Creando una IoT

Un tutorial con Thingspeak, Arduino y NodeMCU

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

Temario

Una Nube: thingspeak

Un microcontrolador: NodeMCU (ESP8266)

Un firmware: Arduino

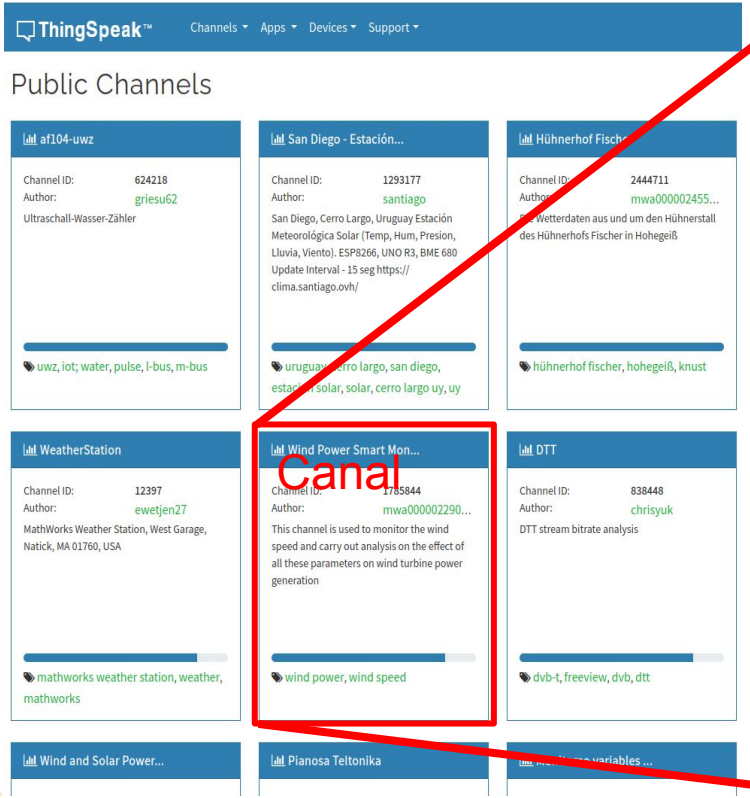
Programemos nuestro microcontrolador

- 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 NodeMCU (ESP8266)

Velocidad de reloj: 80MHz / 160MHz

RAM: 128kB

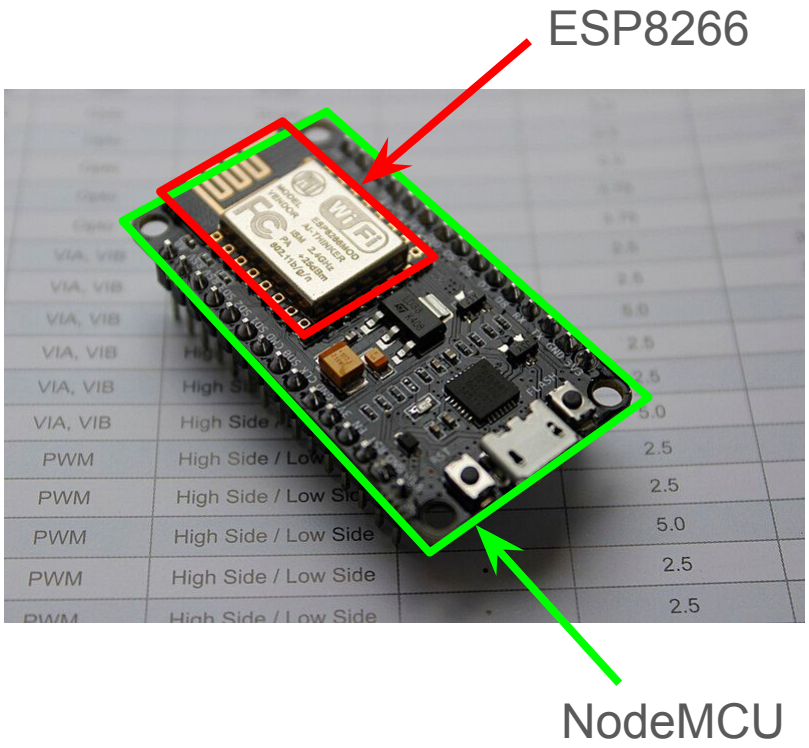
Flash: 4MB

Alimentación: USB (5V)

