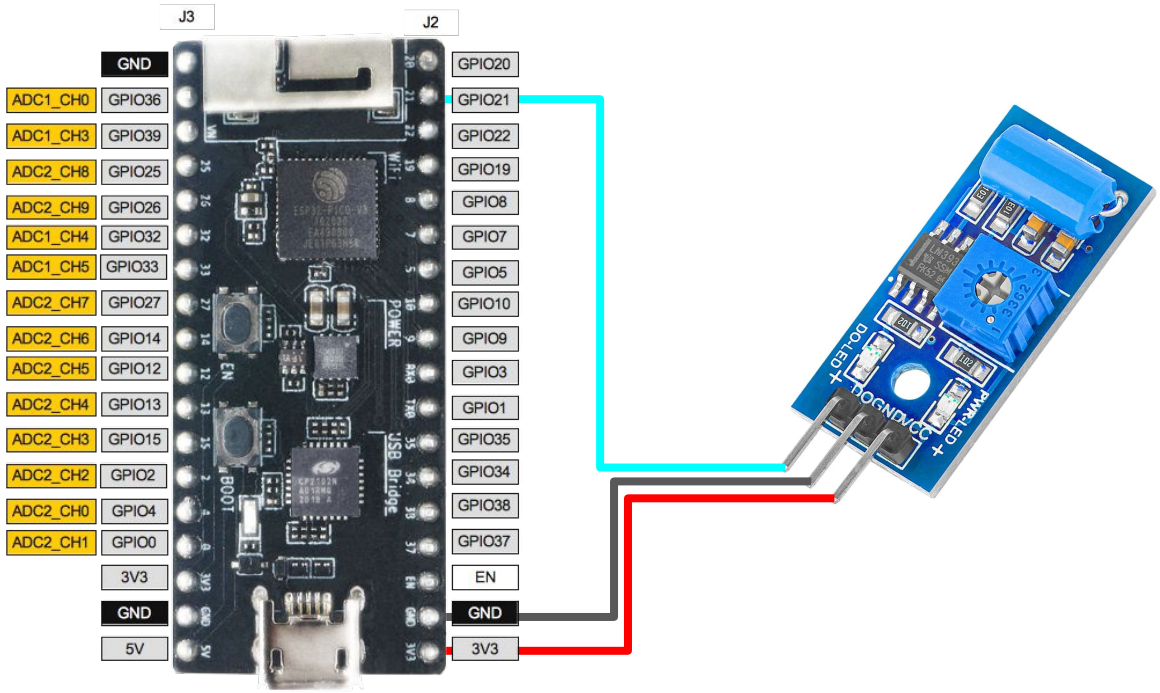
Cómo se lee un sensor digital

Arduino: cablear un sensor digital

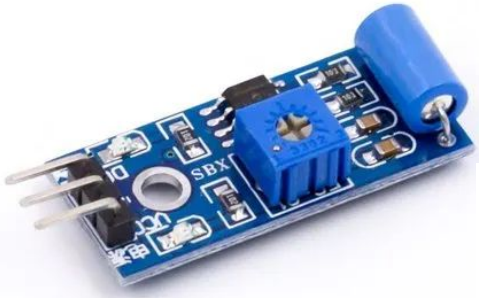




Arduino: cablear un sensor digital



Sensores digitales



Arduino: leer un sensor digital (polling)

```ino
const int buttonPin = 21;

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT_PULLDOWN);
}

void loop() {
  int buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.println("on");
  } else {
    Serial.println("off");
  }
}
```

```ino
const int buttonPin = 21;
int lastState = LOW;

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT_PULLDOWN);
}

void loop() {
  int buttonState = digitalRead(buttonPin);
  if (lastState != buttonState) {
    Serial.println(buttonState);
    lastState = buttonState;
  }
}
```

Arduino: leer un sensor digital (interrupciones)

