



Trabajo práctico integrador - Proyecto de Arduino

Asignatura: Teoría de Control

Año: 2023

Docentes:

- Nicolás Oliva
- José Luis Catalano

Alumnos:

- Joaquin Baccifava - baccifavajoaquin@gmail.com
- Mateo Ferradans - Mferradanss@gmail.com

Índice

Trabajo práctico integrador - Proyecto de Arduino.....	0
Introducción.....	2
Objetivo.....	2
Marco Teórico.....	2
Fundamentos teóricos relacionados con el proyecto:.....	2
Tecnologías o conceptos clave utilizados:.....	2
Esquema de Conexionado.....	3
Conexión por WiFi y comunicación simulada con Packet Sender.....	4
Desarrollo del código.....	4
Resultados.....	7

Introducción

El presente informe tiene como objetivo describir el proceso de aprendizaje y conexión de diversos componentes en una protoboard para lograr el funcionamiento de un Arduino ESP8266 con un sensor de color TCS3200 y un relé SRD-05VDC-SL-C, empleando la comunicación WiFi. Durante el desarrollo de este proyecto, se utilizó el entorno de desarrollo Arduino IDE y el software Packet Sender para simular la comunicación TCP.

Objetivo

El desafío principal consistía en integrar de manera efectiva los componentes mencionados para lograr una comunicación fluida y funcional entre ellos. El objetivo final era permitir que el Arduino ESP8266 recibiera información del sensor de color TCS3200 y, en base a esa información, activara o desactivara el relé SRD-05VDC-SL-C el cual permitía que un led se encendiera usando las señales PWM para que graduara su intensidad. Además de esto lograr la comunicación TCP con el software simulador Packet Sender.

Marco Teórico




Fundamentos teóricos relacionados con el proyecto:

1. **Control de lazo cerrado:** El control de lazo cerrado implica utilizar la retroalimentación para ajustar y corregir el comportamiento de un sistema. En este proyecto, el Arduino recibe información del sensor de color y utiliza esa información para controlar el estado del relé. Al utilizar la retroalimentación del sensor, el control de lazo cerrado permite mantener el relé en un estado específico en respuesta a las condiciones del entorno.
2. **Sistemas de control en tiempo real:** Un sistema de control en tiempo real es aquel que responde y ajusta su salida en función de los cambios en las condiciones del entorno en tiempo real. En este proyecto, el Arduino recibe datos del sensor de color y toma decisiones instantáneas para controlar el estado del relé.

Tecnologías o conceptos clave utilizados:

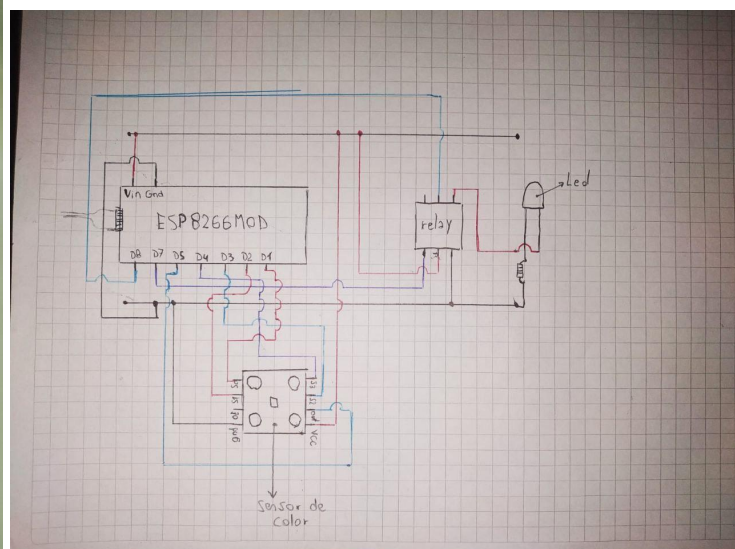
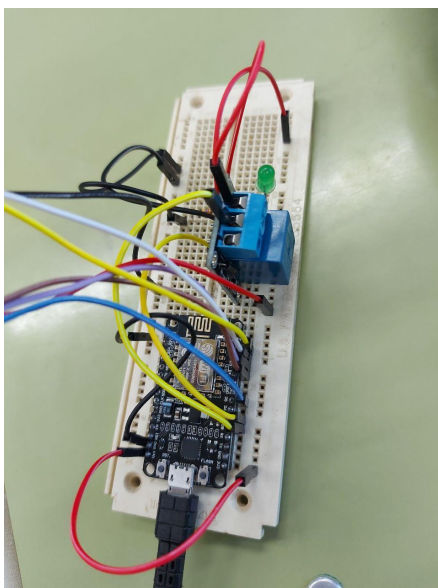
1. **ESP8266:** El ESP8266 es un microcontrolador que ofrece conectividad Wi-Fi, lo que permite a los dispositivos comunicarse a través de redes inalámbricas y acceder a servicios en la nube.
2. **Sensor de color TCS3200:** El sensor de color TCS3200 es un dispositivo que puede detectar y medir con precisión el color de un objeto. Utiliza una matriz de fotodiodos y filtros de color para convertir la luz en señales eléctricas que se pueden procesar. Proporciona información sobre la intensidad de los componentes rojo, verde y azul (RGB) de la luz incidente.
3. **Relé SRD-05VDC-SL-C:** Un relé es un interruptor electromagnético utilizado para controlar circuitos eléctricos mediante una señal eléctrica.
4. **Resistencia:** Una resistencia se utiliza para limitar la corriente que fluye a través de un componente, como un LED, y evitar daños por sobrecarga.
5. **Proteoboard (placa de pruebas):** La proteoboard es una placa de circuito impreso que se utiliza para prototipar y conectar componentes electrónicos de manera temporal sin la necesidad de soldar los componentes.
6. **LED (Diodo Emisor de Luz):** El LED se utiliza como una señal visual para indicar si el relé está encendido o apagado.

7. **Packet Sender:** Packet Sender es una herramienta de software que permite enviar y recibir paquetes de datos a través de diferentes protocolos de red, como TCP, UDP y MQTT.
8. **Arduino IDE:** Arduino IDE (Integrated Development Environment) es un entorno de desarrollo integrado utilizado para programar microcontroladores Arduino.

Sensor de color TCS3200	Relé SRD-05VDC-SL-C	Arduino ESP8266
		

Esquema de Conexión

El proceso de aprendizaje comenzó con el estudio individual de cada componente, comprendiendo su funcionamiento y características específicas. Se revisaron las datasheets del Arduino ESP8266, el sensor de color TCS3200 y el relé SRD-05VDC-SL-C para comprender sus pines de conexión y requerimientos eléctricos.



A continuación, se procedió a conectar los componentes en la protoboard de acuerdo a los siguientes pasos:

```
const int s0 = D1;
const int s1 = D2;
const int s2 = D3;
const int s3 = D4;
const int out = D5;
const int relay = D7;
const int led = D8;
```

Se conectaron los pines de alimentación y tierra de cada dispositivo a la fuente de alimentación de la protoboard, asegurando una correcta conexión eléctrica.

Se establecieron las conexiones necesarias entre los pines de datos del sensor de color TCS3200 y los pines digitales del Arduino ESP8266.

Se conectaron los pines de control del relé SRD-05VDC-SL-C a pines digitales específicos del Arduino ESP8266.

Conexión por WiFi y comunicación simulada con Packet Sender

Para establecer la comunicación por WiFi se configuraron los parámetros de conexión WiFi, como el SSID y la contraseña de la red a la que se deseaba conectar.

A continuación, se implementó el código necesario para recibir los datos del sensor de color TCS3200 y controlar el relé SRD-05VDC-SL-C en base a esos datos.