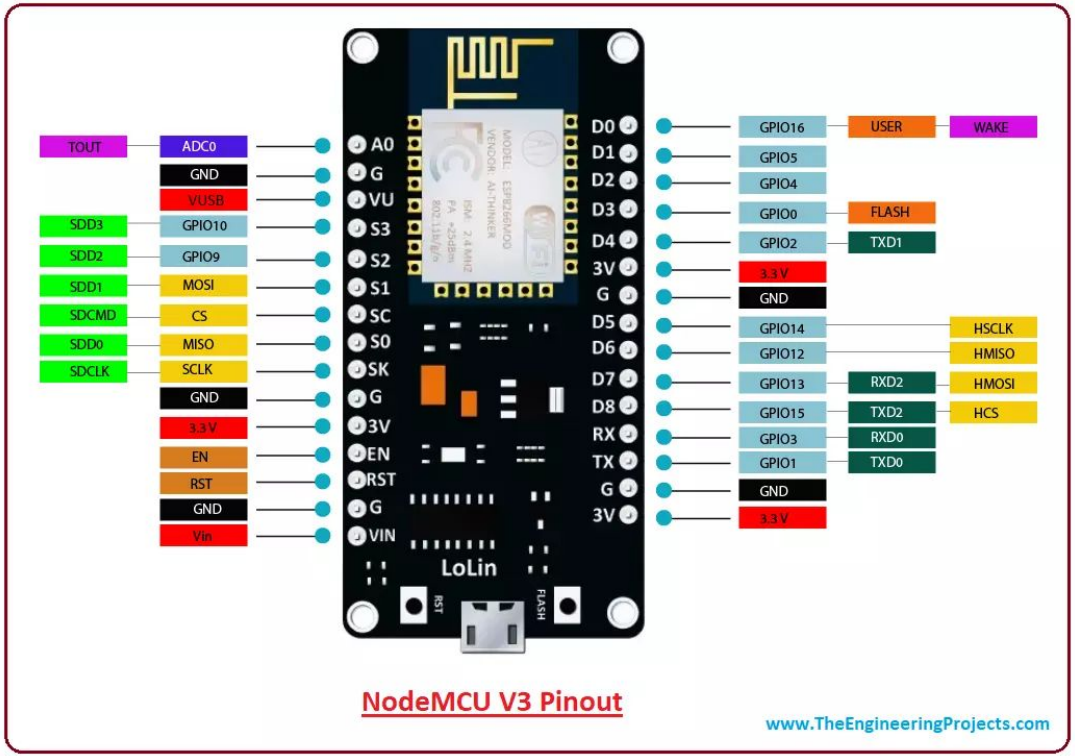
Voltaje de funcionamiento: 3.3V

WiFi integrada (802.11b/g/n)