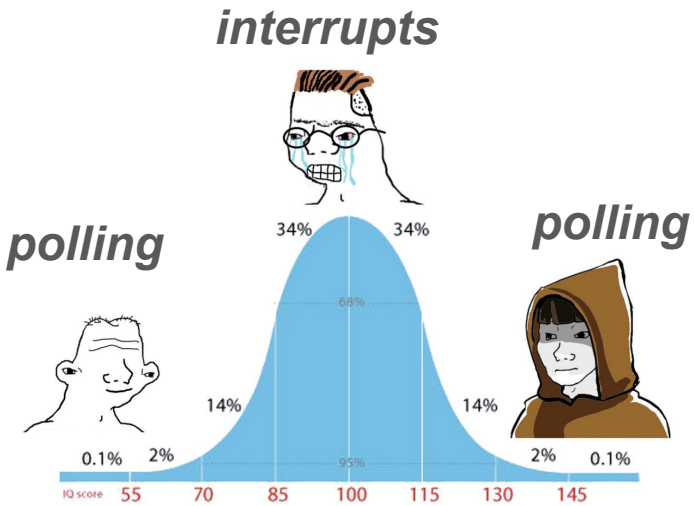
```
const int buttonPin = 21;
volatile byte clicked = 0;

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT_PULLDOWN);
  attachInterrupt(digitalPinToInterrupt(buttonPin), click, RISING);
}
void loop() {
  if (clicked == 1) {
    Serial.println(millis());
    clicked = 0;
  }
}

ICACHE_RAM_ATTR void click() {
  clicked = 1;
}
```

Arduino: leer un sensor digital (interrupciones)

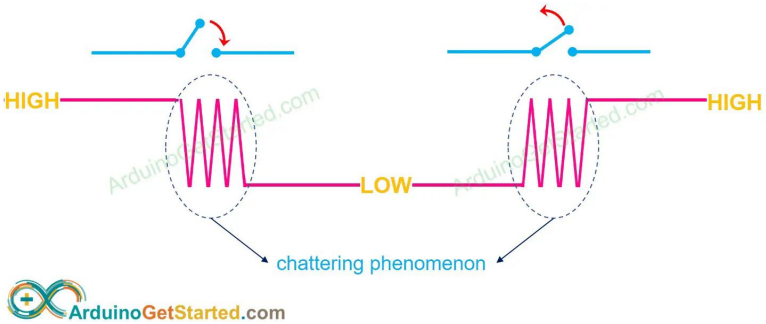
- Interrupciones en `CHANGE` / `FALLING` / `RISING`
- Los handlers de las interrupciones no reciben ni devuelven parámetros
  - Deben comunicarse con el resto del programa mediante variables globales
- Las variables globales que se actualizan en un handler deben ser declaradas `volatile`.
- Los handlers deben ser declarados `ICACHE_RAM_ATTR`.
- Se pueden proteger secciones de código con `noInterrupts()` / `interrupts()`; Usar responsablemente.
- Una interrupción no interrumpe a otra interrupción
  - Durante una interrupción “nada sucede”
    - El tiempo (`millis()`) no avanza
    - El handler debe ser lo más corto y rápido posible
      - No usar `delay()`
      - En lo posible no usar el puerto serial, que incluye la consola

Arduino: leer un sensor digital (interrupciones)

- Polling: fácil, obtuso, tendencia a ineficiente y confuso.
- Interrupciones: elegantes, sutiles, eficientes, peligrosas.



