

ACTIVIDAD 1. Capa de Adquisición

Objetivo

Comprender la funcionalidad de la capa de adquisición de la arquitectura básica de IoT a través del uso de plataformas hardware y sensores.

Conocer e implementar el formato de datos adecuado para la transmisión de los datos.

Herramientas

Se trabajará con las siguientes herramientas:

- ESP 32
- Sensores (MPU6050 u otros)
- Formato de datos JSON

El ESP32

El ESP32 es un microcontrolador de bajo costo y bajo consumo de energía diseñado por la empresa china Espressif Systems. Es uno de los microcontroladores más populares para proyectos de Internet de las cosas (IoT) debido a sus capacidades Wi-Fi y Bluetooth integradas, su amplia variedad de pines de entrada/salida (E/S) y su bajo costo.

El ESP32 se basa en un procesador de doble núcleo Xtensa LX6 de 32 bits con una velocidad de reloj de hasta 240 MHz. Además, cuenta con:

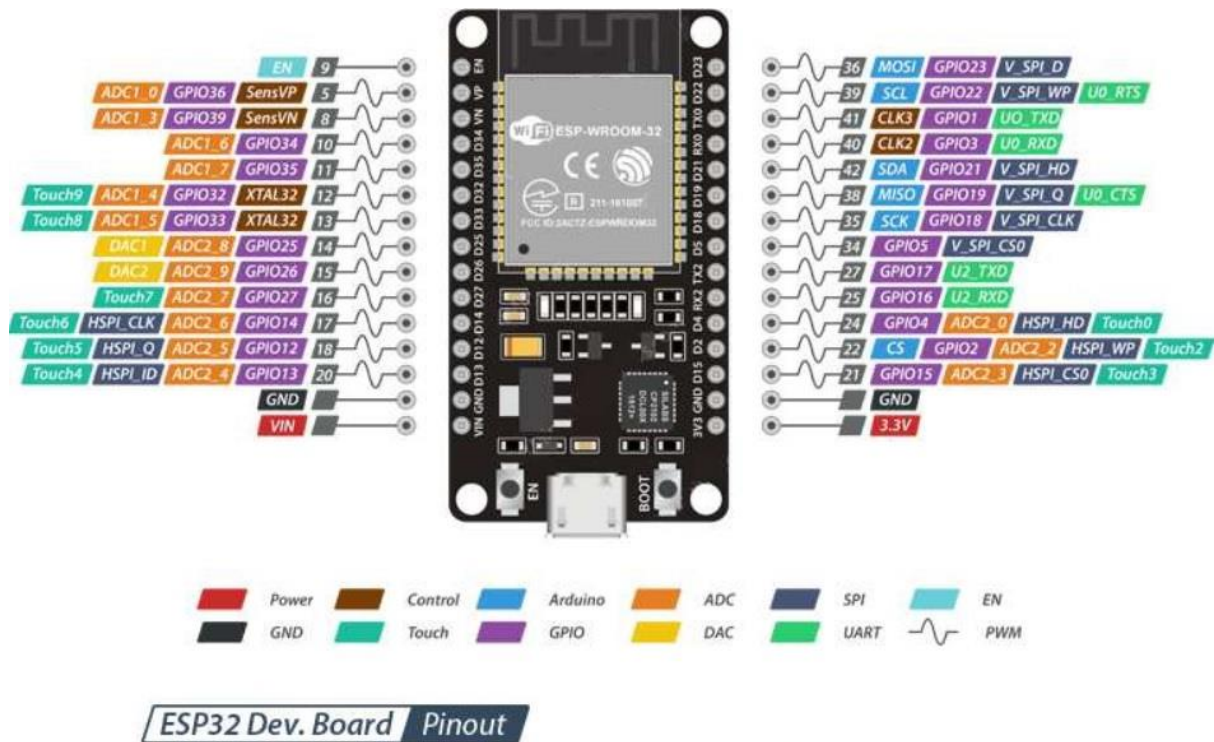
1. Memoria RAM de hasta 520 KB
2. Memoria flash de hasta 16 MB
3. Wi-Fi 802.11 b/g/n/e/i con soporte para estaciones, puntos de acceso y modo mesh
4. Bluetooth Classic y BLE (Bluetooth Low Energy) 4.2
5. Pines de entrada/salida (E/S) con soporte para interfaces SPI, I2C, UART, ADC, DAC, PWM, entre otros.