Costo: 3-5usd



NodeMCU: pines

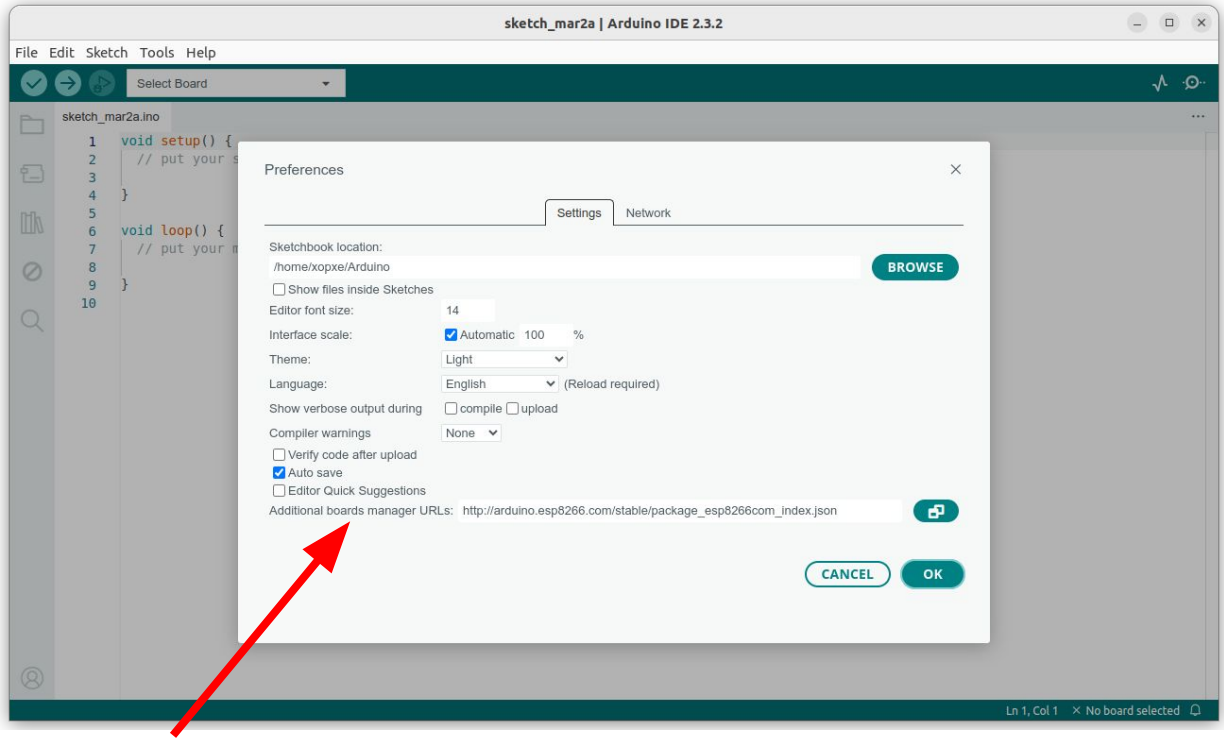


Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/

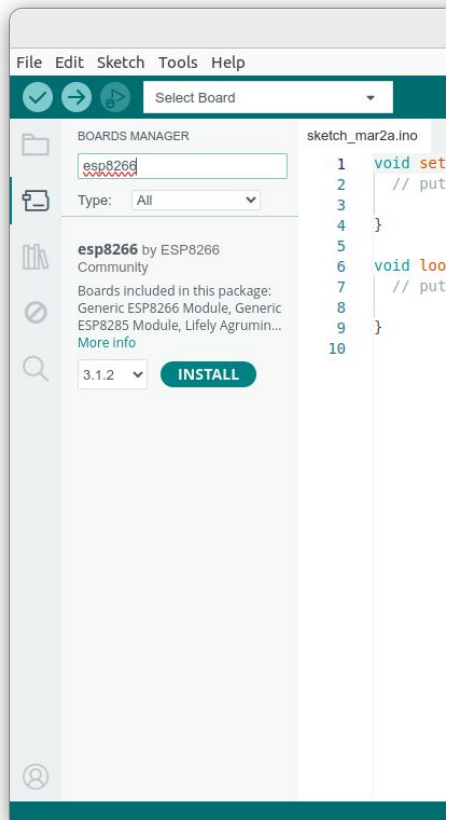
Cómo se programa un MCU

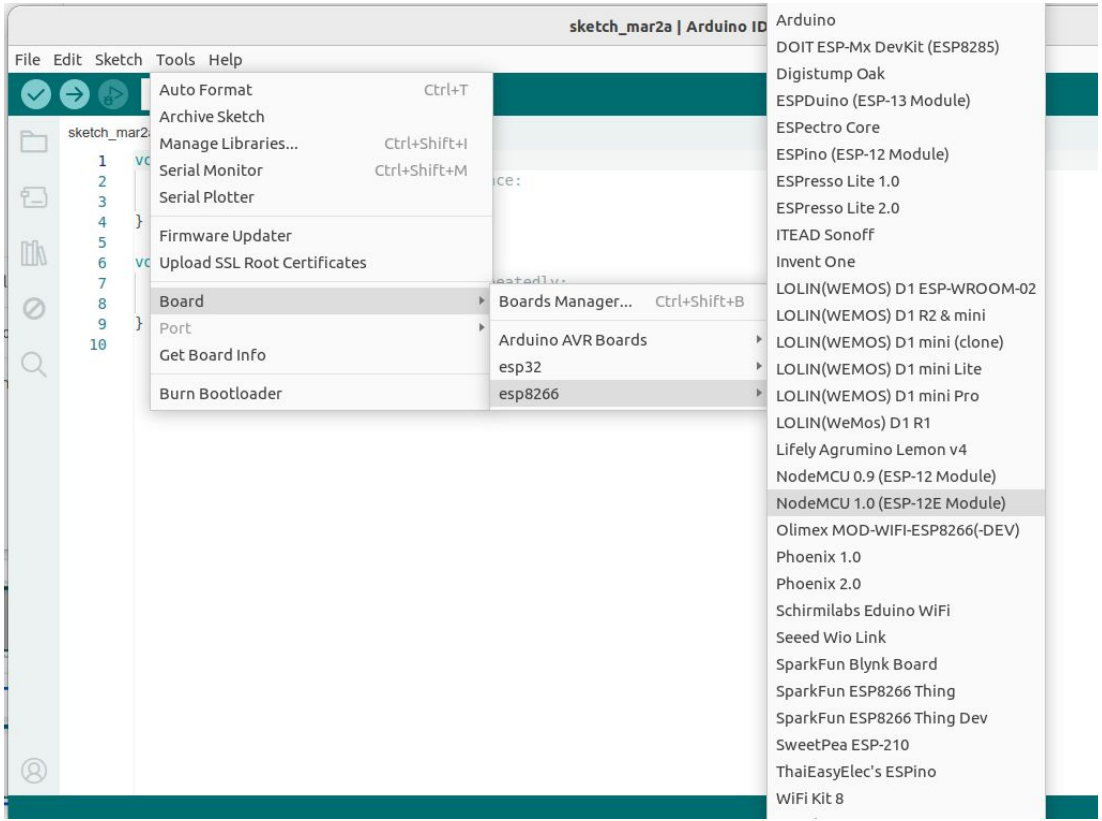
Arduino: agregar soporte para nuestro MCU



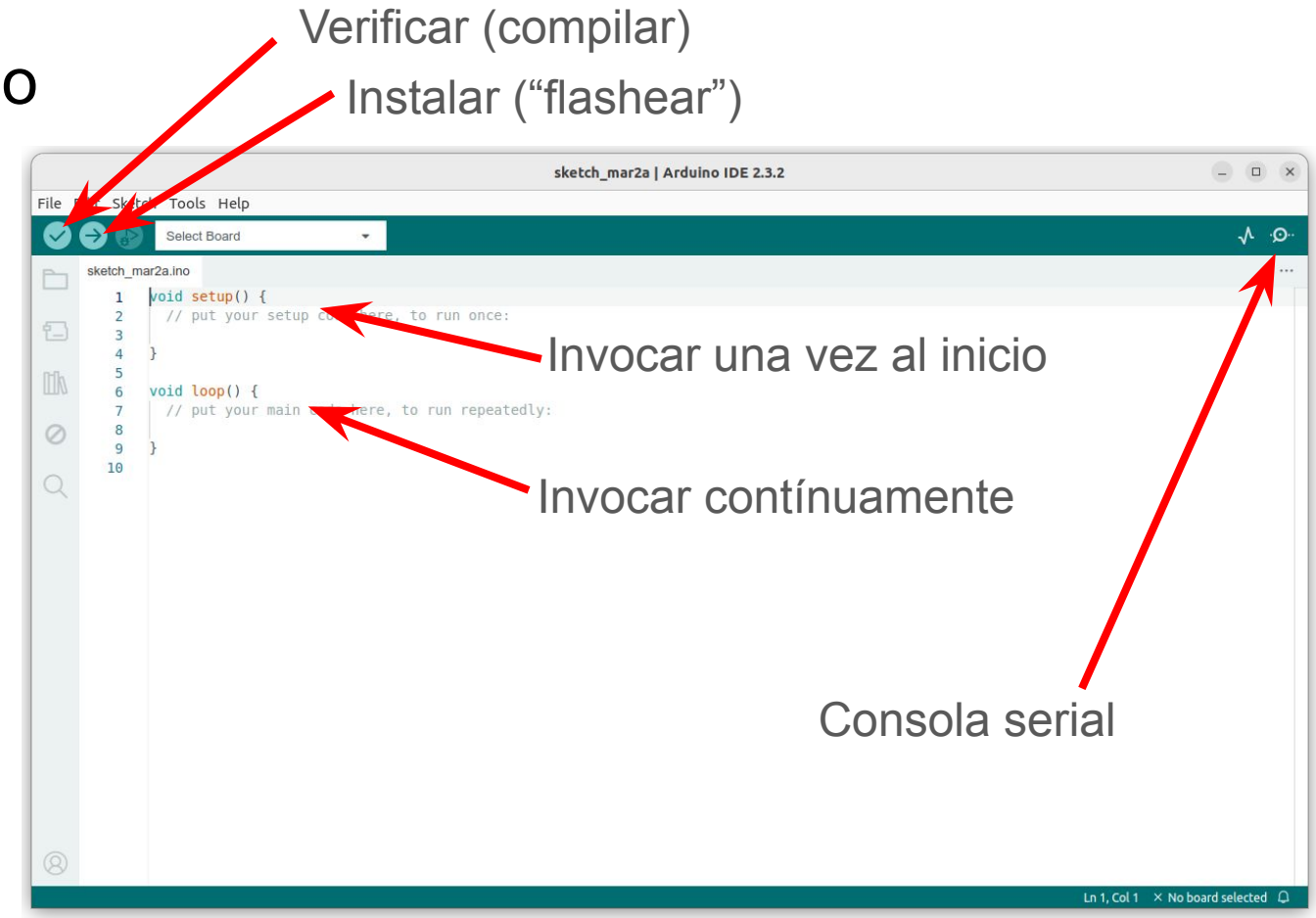
Agregar soporte para NodeMCU: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Arduino: agregar soporte para nuestro MCU