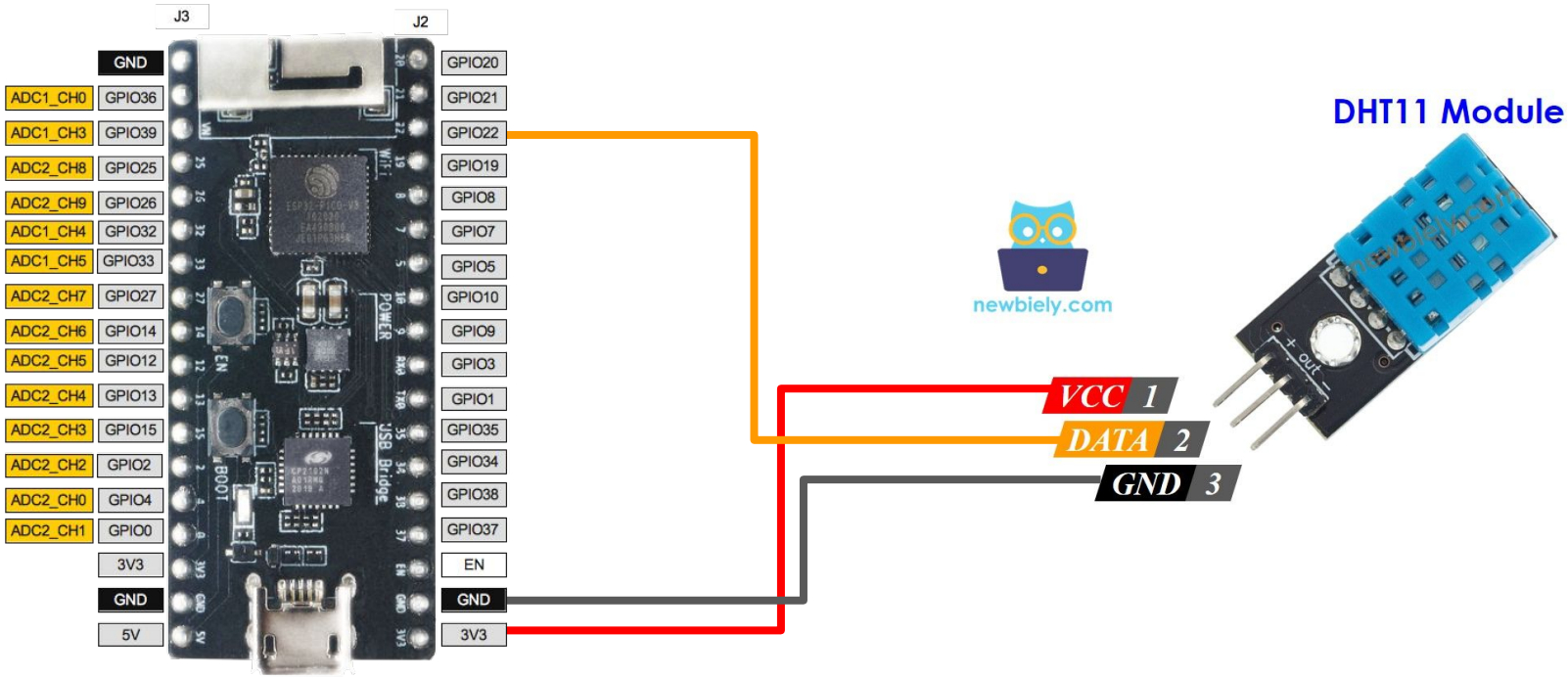
Arduino: leer un sensor digital (*debounce*)



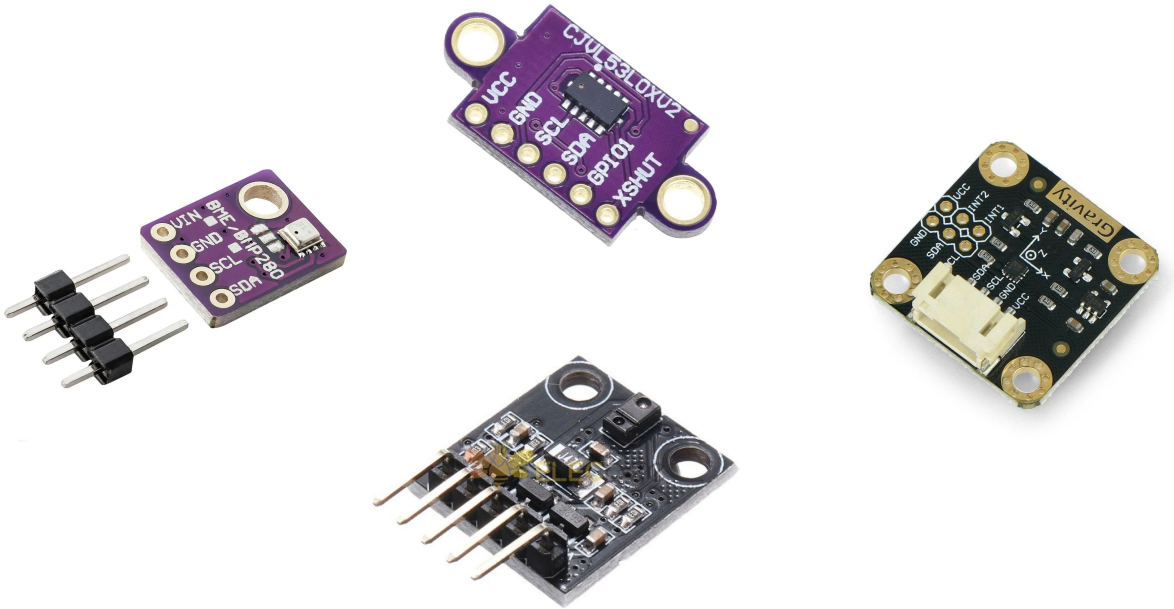

```
#define BOUNCE_TIME 50
static int lastDigitalOn = 0;

...
if( digitalRead(button) && (millis()-lastDigitalOn > BOUNCE_TIME) ) {
  lastDigitalOn = millis();
  // do something
}
...
```

