SEMINARIO DE PROGRAMACION DE SISTEMAS EMBEBIDOS

INGENIERIA FOTONICA

PROFESOR: Daniel Giovanni Martínez Sandoval

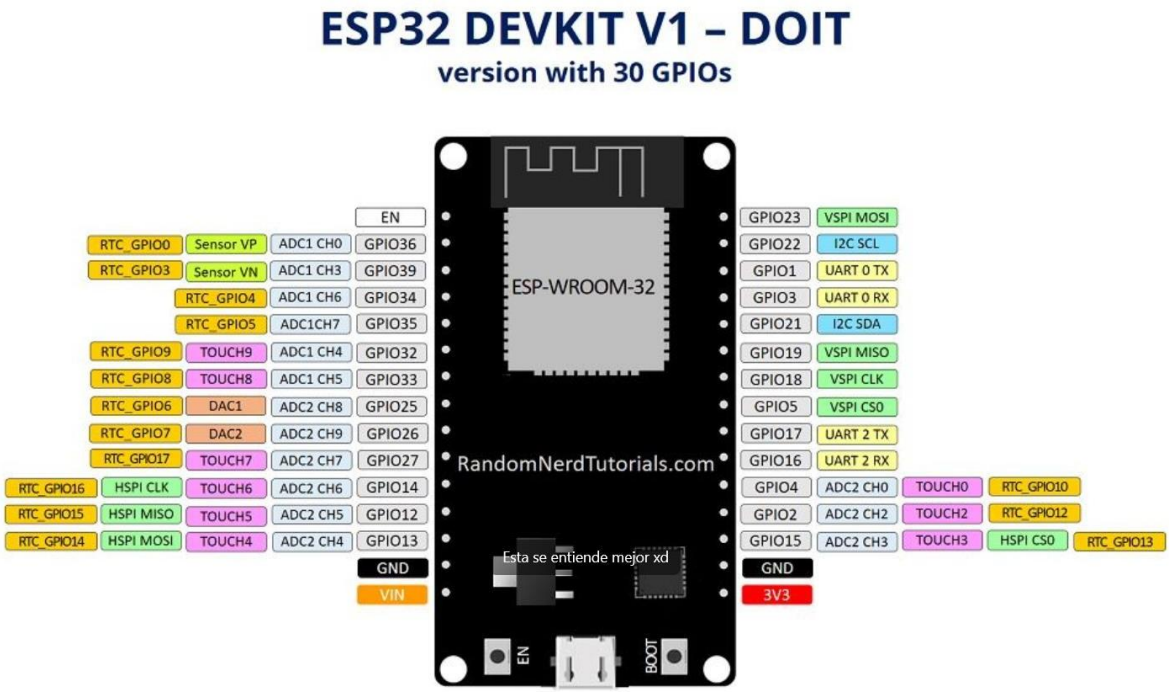
EQUIPO

ALUMNO: Davalos Magaña Jonathan Israel COD. 219339068

ALUMNO: Martinez Flores, Cassandra Frine COD. 218456125

ALUMNA: Cabrera Gómez Sergio Nicolás COD. 222328115

INDICE



1. Objectives

- Understand the basic operation of the ESP32 board, specifically in managing GPIO (General Purpose Input/Output) pins.
- Implement a program in the Arduino IDE development environment to control the built-in LED on the ESP32 board.
- Create an algorithm that allows the LED to blink with an incremental frequency, meaning its blinking speed increases over time.

PROCEDIMIENT2. Development

2.1 Theoretical Framework

The ESP32 is a low-cost, high-performance microcontroller developed by Espressif Systems. It features integrated Wi-Fi and Bluetooth connectivity, making it ideal for IoT (Internet of Things) applications. Among its key features are:

- *Dual-core 32-bit processor.*
- *34 programmable GPIO pins.*
- *Integrated SRAM memory.*
- *Compatibility with multiple communication protocols (SPI, I2C, UART, etc.).*

The built-in LED on the ESP32 board is connected to a specific GPIO pin (usually GPIO2). To control this LED, the pin must be configured as a digital output, and its state must be toggled between HIGH (on) and LOW (off) with a controlled time delay.

