# Manual de Usuario
# Dispositivo Automatizado de Sensores para Arduino
# DASA 2.0

---

PCB v10b

**Miguel Ángel Bañuelos Saucedo**

**Milagros Pacheco Castañeda**

**Rebeca Guillermina Villegas Salas**

**Junio de 2025**

Proyecto PAPIME PE108923

Desarrollo de material didáctico para la asignatura Informática Aplicada a la Ciencia y a la Industria

**Financiamiento: Dirección General de Asuntos del Personal Académico**

Universidad Nacional Autónoma de México
Dirección General de Asuntos del Personal Académico
Proyecto PAPIME PE108923
Ciudad de México, 2025

# Dispositivo Automatizado de Sensores para Arduino DASA 2.0

Miguel Ángel Bañuelos Saucedo[1]

Milagros Pacheco Castañeda[2]

Rebeca Guillermina Villegas Salas[3]

[1] Instituto de Ciencias Aplicadas y Tecnología, Grupo de Desarrollo de Sistemas Electrónicos.

[2] Escuela Nacional Preparatoria, plantel 5.

[3] Escuela Nacional Preparatoria, plantel 7.

## RESUMEN

La enseñanza de elementos básicos de programación y electrónica en los niveles de Enseñanza Media Superior y Superior a menudo enfrenta a los profesores con el reto de revisar circuitos y programas de grupos numerosos. En este trabajo se aborda esa problemática al presentar el diseño de una tarjeta de aprendizaje denominada DASA 2.0 (Dispositivo Automatizado de Sensores para Arduino), la cual se basa en un módulo ESP32 que puede programarse con el mismo ambiente IDE del Arduino. También tiene una pantalla OLED de 0.96 pulgadas, cuatro botones, un zumbador, un LED RGB, una fotorresistencia, un potenciómetro, y varios conectores para insertar módulos de sensores y actuadores. La placa presenta componentes ya conectados y otros que se pueden añadir fácilmente mediante la inserción directa en los receptáculos de que dispone. Esto reduce el nivel de supervisión de parte de los maestros. En este manual se explican las características de la placa, se dan ejemplos de uso de sus componentes y de la aplicación de módulos externos como sensores de temperatura, humedad, presión y flama. Se incluyen diagramas esquemáticos y programas de ejemplo.

# ÍNDICE

## NOMENCLATURA

| Símbolo | Significado |
|---|---|
| ADC | Por las siglas en inglés de Analog to Digital Converter |
| CD | Corriente directa |
| CLK | *Clock* |
| DASA | Dispositivo Automatizado de Sensores para Arduino |
| FLASH | Tipo de memoria no volátil para almacenamiento de programas e información |
| GND | *Ground* |
| I2C | Del inglés: *Inter Integrated Circuit*. Es un protocolo de comunicación serial síncrona. Originalmente conocido como $I^2C$. |
| IDE | *Integrated Development Environment* |
| LCD | *Liquid Crystal Display* |
| LED | *Light Emitting Diode* |
| OLED | *Organic Light Emitting Diode* |
| PAPIME | Programa de Apoyo a Proyectos para Innovar y Mejorar la Educación |
| PWR | *Power* |
| RGB | *Red, Green and Blue* |
| SCK | Serial Clock, en la comunicación I2C |
| SCL | Serial Clock, en la comunicación I2C |
| SDA | Serial Data, en la comunicación I2C |
| USB | *Universal Serial Bus* |
| VCC | Línea de voltaje de alimentación positivo |
| Wi-Fi | *Wireless Fidelity* |

# INTRODUCCIÓN

El uso de la plataforma Arduino para la enseñanza de programación y electrónica ha ampliado su desarrollo, a medida que surgen nuevas tarjetas electrónicas compatibles. Algunas de estas tarjetas ya cuentan con módulos de comunicación inalámbrica (Wi-Fi o Bluetooth), lo que hace posible la interacción con teléfonos inteligentes y el envío de información a través de internet. A mediados del año 2023, fue introducida al mercado una nueva versión de la placa Arduino UNO. Se trata de la revisión 4, que consta de dos versiones, una con comunicación Wi-Fi y Bluetooth (denominada Arduino UNO R4 WiFi) y la otra sin comunicación inalámbrica (denominada Arduino UNO R4 Minima) (Ibrahim, 2023). Ambas versiones cuentan ahora con un microcontrolador de 32 bits, en lugar del chip de 8 bits de la versión anterior. Esto es una evidencia del interés de la comunidad docente y de hacedores de ampliar el desarrollo de aplicaciones para incluir las características de la comunicación inalámbrica. Sin embargo, de momento la nueva tarjeta Arduino UNO R4 WiFi , no cuenta con versiones genéricas, y la versión original cuesta alrededor de $400 pesos (junio de 2024).

Una alternativa al uso de la tarjeta Arduino es la placa ESP32 (Expressif, 2023), de la cual existen muchas versiones, pero son de un costo más accesible. Por ejemplo, la ESP32 WROOM32 NODE-MCU ESP-32S tiene un valor aproximado de $130 pesos (junio de 2025). Esta tarjeta también cuenta con un microcontrolador de 32 bits (Extensa LX6) y capacidad inalámbrica para conexión Wi-Fi y Bluetooth.

Como parte del proyecto PAPIME PE108923 “Desarrollo de material didáctico para la asignatura de Informática aplicada a la ciencia y a la industria”, se decidió diseñar y construir una placa basada en el módulo ESP32, que contara con sensores integrados y la posibilidad de conectarse a sensores y actuadores adicionales. Para su diseño se consideró la experiencia en un proyecto previo donde los autores desarrollaron la placa DASA, que se inserta en una tarjeta Arduino UNO rev3. En ese diseño se integraron en la placa una pantalla LCD, varios sensores y actuadores, y algunos *headers* para expansión. Para el nuevo diseño se contempló la utilización de una pantalla OLED y una mayor capacidad de conexión a sensores externos, para obtener una mayor flexibilidad.

En las siguientes secciones se presentan las características de la placa desarrollada, junto con ejemplos de utilización y programas muestra.

# 1. CARACTERÍSTICAS

## 1.1. Descripción General

La tarjeta DASA 2.0 contiene un módulo ESP32 modelo WROOM NODEMCU ESP-32S de 38 pines formato estrecho. Es importante mencionar que existen varios modelos de tarjetas ESP32, las cuales varían en el número de pines, y en el ancho del módulo, entre otras características. Como ejemplo de estas variantes se presentan los casos de la Figura 1, los cuales varían en el número de pines y en el ancho del módulo. El ESP32 se puede alimentar con 5 V CD mediante una conexión USB; sin embargo, cuenta con un convertidor de voltaje a 3.3 V, por lo que todas las señales se manejan en ese rango.


*Figura 1. Variaciones del módulo ESP32: a) 38 pines, ancho; b) 32 pines, ancho; c) 38 pines, angosto (Elaboración propia).*

La programación del ESP32 se puede realizar mediante el ambiente de programación IDE (*Integrated Development Environment*) del Arduino (Santos, 2023). Para poder utilizarlo es necesario instalar la biblioteca de controladores del ESP32. Ello requiere que en el IDE se abra la opción de instalar nuevas tarjetas como se indica en la Figura 2.

Primero se da clic en el ícono de tarjetas, después se busca ESP32 y se instala la biblioteca esp32 by Expressif Systems.

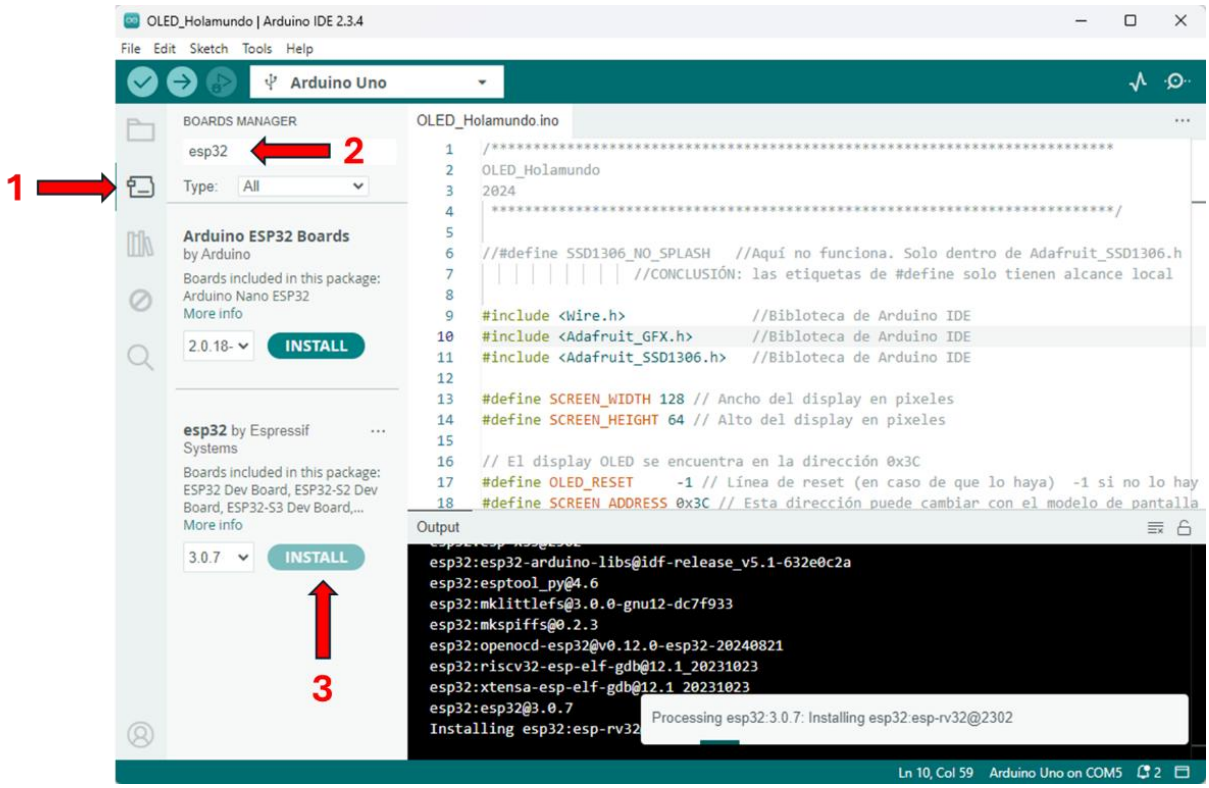



*Figura 2. Instalación de bibliotecas para el ESP32 (Elaboración propia).*

Posteriormente, y con la tarjeta conectada a la computadora mediante un cable USB, se debe buscar las tarjetas ESP32 y seleccionar la ESP32 Dev Module, y su puerto activo (en el ejemplo: COM4), como se muestra en la Figura 3. Cuando se ha establecido comunicación correctamente con la tarjeta, su nombre aparece resaltado en negritas (ver Figura 4).

Para subir el programa al ESP32, es necesario mantener presionado el botón BOOT de la ESP32. Este botón está indicado en la Figura 5. El botón deberá comenzar a presionarse cuando el IDE indique el mensaje de subiendo (*Uploading*…), y deberá permanecer presionado hasta que comience a grabarse la memoria FLASH, lo cual se indica con el porcentaje de avance, según se indica en la Figura 6.