Cómo se lee un sensor basado en un protocolo (ej: 1-wire)

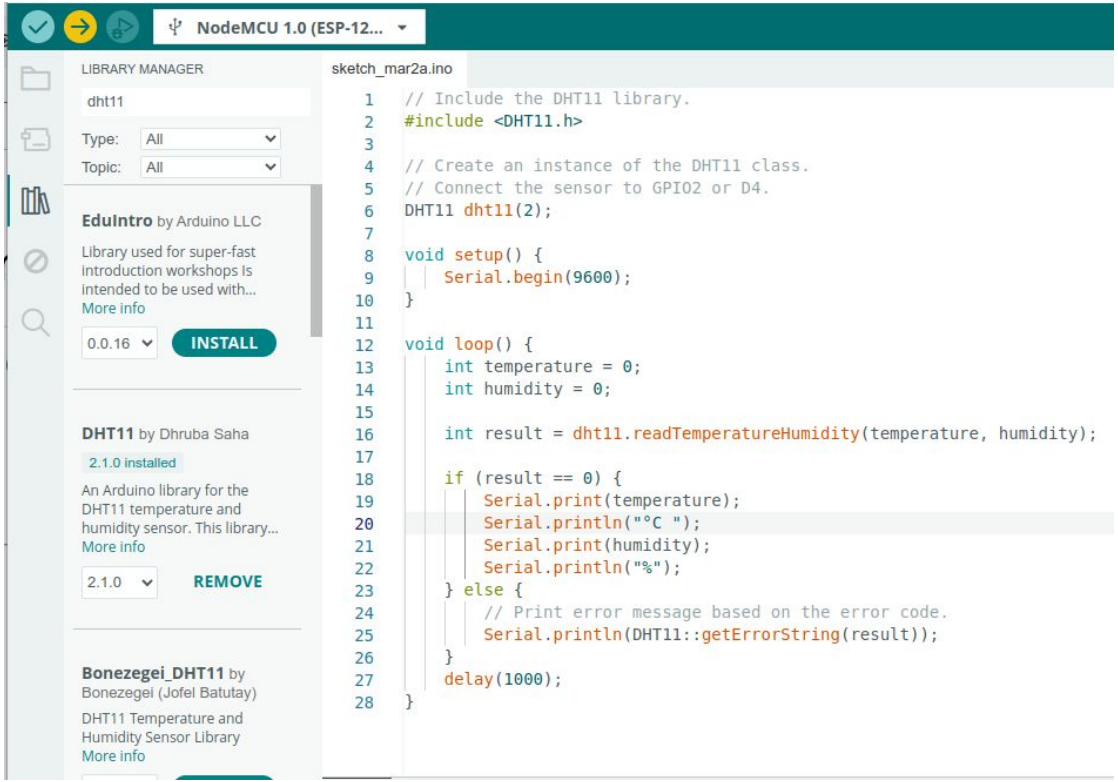
Arduino: cablear un sensor 1-wire



Sensores 1-wire, I²C, SPI…



Arduino: agregar y usar la biblioteca de un sensor

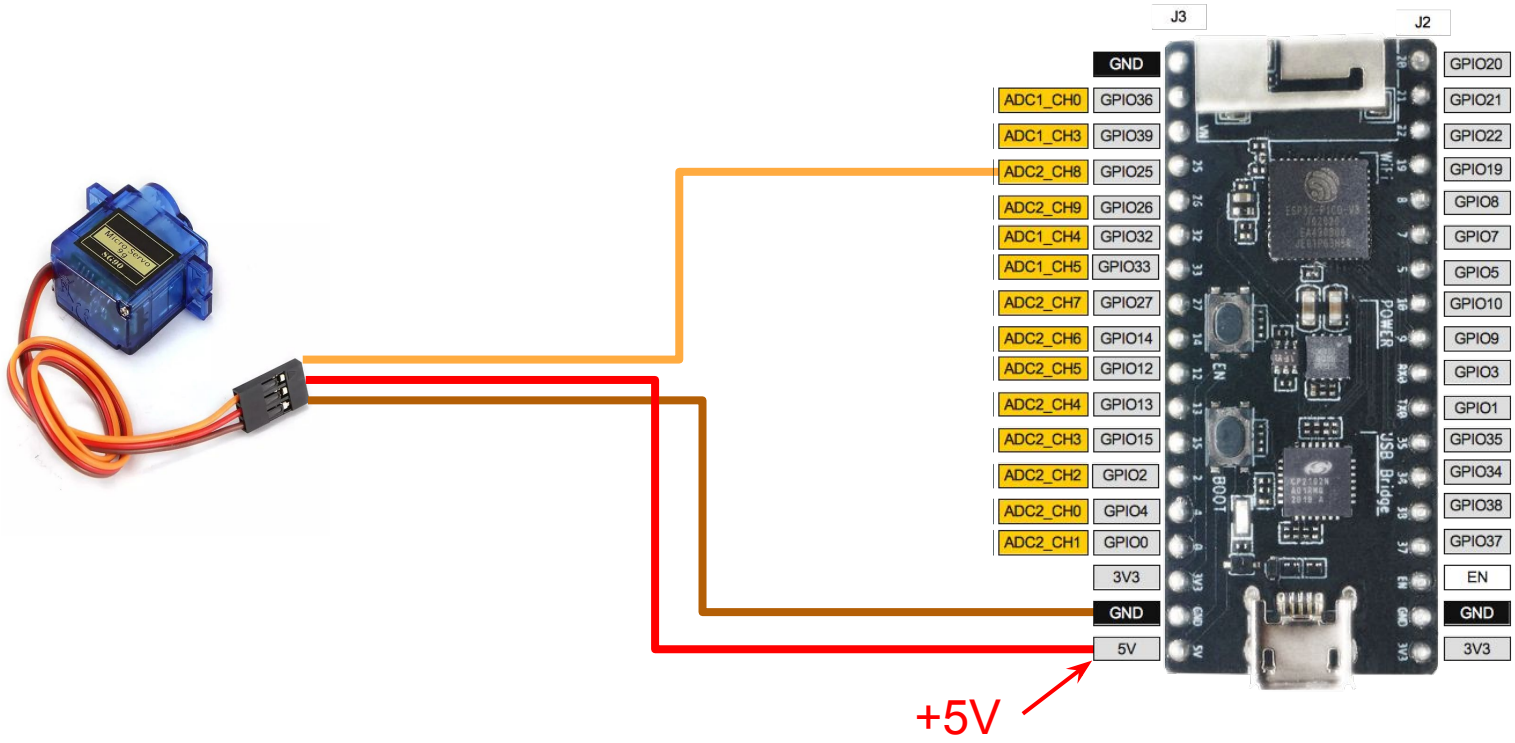


Arduino: leer sensor DHT11

```
#include <DHT11.h>  // Include the DHT11 library.
DHT11 dht11(D4);  // Connect the sensor to GPIO2 (D4)

void setup() {
  Serial.begin(115200);
}

void loop() {
  int temperature = 0;
  int humidity = 0;

  int result = dht11.readTemperatureHumidity(temperature, humidity);
  if (result == 0) {
    Serial.print(temperature); Serial.print("°C ");
    Serial.print(humidity); Serial.println("%");
  } else {
    Serial.println(DHT11::getErrorString(result));
  }
  delay(1000);
}
```

