UNIVERSIDAD NACIONAL DE COLOMBIA

1

Software

Documento de requerimientos de software del sistema SCADA

- Cliente de comunicación con el ESP32 para

Arnold Joseph Merchán Rojas **Santiago Guarín Alfaro**

Tabla de contenido

Documentación del Sistema SCADA para Monitoreo de Alcantarillado

PROPÓSITO

El propósito de este sistema SCADA (Supervisión, Control y Adquisición de Datos) es proporcionar una plataforma de monitoreo en tiempo real para sistemas internos de alcantarillado. El sistema permite supervisar variables críticas como temperatura, nivel de agua y flujo, además de controlar remotamente la activación de una bomba de agua. Esta solución integra hardware (sensores conectados a un ESP32) con una interfaz de usuario intuitiva, proporcionando alertas automáticas cuando los parámetros se desvían de los rangos óptimos, lo que permite una gestión proactiva del sistema de alcantarillado y previene posibles fallos o desbordamientos (el sistema ha sido adaptado a su uso aplicado a un motor DC con múltiples sensores, pero se explicara bajo la premisa de su aplicación a un sistema de alcantarillado).

ALCANCE DEL PRODUCTO

El sistema SCADA fue diseñado originalmente para el monitoreo continuo de una red de alcantarillado. No obstante, debido a restricciones de tiempo, fue necesario adaptar su aplicación al seguimiento de un motor DC, no obstante el sistema puede ser adaptado sin problemas a su aplicación original dentro de un sistema de alcantarillado.

Hardware

- Microcontrolador ESP32 que actúa como servidor para recolección de datos
- Sensores de temperatura: DHT11, nivel de agua: sensor ultrasonido HC-SR04 y flujo: switch de flujo
- Sistema de control de bomba de agua mediante relé utilizamos motor dc

Software

- Cliente de comunicación con el ESP32 para intercambio de datos
- Interfaz gráfica para visualización de datos y control del sistema
- Sistema de autenticación y registro de usuarios mediante Firebase
- Módulo de almacenamiento de datos históricos
- Sistema de alertas basado en umbrales predefinidos

Funcionalidades principales

- Monitoreo en tiempo real de temperatura, nivel de agua y flujo
- Visualización gráfica de tendencias de datos
- Registro e inicio de sesión de usuarios
- Visualización de datos históricos
- Sistema de alertas automáticas

El sistema no incluye el mantenimiento físico del alcantarillado ni la configuración inicial del hardware, centrándose exclusivamente en el monitoreo y control.

CARACTERÍSTICAS DEL USUARIO

El sistema está diseñado para ser utilizado por dos tipos principales de usuarios:

Administradores del sistema de alcantarillado

- **Perfil**: Personal con conocimientos básicos de sistemas de alcantarillado
- **Responsabilidades**: Monitoreo constante del sistema, respuesta ante alertas, control de la bomba de agua revisión de datos históricos para planificación de mantenimiento preventivo
- **Habilidades técnicas**: Familiaridad con interfaces digitales básicas y comprensión de conceptos básicos de sistemas SCADA

---

[*] Revista Argentina de Trabajos Estudiantiles. Patrocinada por la IEEE.

UNIVERSIDAD NACIONAL DE COLOMBIA

se requiere un conocimiento básico en sistemas de alcantarillado y en el manejo de interfaces digitales para un correcto manejo del sistema SCADA