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: consola

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

void loop() {
}
```

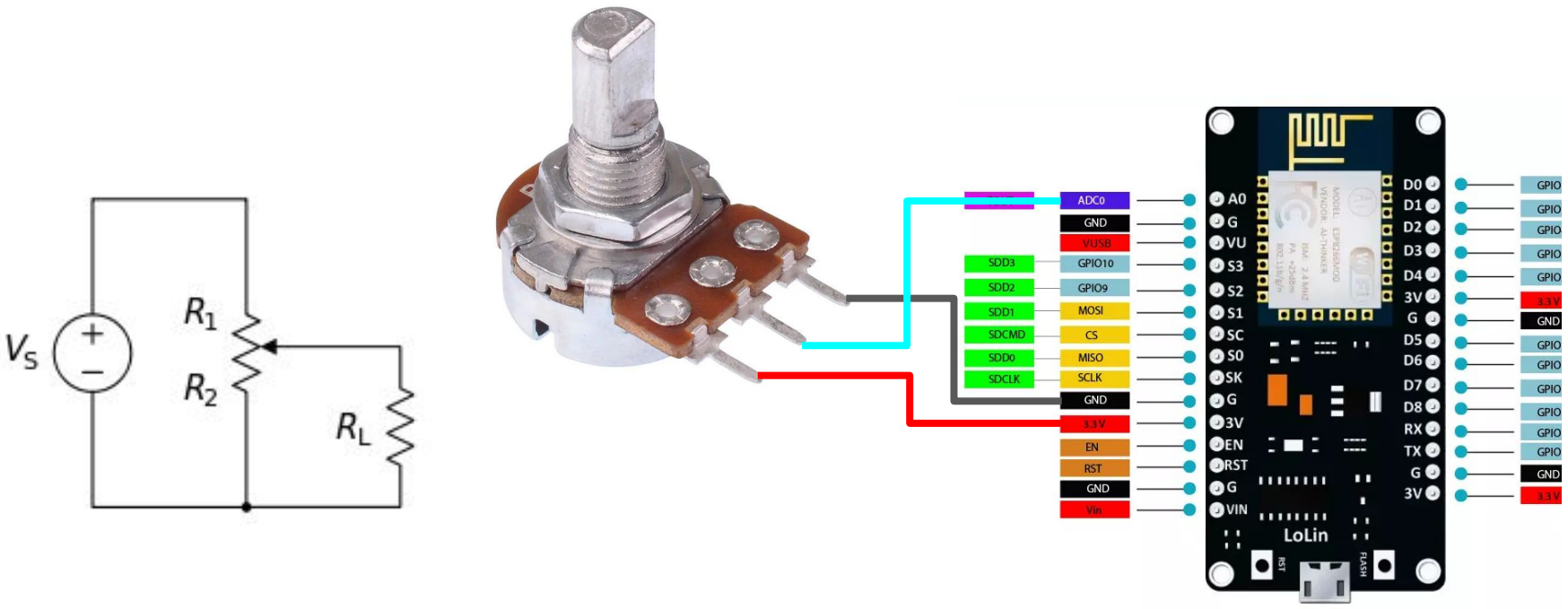
```
void setup() {
 Serial.begin(115200);
}

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

Arduino: escribir en pin digital

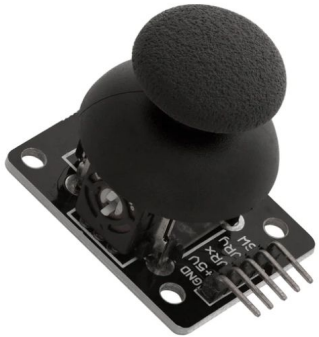
```
void setup() {
 Serial.begin(115200);
 pinMode(D4, OUTPUT);  // LED pin as output.
 Serial.println("\n\nLED pin: " + String(D4));
}

void loop() {
 Serial.print("1 ");
 digitalWrite(D4, HIGH);
 delay(1000);
 Serial.print("0 ");
 digitalWrite(D4, LOW);
 delay(1000);
}
```

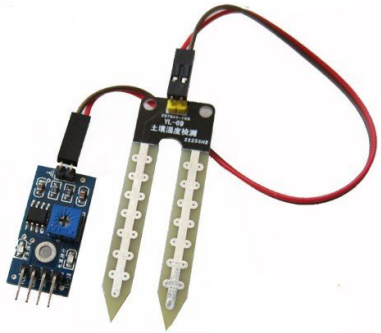
Cómo se lee un sensor analógico

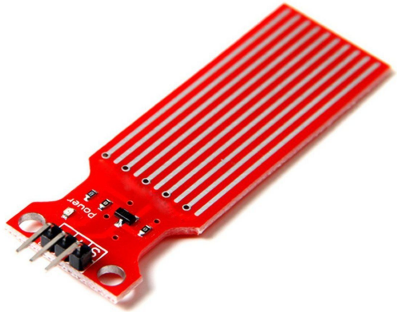
Arduino: cablear una resistencia variable



