UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

LABORATORIO ORGANIZACIÓN COMPUTACIONLA

Manual Técnico

NOMBRE COMPLETO	CARNÉ
Jorge Alejandro De León Batres	202111277
Roberto Miguel Garcia Santizo	202201724
Jose Javier Bonilla Salazar	202200035
Fredy Alexander Esteban Pineda	202110511

Requisitos

- **Hardware**
    - Arduino con conexión en COM8 y configuración a 9600 baudios.
    - Bombas controladas por Arduino (simulación o hardware real).
- **Software**
    - Python 3.x
    - Librerías necesarias: Flask, serial, re
    - Navegador web para acceder a la aplicación.
    - Navegador compatible con HTML5, CSS3 y JavaScript.
    - Bootstrap v5.0.2.

Estructura del proyecto

El proyecto esta dividido en tres archivos.

1. **Server.py:** se encarga de gestionar la lógica del lado del back end de la aplicación, este manipula y gestiona la carga de bombas, el recibimiento de estas y el analizador del archivo que se puede ingresar con las bombas.
2. **Index.html:** se encarga de mostrar la interfaz gráfica para el usuario además de poder ingresar las bombas.
3. **Pr.ino:** es el código que está relacionado al Arduino y maneja dicha lógica

Sever.py

1. **Importación de Módulos**
    a. **Flask:** Framework para desarrollar la aplicación web.
    b. **Serial:** Para la comunicación con Arduino.
    c. **re:** Módulo para trabajar con expresiones regulares.
    d. **time:** Para manejar tiempos de espera entre operaciones.
2. **Estructura del Código**
    El proyecto se organiza en varios endpoints de Flask que manejan las interacciones entre el usuario, la aplicación web, y el Arduino.
    a. **index():** Renderiza la página principal.
    b. **get_bomb_states():** Obtiene el estado actual de las bombas desde el Arduino.
    c. **new_game():** Reinicia el juego enviando un comando al Arduino.
    d. **send_bombs():** Envía posiciones de bombas seleccionadas al Arduino.
    e. **analyze_text():** Procesa texto enviado por el usuario para interpretar instrucciones de posición de bombas.

f. **analyze_text_content():** Analiza el contenido textual y envía las posiciones configuradas al Arduino.

Index.html

1. **Selección de bombas:** El usuario selecciona las posiciones del 1 al 16 haciendo clic en los botones de la cuadrícula.
2. **Nuevo Juego:** Restaura el tablero eliminando todas las selecciones previas.
3. **Carga de archivos:** Permite subir archivos .org para mostrarlos en un área de texto.
4. **Análisis de texto:** Envía el contenido del área de texto al backend para su procesamiento.

Api

Ruta	Método	Descripción
/newGame	POST	Reinicia el tablero
/get_bomb_states	GET	Obtiene las bombas activas
/send_bombs	POST	Envía las posiciones elegidas

Arduino

1. **Estados del Juego:**
   - **CONFIGURANDO:** Se pueden configurar las bombas.
   - **JUGANDO:** Los jugadores intentan desactivar las minas.
   - **GAME_OVER:** El juego finaliza cuando se activa una bomba.
   - **GANASTE:** El jugador gana al desactivar todas las minas seguras.
2. **Funciones Importantes:**
   - **actualizarLCD()**: Actualiza la pantalla LCD con el estado actual del juego.
   - **contarBombasEncendidas()**: Cuenta cuántas bombas están activadas.
   - **posicionYaJugada()**: Verifica si una posición fue jugada.
   - **marcarPosicionJugada()**: Marca una posición como jugada.

3. **Pruebas y Validación**
   - **Conexión del Bluetooth:** Verificar que el módulo Bluetooth se conecta correctamente al dispositivo móvil.

- **Prueba de LEDs:** Asegurar que los LEDs cambian según el estado del juego.
- **Prueba de la Pantalla LCD:** Confirmar que la pantalla muestra los mensajes esperados.
- **Comandos:** Verificar que los comandos enviados por serial y Bluetooth funcionan correctamente.
- **Activación de Bombas:** Probar si el sistema detecta las bombas encendidas y muestra el estado GAME_OVER.