Tarea 5 – Detector de "Llama" Simulado con Respuesta de Emergencia

Introducción (What & When)

El objetivo de esta actividad fue simular la detección de una llama (fuego) utilizando un sensor simple, la fotoresistencia (LDR), y programar una respuesta de emergencia automatizada en un microcontrolador ESP8266 (en este caso, una Wemos D1). La simulación consistía en activar una alarma visual (LED) y desplegar un mecanismo de "supresión" de incendios (Servo SG90).

La tarea fue completada el 22-11-2025 y representó un ejercicio de integración de sensores, actuadores y lógica de control.

Investigación y Referencias (Why)

Para abordar este proyecto, la primera decisión clave fue utilizar la placa **Wemos D1 (ESP8266)**. Esto requirió un proceso inicial de configuración del entorno de desarrollo que incluía:

1. **IDE de Arduino:** Utilicé la versión más reciente para asegurar la compatibilidad.
2. **Controladores (Drivers):** La instalación de los drivers CH340 fue crucial para que el ordenador reconociera la placa.
3. **Integración del ESP8266:** Fue necesario añadir las URLs y los controladores de la tarjeta ESP8266 al gestor de tarjetas del IDE de Arduino.

En cuanto al diseño del circuito, decidí utilizar el software **KiCad** para la creación de los esquemas eléctricos. Mi objetivo al usar KiCad, un software Open Source, fue asegurarme de tener una representación profesional y escalable del circuito, preparando los archivos para una posible fabricación de PCB en el futuro, si el proyecto escalara.

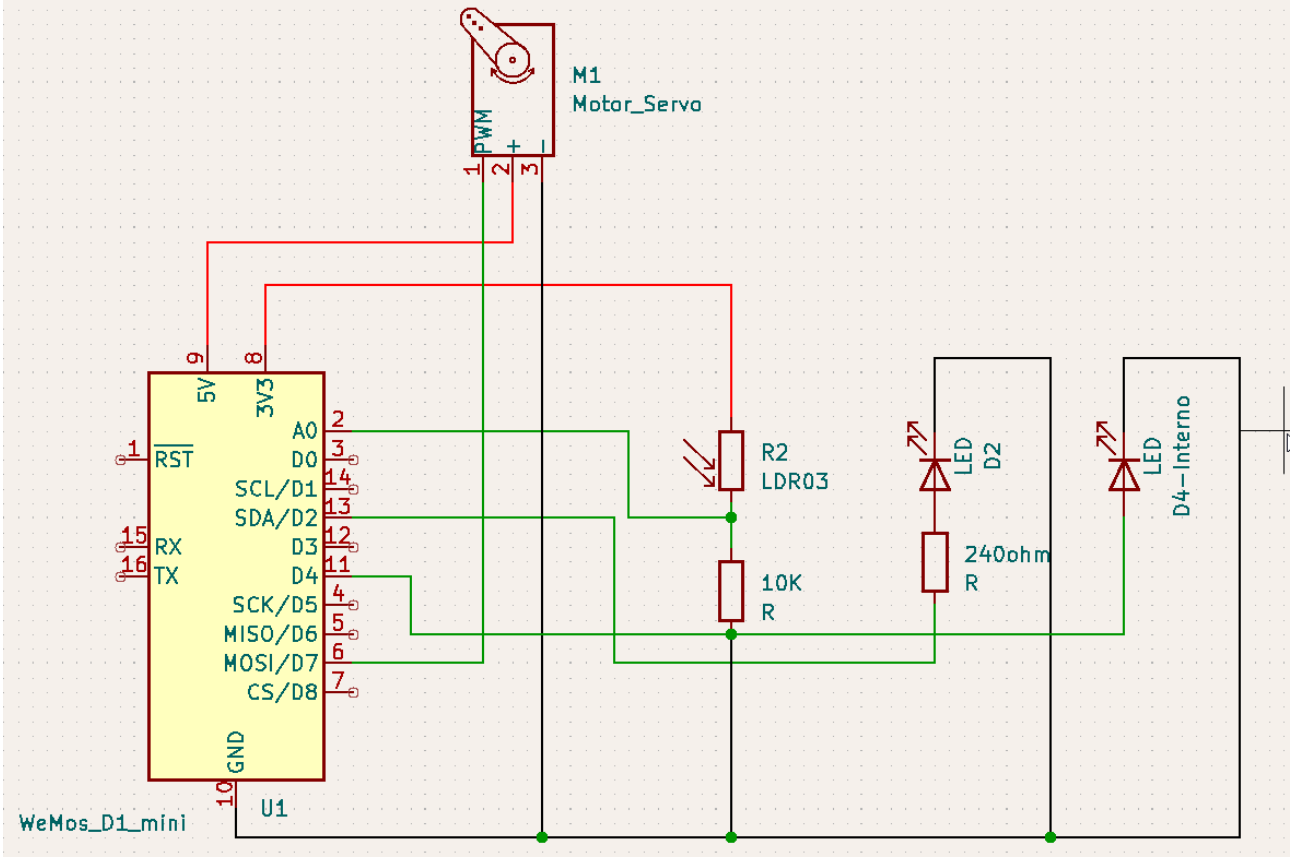
Referencias Utilizadas:

- **Documentación de Arduino:** https://www.arduino.cc/ (Para la sintaxis y funciones básicas).
- **Documentación Wemos D1:**
    - https://www.wemos.cc/en/latest/d1/index.html
    - https://www.wemos.cc/en/latest/tutorials/d1/get_started_with_arduino_d1.html (Guía de inicio).
- **Software de Diseño:** https://www.kicad.org/ (Para el diseño del esquema).

Proceso Paso a Paso (How)

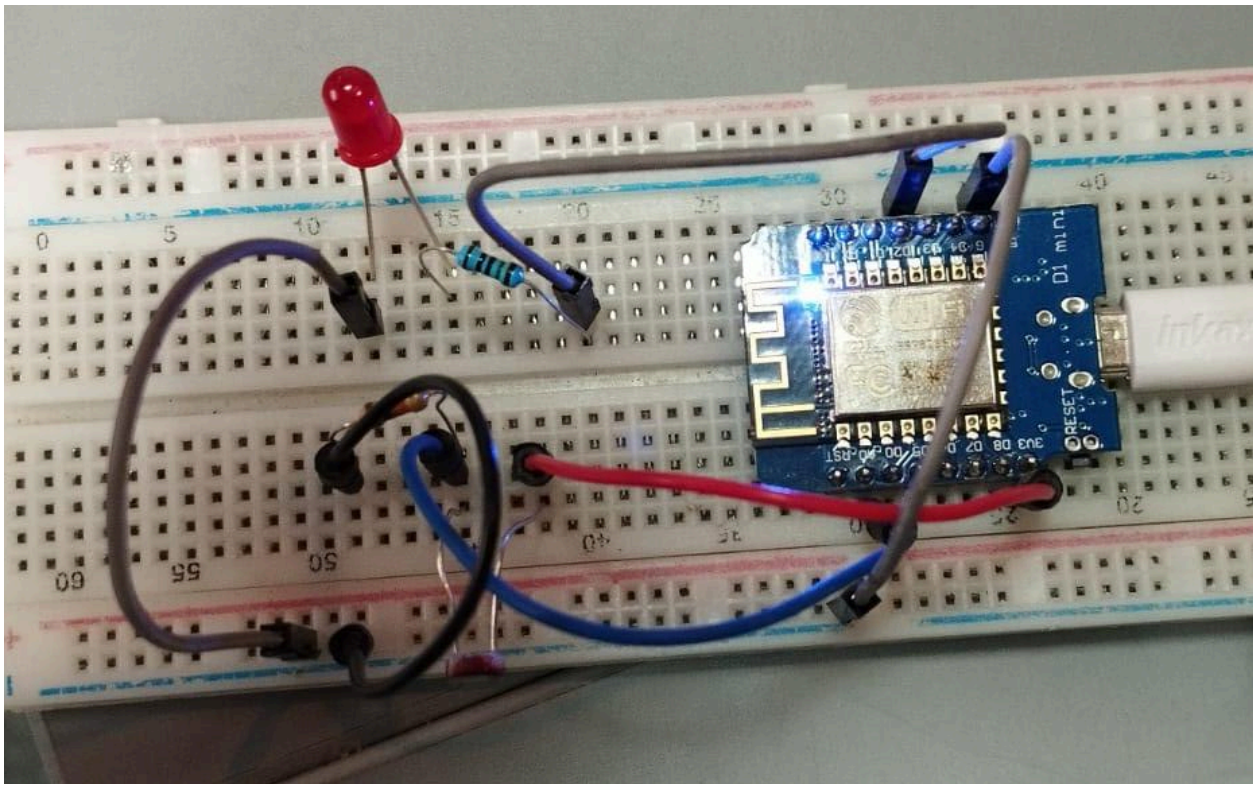
El desarrollo se realizó en fases incrementales para aislar y verificar la funcionalidad de cada componente antes de la integración final.

El esquema del circuito se diseñó primero en KiCad para validar las conexiones, especialmente el divisor de tensión para el LDR y la alimentación del servo.