Sensores analógicos





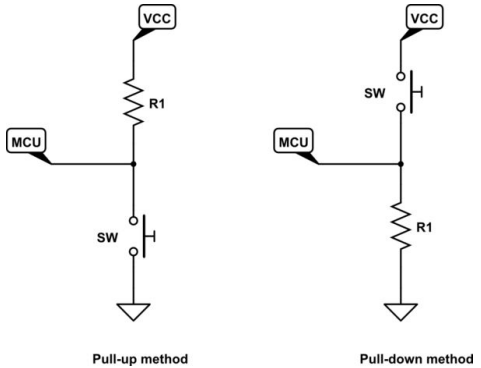


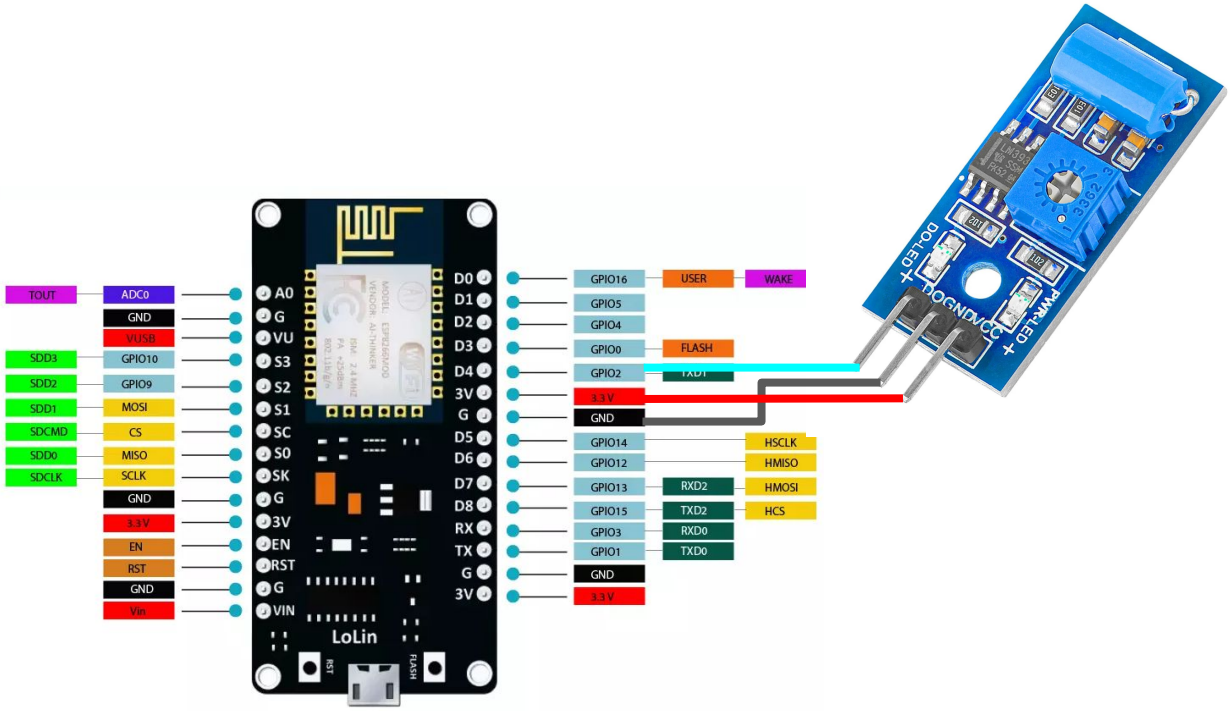
Arduino: conversor analógico-digital (ADC)

```
int last_change = 0;
int led_state = 0;

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

void loop() {
 int now = millis();
 int delay = analogRead(A0);
 Serial.println(String(delay));

 if (now - last_change > delay) {
   last_change = now;
   led_state = 1 - led_state;
   digitalWrite(LED_BUILTIN, led_state);
 }
}
```

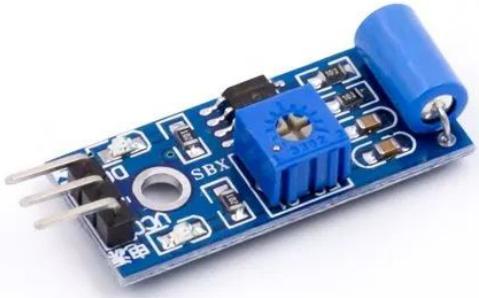
Cómo se lee un sensor digital

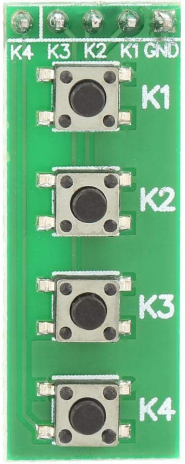
Arduino: cablear un sensor digital





Sensores digitales



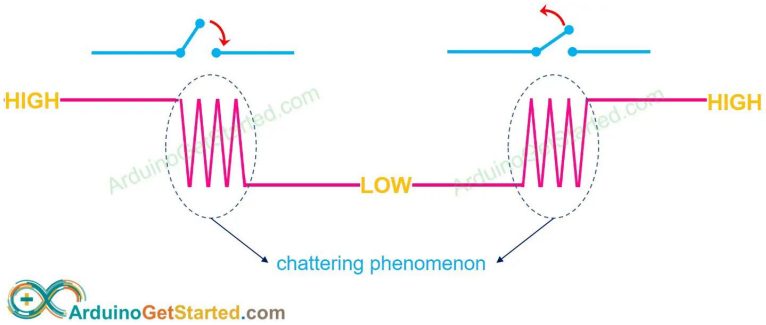


Arduino: leer un sensor digital (polling)

```
const int buttonPin = D2;

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

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

```
const int buttonPin = D2;
int lastState = LOW;

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

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

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
}
...
```

Arduino: leer un sensor digital (interrupciones)

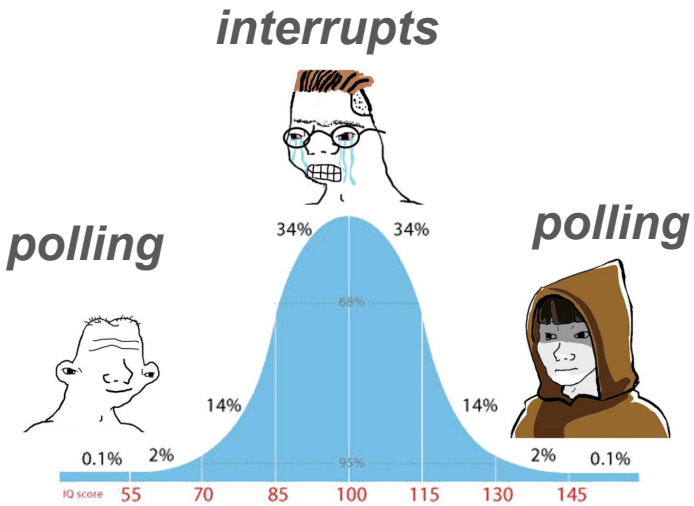
```
const int buttonPin = D2;
volatile byte clicked = 0;