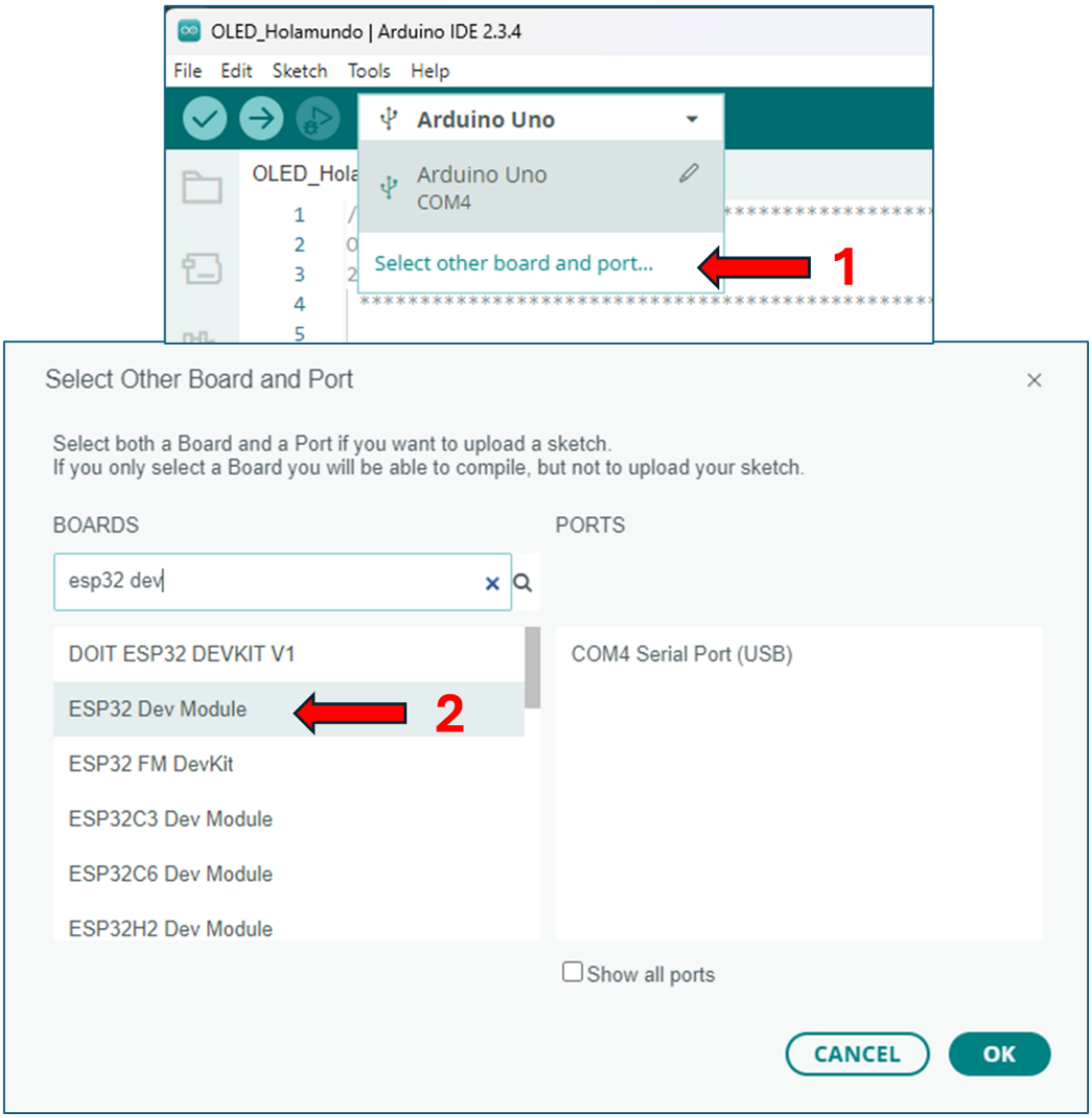



*Figura 3. Selección de nuevas tarjetas en el IDE Arduino (Elaboración propia).*


*Figura 4. Verificación de que el módulo está conectado (Elaboración propia).*


*Figura 5. Botón BOOT para subir un nuevo programa (Elaboración propia).*

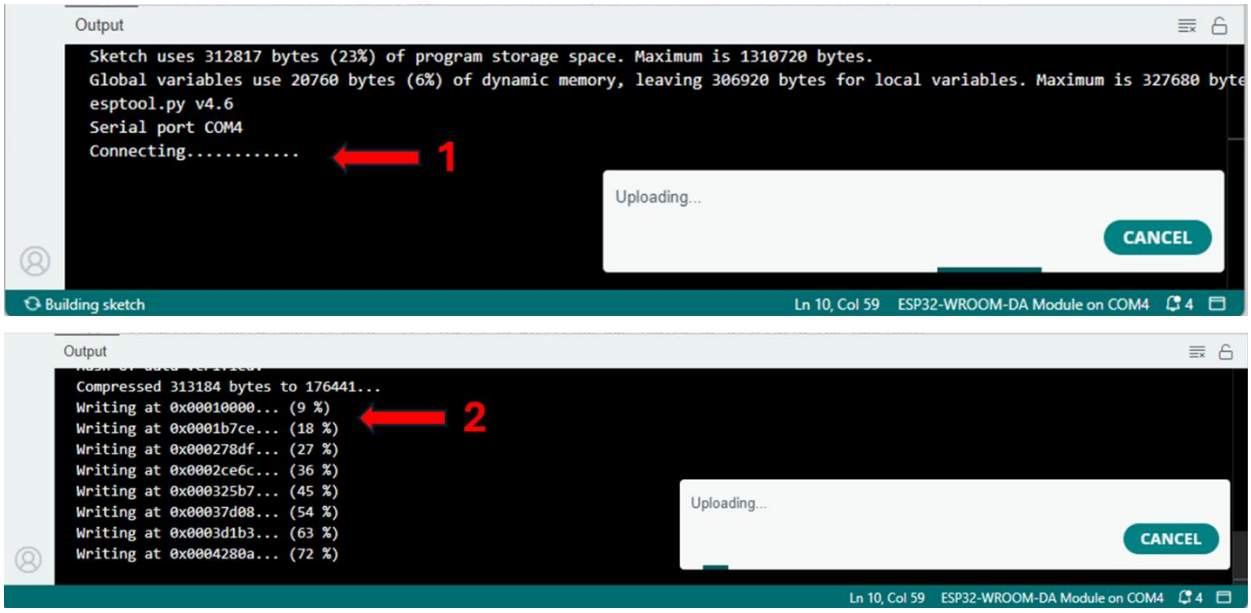



*Figura 6. Secuencia de operación del botón BOOT. 1) Comenzar a presionar. 2) Dejar de presionar (Elaboración propia).*

En la Tabla 1 se presenta la asignación de pines y funciones del ESP32. Con la letra D se indican los puertos de entrada/salidas digitales. Se usa la leyenda (ADC) para indicar aquellos pines que tienen también pueden funcionar como entradas analógicas (ADC, *Analog-to-Digital Converter*). Este convertidor analógico-digital entrega resultados de 12 bits, a diferencia de los 10 bits que entrega la tarjeta Arduino UNO R3. En azul se han marcado las líneas de entrada y salida que se utilizan en la tarjeta DASA 2.0. Se observa que no se han utilizado todas las líneas disponibles, eso es porque algunas de ellas tienen funciones asignadas y otras tienen funciones compartidas con el sistema de

inicialización y programación del microcontrolador. Para no interferir con ellas, o hacer más elaborada la programación, se optó por no utilizarlas.

*Tabla 1. Diagrama de conexiones del ESP32 (Elaboración propia).*

| Función | PIN | ESP-WROOM-32 (38 PINES) | PIN | Función |
|---|---|---|---|---|
| 3.3V | 1 |  | 38 | GND |
| EN | 2 | | 37 | D23 |
| SVP/D36_ADC | 3 | | 36 | D22/SCK |
| SVN/D39_ADC | 4 | | 35 | TX/D1 |
| D34_ADC | 5 | | 34 | RX/D3 |
| D35_ADC | 6 | | 33 | D21/SDA |
| D32_ADC | 7 | | 32 | GND |
| D33_ADC | 8 | | 31 | D19 |
| D25_ADC | 9 | | 30 | D18 |
| D26_ADC | 10 | | 29 | D5 |
| D27_ADC | 11 | | 28 | D17 |
| D14_ADC | 12 | | 27 | D16 |
| D12_ADC | 13 | | 26 | D4_ADC |
| GND | 14 | | 25 | D0_ADC |
| D13_ADC | 15 | | 24 | D2_ADC |
| SD2 | 16 | | 23 | D15_ADC |
| SD3 | 17 | | 22 | SD1 |
| CMD | 18 | | 21 | SD0 |
| 5V | 19 | | 20 | CLK |
| | | | | |



La tarjeta DASA 2.0 cuenta con sensores y actuadores soldados en la placa y 8 conectores para utilizar otros dispositivos. La lista de dispositivos montados en la placa se puede consultar en la Tabla 2 y la lista de conectores en la Tabla 3. La distribución de

los dispositivos y conectores se muestra en la Figura 7. Existen dos líneas digitales: D34 y D35 que solo se pueden utilizar como entrada digital, debido a limitaciones de la tarjeta ESP32; esta información está indicada en la serigrafía de la placa.

*Tabla 2. Dispositivos montados en la placa (Elaboración propia).*

| **Dispositivo** | **Línea de control del ESP32 v10b** |
|---|---|
| Pantalla OLED de 128 x 64 | I2C |
| Fotorresistencia | D13 (Digital) |
| 4 botones de presión | (DEC-) B1=D14<br>(DEC+) B2=D15<br>(MENÚ) B3=D16<br>(OK) B4=D17 |
| Zumbador pasivo | D23 |
| LED RGB | Red=D25 Green=D26 Blue=D27 |
| Potenciómetro 10 kΩ | D36 (ADC1) |

*Tabla 3. Conectores (Elaboración propia).*

| **Conector** | **Función** | **Línea de control del ESP32** |
|---|---|---|
| H1 | Módulo de 3 pines | D32 (ADC1) |
| H2 | Módulo de 3 pines | D33 (ADC1) |
| H3 | Módulo de 3 pines | D34 (ADC1)<br>Digital, solo como entrada |
| H4 (I2C_1) | Módulo I2C | I2C |
| H5 (I2C_2) | Módulo I2C | I2C |
| H6 | Sensor de gas MQ | D35 (ADC1) |
| H7 | Módulo de 4 pines | D4 y D39 (ADC1) |
| H8 | Atornillar sensor de temperatura DS18B20 | D18 |
| J1 | Ventilador de 5 V CD | D19 |

## 1.2. Alimentación

La tarjeta DASA 2.0 está diseñada para ser alimentada mediante un cable USB (5 V CD) que se inserta en el módulo ESP32, como se muestra en la Figura 8. Dependiendo de la versión del ESP32, esta entrada podrá ser de tipo microUSB o USB-C. Este cable también sirve para programar el módulo ESP32 desde una computadora personal. Alternativamente, la DASA 2.0 se puede alimentar mediante un eliminador de baterías de 9 V CD conectado al *jack* PWR1, como se puede observar en la Figura 9.

**ADVERTENCIA:** Sin importar si se alimenta mediante el cable USB o el eliminador de baterías, todos los dispositivos montados en la placa y **los módulos insertados en los conectores recibirán un VOLTAJE DE 3.3 V CD**. Por lo que aquellos dispositivos que requieran 5 V CD para su funcionamiento posiblemente se verán afectados o impedidos de operar.

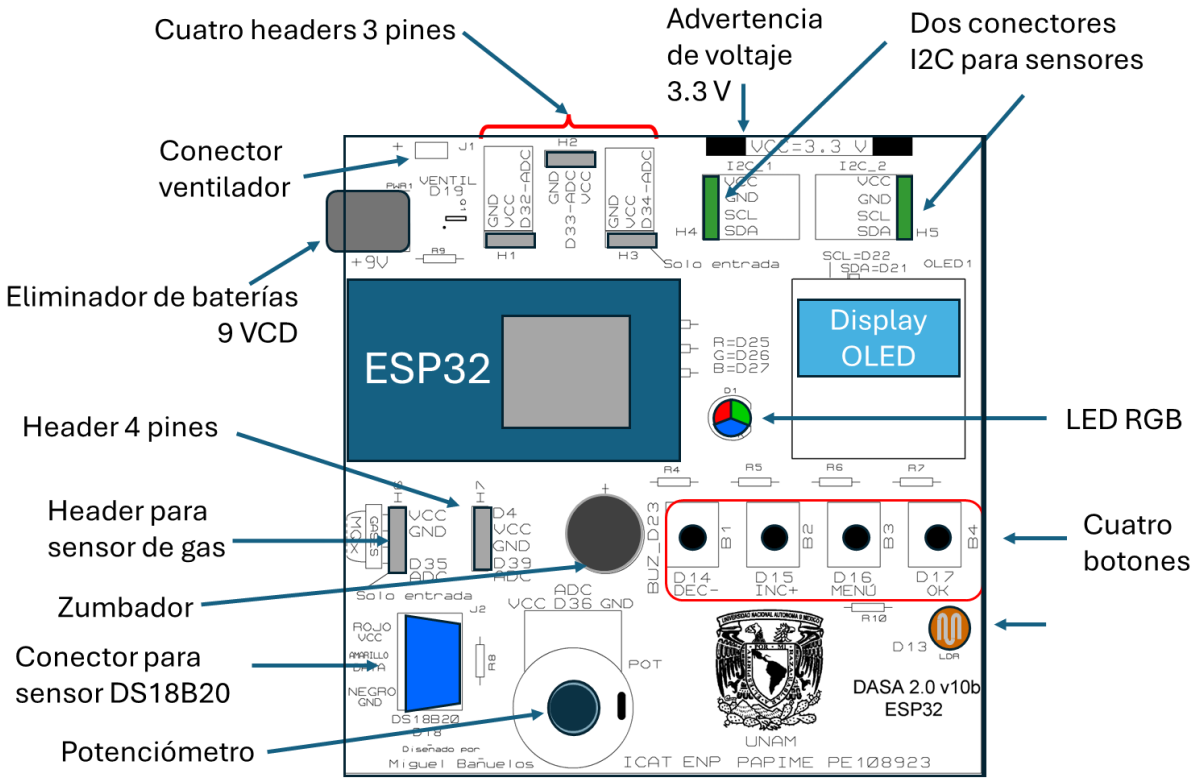



*Figura 7. Esquema de la tarjeta DASA 2.0 v10b (Elaboración propia).*

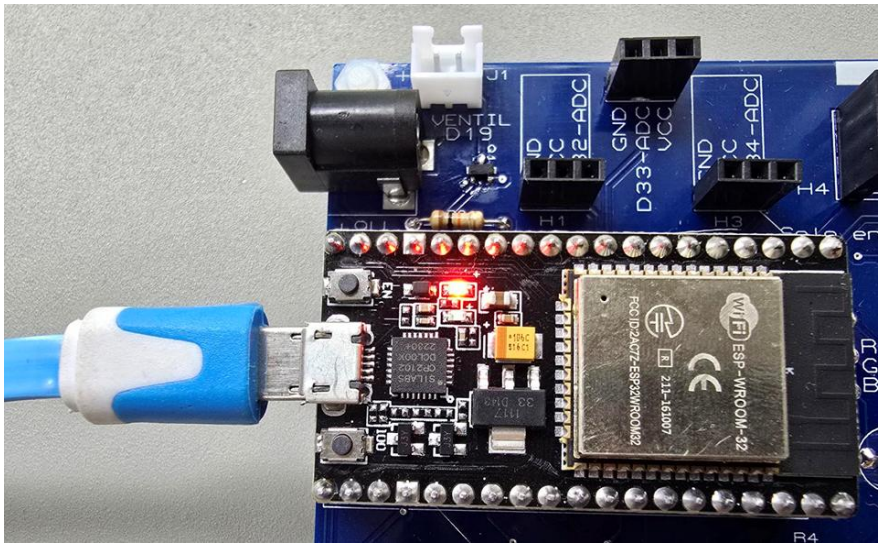



*Figura 8. Conector USB de alimentación (Elaboración propia).*

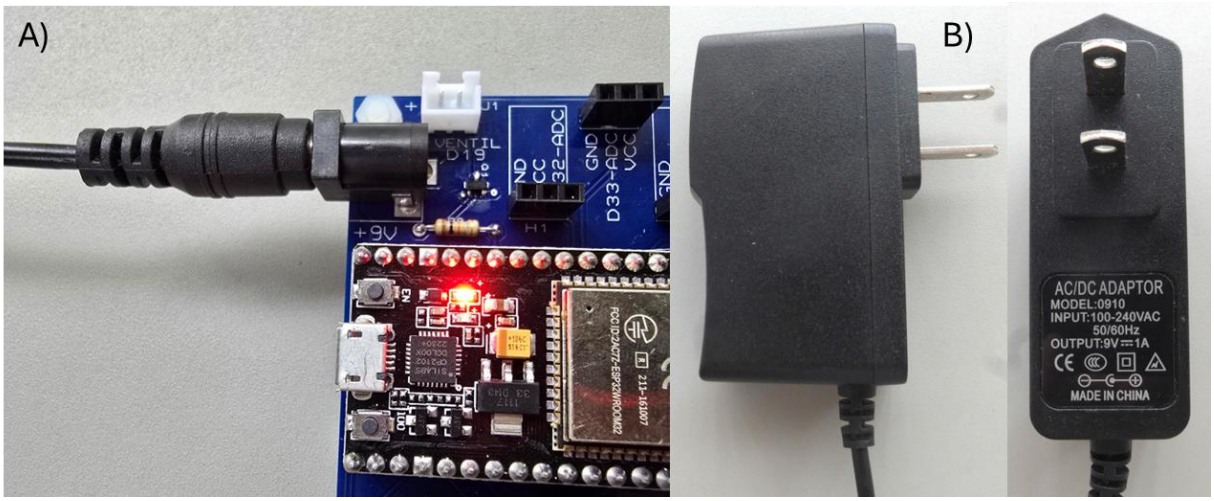



*Figura 9. Alimentación mediante eliminador de voltaje. A) Plug 5 mm. B) Eliminador 9 V CD @ 1 A (Elaboración propia).*

# 2. FUNCIONAMIENTO

## 2.1. Display OLED

La tarjeta incorpora un display OLED de 0.96 pulgadas de diagonal, de 128x64 pixeles, manejado con un controlador SSD1306 mediante el protocolo de comunicación serial I2C (Vishay, 2016). En el mercado existen varios modelos, algunos de los cuales utilizan el

protocolo serial SPI para su manejo. En la Figura 10 se puede ver la imagen de la pantalla OLED utilizada en la tarjeta DASA 2.0. Esta pantalla es del tipo que utiliza comunicación I2C y se distingue por tener cuatro líneas de conexión: GND, VCC, SCL (línea de reloj, **S**erial **CL**ock) y SDA (línea de datos, **S**erial **DA**ta). Las líneas SCL y SDA son características de la comunicación I2C. Algunos display OLED nombran SCK a la señal de reloj.