Materiales y Conexiones

Componente	Función
Wemos D1 (ESP8266)	Microcontrolador central
Protoboard y Cables	Interconexión
Resistencia LDR	Detección de luz (sensor de llama)
Resistencia 10K	Divisor de tensión para el LDR
Resistencia 240 ohm	Límite de corriente para el LED
1 LED Rojo	Alarma visual
Servo SG90	Actuador de "supresión" de fuego
Botón	Reset del sistema



1.

Test de Reconocimiento de Luminosidad (LDR)

Se armó el circuito inicial solo con el LDR conectado a una entrada analógica del Wemos D1. La prueba consistió en monitorear los valores de lectura serial para establecer un umbral.

```
08:31:54.498 -> Lectura LDR: 1008
08:31:55.005 -> Lectura LDR: 862
08:31:55.507 -> Lectura LDR: 860
08:31:56.014 -> Lectura LDR: 859
08:31:56.505 -> Lectura LDR: 859
08:31:57.025 -> Lectura LDR: 859
08:31:57.526 -> Lectura LDR: 859
08:31:58.010 -> Lectura LDR: 864
08:31:58.499 -> Lectura LDR: 865
08:31:59.020 -> Lectura LDR: 866
08:31:59.527 -> Lectura LDR: 866
08:32:00.012 -> Lectura LDR: 868
08:32:00.517 -> Lectura LDR: 894
08:32:00.989 -> Lectura LDR: 877
08:32:01.509 -> Lectura LDR: 879
08:32:02.013 -> Lectura LDR: 880
08:32:02.506 -> Lectura LDR: 883
08:32:02.984 -> Lectura LDR: 981
08:32:03.517 -> Lectura LDR: 994
08:32:04.023 -> Lectura LDR: 996
08:32:04.528 -> Lectura LDR: 997
08:32:05.004 -> Lectura LDR: 1002
08:32:05.523 -> Lectura LDR: 1004
08:32:05.998 -> Lectura LDR: 1003
08:32:06.499 -> Lectura LDR: 1002
08:32:07.013 -> Lectura LDR: 1003
08:32:07.500 -> Lectura LDR: 1003
08:32:08.023 -> Lectura LDR: 1003
08:32:08.527 -> Lectura LDR: 1003
08:32:09.001 -> Lectura LDR: 1003
08:32:09.494 -> Lectura LDR: 1003
08:32:10.021 -> Lectura LDR: 1003
08:32:10.512 -> Lectura LDR: 1003
```

- **Resultado:** El LDR funcionó correctamente, demostrando un cambio brusco en la lectura al exponerlo a una fuente de luz (simulando la llama). Esto permitió establecer un umbral de detección fiable.