Para simular la comunicación, se empleó el software Packet Sender. Se configuraron los mensajes a enviar al Arduino ESP8266 y se enviaron a través de la red WiFi establecida. Se verificó que el Arduino ESP8266 recibiera los mensajes correctamente y respondiera según lo esperado, activando o desactivando el relé según la información recibida.

Desarrollo del código

Incluimos la librería "ESP8266WiFi.h", que proporciona funciones para conectarnos y comunicarnos a través de Wi-Fi. Definimos las variables para configurar la conexión Wi-Fi, como el nombre de la red (SSID) y la contraseña y creamos un objeto de tipo WiFiServer para establecer un servidor en el puerto 3000.

```
// WiFi parameters to be configured
const char* ssid = "Bacci"; // Write
const char* password = "bacci123"; //
WiFiServer server(3000);
```

Declaramos las variables y pines necesarios para el control de colores y otras funciones.

En la función "setup()", realizamos la inicialización del programa. Iniciamos la comunicación e intentamos establecer la conexión Wi-Fi utilizando los parámetros proporcionados. Esperamos hasta que la conexión se establezca correctamente. Luego mostramos por la consola la dirección IP asignada al ESP8266. Iniciamos el servidor y configuramos los pines como entradas o salidas.

La función pinMode() se utiliza para configurar el modo de un pin en el microcontrolador. En este caso, se configuran los pines s0, s1, s2, s3, out, relay y led como salidas o entradas.

La función digitalWrite() se utiliza para establecer el estado de un pin digital en el microcontrolador. En este caso, se utilizan para establecer los pines s0 y s1 en nivel alto y el

pin led en nivel alto para apagar el LED.

```
Serial.println("Server ON");
// Inicializando Server
server.begin();

pinMode(s0, OUTPUT);
pinMode(s1, OUTPUT);
pinMode(s2, OUTPUT);
pinMode(s3, OUTPUT);
pinMode(out, INPUT);
digitalWrite(s0, HIGH);
digitalWrite(s1, HIGH);

pinMode(relay, OUTPUT);
pinMode(led, OUTPUT);
digitalWrite(led, HIGH);
tiempoAnterior=millis();
```

En la función "loop()", verificamos si la conexión Wi-Fi está establecida. Si es así, creamos un objeto de tipo WiFiClient e intentamos conectar a una dirección IP específica en el puerto 8081. Mostramos un mensaje indicando si la conexión se estableció o falló.

Luego, llamamos a la función "color()" que realiza la lectura de los valores de los colores rojo, verde y azul utilizando el sensor conectado al pin "out". Estos valores se almacenan en las variables correspondientes.

```
void color() {
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  rojo = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s3, HIGH);
  azul = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  digitalWrite(s2, HIGH);
  verde = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
}
```

A continuación, realizamos una serie de comprobaciones para determinar el color detectado. Si se detecta el color rojo, enviamos un mensaje al servidor. Si se detecta el color azul, enviamos otro mensaje. Si se detecta el color verde, incrementamos gradualmente la intensidad de un LED conectado al pin "led" en intervalos de tiempo predefinidos. Esta

técnica se llama señales PWM Pulse Width Modulation.

```
} else if (azul < rojo && azul < verde && verde < rojo) {
    alcanzadoVerde100 = false;
    detectoVerde = false;
    client.println("AZUL");
    digitalWrite(relay, LOW);
} else if (rojo > verde && azul > verde) {
    detectoVerde = true;
    // Incrementa la intensidad del LED con el tiempo

    unsigned long tiempoActual = millis();

    Serial.print("V-Tiempo Actual: ");
    Serial.println(tiempoActual);
    Serial.print("V-Tiempo Anterior: ");
    Serial.println(tiempoAnterior);

    if ((tiempoActual - tiempoAnterior) >= t1 && (tiempoActual - tiempoAnterior) <= t2 && !alcanzadoVerde100) {
        client.println("VERDE %25");
        analogWrite(led, 64);

    } else if ((tiempoActual - tiempoAnterior) > t2 && (tiempoActual - tiempoAnterior) <= t3 && !alcanzadoVerde100) {
        client.println("VERDE %50");
        analogWrite(led, 128);
    } else if ((tiempoActual - tiempoAnterior) > t3 && (tiempoActual - tiempoAnterior) <= t4 && !alcanzadoVerde100) {
        client.println("VERDE %75");
        analogWrite(led, 192);
    } else if ((tiempoActual - tiempoAnterior) > t4 && !alcanzadoVerde100) {
        alcanzadoVerde100 = true;
        client.println("VERDE %100");
        analogWrite(led, 254);
        // Reinicia el tiempoAnterior para comenzar el siguiente ciclo
        tiempoAnterior = tiempoActual;
    }
}
```