Existen varios tipos de ESP32 disponibles en el mercado, incluyendo:

1. ESP32-WROOM-32: Es el módulo más común y cuenta con una antena Wi-Fi y Bluetooth integrada. Tiene un tamaño de aproximadamente 18 x 25.5 mm.
2. ESP32-WROVER: Es un módulo más grande que el ESP32-WROOM-32, pero cuenta con más memoria RAM y una memoria flash adicional de 4 MB. También incluye una antena Wi-Fi y Bluetooth integrada.
3. ESP32-PICO-D4: Es un módulo más pequeño que el ESP32-WROOM-32, pero cuenta con una antena externa para Wi-Fi y Bluetooth. Además, tiene una memoria flash integrada de 4 MB y una memoria PSRAM (pseudo-static random-access memory) de 4 MB.

4. ESP32-SOLO-1: Es un módulo más pequeño que el ESP32-WROOM-32 y está diseñado para aplicaciones de bajo consumo. Cuenta con una antena Wi-Fi y Bluetooth integrada, pero tiene menos pines de E/S que otros módulos ESP32.
5. ESP32-S2: Es una versión más reciente del ESP32 que cuenta con Wi-Fi integrado, pero no tiene Bluetooth integrado. Cuenta con un procesador más rápido de 240 MHz, una memoria flash integrada de 2 MB y una memoria RAM de 320 KB.

La distribución de los pines del ESP32 WROOM32 (de 30 pines) es la que se muestra en la siguiente figura:



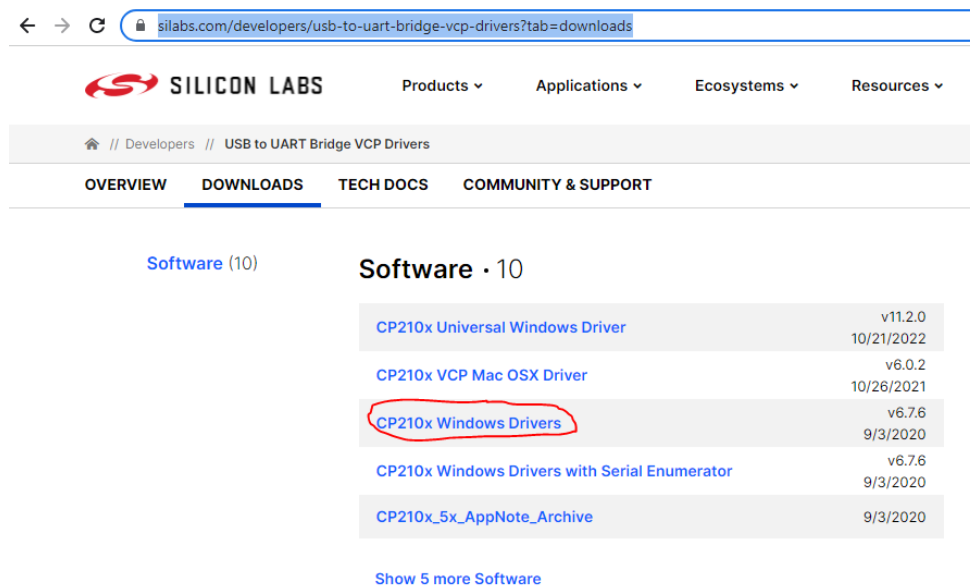
- 18 canales de conversión analógico-digital (ADC)
- 10 GPIO de detección capacitiva
- 3 interfaces UART
- 3 interfaces SPI
- 2 interfaces I2C
- 16 canales de salida PWM
- 2 Convertidores de digital a analógico (DAC)
- 2 interfaces I2S

Configuración del IDE de Arduino para trabajar con el ESP32

Para trabajar con el ESP32 se deben realizar los siguientes pasos:

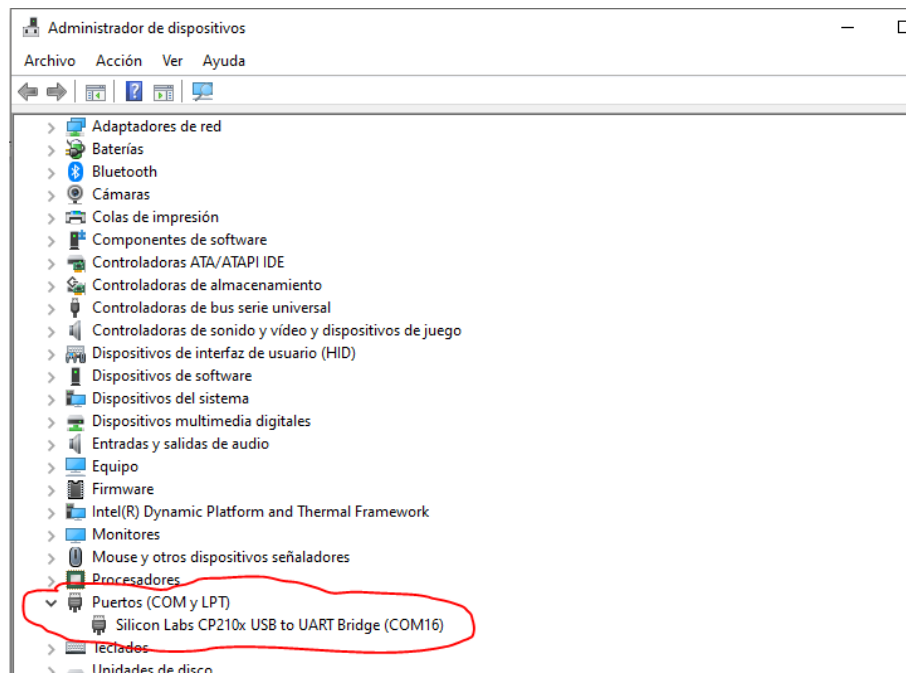
1. Instalación de drivers:

Los drivers para trabajar con el ESP32 se pueden descargar de la siguiente página:
<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>



Después se instala la versión de 64 bits.

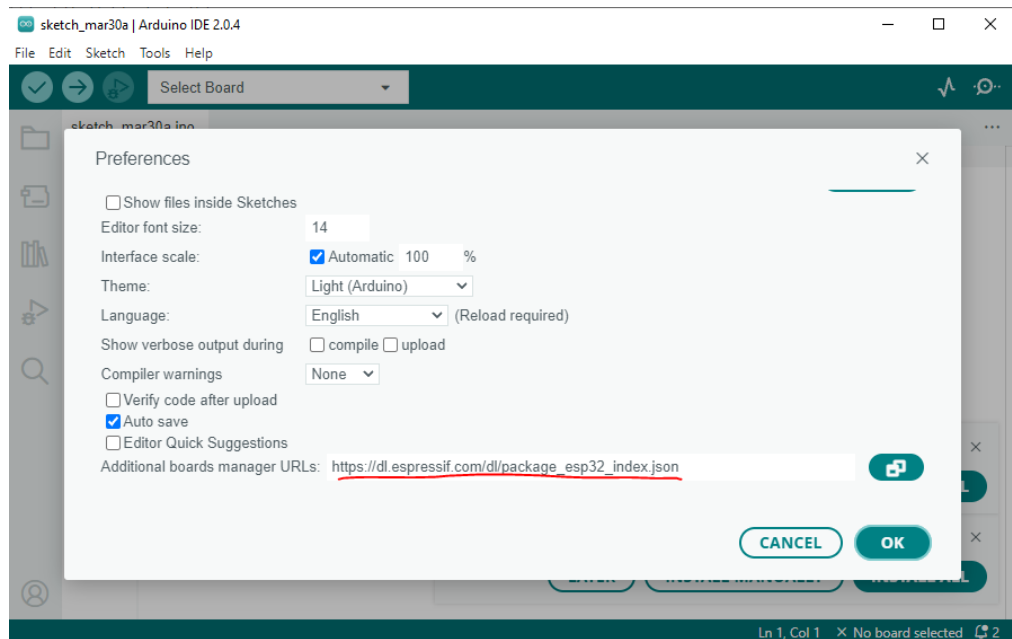
Esto permitirá que cuando se conecte el ESP32 sea reconocido como un dispositivo por Windows y parecerá de la siguiente manera en el administrador de dispositivos:



2. Agregar la tarjeta de desarrollo en la preferencias del IDE de Arduino.

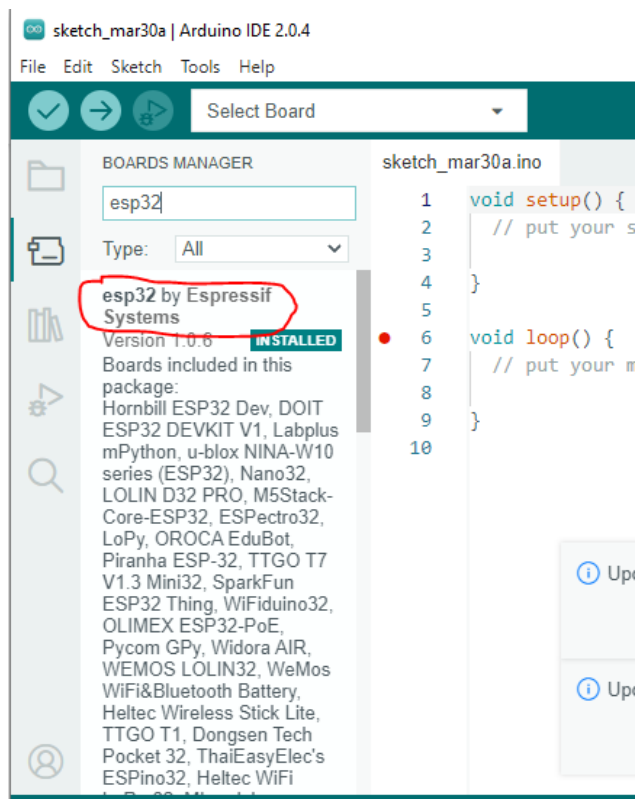
Esto se hace agregando el siguiente enlace:

https://dl.espressif.com/dl/package_esp32_index.json



Luego de esto se cierra y vuelve a abrir el IDE de Arduino.

3. Se procede a instalar con el gestor de tarjetas de Arduino el ESP32



Se cierra el IDE y se vuelve a abrir.

4. Probamos que funcione con el siguiente sketch: blink

```
/*
 * Ejemplo blink ESP32s
 */
```

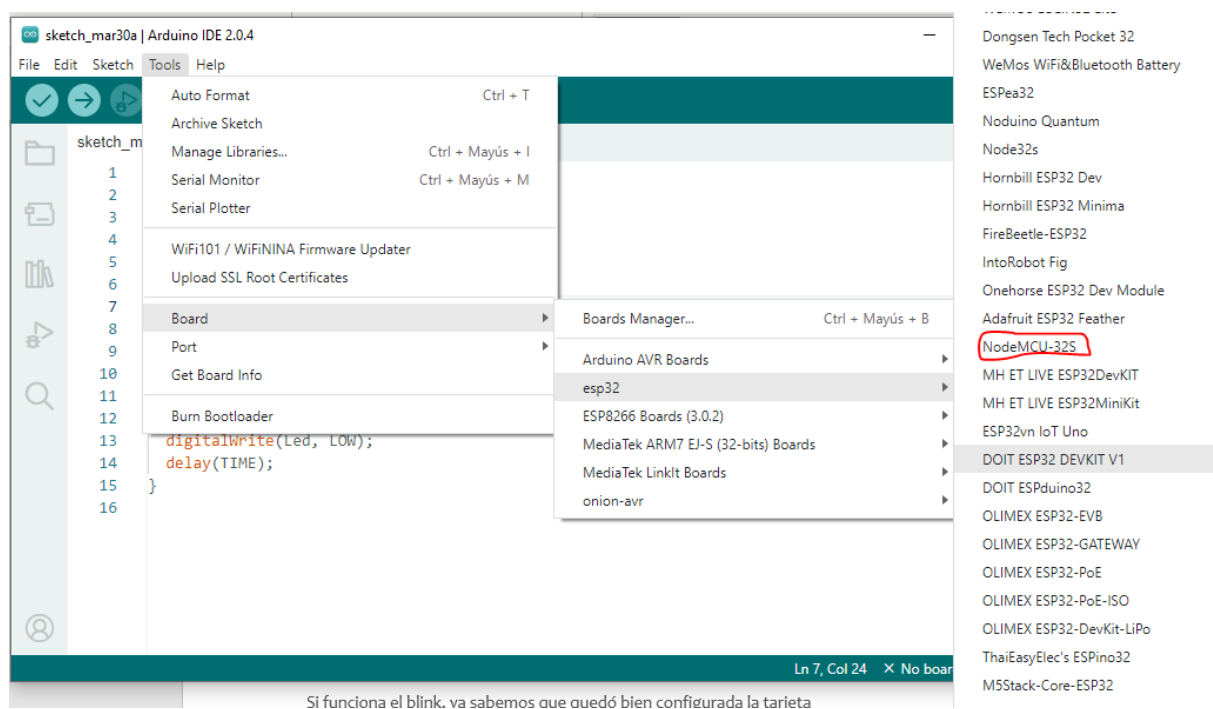
```

#define Led 2
const int TIME = 1000;
void setup() {
    pinMode(Led, OUTPUT);
}

void loop() {
    digitalWrite(Led, HIGH);
    delay(TIME);
    digitalWrite(Led, LOW);
    delay(TIME);
}