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

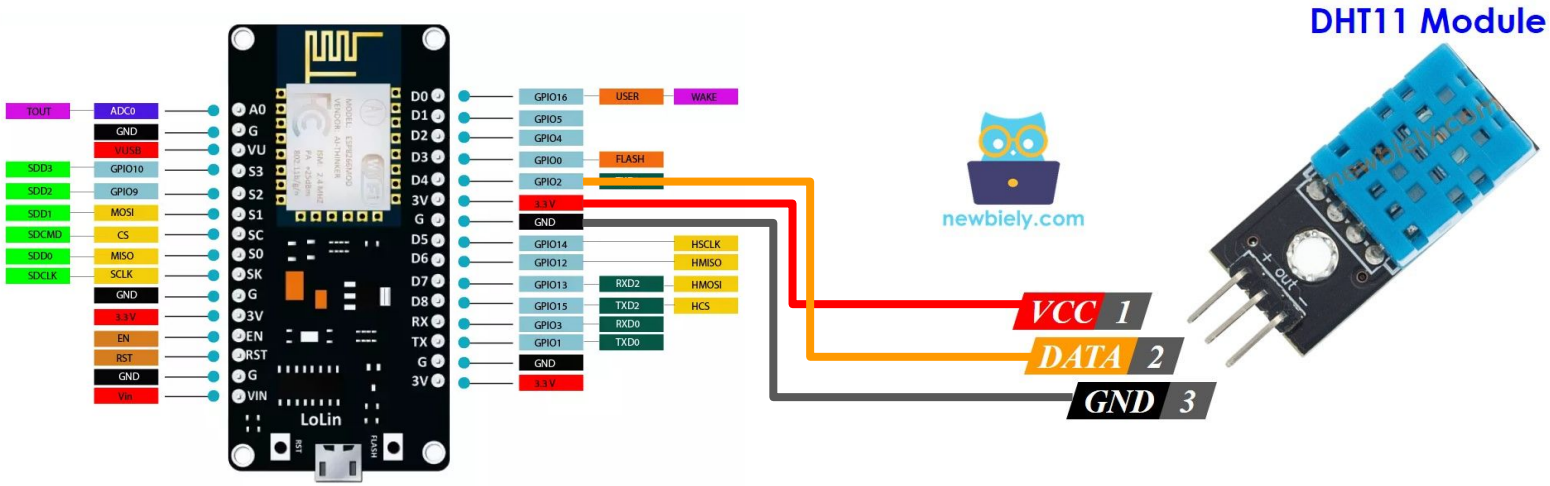
Arduino: leer un sensor digital (interrupciones)

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

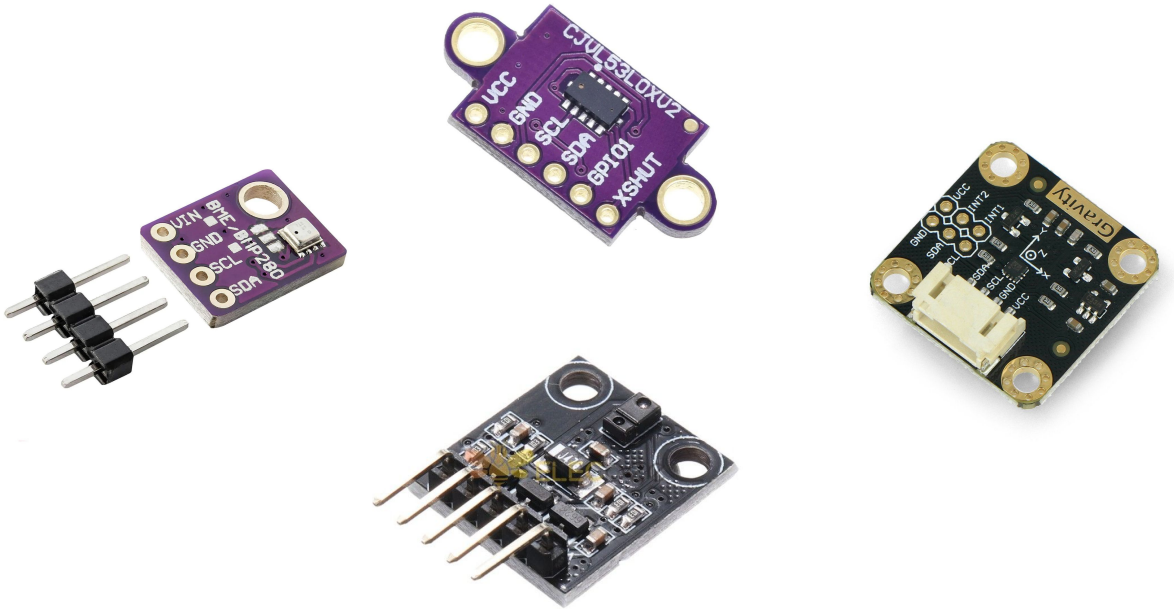


Cómo se lee un sensor basado en un protocolo (ej: 1-wire)