REQUERIMIENTOS FUNCIONALES

1. Autenticación de usuarios

- El sistema permitirá el registro de nuevos usuarios con nombre de usuario y contraseña
- El sistema verificará las credenciales de los usuarios registrados durante el inicio de sesión
- El sistema almacenará los datos de usuario en la base de datos Firebase

```
def inicio_sesion():
    ventana_inicio=tk.Toplevel(ventana)
    ventana_inicio.attributes('-fullscreen',1)
    ventana_inicio.config(bg='white')
    iniciar= tk.Label(ventana_inicio,text='inicie sesión',font=(fuentetitbrick),bg='white',fg='black')
    iniciar.place(x=200,y=20)
    boton_volver=tk.Button(ventana_inicio,text='volver',font=(fuentebotonesventanas),bg='#000000',fg='white',relief='raised',command=ventana_i
    boton_volver.place(x=1180,y=20)
    usuario=tk.Label(ventana_inicio,text='usuario',font=(fuenteusucontra),bg='white',fg='black')
    usuario.place(x=500,y=220)
    caja_texto_usuario = tk.Text(ventana_inicio, height=1, width=40)
    caja_texto_usuario.place(x=475,y=310)
    contraseÑa=tk.Label(ventana_inicio,text='contraseña',font=(fuenteusucontra),bg='white',fg='black')
    contraseña.place(x=450,y=350)
    caja_texto_contraseña = tk.Text(ventana_inicio, height=1, width=40)
    caja_texto_contraseña.place(x=475,y=425)

    text_area = tk.Text(ventana_inicio,height=3,width=50)
    text_area.place(x=500,y=450)
    def inicio():
        texto_usuario=caja_texto_usuario.get("1.0", tk.END).strip()
        texto_contraseña=caja_texto_contraseña.get("1.0", tk.END).strip()
        referencia= db.reference(f'usuarios/{texto_usuario}')
        name=caja_texto_usuario.get("1.0", tk.END).strip()
        dato_usuario=referencia.get()
        contr_usua=dato_usuario.get('contraseña')
        print(contr_usua)
        print(texto_contraseña)
        if (contr_usua== texto_contraseña):

            text_area.insert(tk.END,f'Hola {name}, Bienvenido desde la siguiente pestaña podrá observar lo referente a su sistema de alcantal
            def destruir_inicio():
                ventana_inicio.destroy
            #ventana_inicio.after(5000,destruir_inicio)
            #ventana_inicio.destroy
            #destruir_inicio()
            Scada()
```

2. Visualización de datos en tiempo real

- El sistema mostrará los valores actuales de temperatura, nivel de agua y flujo
- El sistema presentará gráficas de tendencias para temperatura y nivel de agua
- El sistema actualizará los datos visualizados cada pocos segundo
- El sistema incluirá una animación que represente el funcionamiento de la bomba

```
    else:
        text_area.insert(tk.END,f'Hola {name}, el usuario o la contraseñaa estan incorrectos')
boton_sesion=tk.Button(ventana_inicio,text='iniciar sesion',font=(fuentebotones),bg='#000000',fg='white',relief='raised',command=inicio)
boton_sesion.place(x=500,y=550)
```

```
frame_temperatura = Frame(canvas_graficas,  bg='blue',pady=20,padx=20)
frame_temperatura.grid(column=0,row=0, sticky='nsew')
def cambios_de_valores_temperatura():
    plt.cla()
    if None in eje_y_grafica_temp:
        eje_y_grafica_temp.remove(None)
        eje_y_grafica_temp.append(last_line_T)
    else:
        eje_y_grafica_temp.pop(0)
        eje_y_grafica_temp.append(last_line_T)
    figura, axs = plt.subplots( dpi=80, figsize=(5,4),
    sharey=True, facecolor='#00f9f844')
    figura.suptitle('Grafica de Temperatura')
    axs.plot(eje_x_grafica_temp, eje_y_grafica_temp, color = 'm')
    canvas = FigureCanvasTkAgg(figura, master = frame_temperatura)  # Crea el area de dibujo en Tkinter
    canvas.draw()
    canvas.get_tk_widget().grid(column=0, row=0)
    ventana.after(1000,cambios_de_valores_temperatura)
    plt.close(figura)
frame_temperatura.after(100,cambios_de_valores_temperatura)

frame_nivel = Frame(canvas_graficas,  bg='blue',pady=20,padx=20)
frame_nivel.grid(column=1,row=0, sticky='nsew')
def cambios_de_valores_nivel():
    if None in eje_y_grafica_nivel:
        eje_y_grafica_nivel.remove(None)
        eje_y_grafica_nivel.append(last_line_N)
    else:
        eje_y_grafica_nivel.pop(0)
        eje_y_grafica_nivel.append(last_line_N)
    figura, axs = plt.subplots( dpi=80, figsize=(5,4),
    sharey=True, facecolor='#00f9f844')
    figura.suptitle('Grafica de Nivel del Agua')
    axs.plot(eje_x_grafica_nivel, eje_y_grafica_nivel, color = 'm')
    canvas = FigureCanvasTkAgg(figura, master = frame_nivel)  # Crea el area de dibujo en Tkinter
    canvas.draw()
    canvas.get_tk_widget().grid(column=1, row=0)
```