```

Seleccionamos la tarjeta nodeMCU-32s y el puerto en el que se conectó el ESP32.



Se procede a cargar el sketch y si funciona el blink, ya sabemos que quedó bien configurada la tarjeta

Formato JSON

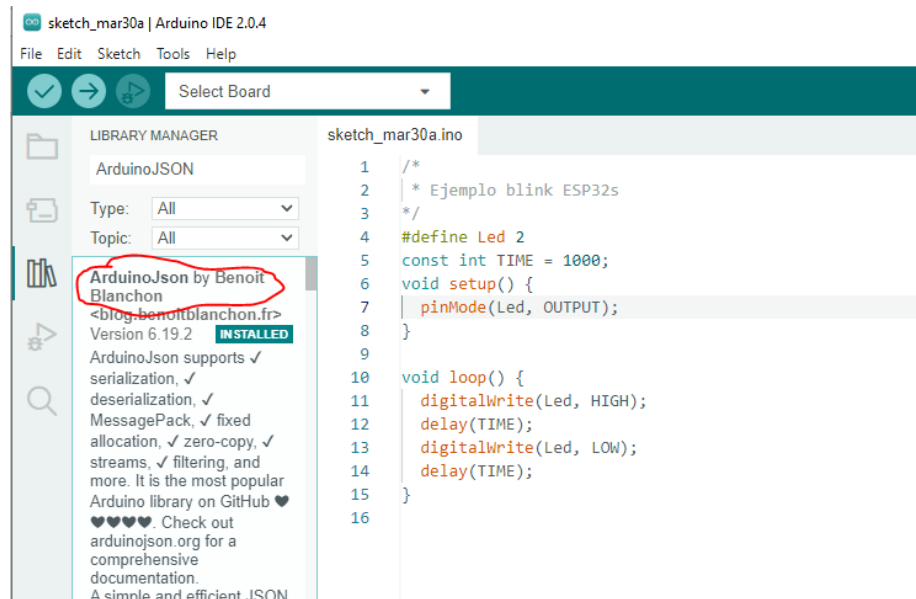
(El siguiente texto es tomado de la página oficial de JSON: <https://www.json.org/json-es.html>)

JSON (Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

JSON está constituido por dos estructuras:

JSON en Arduino

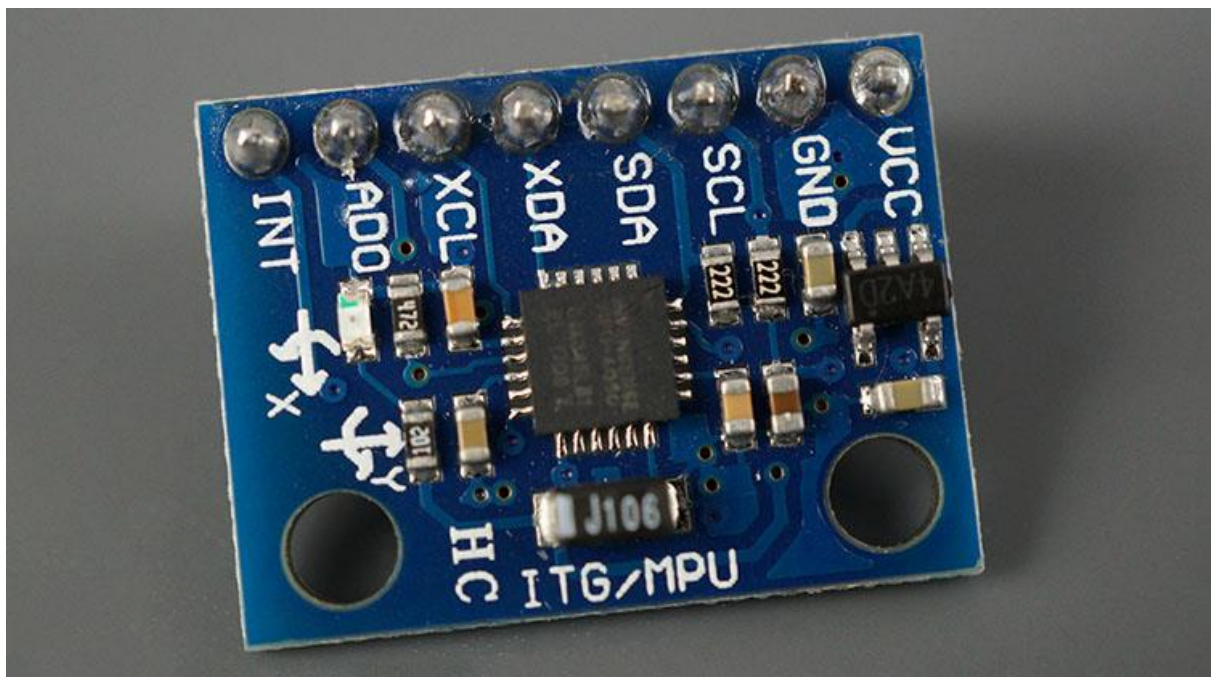
Para trabajar con el formato JSON en Arduino existe una librería denominada ArduinoJSON, la cual debemos instalar:



En el siguiente enlace: <https://www.luisllamas.es/arduino-json/> se da una explicación clara de cómo usar la librería para serializar y deserializar json.

Sensor (ejemplo con MPU6050)

El MPU-6050 IMU (Unidad de medición inercial) es un acelerómetro de 3 ejes y un sensor de giroscopio de 3 ejes. El acelerómetro mide la aceleración gravitacional y el giroscopio mide la velocidad de rotación. Además, este módulo también mide la temperatura. Este sensor es ideal para determinar la orientación de un objeto en movimiento.



La configuración de pines es la siguiente:

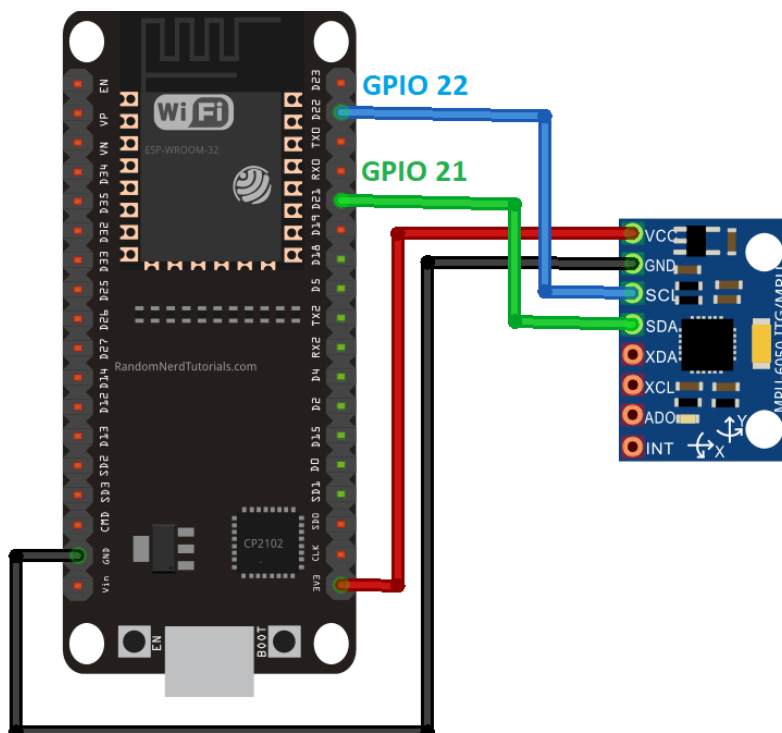
CCV	Alimentar el sensor (3.3V o 5V)
TIERRA	Tierra común
SCL	Pin SCL para comunicación I2C (GPIO 22)
ASD	Pin SDA para comunicación I2C (GPIO 21)
XDA	Se utiliza para interconectar otros sensores I2C con el MPU-6050
XCL	Se utiliza para interconectar otros sensores I2C con el MPU-6050
AD0	Use este pin para cambiar la dirección I2C
EN T	Pin de interrupción: se puede utilizar para indicar que hay nuevos datos de medición disponibles