Con la comunicación serial I2C, es posible conectar varios dispositivos al mismo par de líneas de comunicación (esto es a la línea de reloj y a la de datos). Para diferenciarse entre ellos tienen asociada una dirección cada uno. En el caso de la pantalla OLED esta dirección es la 0x3C (hexadecimal), a pesar de que en la parte posterior del módulo su serigrafía indica que su dirección es la 0x78 (hexadecimal). Esto es importante de considerar ya que utilizar la dirección incorrecta impide la comunicación con la pantalla. **NOTA:** no es posible conectar otra pantalla OLED que tenga asignada la misma dirección 0x3C, debido a que provocará una colisión de la comunicación.

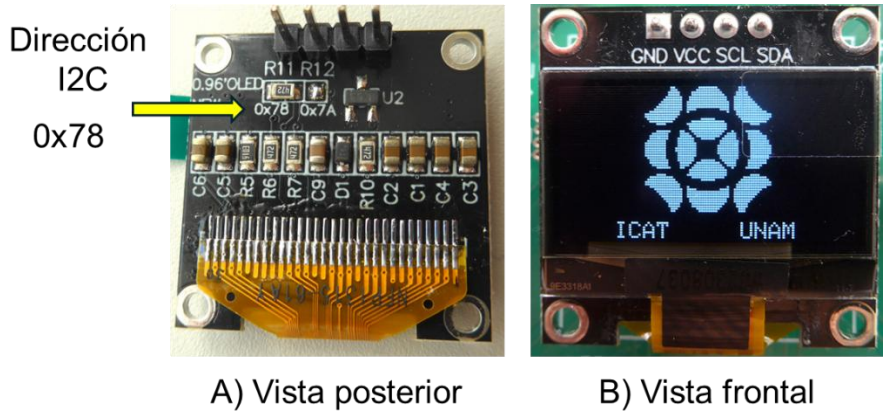



*Figura 10. Display OLED de 0.96 pulgadas (Elaboración propia).*

Como se mencionó, el *display* que utiliza el DASA 2.0 tiene el controlador SSD1306 y cuenta con 128 x 64 pixeles. Cada pixel responde a las coordenadas según se indica en la Figura 11. Para posicionar el cursor se utiliza la instrucción `display.setCursor(X, Y),` donde **X** es la columna (0…127) y **Y** es el renglón (0…63).

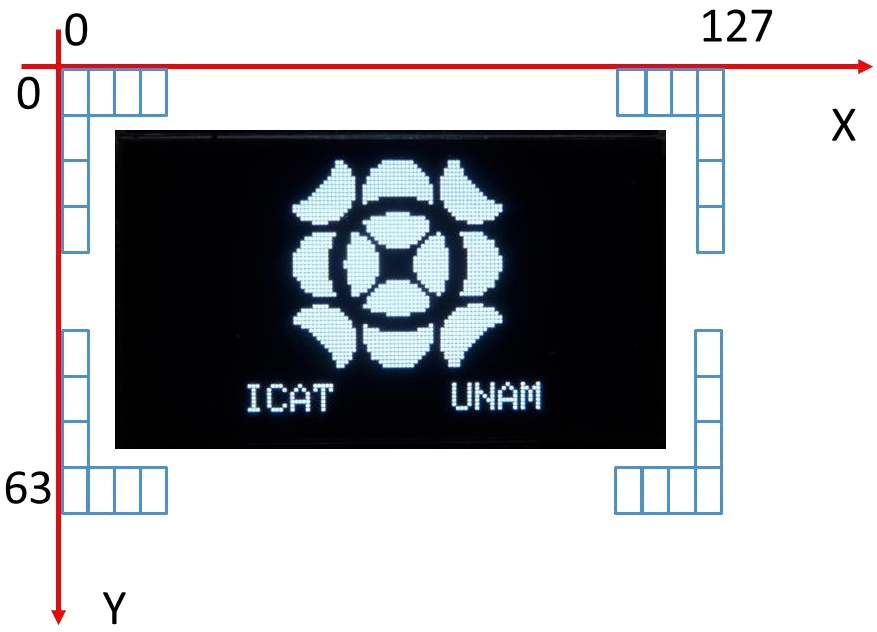



*Figura 11. Sistema de coordenadas de los pixeles en la pantalla OLED (Elaboración propia).*

En la Figura 12, se indican las conexiones de la pantalla OLED con la tarjeta ESP32, tal como se utilizan en la placa DASA 2.0. Un ejemplo de despliegue de texto en la pantalla se encuentra en el programa de la Tabla 4, el cual envía los mensajes UNAM 2025 y Hola Mundo.

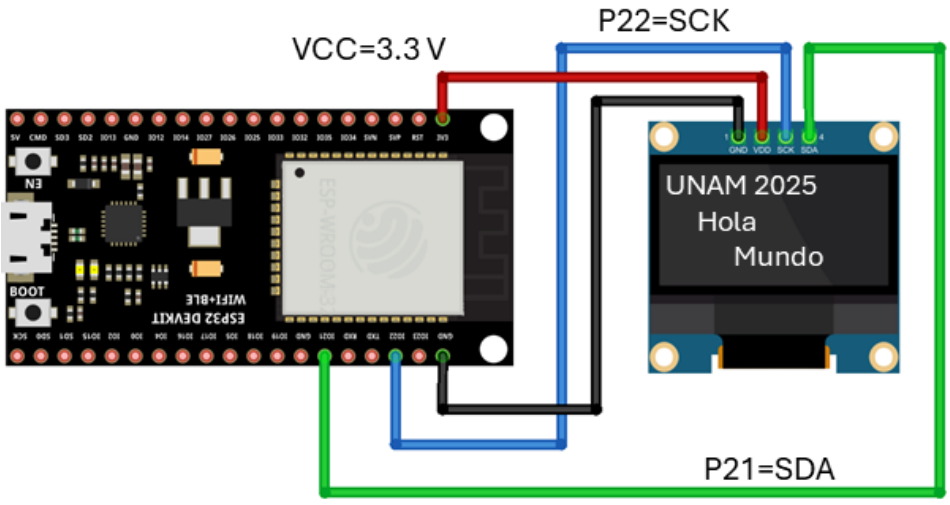



*Figura 12. Diagrama de conexiones de la pantalla OLED (Elaboración propia).*

*Tabla 4. Ejemplo del manejo de la pantalla OLED (Elaboración propia).*

```
/*************************************************************************
OLED_Holamundo
2024
 *************************************************************************/

#include <Adafruit_GFX.h>      //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>  //Biblioteca de Arduino IDE by Adafruit
#define SCREEN_WIDTH 128 // Ancho del display en pixeles
#define SCREEN_HEIGHT 64 // Alto del display en pixeles

// El display OLED se encuentra en la dirección 0x3C
#define OLED_RESET     -1 // Línea de reset (en caso de que lo haya)  -1 si no lo hay
#define SCREEN_ADDRESS 0x3C // Esta dirección puede cambiar
//#define SCREEN_ADDRESS 0x78   // Esta es la dirección que indica la serigrafía
(no funciona)

// Configuración de la pantalla OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

void setup() {
  Serial.begin(9600);   //Inicializa comunicación serial a la PC
  // SSD1306_SWITCHCAPVCC = La OLED en modo compatible con VCC = 5 V
  // SSD1306_SWITCHCAPVCC = Genera VCC = 3.3V internamente
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 error de comunicación"));
    for(;;); // Loop infinito, aquí para el programa
  }

  display.clearDisplay();   // Limpia el búfer del display//EVita el logo de Adafruit
  display.display();    // Actualiza la memoria del display a cero imagen
    display.setTextSize(2);   // Selecciona el tamaño de letra
  display.setTextColor(WHITE);  //Selecciona WHITE o BLACK
  display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
  display.print("UNAM 2025"); // Envía letrero a la memoria de la OLED
  display.setCursor(0,20);    // Nueva posición del cursor
  display.print("  Hola");    // Envía letrero a la memoria de la OLED
  display.setCursor(0,40);    // Nueva posición del cursor
  display.print("    Mundo"); // Envía letrero a la memoria de la OLED
  display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
}

void loop() {
}
```

## 2.2. Potenciómetro

La placa DASA 2.0 cuenta con un potenciómetro de 10 kΩ, el cual está conectado a la línea **D36** del ESP32, que también sirve como entrada analógica. El potenciómetro puede ser utilizado para visualizar el comportamiento del convertidor analógico-digital, o para desarrollar prácticas de control de intensidad del LED RGB, entre otras funciones. Para leer el voltaje del potenciómetro se debe utilizar el canal **36** en el comando del convertidor analógico-digital. En la Figura 13, se indican las conexiones para utilizar la pantalla OLED y un potenciómetro, de acuerdo con el diseño de la tarjeta DASA 2.0. Por otro lado, en la Tabla 5 se muestra un programa ejemplo que hace un promedio de 100 lecturas del convertidor y despliega el resultado en la pantalla OLED.

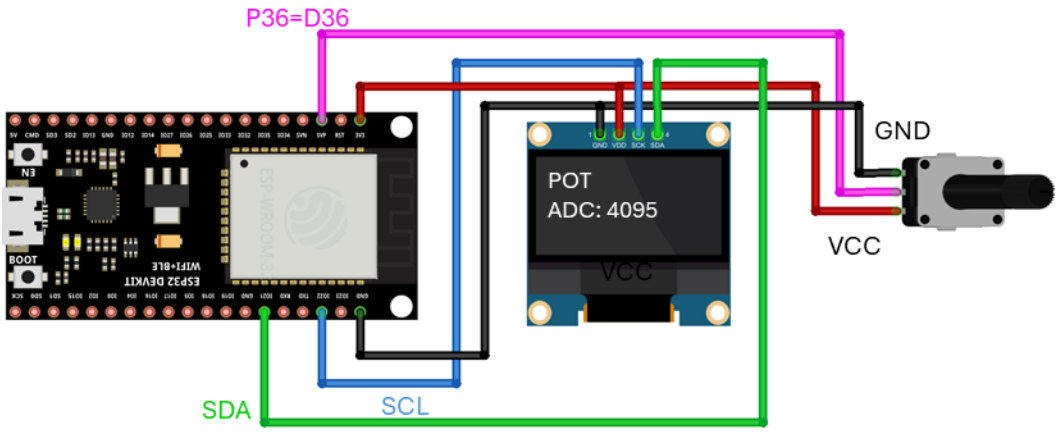



*Figura 13.Diagrama de conexiones de una pantalla OLED y un potenciómetro (Elaboración propia).*

*Tabla 5. Programa ejemplo de lectura analógica del potenciómetro (Elaboración propia).*

```
/*************************************************************************
OLED_POT
Programa que muestra en la pantalla el valor de la conversión A/D al girar
el potenciómetro
2024
*************************************************************************/

#include <Adafruit_GFX.h>      //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>  //Biblioteca de Arduino IDE by Adafruit

#define SCREEN_WIDTH 128 // Ancho del display en pixeles
#define SCREEN_HEIGHT 64 // Alto del display en pixeles

// El display OLED se encuentra en la dirección 0x3C
#define OLED_RESET     -1 // Línea de reset (en caso de que lo haya)  -1 si no lo hay
#define SCREEN_ADDRESS 0x3C // Esta dirección puede cambiar con el modelo de
pantalla OLED
//#define SCREEN_ADDRESS 0x78   // Esta es la dirección que indica la serigrafía
(no funciona)

// Configuración de la pantalla OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

/*****************  DECLARACIÓN DE VARIABLES  ************************/
const int pinPotenciometro = 36;    // D36 pin del potenciómetro
int valPot = 0;    // variable para guardar el valor del potenciometro
long suma=0;       // Variable auxiliar para calcular el promedio
int promedio=0;  // Resultado del cálculo del promedio
/************** INICIALIZACIÓN  *************************************/
void setup() {
  Serial.begin(9600);   //Inicializa comunicación serial a la PC

  // SSD1306_SWITCHCAPVCC = La OLED en modo compatible con VCC = 5 V

  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 error de comunicación"));
    for(;;); // Loop infinito, aquí para el programa
  }

  display.clearDisplay();   // Limpia el búfer del display//EVita el logo de Adafruit
  display.display();    // Actualiza la memoria del display a cero imagen

  display.setTextSize(2);   // Selecciona el tamaño de letra
```

```
  display.setTextColor(WHITE);  //Selecciona WHITE o BLACK
}
/**************** PROGRAMA PRINCIPAL  ********************************/
void loop() {
  suma=0;
  for (int i=1;i<=100;i++){
    suma=analogRead(pinPotenciometro)+suma;  // 100 lecturas del convertidor A/D
  }
  valPot=suma/100;      // Calcula el promedio

  display.clearDisplay();   // Limpia el búfer del display
  display.setCursor(0,0);   // X, Y => HOR, VER
  display.print("POT");
  display.setCursor(0,20);
  display.print("ADC:");
  display.print(valPot);
  display.display();
}
```

## 2.3. Zumbador

Otro de los actuadores que se incluyen en la placa es un zumbador pasivo (Handson, 2021), el cual está conectado a la línea digital D23 del ESP32. Se puede emplear para producir un tono mediante la función **tone()**. El diagrama de conexión utilizado por la tarjeta DASA 2.0 se muestra en la Figura 14. Como ejemplo de su utilización se proporciona el código de la Tabla 6, que produce un tono de dos segundos de duración. No es un tono puro, debido a que se produce mediante una señal cuadrada en lugar de senoidal.

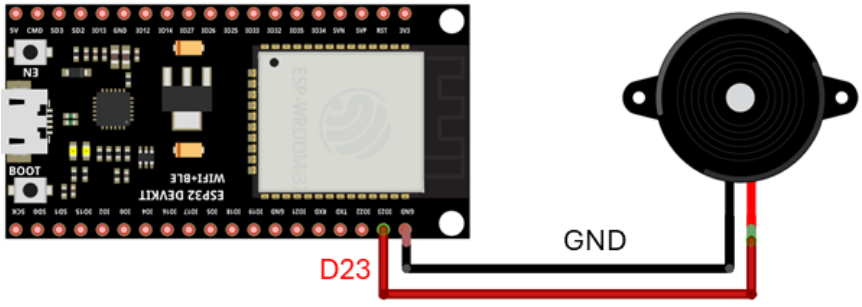



*Figura 14. Conexión de un zumbador pasivo (Elaboración propia).*

*Tabla 6. Ejemplo de utilización del zumbador (Elaboración propia).*

```
/* Buzzer23
 * Programa que produce un tono mediante un
 * zumbador pasivo conectado al canal 23
 * ESP32S
 */

#define buzzer 23  // GPIO23 pin del zumbador
void setup() {
  // Se configura el canal A1 como salida digital
  pinMode(buzzer, OUTPUT);
  // Se genera un tono de 440 Hz (nota musical LA 4)
  // El tono dura 2 segundos y se apaga
  tone(buzzer,440);
  delay(2000);
  noTone(buzzer);  // Se apaga el zumbador
}

void loop() {
}
```

## 2.4. LED RGB

El diseño cuenta también con un LED RGB de cátodo común, que puede servir para visualización de eventos, alarmas o como ejemplo de control de iluminación. Los colores se controlan mediante terminales digitales de acuerdo con la Tabla 7. El diagrama de conexiones se muestra en la Figura 15. Como ejemplo del funcionamiento se incluye el código de la Tabla 8, el cual enciende en secuencia el color rojo, verde y azul, con pausas de dos segundos.