3. Sistema de alertas

- El sistema mostrará advertencias cuando la temperatura supere los 25°C
- El sistema mostrará advertencias cuando la diferencia entre la distancia del sensor y el nivel de agua sea inferior a 10 unidades y el flujo sea 0
- El sistema presentará las alertas en un área específica de la interfaz con resaltado en color rojo

```
def advertencia_de_temperatura():
    if (last_line_T >25):
        Advertencia_2.config(text='¡¡¡ADVETENCIA!!!\n La temperatura  actual\nsobrepasa la temperatura\noptima',fg='red')
    else:
        Advertencia_2.config(text='')
    ventana.after(100,advertencia_de_temperatura)
ventana.after(100,advertencia_de_temperatura)
def advertencia_de_flujo():
    if ((last_line_N<10) and (last_line_F==0)):
        Advertencia.config(text='¡¡¡ADVETENCIA!!!\n El flujo de agua actual\nes menor al flujo\noptimo recomendado',fg='red')
    else:
        Advertencia.config(text='aquí se presentaran las\nadvertencias presentadas por el \nsistema',font=(fuentescada,15),bg='white',fg=
    ventana.after(100,advertencia_de_flujo)
ventana.after(100,advertencia_de_flujo)
```

4. Registro histórico de datos

- El sistema almacenará todos los datos
- de datos históricos en formato texto
- El sistema mostrará el número de registros almacenados

```
    ventana.after(1000,cambios_de_valores_nivel)
    plt.close(figura)
frame_nivel.after(100,cambios_de_valores_nivel)
```

```
def lectura_temp():
    with open(temperatura_file, 'r')as f:
        global temp_line
        for i in f:
            temp_line=i
Lectemphilo=threading.Thread(target=lectura_temp)
Lectemphilo.start()
Lec_Est_hilo=threading.Thread(target=lectura_estado)
Lec_Est_hilo.start()
```

UNIVERSIDAD NACIONAL DE COLOMBIA

```
def historico():
    cerrar()
    ventana_historico=tk.Toplevel(ventana)
    ventana_historico.attributes('-fullscreen',1)
    ventana_historico.config(bg='white')
    LDAtos= tk.Label(ventana_historico,text='Datos historicos de sensores',font=(fuenteusucontra),bg='white',fg='black')
    LDAtos.place(x=150,y=35)
    boton_volver=tk.Button(ventana_historico,text='volver',font=(fuentebotonesventanas),bg='#000000',fg='white',relief='raised',command=ve
    boton_volver.place(x=1180,y=20)
    canvas_camptext_histo =tk.Canvas(ventana_historico,width=500,height=500,bg='white')
    canvas_camptext_histo.place(x=450,y=200)
    histo_area = scrolledtext.ScrolledText(ventana_historico, width=60, height=10)
    histo_area.pack()
    canvas_camptext_histo.create_window(250,250, window=histo_area)

    def lectura_estado():
        global estado_line
        with open(estado, 'r')as f:
            for i in f:
                estado_line=i
    def lectura_flujo():
        with open(flujo_file, 'r')as f:
            global flujo_line
            for i in f:
                flujo_line=i
    def lectura_nivel():
        global nivel_line
        a=0
        with open(nivel_file, 'r')as f:
            for i in f:
                a=1
                nivel_line=i
                histo_area.insert(tk.END,f'Datos Registrados: {a}Temp={temp_line} /// Flujo={flujo_line} /// Nivel={nivel_line} /// Estado
```