Para su uso se deben instalar las siguientes librerías en el IDE de Arduino:

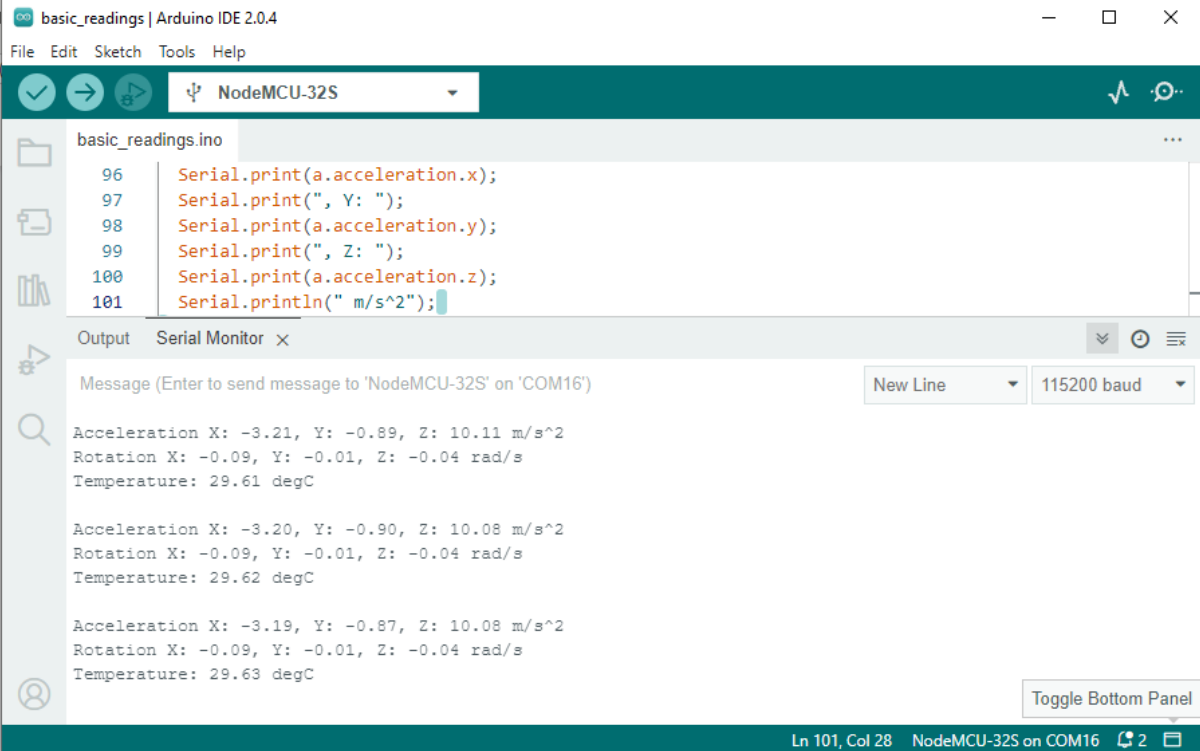
1. Adafruit mpu6050
2. Sensor unificado de Adafruit
3. Adafruit Bus IO

Después de instaladas se cierra el IDE y se vuelve a abrir.