*Tabla 7. Conexiones del LED RGB (Elaboración propia).*

| **Color** | **Pin de control** |
|---|---|
| Rojo | D25 |
| Verde | D26 |
| Azul | D27 |

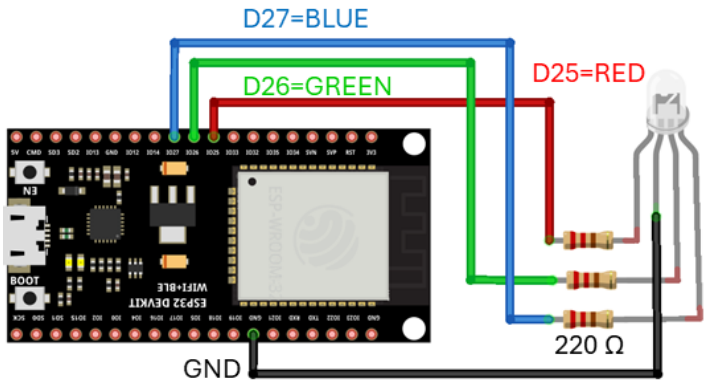



*Figura 15. Conexión de un LED RGB (Elaboración propia).*

*Tabla 8. Programa de prueba del LED RGB (Elaboración propia).*

```
/*   RGB_test
Prueba el LED RGB
ESP32 DASA v2.10b
2024
*/

#define RGB_R 25    //LED Rojo en D25
#define RGB_G 26    //LED Verde en D26
#define RGB_B 27     //LED Azul en D27

void setup() {
  pinMode(RGB_B, OUTPUT);  //Declara PIN como salida
  pinMode(RGB_G, OUTPUT);
  pinMode(RGB_R, OUTPUT);
}

void loop() {
    digitalWrite(RGB_R, HIGH);  //LED ROJO encendido
    digitalWrite(RGB_G, LOW);
    digitalWrite(RGB_B, LOW);
  delay(2000);
    digitalWrite(RGB_G, HIGH);  //LED VERDE encendido
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_B, LOW);
  delay(2000);
    digitalWrite(RGB_B, HIGH);  //LED AZUL encendido
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_G, LOW);
  delay(2000);
}
```

## 2.5. Fotorresistencia

La fotorresistencia es un dispositivo sensible a la luz cuya resistencia en la oscuridad es máxima y su resistencia va disminuyendo de manera no lineal a medida que se expone a mayores intensidades de luz. Para detectar su variación se utiliza conectada en serie con una resistencia, en este caso de 4.7 kΩ. El punto medio de esa conexión está conectado a la entrada D13 del módulo ESP32 como se señala en la Figura 16. Como ejemplo de programación se presenta el listado de la Tabla 9, donde al disminuir la iluminación se apaga el LED RGB Rojo. Existen diversos modelos de fotorresistencia, los cuales requerirán un valor distinto de resistencia conectada en serie. La placa DASA 2.0 v10b utiliza fotorresistencias modelo 5516.

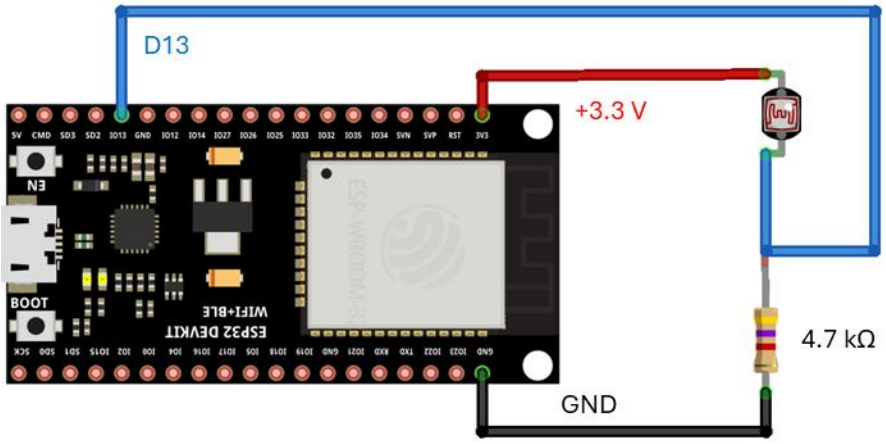



*Figura 16. Conexión de una fotorresistencia en la tarjeta DASA 2.0 (Elaboración propia).*

*Tabla 9. Programa de prueba de la fotorresistencia (Elaboración propia).*

```
/* Fotorres_LED
 * Programa que enciende un LED dependiendo de la
 * iluminación que recibe la fotoresistencia
 * DASA 2.10b ESP32
 * 2025  */

#define fotor 13    // Fotorresistencia en D13
#define LED 14    //LED Rojo en D14

void setup() {
   pinMode(fotor, INPUT);  // Declara PIN como entrada digital
```

```
    pinMode(LED, OUTPUT);  //Declara PIN como salida digital
}

void loop() {
  if(digitalRead(fotor))  //Ajustar el valor 1500 según la iluminación presente
    digitalWrite(LED, HIGH);  //LED RGB encendido
  else
    digitalWrite(LED, LOW); // LED se apaga
}
```

## 2.6. Botones

En la placa DASA 2.0 se incluyen cuatro botones de presión (*push button*) que pueden ser utilizados para programar menús de configuración u otro tipo de actividades. Los botones están conectados al módulo ESP32 de acuerdo con la Tabla 10. El diagrama de conexiones de los botones se muestra en la Figura 17. Al presionar los botones se enviará un valor de voltaje alto (HIGH) a la terminal del ESP32. Mientras no se presionen se leerán como voltaje bajo (LOW).

*Tabla 10. Conexión de los botones (Elaboración propia).*

| **Botón** | **Función pre-asignada** | **PIN** |
|---|---|---|
| B1 | DEC- (decrementar) | D14 |
| B2 | INC+ (incrementar) | D15 |
| B3 | MENÚ | D16 |
| B4 | OK | D17 |

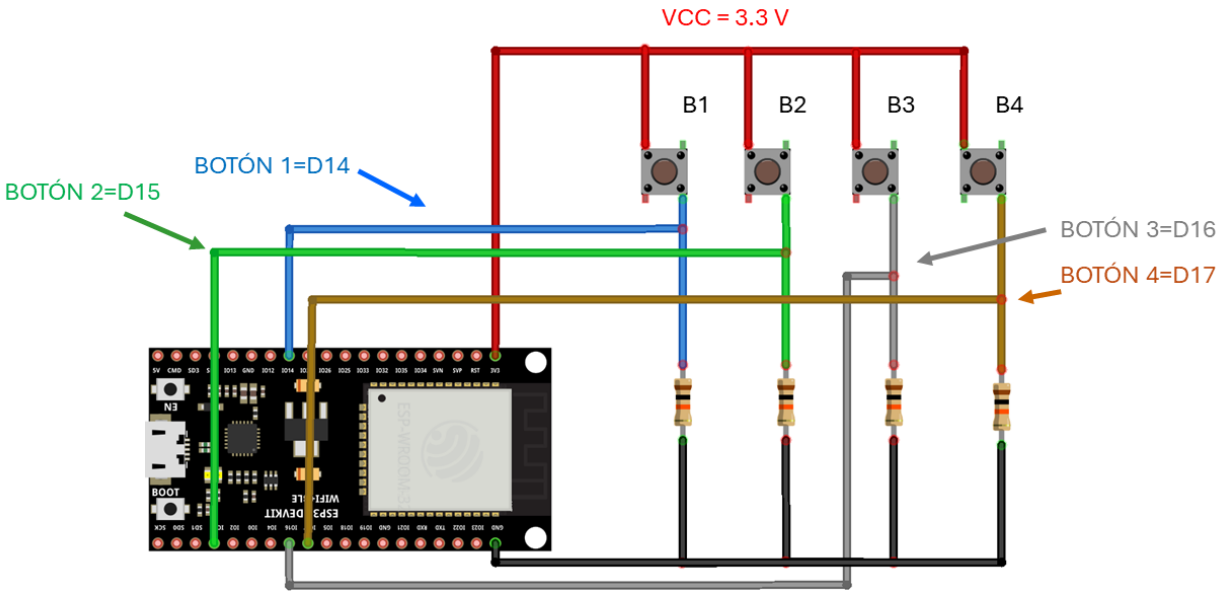



*Figura 17. Diagrama de conexión de los botones (Elaboración propia).*

Un programa de ejemplo de operación de los botones de presión se incluye en la Tabla 11. Al presionar alguno de los botones B1 a B3 se encenderá un color del LED RGB, y si se presiona el botón B4 se apagará cualquier LED que esté encendido.

*Tabla 11. Código de ejemplo de operación de los botones (Elaboración propia).*

```
/*   RGB_boton
Prueba el LED RGB y los botones
DASA v2.10b   ESP32
2025
*/
#define boton1    14  //Botón 1 en D14
#define boton2    15  //Botón 2 en D15
#define boton3    16  //Botón 3 en D16
#define boton4    17  //Botón 4 en D17
#define RGB_R     25  //LED RGB Rojo en D25
#define RGB_G     26  //LED RGB Verde en D26
#define RGB_B     27  //LED RGB Azul en D27

void setup() {
  pinMode(boton1, INPUT); //Botones como entradas
  pinMode(boton2, INPUT);
  pinMode(boton3, INPUT);
  pinMode(boton4, INPUT);
  pinMode(RGB_B, OUTPUT); //LED RGB como salidas
```

```
  pinMode(RGB_G, OUTPUT);
  pinMode(RGB_R, OUTPUT);
}

void loop() {
  if(digitalRead(boton1)){  //Si se presiona Botón 1 enciende ROJO
    digitalWrite(RGB_R, HIGH);
    digitalWrite(RGB_G, LOW);
    digitalWrite(RGB_B, LOW);
  }

  if(digitalRead(boton2)){  //Si se presiona Botón 2 enciende VERDE
    digitalWrite(RGB_G, HIGH);
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_B, LOW);
  }

  if(digitalRead(boton3)){  //Si se presiona Botón 3 enciende AZUL
    digitalWrite(RGB_B, HIGH);
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_G, LOW);
  }
  if(digitalRead(boton4)){  //Si se presiona Botón 4 apaga TODOS
    digitalWrite(RGB_B, LOW);
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_G, LOW);
  }
}
```

## 2.7. Sensor de temperatura DS18B20

Como ejemplo de operación de un sensor de temperatura, se puede utilizar el DS18B20 (Maxim, 2019), el cual es un circuito integrado que incluye un convertidor analógico-digital y envía el dato digital de la temperatura mediante un protocolo de comunicación serial denominado 1-wire. La tarjeta DASA 2.0 admite la inserción directa de un módulo comercial de tres pines (ver sección 2.10), o un sensor con terminales de cable (en el borne azul J2). En esta sección se cubrirá este último caso. En la Figura 18, se muestra el diagrama de conexiones del sensor. En este caso la herramienta de dibujo de diagramas no cuenta con una representación de la versión cableada del sensor, por lo

que se utilizó la versión circuito integrado. En la Tabla 12 se presenta un programa que lee la temperatura medida por el sensor DS18B20 (conectado en el borne) y muestra el resultado en la pantalla OLED. Adicionalmente envía los datos por puerto serial a la PC.

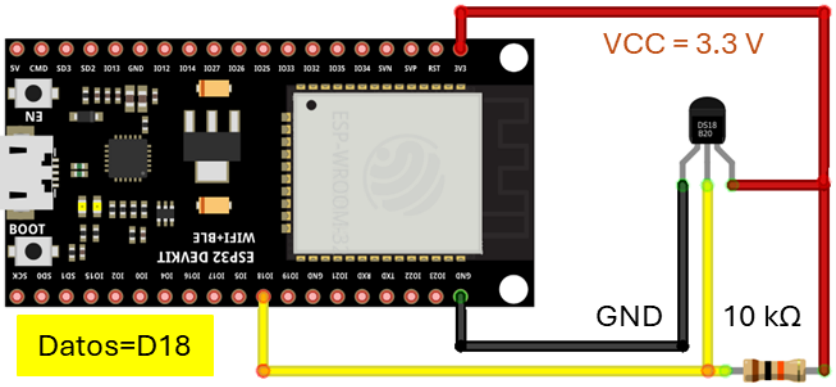



*Figura 18. Diagrama de conexiones de un sensor de temperatura DS18B20 (Elaboración propia).*

En la Figura 19, se muestra un detalle de la conexión del sensor de temperatura DS18B20 al borne de la placa DASA 2.0.