5. Medición de Temperatura

- El sistema utilizará sensores DHT11 para obtener la temperatura.
- El sistema actualizará la medición de la temperatura en intervalos regulares
- el sistema mostrará los valores en la consola.

```
pin_04 = dht.DHT22(Pin(15))           # crea el objeto pin_04 para un módulo DHT22 en el pin GPIO 04

pin_04.measure()                      # realiza la lectura de la temperatura y humedad
temperatura = pin_04.temperature()    # asigna a la variable "temperatura" la temperatura
print ("La temperatura es de " + str(temperatura) + "ºC ")
```

6. Medición de Flujo

- El sistema utilizará un sensor de flujo configurado como entrada digital con resistencia pull-up.
- El sensor leerá continuamente su estado (activo/inactivo) para detectar la presencia de flujo.
- El valor obtenido del sensor de flujo se registrará y se utilizará en la lógica de control del actuador.

7. Medición de Distancia con Sensor Ultrasónico

- El sistema utilizará un sensor ultrasónico HC-SR04, conectados mediante pines específicos para trigger y echo.
- La distancia se calculará en centímetros, utilizando la medición del tiempo de eco y la velocidad del sonido en el aire.
- Se deberá implementar un mecanismo de validación y timeout para gestionar lecturas fuera del rango permitido.

```
class HCSR04:
    def __init__(self, trigger_pin, echo_pin, echo_timeout_us=500*2*30):
        self.echo_timeout_us = echo_timeout_us
        self.trigger = Pin(trigger_pin, mode=Pin.OUT, pull=None)
        self.trigger.value(0)
        self.echo = Pin(echo_pin, mode=Pin.IN, pull=None)
    def _send_pulse_and_wait(self):
        self.trigger.value(0)
        time.sleep_us(5)
        self.trigger.value(1)
        # Send a 10us pulse.
        time.sleep_us(10)
        self.trigger.value(0)
        try:
            pulse_time = machine.time_pulse_us(self.echo, 1, self.echo_timeout_us)
            return pulse_time
        except OSError as ex:
            if ex.args[0] == 110:
                raise OSError('Out of range')
            raise ex
    def distance_mm(self):
        pulse_time = self._send_pulse_and_wait()
        mm = pulse_time * 100 // 582
        return mm
    def distance_cm(self):
        pulse_time = self._send_pulse_and_wait()
        cms = (pulse_time / 2) / 29.1
        return cms
```

8. Control de Actuadores (Relé)

- El sistema controlará un relé conectado a un pin de salida digital para activar o desactivar dispositivos externos.
- La activación del relé se basará en condiciones específicas derivadas de las lecturas de sensores.
- El estado del relé se deberá comunicar claramente mediante mensajes en la consola o una interfaz remota.

9. Comunicación y Control Remoto

- El sistema implementará un servidor TCP que escuche en un puerto predefinido para aceptar conexiones remotas.
- Se recibirá validará y procesará comandos de control enviados en formato numérico por clientes remotos.
- Se enviarán periódicamente a los clientes conectados los datos de los sensores (temperatura, flujo, distancia) y el estado actual del relé.

UNIVERSIDAD NACIONAL DE COLOMBIA

```
while True:
    conn, addr = server_socket.accept()
    print(f"Conexión establecida desde {addr}")
    try:
        while True:
            # Recibir comando
            data = conn.recv(1024).decode().strip()
            if not data:
                break
            try:
                comando = int(data)  # Convertir a entero
            except ValueError:
                print(f"Error: Comando no válido recibido ({data})")
                continue  # Ignorar y seguir esperando comandos válidos
            print(f"Comando recibido: {comando}")
            pin_temp.measure()
            temperatura = pin_temp.temperature()
            flujo = p2.value()
            ultra1 = distancia1.distance_cm()
            # Control de la bomba
            if (ultra1 < 10) and (flujo == 1) and (temperatura < 26)and(comando==1):
                rele.value(1)  # Activa el relé
                estado_rele = "Encendido"
            else:
                rele.value(0)  # Desactiva el relé
                estado_rele = "Apagado"
                # Enviar datos al cliente
            datos = f"{temperatura},{flujo},{ultra1},{estado_rele}\n"
            conn.sendall(datos.encode())
            print(f"Datos enviados: {datos}")
            time.sleep(3)  # Esperar antes de la siguiente lectura
    except Exception as e:
        print(f"Error: {e}")
    finally:
        conn.close()
        print("Cliente desconectado, esperando nueva conexión...")
```