La conexión del sensor al ESP32 se hace a través de los puertos I2C de la siguiente manera:



Para probar el funcionamiento del sensor cargamos el sketch de ejemplo `basic_readings` que viene con la librería del MPU6050. Este ejemplo muestra el resultado de la lectura del acelerómetro, del giroscopio y de la temperatura:



The screenshot shows the Arduino IDE 2.0.4 interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar shows icons for running, uploading, and other functions. The sketch is named 'basic_readings.ino' and is being compiled for a 'NodeMCU-32S' board. The code in the editor is as follows:

```
96 Serial.print(a.acceleration.x);
97 Serial.print(", Y: ");
98 Serial.print(a.acceleration.y);
99 Serial.print(", Z: ");
100 Serial.print(a.acceleration.z);
101 Serial.println(" m/s^2");
```

The Serial Monitor is open, showing the output of the sketch. The output is as follows:

```
Message (Enter to send message to 'NodeMCU-32S' on 'COM16')
Acceleration X: -3.21, Y: -0.89, Z: 10.11 m/s^2
Rotation X: -0.09, Y: -0.01, Z: -0.04 rad/s
Temperature: 29.61 degC

Acceleration X: -3.20, Y: -0.90, Z: 10.08 m/s^2
Rotation X: -0.09, Y: -0.01, Z: -0.04 rad/s
Temperature: 29.62 degC

Acceleration X: -3.19, Y: -0.87, Z: 10.08 m/s^2
Rotation X: -0.09, Y: -0.01, Z: -0.04 rad/s
Temperature: 29.63 degC
```

The status bar at the bottom indicates 'Ln 101, Col 28' and 'NodeMCU-32S on COM16'.

Procedemos a modificar el ejemplo para que los datos sean mostrados en el formato JSON que requerimos.

Asumamos que el json que requerimos debe tener el siguiente formato:

```
{
  "accx": -3.21,
  "accy": -0.89,
  "accz": 10.11,
  "rotx": -0.09,
  "roty": -0.01,
  "rotz": -0.04,
  "temp": 29.62
}
```

El código quedaría de la siguiente manera:

```
// Basic demo for accelerometer readings from Adafruit MPU6050

#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <ArduinoJson.hpp>
#include <ArduinoJson.h>
```

```

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+16G");
    break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
    Serial.println("+ 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+ 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+ 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+ 2000 deg/s");
    break;
  }
}

```

```

}

mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
Serial.print("Filter bandwidth set to: ");
switch (mpu.getFilterBandwidth()) {
case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
}

Serial.println("");
delay(100);
}

void loop() {

    /* Get new sensor events with the readings */
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);

    String json;
    StaticJsonDocument<300> doc;
    doc["accx"] = a.acceleration.x;
    doc["accy"] = a.acceleration.y;
    doc["accz"] = a.acceleration.z;
    doc["rotx"] = g.gyro.x;
    doc["roty"] = g.gyro.y;
    doc["rotz"] = g.gyro.z;
    doc["temp"] = temp.temperature;

    serializeJson(doc, json);

```

```

Serial.println(json);

delay(5000);
}

```

Después de subir el sketch el resultado en el monitor serial será el siguiente:

```

MPU6050 Found!
Accelerometer range set to: +-8G
Gyro range set to: +- 500 deg/s
Filter bandwidth set to: 21 Hz

{"accx":-0.263362199,"accy":0.375889659,"accz":10.49618053,"rotx":-0.090064317,"roty":-0.017320061,"rotz":-0.034107197,"temp":35.76235199}
{"accx":-0.244208574,"accy":0.387860686,"accz":10.49618053,"rotx":-0.086067379,"roty":-0.023448698,"rotz":-0.033041347,"temp":35.92705917}
{"accx":-0.275333196,"accy":0.368707061,"accz":10.48660374,"rotx":-0.085534461,"roty":-0.016787136,"rotz":-0.034640122,"temp":35.95941162}
{"accx":-0.265756398,"accy":0.380678058,"accz":10.53927612,"rotx":-0.087666154,"roty":-0.018652374,"rotz":-0.033840735,"temp":35.95941162}
{"accx":-0.270544797,"accy":0.387860686,"accz":10.47463226,"rotx":-0.087133229,"roty":-0.016787136,"rotz":-0.031975497,"temp":35.93588257}
{"accx":-0.270544797,"accy":0.366312861,"accz":10.47702694,"rotx":-0.085534461,"roty":-0.018119449,"rotz":-0.034107197,"temp":35.94764709}
{"accx":-0.282515794,"accy":0.378283858,"accz":10.50815105,"rotx":-0.086600304,"roty":-0.014921899,"rotz":-0.033041347,"temp":35.90058899}

```