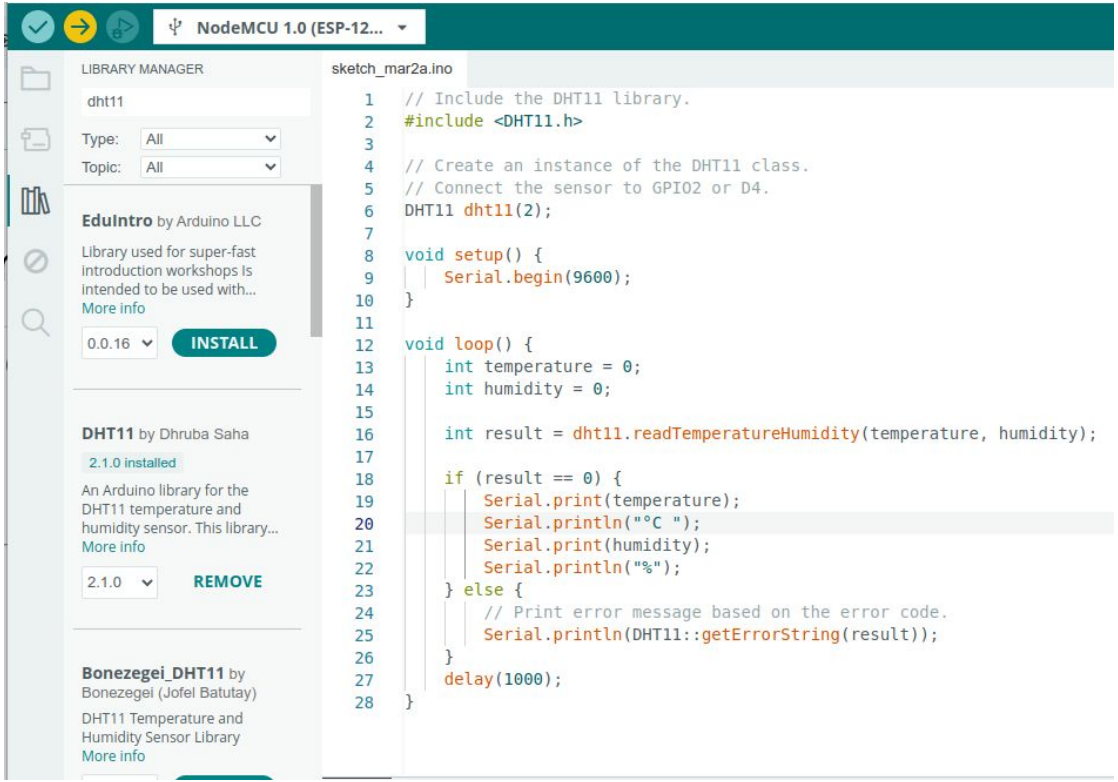
Arduino: cablear un sensor 1-wire



Sensores 1-wire, I²C, SPI…



Arduino: agregar y usar la biblioteca de un sensor

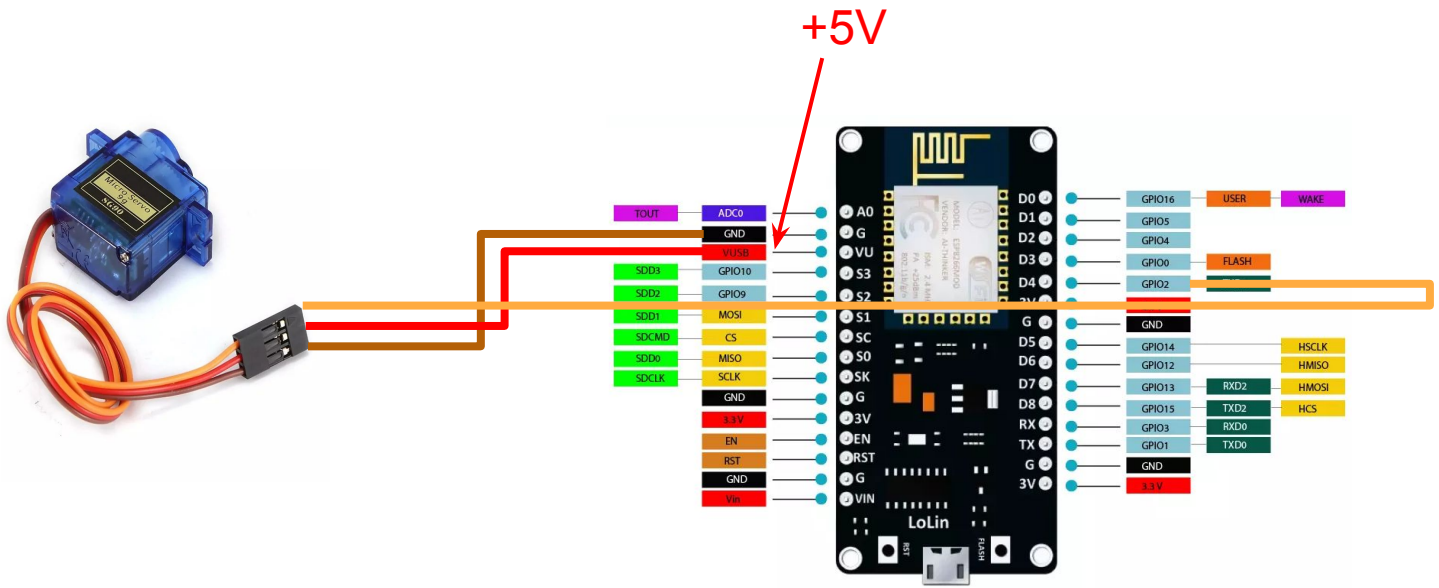


Arduino: leer sensor DHT11

```
#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

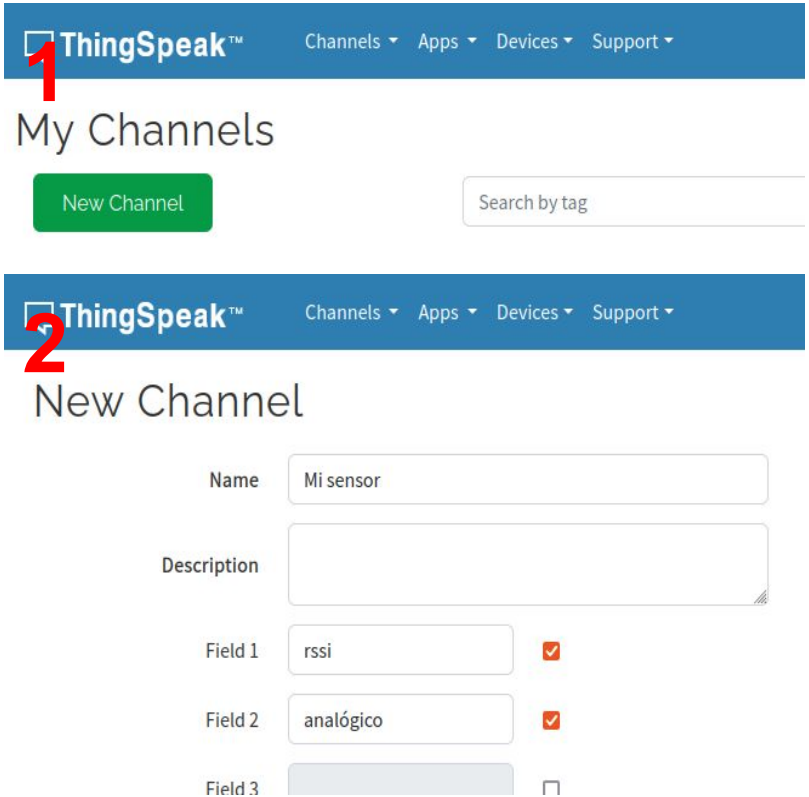


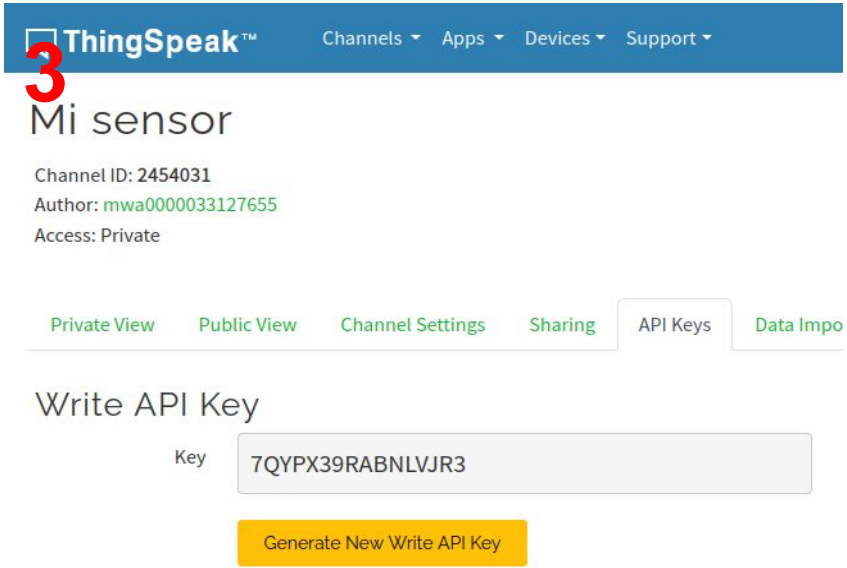
Arduino: controlar un servo-motor

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

void setup() {