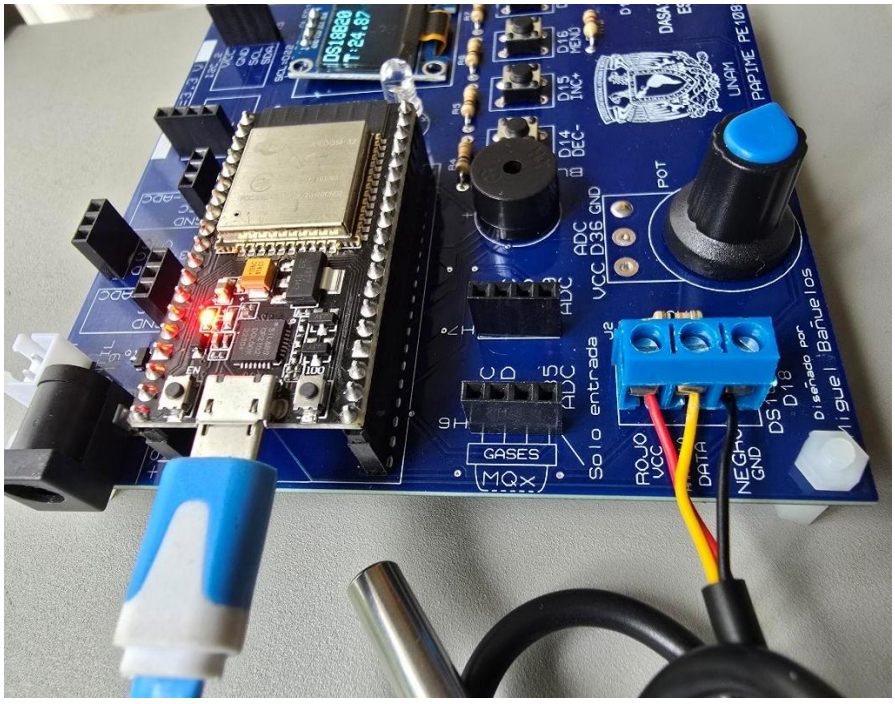



*Figura 19. Imagen de la conexión del sensor DS18B20 al borne (Elaboración propia).*

*Tabla 12. Programa para el sensor DS18B20 (Elaboración propia).*

```
/*  OLED_DS18B20_D18
Lee el sensor y despliega la temperatura en OLED
DASA v2.0 ESP32
2024
*/
#include <Adafruit_GFX.h>      //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>  //Biblioteca de Arduino IDE by Adafruit

#include <OneWire.h>          //Biblioteca de Arduino IDE by Jim Studt et al v2.3.8
#include <DallasTemperature.h>  //Bibioteca de Arduino IDE by Miles Burton v4.0.3

// Se seleciona el pin para lectura del sensor
#define ONE_WIRE_BUS 18  //Datos en D16
OneWire oneWire(ONE_WIRE_BUS);        //Para manejar el bus 1-WIRE
DallasTemperature ds18b20(&oneWire);  //Para manejar al sensor DS18B20

#define SCREEN_WIDTH 128 // Ancho de la pantalla OLED en pixeles
#define SCREEN_HEIGHT 64 // Altura de la pantalla OLED en pixeles

// Declaración para un display OLED SSD1306 conexión I2C
#define OLED_RESET     -1 // Pin de RESET. (-1 cuando el display no cuenta con
esa conexión)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
float t = 0;    //temperatura

void setup()
{
  ds18b20.begin();
  Serial.begin(9600);   //Inicializa comunicación serial con la PC
  // El display trabaja en la dirección a 0x3C
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { //
    Serial.println(F("no se detectó SSD1306"));
    for(;;); // Si no se detecta, loop infinito
  }
  display.clearDisplay();   // Limpia el búfer del display y borra logo Adafruit
  display.display();  //Actualiza el búfer del display
  display.setTextSize(2);
  display.setTextColor(WHITE);
}

void loop()
{
  ds18b20.requestTemperatures();
```

```
  t = ds18b20.getTempCByIndex(0);
  display.clearDisplay();   // Limpia el búfer del display
  display.setCursor(0,0);   // X, Y => HOR, VER
  display.print("DS18B20");
  display.setCursor(0,20);
  display.print("T:");
  display.print(t);

  display.print(" ");
  display.print(char(0xF7));   //símbolo de grado SSD1306
  display.print("C");
  display.display();

  Serial.println(t);  //Envía la temperatura a la PC
  //delay(1000); //No es necesario añadir un retardo porque cada conversión toma 750
ms
}
```

## 2.8. Sensor de gas MQ

Es posible utilizar de manera directa un sensor de gas tipo MQ-2 (Hanwei, 2021), al insertarlo en el receptáculo H6. Este tipo de sensores está diseñado para operar con un voltaje de alimentación de 5 V CD. La tarjeta DASA 2.0 le suministra un voltaje de 3.3 V CD por compatibilidad con los intervalos de voltaje del módulo ESP32. Esto puede hacer que las mediciones que entregue el sensor tengan una menor exactitud y requieran calibración; sin embargo, se pueden utilizar para detectar umbrales que indiquen un exceso de gas y con ello disparar una señal de alarma. El diagrama de conexión se presenta en la Figura 20. La señal analógica de salida del sensor se conecta al canal D35 del ESP32. En la Figura 21 se presenta una imagen de la inserción del sensor en el conector de la tarjeta DASA 2.0. Este conector también es compatible con otros sensores de gas de la familia MQ-n. Un programa muestra del funcionamiento del sensor se presenta en la Tabla 13.

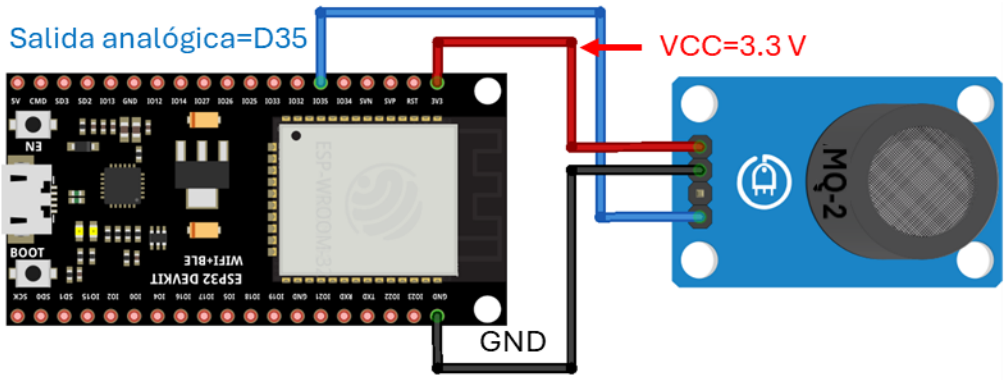



*Figura 20. Conexión de un sensor de gas MQ-2 (Elaboración propia).*

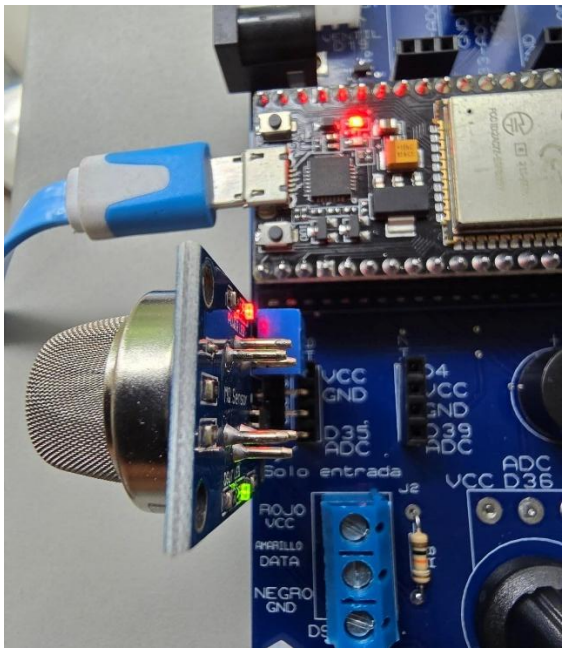



*Figura 21. Conexión de un sensor MQ-2 a la tarjeta DASA 2.0 (Elaboración propia).*

*Tabla 13. Programa ejemplo para el sensor de gas MQ-2 (Elaboración propia).*

```
/*  OLED_MQ2
 * Programa basado en
 * Arduino IDE ejemplo MQUnifiedsensor Library - reading an MQ2
 * DASA v2.0  ESP32      2024   */

#include <Adafruit_GFX.h>       //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>   //Biblioteca de Arduino IDE by Adafruit

#define pinMQ 35    //Canal analógico del sensor de gas
```

```
#define SCREEN_WIDTH 128 // Ancho de la pantalla OLED en pixeles
#define SCREEN_HEIGHT 64 // Altura de la pantalla OLED en pixeles

// Declaración para un display OLED SSD1306 conexión I2C
#define OLED_RESET     -1 // Pin de RESET. (-1 cuando el display no cuenta con
esa conexión)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

long suma;      // Para calcular el promedio
float RS_air; // Valor de RS en aire limpio
float R0; // Valor de R0 via H2
float sensor_volt;
float RS_gas; // Get value of RS in a GAS
float ratio; // Get ratio RS_GAS/RS_air
float sensorValue;

void setup() {
    Serial.begin(9600);
  // SSD1306_SWITCHCAPVCC = genera internamente el voltaje del display de 3.3 V
  // El display trabaja en la dirección a 0x3C
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("no se detectó SSD1306"));
    for(;;); // Si no se detecta, loop infinito
  }
  display.clearDisplay();   // Limpia el búfer del display y borra logo Adafruit
  display.display();  //
  display.setTextSize(2);
  display.setTextColor(WHITE);
}

void loop() {
/*--- Calcula el promedio de 100 lecturas ---*/
  suma = 0;
  for(int x = 0 ; x < 100 ; x++){
    suma = suma + analogRead(pinMQ);
  }
  sensorValue = suma/100.0;
  sensor_volt = sensorValue/4096*3.3;

// Comentar las líneas siguientes según corresponda
  RS_air = (5.0-sensor_volt)/sensor_volt; // Para calibrar en aire limpio
//  RS_gas = (5.0-sensor_volt)/sensor_volt; // Para hacer la medición
  R0 = RS_air/10.0; // La proporción RS/R0 es 10 en aire limpio // Para calibrar
/*-Reemplace el nombre "R0" con el valor de R0 en el demo First Test -*/
```

```
  ratio = RS_gas/R0; // ratio = RS/R0
  display.clearDisplay();   // Limpia el búfer del display
  display.setCursor(0,0);   // X, Y => HOR, VER
  display.print("  MQ-2");
  display.setCursor(0,20);
  display.print("V=");
  display.print(sensor_volt);
  display.print(" V");
  display.setCursor(0,40);
// Para calibración
  display.print("R0=");
  display.print(R0);
  display.print("");
// Para medición (requiere calibración)
  // display.print("Rs/R0=");
  // display.print(ratio);
  // display.print("");
  display.display();
  delay(1000);
}
```

## 2.9. Módulos I2C

Existen en el mercado algunos sensores digitales con comunicación I2C y que cuentan con 4 pines de conexión con la secuencia: VCC, GND, SCL(reloj), SDA(datos). La tarjeta DASA 2.0 dispone de dos conectores para aceptar ese tipo de módulos.

El protocolo de comunicación serial I2C o IIC (Inter-Integrated Circuit) es una forma de comunicación entre dispositivos que se encuentran a pocos centímetros entre sí. Es una variante de comunicación serial que se denomina de tipo síncrono; es decir, que se requiere una línea de comunicación dedicada exclusivamente a enviar pulsos de reloj. Los pulsos de reloj los genera la tarjeta principal del sistema (maestro) y es recibida por los demás dispositivos (esclavos). Además, se utiliza una línea para transmisión de datos en forma bidireccional; esto es, el maestro puede enviar datos a los demás dispositivos, y los dispositivos esclavos pueden enviar datos al maestro. Para que no haya colisión de los datos (dos dispositivos enviando señales al mismo tiempo), el protocolo I2C contiene bits que señalan el inicio y fin de una comunicación (por lo general, palabras de 7 u 8 bits). En la Figura 22, se muestra un ejemplo de conexión de un módulo ESP32 con tres

dispositivos que utilizan la comunicación I2C. Por simplicidad se han omitido las conexiones de VCC y GND. La señal de reloj la genera el ESP32 en la línea D22, mientras que la línea de datos es la D21. El ESP32 produce los pulsos de reloj y los envía a los tres dispositivos (OLED, SENSOR1 y SENSOR2). La comunicación la debe iniciar el dispositivo maestro, que en este caso es el ESP32, y enviará un código binario (dirección) con lo cual solo el dispositivo direccionado responderá. Es importante mencionar que no se pueden tener dos dispositivos que tengan la misma dirección asignada, ya que se ocasionará una colisión de los datos y posiblemente un cortocircuito. En la Figura 22 se indican las direcciones a las que responde cada dispositivo. El prefijo **0x** indica que se trata de valores hexadecimales. Un ejemplo de visualización de las señales de reloj y datos en un osciloscopio se puede ver en la Figura 23. En este caso, la señal de pulsos de reloj tiene una frecuencia de 100 kHz. En las siguientes subsecciones se mostrará el uso de algunos sensores que cuentan con comunicación I2C.

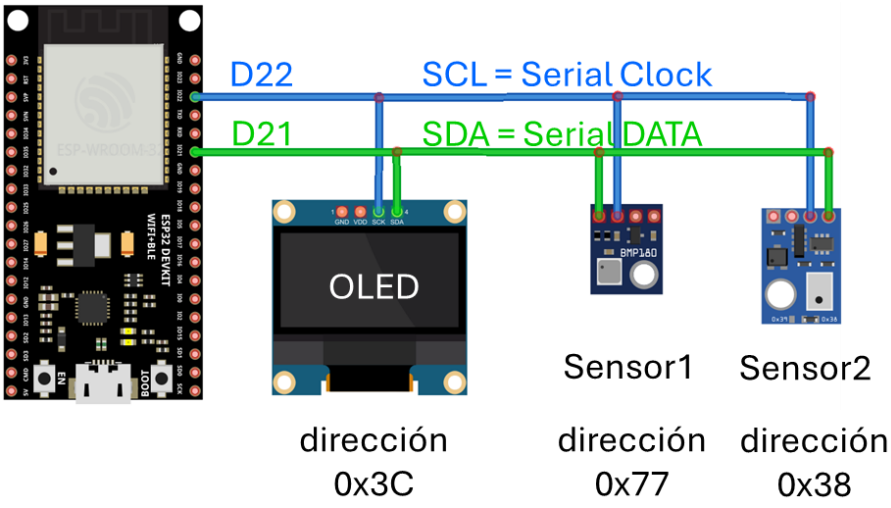



*Figura 22. Dispositivos con comunicación I2C conectados a un módulo ESP32 (Elaboración propia).*

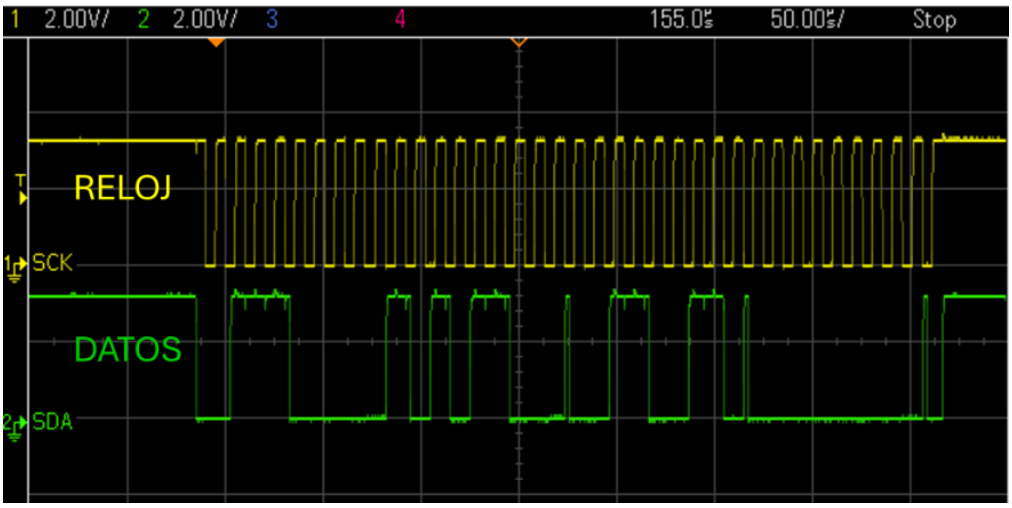



*Figura 23. Las señales de reloj y datos de una comunicación I2C vistas en un osciloscopio (Elaboración propia).*

### 2.9.1. Sensor de temperatura y humedad AHT10 y AHT20

Los sensores AHT10 y AHT20 se consiguen en pequeños módulos con conexión I2C de cuatro pines (Asair, 2018). Son sensores de temperatura y humedad con mayor exactitud que el DHT11 (Sunrom, 2012). El diagrama de conexiones se muestra en la Figura 24. Como ejemplo para el AHT10 y AHT20 se tiene el programa de la Tabla 14, donde se utiliza la pantalla OLED para mostrar la temperatura y la humedad relativa. El programa se puede utilizar también con un sensor AHT20.