Cómo se controla un servo-motor

Arduino: cablear un servo-motor

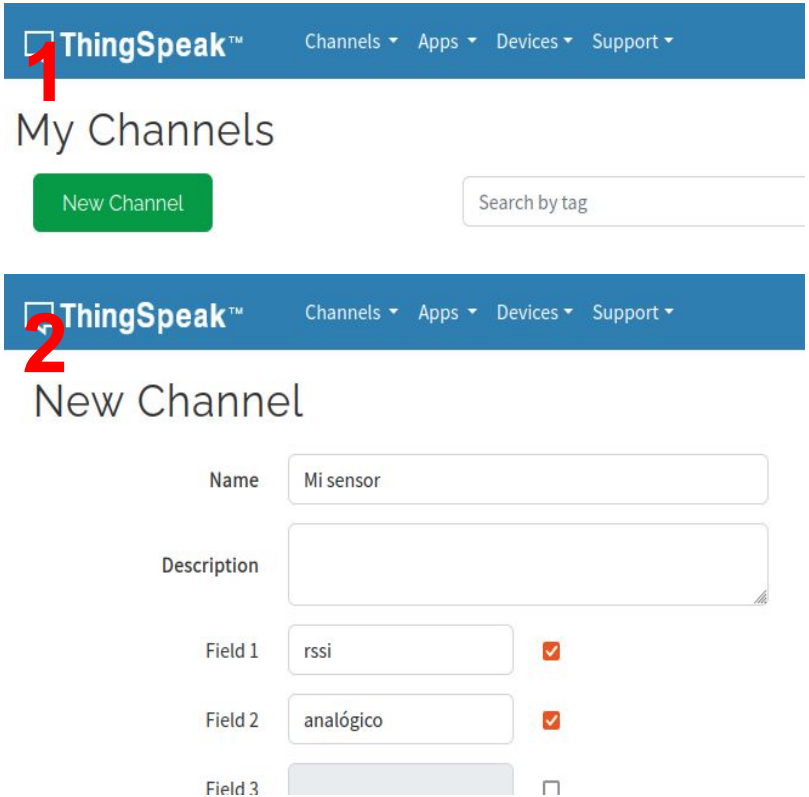


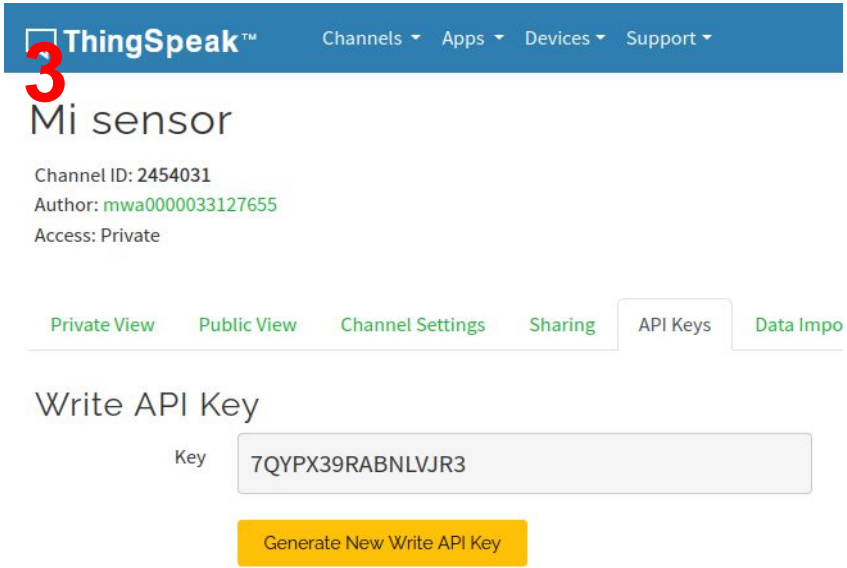
Arduino: controlar un servo-motor

```
#include <Servo.h>
Servo servo1;

void setup() {
  Serial.begin(115200);
  servo1.attach(25);  // servo attached to GPIO25 pin
}

void loop() {
  Serial.println(0);
  servo1.write(0);
  delay(1000);

  Serial.println(180);
  servo1.write(180);
  delay(1000);
}
```