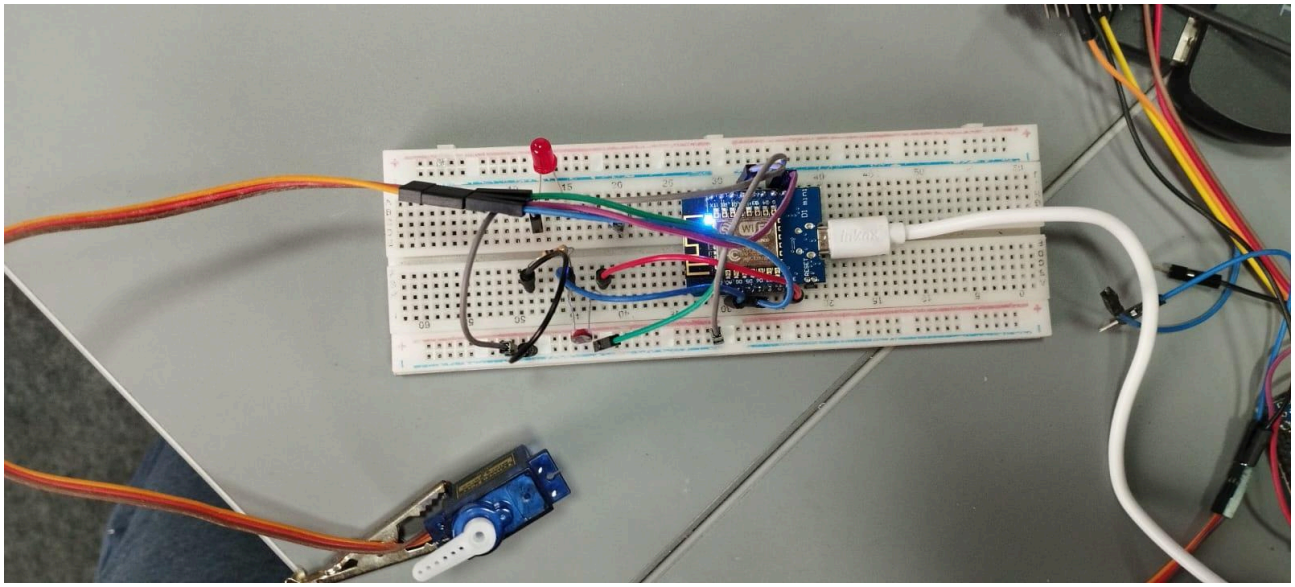
2. Integración del LED de Alarma

En esta etapa, se añadió el LED Rojo. La lógica de control se programó para que, si la lectura del LDR superaba el umbral (detección de llama), el LED se encendiera inmediatamente.

- **Resultado:** El test fue exitoso. El LED respondía instantáneamente al cambio de luminosidad, confirmando que la lógica de detección y la respuesta visual estaban operativas.

3. Integración Final del Servo (Actuador)

Finalmente, se incorporó el Servo SG90, junto con el botón de *reset*. El servo se integró al *firmware* para que, al detectarse la "llama", este realizará una "barrida" o un giro para simular el mecanismo de extinción, y solo regresará a su posición de vigilancia al presionar el botón de *reset*.



Resultados y Errores (Resultados & Fracasos)

El circuito completo funcionó tal como se esperaba:

1. **Estado de Vigilancia:** El sistema se mantiene en reposo, monitoreando el LDR.
2. **Detección:** Al acercar una luz, el LED se enciende como alarma y el servo realiza un movimiento.
3. **Reset:** El sistema vuelve al estado de vigilancia tras presionar el botón.

Fallo Documentado: Limitación Física del Servo SG90

El error más significativo se encontró en el **firmware** durante la fase de control del actuador.

- **Descripción del Error:** Inicialmente, el código de programación se configuró para que el servo realizará una barrida entre **45 y 180 grados**.
- **Realidad:** Este servo SG90 en particular, debido a sus limitaciones físicas internas o calibración, sólo alcanza un ángulo máximo de aproximadamente **100 grados**. Al intentar

forzarlo a 180 grados, el motor vibraba y no se movía, lo que generaba un ruido audible y un consumo de energía innecesario.
- **Solución Aplicada:** Tuve que modificar el código para limitar el rango de movimiento a un máximo seguro y funcional, ajustando la barrida entre 45° y 100° para evitar el sobreesfuerzo del SG90.

Reflexión

Este proyecto me enseñó el conocimiento teórico de un componente (ej. un servo que generalmente permite 180°) a menudo debe ser ajustado por las **limitaciones físicas y calibraciones del *hardware* específico** que se está utilizando en el momento. No siempre la hoja de datos se traduce en un rendimiento idéntico en todos los modelos o lotes.

Si repitiera el ejercicio, **mejoraría la calibración del LDR** creando un promedio de lecturas ambientales al inicio del programa, en lugar de utilizar un umbral fijo. Esto haría que el detector fuera más adaptativo a diferentes entornos de iluminación.

Archivos (Where)

Todos los archivos del proyecto, incluyendo el esquema de KiCad (en formato editable y PDF), las capturas de pantalla de las pruebas, y el *firmware* (código Arduino), se encuentran disponibles en el repositorio:

Firmware (Código Arduino):
https://github.com/Juandeleon-utec/Juan_de_Leon/blob/main/docs/arduino_bombero/arduino_bombero.ino

Archivos de Diseño (KiCad):
https://github.com/Juandeleon-utec/Juan_de_Leon/tree/main/docs/kicad/bomberos/bomberos.7z

Ubicacion de la carpeta de Github, con el material y videos:
https://juandeleon-utec.github.io/Juan_de_Leon/tecnicos/mt04/

Atribución y Ética

Todo el código y el diseño de la lógica fue desarrollado de forma individual en base a conocimientos ya adquiridos en trabajos previos y cursos anteriores.

La referencia para la configuración inicial de la placa Wemos D1 se obtuvo directamente de los tutoriales oficiales de la comunidad (enlaces citados en la sección de Investigación y Referencias).

CCL – Cognitive Contribution Label (uso de IA)

Herramienta: ChatGPT

Propósito:

● Reorganizar el texto original en un documento claro y alineado con criterios de documentación académica.

● Mejorar coherencia narrativa y estructura.

Aportes no delegados:

○ Todas las decisiones técnicas, ideas, contexto, imágenes, análisis y reflexiones

provienen del proyecto original.