Después de enviar los mensajes al servidor, esperamos a recibir una respuesta y la mostramos por la consola.

```
    digitalWrite(relay, HIGH);
} else {
    alcanzadoVerde100 = false;
    detectoVerde = false;
    digitalWrite(relay, LOW);
}

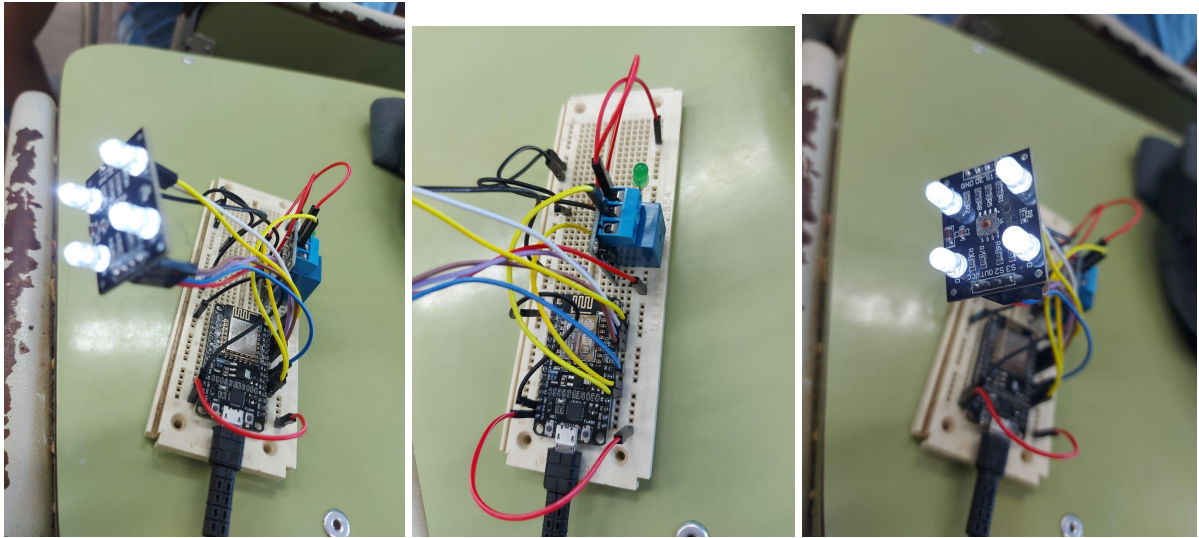
// Espera y lee la respuesta del servidor
while (client.available()) {
    Serial.println("Client Available");
    char c = client.read();
    Serial.print(c);
}

delay(900);
// Cierra la conexión
client.stop();
}
```

Finalmente, esperamos un tiempo de 900 milisegundos y cerramos la conexión. Luego, verificamos si hay una nueva conexión entrante al servidor y procesamos la solicitud.

También tenemos una función adicional que se encarga de enviar mensajes TCP al servidor en una dirección IP específica y en el puerto 8081.

Resultados



Los resultados fueron correctos, el sensor de color prende sus 4 leds blancos para alumbrar a la superficie que se le acerque y poder detectar el color correcto. En nuestro código al programar hicimos que cuando detecte el color verde pueda encender el led que le da acceso el relé y vaya incrementando la intensidad del led mediante el tiempo pasa.