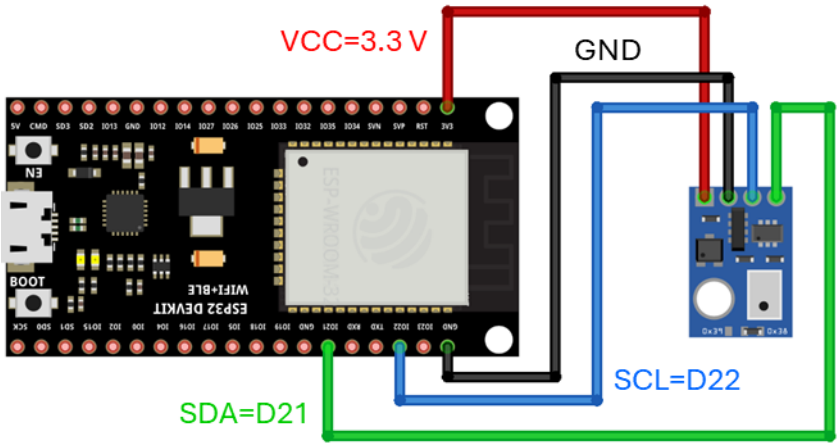



*Figura 24. Diagrama de conexión de un módulo AHT10 o AHT20 (Elaboración propia).*

Un ejemplo de conexión física a la tarjeta DASA 2.0 se puede observar en la Figura 25. Y su uso sería idéntico en el caso de un sensor AHT20.

**PRECAUCIÓN**: siempre se deberá respetar la secuencia de conexión (VIN/VCC, GND, SCL, SDA) según se indica en la serigrafía del módulo y de la tarjeta DASA 2.0. Una conexión incorrecta puede dañar al módulo o la tarjeta DASA. **NOTA: No se deberán conectar dos módulos AHTxx al mismo tiempo,** debido a que el programa no los puede diferenciar, ya que ambos responden a la misma dirección. Por la misma razón, tampoco se pueden combinar AHT10 con AHT20.


*Figura 25. Conexión de un módulo AHT10 (Elaboración propia).*

*Tabla 14. Ejemplo de lectura del sensor **AHT10 y AHT20** con despliegue en la pantalla OLED (Elaboración propia).*

```
/*  OLED_AHT20.ino
  Sensor de temperatura y humedad con despliegue en la pantalla OLED
  DASA 2.0 ESP32   2024   */
#include <Adafruit_GFX.h>      //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>  //Biblioteca de Arduino IDE by Adafruit

#include <AHTxx.h>  //https://github.com/enjoyneering/AHTxx
//Biblioteca para el manejo del sensor AHT10 y AHT20

#define SCREEN_WIDTH 128 // Ancho de la pantalla OLED en pixeles
#define SCREEN_HEIGHT 64 // Altura de la pantalla OLED en pixeles
```

```
// Declaración para un display OLED SSD1306 conexión I2C
#define OLED_RESET     -1 // Pin de RESET. (-1 cuando el display no cuenta con esa conexión)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
float ahtTemp;      //Temperatura leída
float ahtRH;        //Humedad relativa leída
AHTxx aht20(AHTXX_ADDRESS_X38, AHT2x_SENSOR); //Dirección del sensor, tipo de sensor

void setup(){
  Serial.begin(9600);   //Inicializa comunicación serial con la PC
  while (aht20.begin() != true) //for ESP-01 use aht10.begin(0, 2);
  {
    Serial.println(F("AHT20 no se conectó o error al cargar coeficiente de calibración"));
    //(F()) guarda la cadena en memoria flash y libera la memoria dinámica
    delay(5000);
  }
  Serial.println(F("AHT20 OK"));

  // SSD1306_SWITCHCAPVCC = genera internamente el voltaje del display de 3.3 V
  // El display trabaja en la dirección a 0x3C
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("no se detectó SSD1306"));
    for(;;); // Si no se detecta, loop infinito
  }
  display.clearDisplay();   // Limpia el búfer del display y borra logo Adafruit
  display.display();  //
  display.clearDisplay();   // Limpia el búfer del display
  display.setTextSize(2);
  display.setTextColor(WHITE);
}

void loop(){
  ahtTemp = aht20.readTemperature(); //Lee 6-bytes vía I2C, le toma 80 milisegundos
  ahtRH = aht20.readHumidity();
  display.clearDisplay();   // Limpia el búfer del display
  display.setCursor(0,0);   // X, Y => HOR, VER
  display.print("AHT20/20");
  display.setCursor(0,20);
  display.print("T:");
  display.print(ahtTemp);

  display.print(" ");
```

```
  display.print(char(0xF7));   //símbolo de grado SSD1306
  display.print("C");
  display.setCursor(0,40);
  display.print("HR:");
  display.print(ahtRH);
  display.print(" %");
  display.display();
  delay(1000);
}
```

### 2.9.2. Sensor de presión atmosférica y temperatura BMP180

El sensor BMP180 es un práctico detector de presión atmosférica y temperatura ambiental (Bosch, 2015). También se puede conseguir en un módulo con comunicación I2C e insertarse en alguno de los dos receptáculos I2C disponibles. La conexión del módulo se muestra en la Figura 26, y un ejemplo de los datos presentados en la pantalla OLED en la Figura 27, los cuales fueron generados con el programa de la Tabla 15.

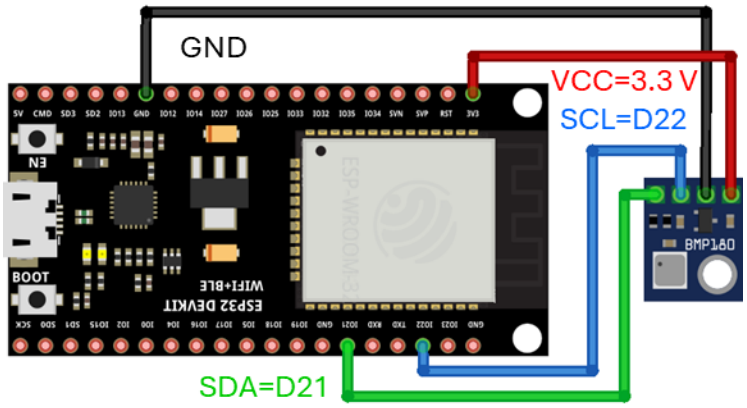



*Figura 26. Diagrama de conexión del sensor BMP180.*

**NOTA IMPORTANTE**: siempre se deberá respetar la secuencia de conexión (VIN/VCC, GND, SCL, SDA) según se indica en la serigrafía del módulo y de la tarjeta DASA 2.0.

Una conexión incorrecta puede dañar al módulo o la tarjeta DASA. **NOTA: No se deberán conectar dos módulos iguales al mismo tiempo.**

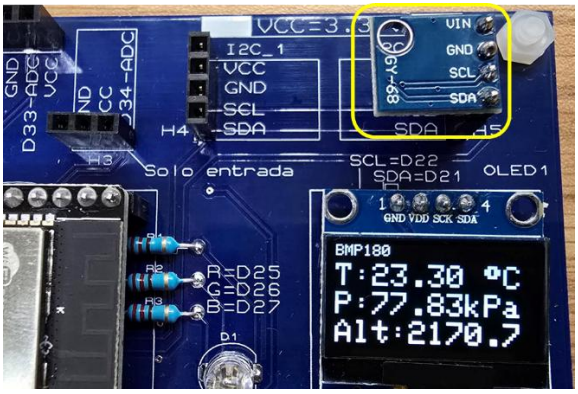



*Figura 27. Conexión del módulo de presión atmosférica BMP180 (Elaboración propia).*

*Tabla 15. Programa ejemplo del uso del sensor de presión BMP180 (Elaboración propia).*

```
#include <Adafruit_GFX.h>       //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>   //Biblioteca de Arduino IDE by Adafruit
#include <Adafruit_BMP085.h>
// Biblioteca de Arduino IDE (Adafruit BMP085 Library) by Adafruit v1.2.4 with dependancies

#define SCREEN_WIDTH 128 // Ancho de la pantalla OLED en pixeles
#define SCREEN_HEIGHT 64 // Altura de la pantalla OLED en pixeles
#define OLED_RESET     -1 // Pin de RESET. (-1 cuando el display no cuenta con esa conexión)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
float BMPTemp;      //Almacena la temperatura en grados Celsius
int BMPPresion;   //Almacena la presión en Pascales
float BMPAltitud;   //Almacena la altitud en metros
Adafruit_BMP085 bmp; //Crea el objeto bmp para manejo del sensor BMP180

void setup(){
  Serial.begin(9600);
  if (!bmp.begin()) {
  Serial.println("Could not find a valid BMP085 sensor, check wiring!");
  while (1) {}
  }
  // El display trabaja en la dirección a 0x3C
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { //
    Serial.println(F("no se detectó SSD1306"));
```

```
    for(;;); // Si no se detecta, loop infinito
  }
  display.clearDisplay();   // Limpia el búfer del display y borra logo Adafruit
  display.display();  //Actualiza la memoria del display OLED
  display.setTextSize(2);
  display.setTextColor(WHITE);
}

void loop(){
  BMPTemp = bmp.readTemperature(); // Lee la temperatura en grados Celsius
  BMPPresion = bmp.readPressure();  // Lee la presión en kPascales
  BMPAltitud = bmp.readAltitude();  // Lee la altitud en metros MSNM
  display.clearDisplay();   // Limpia el búfer del display
  display.setCursor(0,0);   // X, Y => HOR, VER
  display.setTextSize(1.5);
  display.print("BMP180");
  display.setCursor(0,12);
  display.setTextSize(2);
  display.print("T:");
  display.print(BMPTemp);
  display.print(" ");
  display.print(char(0xF7));   //símbolo de grado SSD1306
  display.print("C");
  display.setCursor(0,30);
  display.print("P:");
  display.print(BMPPresion/1000);
  display.print(" kPa");
  display.setCursor(0,48);
  display.print("Alt:");
  display.print(BMPAltitud);
  display.print(" m");
  display.display();
  delay(1000);
}
```

En el programa de la Tabla 15, la altura de lugar se estima a partir de la presión atmosférica absoluta determinada por el sensor (Bosch, 2015). Para ello se utiliza la ecuación (1)

$$altura\ [m] = 44330 * \left[1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right] \quad (1)$$

Donde p=presión atmosférica determinada por el sensor, y $p_0$=presión atmosférica al nivel del mar.

Si se desea implementar una estación meteorológica, se debe reportar la presión reducida (Conagua, 2010). Así se denomina a la presión que considera que todas las estaciones meteorológicas se encuentran al nivel del mar, lo cual facilita la comparación de datos. La presión reducida se puede calcular con la expresión (2)

$$Pr = 10^{\left[\left(\frac{1}{\frac{273.15+TBS}{0.01478*H}+0.202977}\right)+\log(\mathrm{Pe})\right]} \quad (2)$$

Donde

Pr es la presión reducida en mbar

TBS es la temperatura promedio de bulbo seco de las últimas 12 horas en °C

H es la altura a la que se encuentra la estación en m

Pe es la presión atmosférica reportada por el sensor en mbar

El código de la Tabla 15 se puede modificar para que se calcule la presión reducida, obteniéndose el programa que se muestra en la Tabla 16. Se considera la altura de 2300 metros sobre el nivel del mar (aproximación de la altura en Ciudad Universitaria). Este valor se define en la línea

```
const int altura =2300;     // Altura de la estación meteorológica en metros
```

y deberá ser ajustado al valor del lugar de la medición. Por otro lado, para el valor de la temperatura promedio de bulbo seco se consideró 20 °C.

*Tabla 16. Uso del BMP180 para una estación meteorológica (Elaboración propia).*

```
#include <Adafruit_GFX.h>         //Biblioteca Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>     //Biblioteca Arduino IDE by Adafruit
#include <Adafruit_BMP085.h>
// Biblioteca de Arduino IDE (Adafruit BMP085 Library) by Adafruit v1.2.4 with
dependancies
#define SCREEN_WIDTH 128 // Ancho de la pantalla OLED en pixeles
#define SCREEN_HEIGHT 64 // Altura de la pantalla OLED en pixeles
// Declaración para un display OLED SSD1306 conexión I2C
#define OLED_RESET     -1 // Pin de RESET. (-1 cuando el display no cuenta con
esa conexión)
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
float BMPTemp;      //Almacena la temperatura en grados Celsius
float BMPPresion;   //Almacena la presión en Pascales
float BMPAltitud;   //Almacena la altitud en metros
float BMPmbar;      //Presión en mbar
float Pr;  // Presión reducida (para estación meteorológica en mbar)
const int TBS = 20;     // Temperatura de referencia (bulbo seco en grados Celsius)
const int altura =2300;     // Altura de la estación meteorológica en metros
Adafruit_BMP085 bmp; //Crea el objeto bmp para manejo del sensor BMP180
void setup(){
  Serial.begin(9600);
  if (!bmp.begin()) {
  Serial.println("Could not find a valid BMP085 sensor, check wiring!");
  while (1) {}
  }
  // SSD1306_SWITCHCAPVCC = genera internamente el voltaje del display de 3.3 V
  // El display trabaja en la dirección a 0x3C
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("no se detectó SSD1306"));
    for(;;); // Si no se detecta, loop infinito  }
  display.clearDisplay();   // Limpia el búfer del display y borra logo Adafruit
  display.display();  //Actualiza la memoria del display OLED
  display.setTextSize(2);
  display.setTextColor(WHITE);}
void loop(){
  BMPTemp = bmp.readTemperature(); // Lee la temperatura en grados Celsius
  BMPPresion = bmp.readPressure();  // Lee la presión en Pascales
  BMPmbar = BMPPresion/100;    // Convierte a mbar
  BMPAltitud = bmp.readAltitude();  // Lee la altitud en metros MSNM
  Pr = pow(10,((1/(((273.15+TBS)/(0.01478*altura))+0.202977)+log10(BMPmbar))));
  display.clearDisplay();   // Limpia el búfer del display
  display.setCursor(0,0);   // X, Y => HOR, VER
  display.setTextSize(1.5);
  display.print("BMP180  ");
  display.print(BMPmbar);
  display.print("mbar");
  display.setCursor(0,12);
  display.setTextSize(2);
  display.print("T:");
  display.print(BMPTemp);
  display.print(" ");
  display.print(char(0xF7));   //símbolo de grado SSD1306
  display.print("C");
  display.setCursor(0,30);
  display.print("Pr:");
```

```
display.print(Pr);
display.setCursor(0,48);
display.print("Alt:");
display.print(BMPAltitud);
display.print(" m");
display.display();
delay(1000); }
```

## 2.10. Módulos de 3 pines

La tarjeta DASA 2.0 cuenta con tres receptáculos (H1, H2 y H3) que pueden aceptar sensores comerciales de tres pines, siempre y cuando tengan la secuencia de terminales: GND, VCC, SEÑAL. La señal puede ser analógica o digital, pues las líneas utilizadas (D25, D26, D27) también funcionan como entrada al convertidor analógico-digital. En la Figura 28 se muestra un módulo con un sensor de temperatura digital tipo DS18B20, el cual se puede encontrar en kits de 37 sensores para Arduino.

