ESCUELA: Universidad Politécnica de Chiapas.

CARRERA: Ingeniería en Software.

TEMA: Práctica 2 - ESP32 – LED EXTERNO.

ASIGNATURA: Electricidad y Magnetismo.

GRADO Y GRUPO: “4° B”

INTEGRANTES:

Sujey Calderón Martínez. 233291

Hannia Paola De Los Santos Bautista. 233273

Victor Fabricio Pérez Constantino. 233394

Ameth De Jesús Méndez Toledo. 233363

Joaquín Esaú Pérez Diaz. 233412

PROFESOR:

Juan Manuel Martínez Constantino

FECHA DE ENTREGA: viernes 1 de Noviembre de 2024.

TABLA DE CONTENIDO

PRÁCTICA 2 - ESP32 - LED EXTERNO

ANTECEDENTES

El uso de microcontroladores en aplicaciones de control y automatización ha crecido exponencialmente. En este artículo, se presenta el desarrollo de una práctica con la placa ESP32 para el control de un LED externo. Este proyecto muestra las posibilidades de aplicaciones básicas de este dispositivo, junto con los componentes necesarios y el proceso paso a paso.

INTRODUCCIÓN

El ESP32 es un microcontrolador de bajo costo con capacidades WiFi y Bluetooth integrado, lo que lo convierte en una herramienta versátil para aplicaciones en domótica, IoT, y automatización. En esta práctica, se ha utilizado un ESP32 para el control de un LED externo, lo cual permite observar la interacción básica entre el microcontrolador y dispositivos externos a través de puertos GPIO.

DESARROLLO:

MARCO TEÓRICO

El ESP32 permite la interacción con sensores y actuadores externos mediante sus pines GPIO. Cada uno de estos pines puede ser configurado para enviar o recibir señales digitales y analógicas, lo cual permite implementar una gran variedad de aplicaciones, desde control de luces hasta monitoreo de datos.

PROCEDIMIENTOS

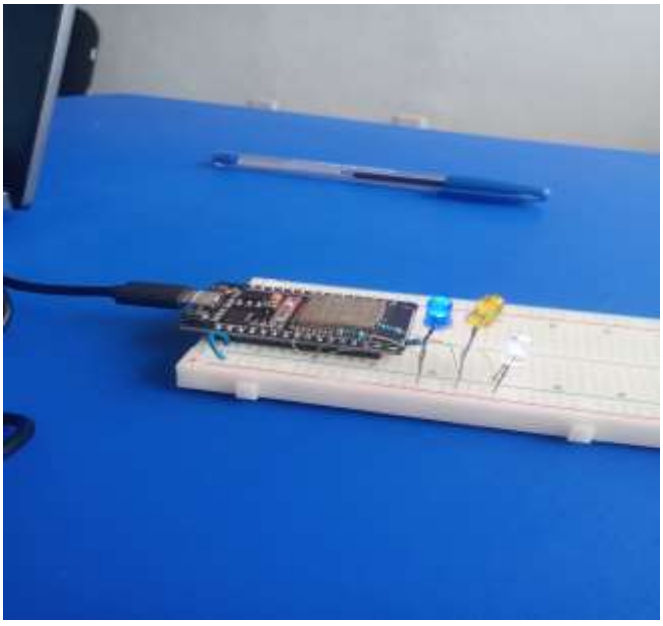
Primero descargamos y configuramos el IDE de Arduino. Después, conectamos el cable de datos USB a la placa ESP32 y a la computadora. Descargamos las librerías correspondientes para la placa, buscamos y seleccionamos la placa ESP32 en el IDE de Arduino, y utilizamos un código básico para hacer parpadear un LED. Este código es un programa sencillo que permite hacer parpadear un LED conectado al

pin 2 del ESP32. Luego, modificamos el tiempo de encendido y apagado cambiándolo a una cantidad menor para ajustar la velocidad de parpadeo del LED.

LISTA DE MATERIALES.

- Placa ESP32
- LED externo
- Resistencia de 300 ohmios
- Cableado para conexiones
- Cable de datos USB
- Laptop

RESULTADOS DE LA PRÁCTICA.



La foto muestra el momento en que el LED está encendido, como resultado del código de parpadeo cargado en el ESP32. Este montaje demuestra que la placa está funcionando correctamente y controlando el estado del LED mediante las instrucciones de encendido y apagado en el código.

CÓDIGO

```
1  int LED = 23;
2  int buttonPin = 22;
3  int buttonState = 0;
4
5  void setup() {
6    Serial.begin(115200);
7
8    pinMode(LED,OUTPUT);
9    pinMode(buttonPin, INPUT_PULLUP);
10 }
11
12 void loop() {
13   buttonState = digitalRead(buttonPin);
14   Serial.println(buttonState);
15
16   if(buttonState == LOW){
17     digitalWrite(LED,HIGH);
18   }
19   else{
20     digitalWrite(LED,LOW);
21   }
22 }
23
```

DECLARACIÓN DE VARIABLES:

int LED = 23; Define el pin 23 del ESP32 como el pin donde estará conectado el LED.

int buttonPin = 22;Define el pin 22 como el pin donde se conecta un botón.

int buttonState = 0;Declara la variable buttonState, que almacenará el estado del botón (presionado o no).

Función setup():Esta función se ejecuta una vez al iniciar el ESP32 y configura los pines y la comunicación serial.

Serial.begin(115200);
Inicia la comunicación serial a una velocidad de 115200 baudios para poder ver en el monitor serial el estado del botón.

pinMode(LED, OUTPUT);Configura el pin 23 (donde está el LED) como una salida.

pinMode(buttonPin, INPUT_PULLUP);Configura el pin del botón (22) como entrada con un resistor de pull-up interno. Esto asegura que el pin esté en un estado estable (HIGH) cuando el botón no está presionado.

Función loop():Esta función se ejecuta en un bucle infinito mientras el ESP32 esté encendido.

buttonState = digitalRead(buttonPin);Lee el estado actual del botón (si está presionado o no) y almacena el valor en la variable buttonState.

Serial.println(buttonState);
Imprime el estado del botón en el monitor serial (0 cuando el botón está presionado y 1 cuando no lo está, debido a la configuración INPUT_PULLUP).

Este código controla un LED con un botón en un ESP32. Al presionar el botón, el LED conectado al pin 23 se enciende, y al soltarlo, el LED se apaga. El botón está conectado al pin 22 y configurado con una resistencia interna INPUT_PULLUP para estabilizar su estado. Además, el estado del botón se imprime en el monitor serial para verificar su funcionamiento.

CONCLUSIONES

El control de un LED externo a través de un ESP32 permite entender los principios básicos de configuración de pines y control de dispositivos externos. A través de esta práctica, se ha desarrollado una comprensión del uso de GPIO en aplicaciones prácticas y su potencial en proyectos de automatización.

BIBLIOGRAFÍAS:

https://www.sovi.es/tutorial-de-led-intermitente-esp32-usando-control-gpio-con-arduino-ide/

https://www.programadornovato.com/esp32-encender-un-led/

https://esp32io.com/tutorials/esp32-led-strip