Pines GPIO (General Purpose Input/Output)

Qué son y cómo usar los GPIO en el ESP32

Los pines GPIO (General Purpose Input/Output), son puertos digitales que pueden ser configurados tanto como entradas como salidas en un microcontrolador.

Las entradas son utilizadas para leer señales del exterior, como sensores, interruptores o cualquier dispositivo que genere una señal eléctrica.

Las salidas, por otro lado, permiten al microcontrolador controlar dispositivos como LEDs, motores y otros componentes.

Uso de las entradas y salidas digitales en el ESP32

En el entorno de Arduino para ESP32, el manejo de los pines GPIO es bastante sencillo. Básicamente, es exactamente igual al que tendríamos en el caso de un Arduino “normal”.

Para configurar un pin como entrada o salida, se utiliza la función pinMode(pin, mode) donde pin es el número del pin que se quiere configurar y mode puede ser **INPUT** para configurarlo como entrada o **OUTPUT** para configurarlo como salida.

Modos de los pines GPIO

Además de los modos INPUT y OUTPUT, los pines GPIO en el ESP32 también pueden configurarse en otros modos que permiten funcionalidades más avanzadas. Algunos de estos modos incluyen:

- **INPUT_PULLUP**: Configura el pin como entrada con resistencia pull-up interna activada. Esto es útil para detectar cambios en interruptores o botones.
- **INPUT_PULLDOWN**: Similar al anterior, pero con resistencia pull-down interna activada.

- **OUTPUT_OD**: Configura el pin como salida de drenaje abierto, que es útil cuando se trabaja con circuitos externos.

Resistencias internas de pull-up y pull-down

Los pines GPIO en el ESP32 tienen resistencias pull-up y pull-down internas que pueden activarse según sea necesario.

Estas resistencias son útiles cuando se trabaja con entradas, como interruptores o botones, ya que ayudan a establecer un estado lógico predefinido cuando no hay una señal externa.

Ejemplos de código

Configurar un pin como entrada y leer su valor

```
const int sensorPin = 13; // Ejemplo de pin para sensor
int sensorValue;

void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  sensorValue = digitalRead(sensorPin);
  Serial.println(sensorValue);
  delay(1000);
}
```

Uso de una entrada digital en el ESP32

```
const int buttonPin = 12;  // Pin para el botón
int buttonState;           // Variable para almacenar el estado del botón

void setup() {
  pinMode(buttonPin, INPUT);  // Configurar el pin como entrada
  Serial.begin(9600);
}

void loop() {
  buttonState = digitalRead(buttonPin);  // Leer el estado del botón
  Serial.println(buttonState);           // Imprimir el estado en el puerto serie
  delay(100);                            // Pequeña pausa para evitar lecturas erráticas
}
```

Uso de una salida digital en el ESP32

```
const int ledPin = 13;  // Pin para el LED
int ledState = LOW;     // Estado inicial del LED

void setup() {
  pinMode(ledPin, OUTPUT);  // Configurar el pin como salida
  Serial.begin(9600);
}

void loop() {
  digitalWrite(ledPin, ledState);  // Establecer el estado del LED
  Serial.println(ledState);        // Imprimir el estado en el puerto serie
  delay(1000);                     // Esperar un segundo
  ledState = !ledState;            // Cambiar el estado del LED
}
```

Configurar un botón con resistencia pull-up interna

```
const int buttonPin = 12;
int buttonState;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) {
    Serial.println("Botón presionado");
  }
  delay(100);
}
```