El programa de lectura del sensor con despliegue en la pantalla OLED es similar al de la tabla 9. La única diferencia consistiría en adecuar la línea

```
#define ONE_WIRE_BUS 18  //Datos en D18
```

para que haga referencia al pin correspondiente al receptáculo utilizado (p. ej. 32, en el header H1, como en la Figura 28).

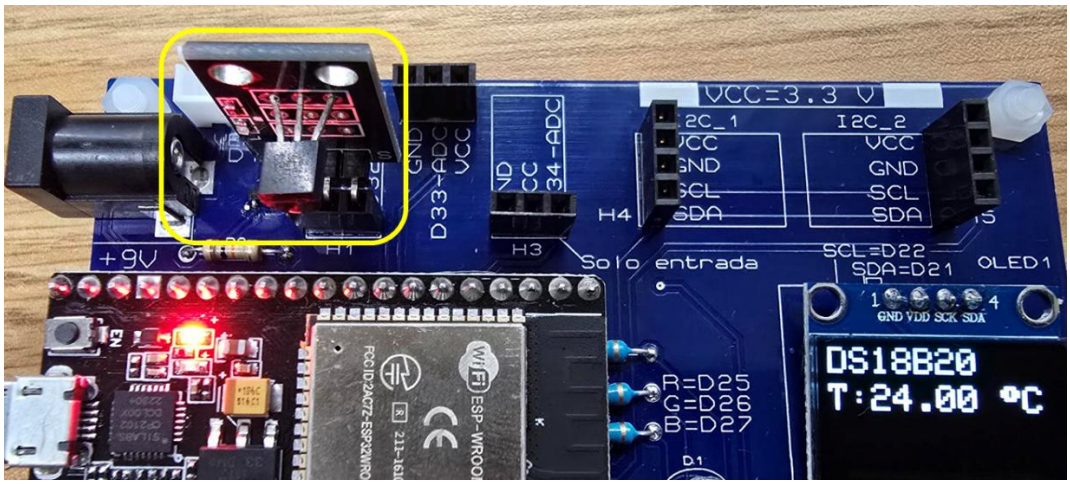



*Figura 28. Módulo DS18B20 conectado a un receptáculo de 3 pines (Elaboración propia).*

## 2.11. Módulos de 4 pines

Además de los conectores para sensores comerciales de 3 pines, la placa cuenta con un conector de 4 pines (H7) con la siguiente secuencia de conexión: salida analógica (conectada a D39_SVN), GND, VCC, y una salida digital (conectada a D4). Existen varios incluidos en los kits de 37 sensores para Arduino. Uno de estos sensores es el detector de flama KY-026, cuya conexión se muestra en la Figura 29.

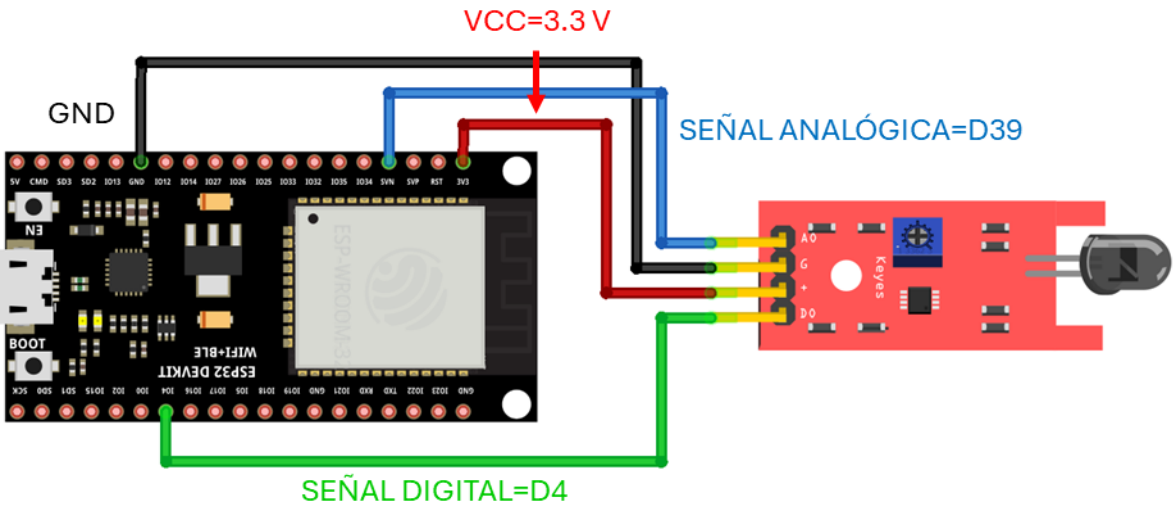



*Figura 29. Diagrama de conexiones con un sensor de flama KY-026.*

**NOTA IMPORTANTE**: siempre se deberá respetar la secuencia de conexión (Señal analógica, VCC, GND, Señal digital) según se indica en la serigrafía del módulo y de la tarjeta DASA 2.0. Una conexión incorrecta puede dañar al módulo o la tarjeta DASA.
El sensor de flama se inserta en el receptáculo H7 según se muestra en la imagen de la Figura 30. El sensor cuenta con una salida analógica y una salida digital. Sin embargo, se encontró que es más fácil trabajar con la señal analógica. La señal digital requiere del ajuste del potenciómetro azul (trimpot) y eso complica su operación.

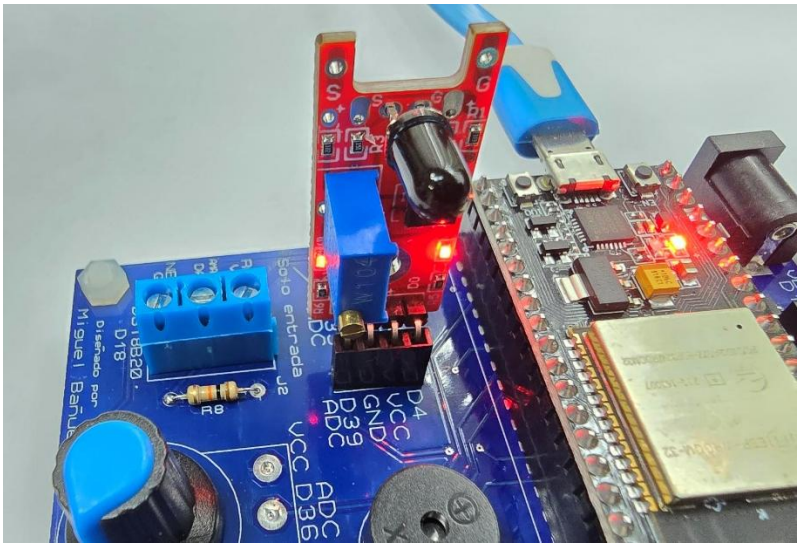



*Figura 30. Colocación del sensor de flama en la tarjeta DASA 2.0 (Elaboración propia).*

En la tabla 14, se enlista un ejemplo de programa para el uso del sensor de flama. En este, la pantalla OLED solo despliega un mensaje identificando al sensor. Cuando el fototransistor detecta una radiación infrarroja de cierta intensidad, el voltaje que envía al ESP32 disminuye. En este caso se fijó un umbral en 200, por debajo del cual se encenderá el LED ROJO indicando la presencia de la flama. Es importante mencionar que el detector también es sensible al calor ambiente o al de una persona, por lo que se puede ajustar la sensibilidad variando el parámetro en la instrucción
`if(valorFlama < 200)`.

Con el parámetro en 200, el sensor pudo detectar una flama de un encendedor a una distancia de 12 cm. NOTA: Al operar los encendedores piezoeléctricos emiten radiación electromagnética que podría interferir con la operación de la tarjeta DASA.

*Tabla 17. Ejemplo del uso del sensor de flama (Elaboración propia).*

```
/*************************************************************************
OLED_Flama
Utiliza el detector de flama KY-026
DASA 2.0 ESP32
2024
 *************************************************************************/

#include <Adafruit_GFX.h>        //Biblioteca Arduino IDE by Adafruit
#include <Adafruit_SSD1306.h>    //Biblioteca Arduino IDE by Adafruit

#define SCREEN_WIDTH 128 // Ancho del display en pixeles
#define SCREEN_HEIGHT 64 // Alto del display en pixeles

// El display OLED se encuentra en la dirección 0x3C
#define OLED_RESET     -1 // Línea de reset (en caso de que lo haya)  -1 si no lo hay
#define SCREEN_ADDRESS 0x3C // Esta dirección puede cambiar con el modelo de pantalla OLED
//#define SCREEN_ADDRESS 0x78   // Esta es la dirección que indica la serigrafía (no funciona)

// Configuración de la pantalla OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define RGB_R 17    //LED Rojo en D17
#define flamadigital 4   //Señal digital del sensor en línea D4
#define flamaanalogica 39  //Señal analógica de la flama en línea D39
int valorFlama = 0;       //Valor analógico de la flama

void setup() {
  Serial.begin(9600);   //Inicializa comunicación serial a la PC
  pinMode(flamadigital, INPUT);  // Línea como entrada
  pinMode(RGB_R, OUTPUT);   //usa LED RGB ROJO
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 error de comunicación"));
    for(;;); // Loop infinito, aquí para el programa
  }

  display.clearDisplay();   // Limpia el búfer del display//EVita el logo de Adafruit
```

```
  display.display();    // Actualiza la memoria del display a cero imagen

  display.setTextSize(2);   // Selecciona el tamaño de letra
  display.setTextColor(WHITE);  //Selecciona WHITE o BLACK
  display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
  display.print("DETECTOR"); // Envía letrero a la memoria de la OLED
  display.setCursor(0,20);    // Nueva posición del cursor
  display.print("  DE");    // Envía letrero a la memoria de la OLED
  display.setCursor(0,40);    // Nueva posición del cursor
  display.print("    FLAMA"); // Envía letrero a la memoria de la OLED
  display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
  digitalWrite(RGB_R, LOW);
}

void loop() {
  valorFlama = analogRead(flamaanalogica);
  Serial.println(valorFlama);
  if(valorFlama < 200)        //Ajustar el valor 200 dependiendo de las condiciones de
uso
    digitalWrite(RGB_R, HIGH);  //
  else
    digitalWrite(RGB_R, LOW);
  delay(300);
}
```

## 2.12. Ventilador

Como ejemplo de operación de otro actuador, la tarjeta DASA 2.0 cuenta con un conector (J1) para insertar un ventilador de 5 V. Debido a que un ventilador requiere más corriente de la que puede proporcionar un pin de salida de la tarjeta ESP32, es necesario incluir un transistor, para obtener la alimentación a través de la línea de 5 V, es decir, directamente del cable USB. El diagrama esquemático se muestra en la Figura 31 y una fotografía de la conexión en la Figura 32. Se ha incluido un potenciómetro (canal analógico 36), para poder efectuar el control de velocidad del ventilador, utilizando una señal PWM (salida D19). En la Tabla 18 se encuentra un código de ejemplo.

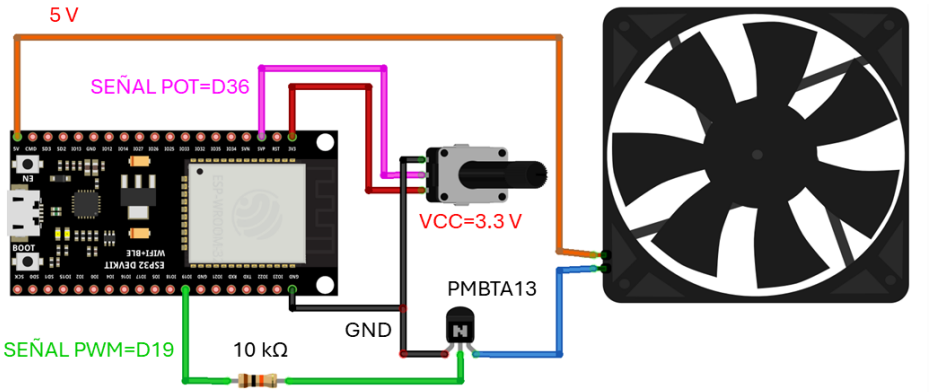



*Figura 31. Conexión de un ventilador y un potenciómetro.*

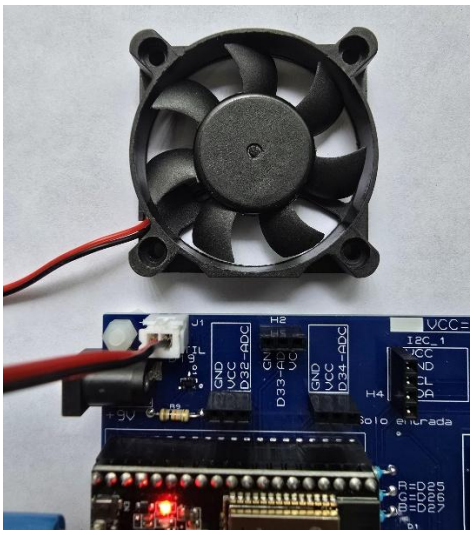


*Figura 32. Conexión del ventilador a la tarjeta DASA 2.0.*

*Tabla 18. Programa para controlar el ventilador con un potenciómetro (Elaboración propia).*

```
/* Ventil_PWM
 * Programa que controla la velocidad del ventilador
 * mediante el potenciómetro
 * DASA 2.0 ESP32
 * 2025  */

#define VENT 19   //Ventilador en D19
#define POT 36  //Canal analógico del potenciómetro

int valPOT = 0; //Guarda el valor del potenciómetro
int valPWM = 0; //Valor de la señal PWM (0-255)
```

```
void setup() {
   pinMode(VENT, OUTPUT);  //Declara PIN como salida digital
}

void loop() {
   valPOT = analogRead(POT); //Obtiene 12 bits del canal analógico
   valPWM = map(valPOT, 0, 4094, 0, 255);  //Hace un cambio de escala
   analogWrite(VENT, valPWM);
   delay(50);
 }
```

## AGRADECIMIENTOS

El presente trabajo fue financiado por la Dirección General de Asuntos del Personal Académico mediante el proyecto PAPIME PE108923, Desarrollo de material didáctico para la asignatura de Informática Aplicada a la Ciencia y a la Industria.

## REFERENCIAS

Asair. (2018). *AHT10 Technical Manual*. Aosong Electronic Co.

Bosch. (2015). *BMP180 digital pressure sensor*. Bosch Sensortec GmbH.

Conagua. (2010). *Manual teórico práctico del observador meteorológico de superficie*. Comisión Nacional del Agua. Semarnat.