O

int LED1 = 23;

2.2 Methodology

1. **Development Environment Setup:**
    - The Arduino IDE was used to program the ESP32.
    - Support for the ESP32 board was installed in the IDE using the board manager.
2. **Connections:**
    - No external connections were required, as the built-in LED on the ESP32 board (connected to GPIO2) was used.
3. **Programming:**
    - A program was developed in C++ using Arduino libraries to control the LED.
    - The algorithm increases the LED blinking speed in each iteration by reducing the delay time between turning the LED on and off.

```
void setup(){
  Serial.begin(115200);
  pinMode(LED1,OUTPUT);
  }
void loop(){
  //Parpadeo LED
  digitalWrite(LED1,HIGH);
  delay(1000);
  digitalWrite(LED1,LOW);
  delay(1000);
```

Line-by-Line Explanation:

1. int LED1 = 23;
    - Here, an integer variable named LED1 is declared and assigned the value 23. This value represents the GPIO (General Purpose Input/Output) pin of the ESP32 to which the LED is connected. In this case, the LED is connected to GPIO pin 23.
2. void setup(){
    - This line defines the setup() function, which is a special function in Arduino and ESP32. This function runs only once when the program starts or when the device is reset. It is used to configure initial settings.
3. Serial.begin(115200);
    - This line initializes serial communication at a baud rate of 115200. This allows the ESP32 to communicate with a computer or another device via the serial port. It is useful for sending debug messages or receiving commands.
4. pinMode(LED1, OUTPUT);

- Here, the GPIO pin 23 (defined by the variable LED1) is configured as an output (OUTPUT). This means the pin can send electrical signals to control external devices, such as an LED.

5. }
    - This closing brace marks the end of the setup() function.
6. void loop(){
    - This line defines the loop() function, another special function in Arduino and ESP32. This function runs repeatedly in an infinite loop after setup() has finished. It contains the code that should run continuously.
7. // Parpadeo LED
    - This is a comment. Comments in code start with // and are ignored by the compiler. They are used to explain what the code does. In this case, the comment indicates that the following code will make the LED blink.
8. digitalWrite(LED1, HIGH);
    - This line turns on the LED. digitalWrite() is a function that sets the state of a digital pin. Here, the pin LED1 (GPIO 23) is set to HIGH, meaning a high voltage (typically 3.3V on the ESP32) is applied to the pin, turning on the LED.
9. delay(1000);
    - This line pauses the program execution for 1000 milliseconds (1 second). During this time, the LED remains on.
10. digitalWrite(LED1, LOW);
    - This line turns off the LED. Here, the pin LED1 is set to LOW, meaning a low voltage (0V) is applied to the pin, turning off the LED.
11. delay(1000);
    - This line pauses the program execution for another 1000 milliseconds (1 second). During this time, the LED remains off.

12. }

    - This closing brace marks the end of the loop() function. After the code inside loop() is executed, the program returns to the beginning of loop() and repeats indefinitely.

Summary:

- The code configures GPIO pin 23 as an output and then enters an infinite loop where it turns the LED connected to that pin on and off, with a 1-second interval between each change. This causes the LED to blink continuously.

Resumen:

- El código configura el pin GPIO 23 como una salida y luego entra en un bucle infinito donde enciende y apaga el LED conectado a ese pin, con un intervalo de 1 segundo entre cada cambio. Esto hace que el LED parpadee continuamente.

1. Testing:
    - The program was uploaded to the ESP32, and the LED behavior was observed.
    - It was verified that the LED blinked with an incremental frequency until reaching a minimum delay of 100 ms.

---

3. Results

- The built-in LED on the ESP32 blinked correctly, increasing its blinking speed with each iteration.
- The delay between turning the LED on and off decreased from 1000 ms to 100 ms in steps of 100 ms.
- The program ran stably without execution errors.

---

4. Conclusion

This practice allowed for an understanding of the basic operation of the ESP32, specifically in managing GPIO pins to control an LED. An algorithm was successfully implemented to increase the LED blinking speed, demonstrating the ESP32's capability to perform real-time

control tasks. This exercise lays the foundation for more complex projects, such as controlling external devices or implementing advanced communication protocols.