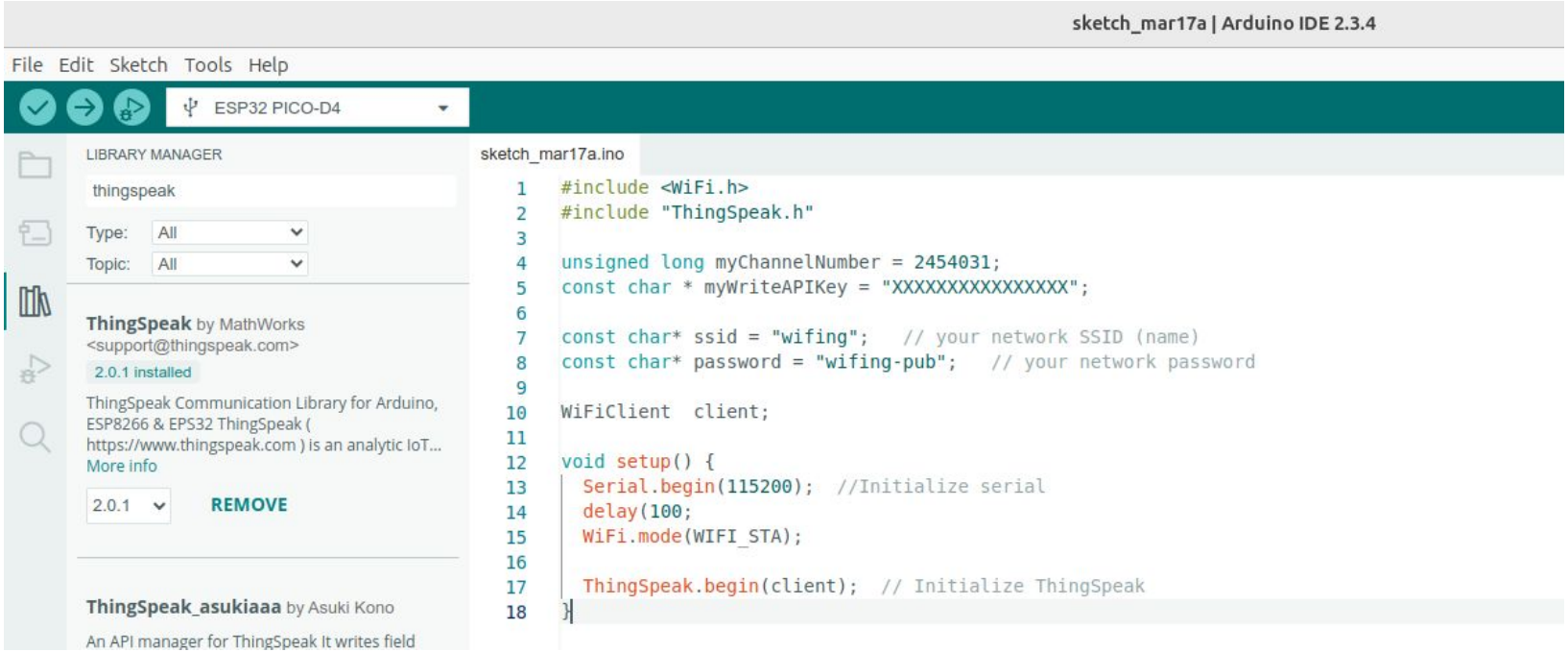
Cómo se envían datos a la nube

Creemos un canal





Publiquemos en thingspeak



Publiquemos en thingspeak (1 campo)

```
#include "ThingSpeak.h"
unsigned long myChannel = 2454031;
const char* writeAPIKey = "7QYPX39RABNLVJR3";

#include <WiFi.h>
char ssid[] = "wifing";
char pass[] = "wifing-pub";
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(100);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
}
```

```
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to Wifi...");
    while (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);
      delay(5000);
    }
  }

  long rssi = WiFi.RSSI();
  int httpCode = ThingSpeak.writeField(myChannel, 1, rssi, writeAPIKey);
  if (httpCode == 200) {
    Serial.println("Channel write successful.");
  } else {
    Serial.println("HTTP error code " + String(httpCode));
  }

  delay(20000);
}
```

Publiquemos en thingspeak (varios campos)

```
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    while (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);
      delay(5000);
    }
  }

  long rssi = WiFi.RSSI();
  int valueA0 = analogRead(A0);
  ThingSpeak.setField(1, rssi);
  ThingSpeak.setField(2, valueA0);

  int httpCode = ThingSpeak.writeFields(myChannel, myWriteAPIKey);
  if (httpCode == 200) {
    Serial.println("Channel write successful.");
  } else {
    Serial.println("Problem writing to channel. HTTP error code " + String(httpCode));
  }
  delay(20000);
}
```

Fin