Expressif. (2023). *ESP32-WROOM-32 Datasheet*. Expressif Systems.

Handson. (2021). *Passive buzzer module KY-006 datasheet*. Handson Technology.

Hanwei. (2021). *MQ-2 Gas sensor datasheet*. Hanwei Electronics Co.

Ibrahim, D. (2023). *Mastering the Arduino UNO R4*. Elektor International Media B. V.

Maxim. (2019). *DS18B20 Digital Thermometer datasheet*. Maxim Integrated.

Santos, R., & Santos, S. (2023). *Learn ESP32 with Arduino IDE* (Second). Random Nerd Tutorials.

Sunrom. (2012). *DHT11 Humidity and temperature sensor*. Sunrom Technologies.

Vishay. (2016). *OLED-128O064D Datasheet*. Vishay Intertechnology, Inc.

## ANEXO A. Programa de prueba básico

```
/*************************************************************************
PRUEBA_simple.ino    DASA 2.0 v10b
2025
 *************************************************************************/

#include <Adafruit_GFX.h>      //Bibloteca de Arduino IDE by Adafruit v1.11.11
#include <Adafruit_SSD1306.h>   //Bibloteca de Arduino IDE by Adafruit v2.5.13

#define SCREEN_WIDTH 128 // Ancho del display en pixeles
#define SCREEN_HEIGHT 64 // Alto del display en pixeles

// El display OLED se encuentra en la dirección 0x3C
#define OLED_RESET     -1 // Línea de reset (en caso de que lo haya)  -1 si no lo hay
#define SCREEN_ADDRESS 0x3C // Esta dirección puede cambiar con el modelo de pantalla OLED
//#define SCREEN_ADDRESS 0x78   // Esta es la dirección que indica la serigrafía (no funciona)
#define fotor     13  // Fotorresistencia en D13
#define boton1    14  //Botón 1 en D14
#define boton2    15  //Botón 2 en D15
#define boton3    16  //Botón 3 en D16
#define boton4    17  //Botón 4 en D17
#define buzzer    23  // GPIO23 pin del zumbador
#define RGB_R     25  //LED RGB Rojo en D25
#define RGB_G     26  //LED RGB Verde en D26
#define RGB_B     27  //LED RGB Azul en D27
#define pot       36  // Potenciómetro

// Configuración de la pantalla OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
int valPOT = 0;     // Valor del pot
bool btn1 = false;  //¿Ya se presionó el botón?
bool btn2 = false;
bool btn3 = false;
bool btn4 = false;

void setup() {
  Serial.begin(9600);   //Inicializa comunicación serial a la PC
  pinMode(fotor, INPUT);
  pinMode(boton1, INPUT); //Botones como entradas
  pinMode(boton2, INPUT);
  pinMode(boton3, INPUT);
  pinMode(boton4, INPUT);
```

```
  pinMode(RGB_B, OUTPUT); //LED RGB como salidas
  pinMode(RGB_G, OUTPUT);
  pinMode(RGB_R, OUTPUT);
  pinMode(buzzer, OUTPUT);
  // SSD1306_SWITCHCAPVCC = La OLED en modo compatible con VCC = 5 V
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 error de comunicación"));
    for(;;); // Loop infinito, aquí para el programa
  }
  display.clearDisplay();   // Limpia el búfer del display//EVita el logo de Adafruit
  display.display();    // Actualiza la memoria del display a cero imagen

  display.setTextSize(2);   // Selecciona el tamaño de letra
  display.setTextColor(WHITE);  //Selecciona WHITE o BLACK
  display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
  display.print("PRUEBA"); // Envía letrero a la memoria de la OLED
  display.setCursor(0,20);    // Nueva posición del cursor
  display.print("SIMPLE");    // Envía letrero a la memoria de la OLED
  display.setCursor(0,40);    // Nueva posición del cursor
  display.print("    "); // Envía letrero a la memoria de la OLED
  display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
  valPOT = analogRead(pot); // Lee el valor del potenciómetro
}

void loop() {
  tone(buzzer, 440); //440 Hz
  delay(500);
  tone(buzzer, 494);
  delay(500);
  noTone(buzzer);

  digitalWrite(RGB_R, HIGH);
  digitalWrite(RGB_G, LOW);
  digitalWrite(RGB_B, LOW);
  delay(1000);

  digitalWrite(RGB_G, HIGH);
  digitalWrite(RGB_R, LOW);
  digitalWrite(RGB_B, LOW);
  delay(1000);

  digitalWrite(RGB_B, HIGH);
  digitalWrite(RGB_R, LOW);
  digitalWrite(RGB_G, LOW);
  delay(1000);
```

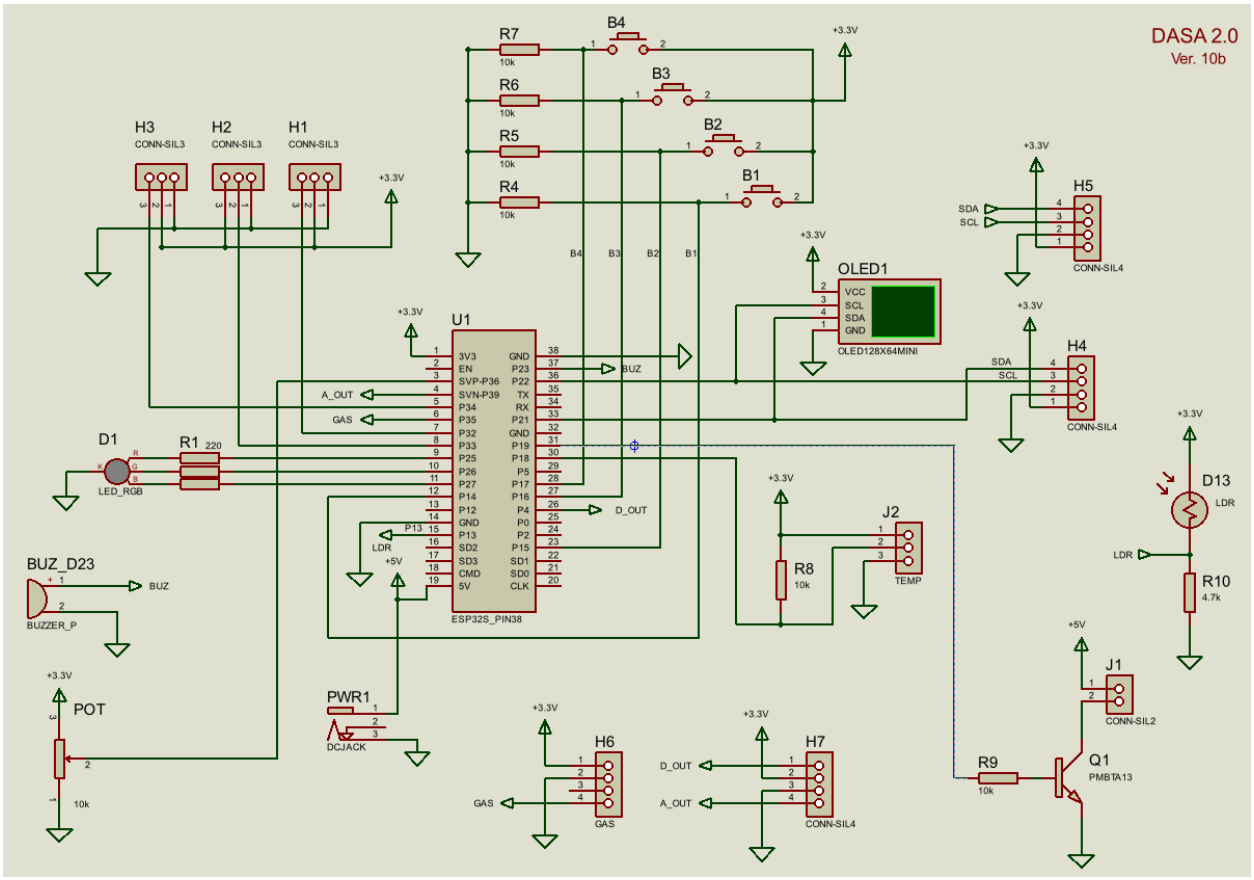
```
digitalWrite(RGB_B, LOW);
digitalWrite(RGB_R, LOW);
digitalWrite(RGB_G, LOW);

display.clearDisplay();   //Borra la memoria de la pantalla OLED
display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
display.print("Gire POT"); // Envía letrero a la memoria de la OLED
display.setCursor(0,20);    // Nueva posición del cursor
display.print("a la izq");    // Envía letrero a la memoria de la OLED
display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
while(analogRead(pot)>10);

display.clearDisplay();   //Borra la memoria de la pantalla OLED
display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
display.print("Gire POT"); // Envía letrero a la memoria de la OLED
display.setCursor(0,20);    // Nueva posición del cursor
display.print("a la der");    // Envía letrero a la memoria de la OLED
display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
while(analogRead(pot)<4000);

display.clearDisplay();   //Borra la memoria de la pantalla OLED
display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
display.print("Bloquee"); // Envía letrero a la memoria de la OLED
display.setCursor(0,20);    // Nueva posición del cursor
display.print("el LDR");    // Envía letrero a la memoria de la OLED
display.display();    //Actualiza el display (Esto hace que se escriban los letreros)

while(analogRead(fotor)>1500){    //Ajustar el valor 1500 según la iluminación presente
  digitalWrite(RGB_R, HIGH);
}
digitalWrite(RGB_R, LOW);

display.clearDisplay();   //Borra la memoria de la pantalla OLED
display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
display.print("Presiona"); // Envía letrero a la memoria de la OLED
display.setCursor(0,20);    // Nueva posición del cursor
display.print("cada boton");    // Envía letrero a la memoria de la OLED
display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
while(!btn1 || !btn2 || !btn3 || !btn4){
if(digitalRead(boton1)){  //Si se presiona Botón 1 enciende ROJO
  digitalWrite(RGB_R, HIGH);
  digitalWrite(RGB_G, LOW);
  digitalWrite(RGB_B, LOW);
  btn1=true;
}
```

```
  if(digitalRead(boton2)){  //Si se presiona Botón 2 enciende VERDE
    digitalWrite(RGB_G, HIGH);
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_B, LOW);
    btn2=true;
  }

  if(digitalRead(boton3)){  //Si se presiona Botón 3 enciende AZUL
    digitalWrite(RGB_B, HIGH);
    digitalWrite(RGB_R, LOW);
    digitalWrite(RGB_G, LOW);
    btn3=true;
  }
 if(digitalRead(boton4)){  //Si se presiona Botón 4 apaga TODOS
    digitalWrite(RGB_B, HIGH);
    digitalWrite(RGB_R, HIGH);
    digitalWrite(RGB_G, HIGH);
    btn4=4;     //Noten que no puse TRUE ¿por qué?
  }

  }
  digitalWrite(RGB_B, LOW);
  digitalWrite(RGB_R, LOW);
  digitalWrite(RGB_G, LOW);

  display.clearDisplay();   //Borra la memoria de la pantalla OLED
  display.setCursor(0,0);   // X, Y => HOR, VER  Posición del cursor
  display.print("Prueba"); // Envía letrero a la memoria de la OLED
  display.setCursor(0,20);    // Nueva posición del cursor
  display.print("finalizada");    // Envía letrero a la memoria de la OLED
  display.display();    //Actualiza el display (Esto hace que se escriban los letreros)
  while(1);
}
```

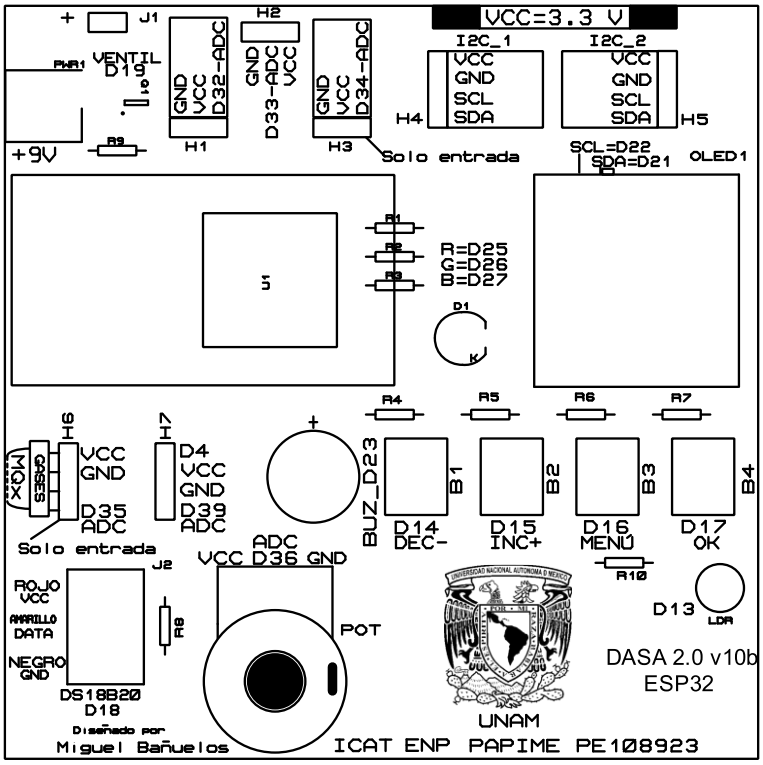
## ANEXO B. Lista de partes PCB v2.10b

| Número de parte | Descripción | Cantidad |
|---|---|---|
| R1, R2, R3 | Resistencia carbón 220 Ω 5% ½ W | 3 |
| R4, R5, R6, R7, R8, R9 | Resistencia carbón 10 kΩ 5% ½ W | 5 |
| R10 | Resistencia carbón 4.7 kΩ 5% ½ W | 1 |
| POT | Mini potenciómetro de carbón POTWH-10K 10 kΩ | 1 |
| RGB | LED RGB cátodo común 5 mm | 1 |
| B1, B2, B3, B4 | Push button 4 pines | 4 |
| Q1 | Transistor Darlington NPN PMBTA13 | 1 |
| U1 | Módulo ESP32 WROOM-32 MCU Node 32S 38 pines | 1 |
| Buzzer | Zumbador pasivo | 1 |
| LDR | Fotorresistencia 5516 | 1 |
| OLED1 | Pantalla OLED 0.96” 128x64 pixeles SSD1306 | 1 |
| B1, B2, B3, B4 | Push-button cuatro terminales | 4 |
| | Header receptáculo 19 posiciones | 2 |
| H1, H2, H3 | Header receptáculo 3 posiciones | 3 |
| H4, H5, H6, H7 | Header receptáculo 4 posiciones | 4 |
| J1 | Conector JST XH2.54 pin | 1 |
| J2 | Bloque de terminales con tornillo TRT-03 | 1 |
| | Poste para circuito impreso M3 10 mm con tuerca | 4 |
| PWR1 | Jack invertido 3.5 mm para PCB | 1 |
| DASA 2.0 v8 | Placa de circuito impreso | 1 |

# ANEXO C. Diagrama esquemático



## ANEXO D. Serigrafía



## ANEXO E. CIRCUITO IMPRESO