 Serial.begin(115200);
 servo1.attach(D4);  // servo attach D4 pin of arduino
}

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

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

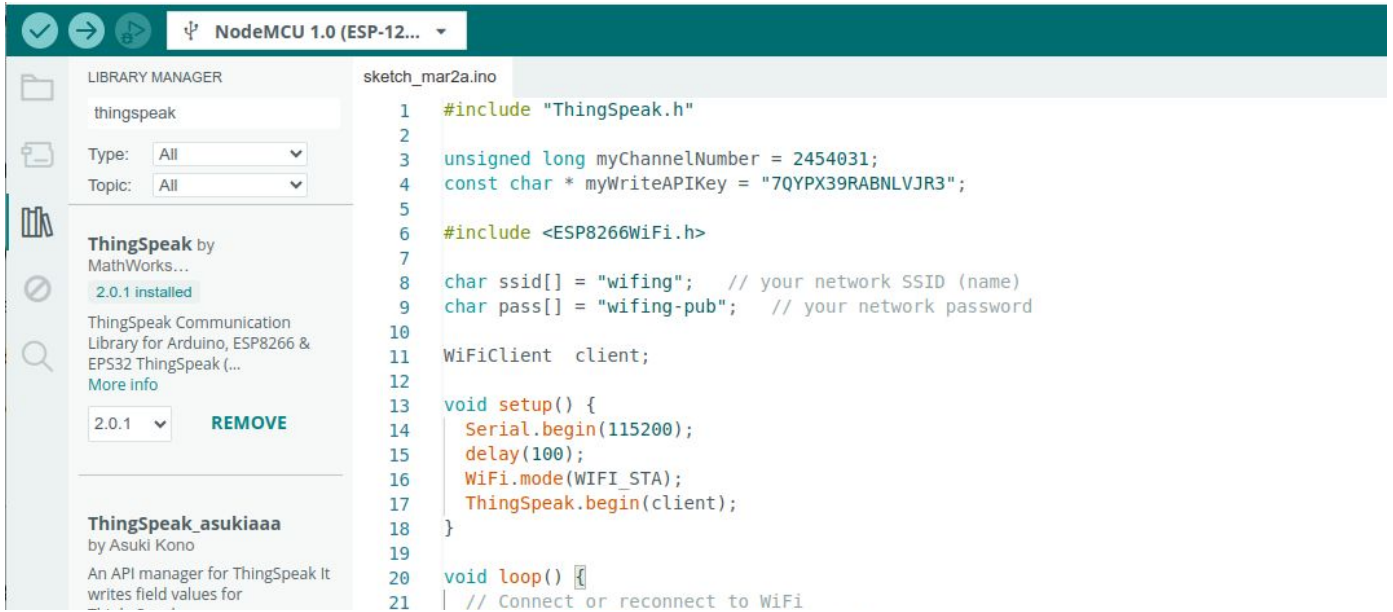
Cómo se envían datos a la nube

Creemos un canal





Publiquemos en thingspeak



Publiquemos en thingspeak (1 campo)

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

#include <ESP8266WiFi.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