10. Gestión de Múltiples Sensores

- El sistema soportará la integración simultánea de múltiples sensores para obtener lecturas de diferentes puntos.
- Cada sensor operará de forma independiente sin interferir con las lecturas de los demás, permitiendo su correcta integración en el sistema.

```
p2 = Pin(17, Pin.IN, Pin.PULL_UP)
pin_temp = dht.DHT11(Pin(19))
distancia1 = HCSR04(5, 18)
rele = machine.Pin(16, machine.Pin.OUT)
while True:
    sleep(3)
    pin_temp.measure()
    temperatura = pin_temp.temperature()
    print("La temperatura es de " + str(temperatura) + "ºC" )
    sleep(2)
    # read and print the pin value sensor flujo
    print(f'sensor flujo{p2.value()}')
    sleep(2)
    ultra1=distancia1.distance_cm()
    print ("La distancia promedio es de " + str(ultra1))
    if (ultra1<10)and (p2.value()==1)and(temperatura<26):
        print("Encendiendo relé")
        rele.value(0)  # Activa el relé
        time.sleep(2)  # Espera 2 segundos
        print("Apagando relé")
    if (ultra1>10)or (p2.value()==0)or(temperatura>26):
        rele.value(1)  # Desactiva el relé
        time.sleep(2)
    datos=(f'Temperatura:{temperatura} ºC ///, Flujo:{p2.value()}///, Nivel: {ultra1} ')
```

11. Manejo de Errores y Robustez del Sistema

- Se implementarán mecanismos para detectar errores en las mediciones, como tiempos de espera excesivos.
- El sistema generará y registrará mensajes de error detallados para facilitar el diagnóstico y la corrección de fallos.
- Ante la detección de un error, el sistema deberá continuar operando y mantener el monitoreo general sin interrupciones.

12. Sincronización y Periodicidad de Operaciones

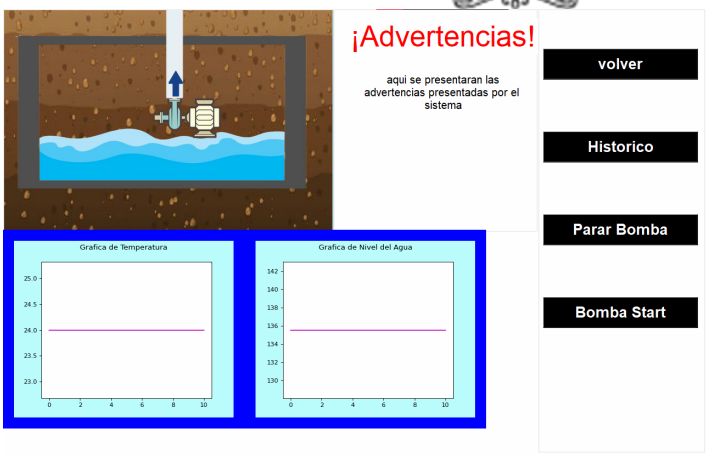
- Las operaciones de lectura de sensores se realizan en intervalos fijos predefinidos para asegurar la estabilidad.
- La actualización de datos y el envío de información a clientes remotos se sincronizan para mantener la coherencia en la transmisión de datos.
- El sistema permitirá la configuración flexible de los intervalos de tiempo, adaptándose a distintos escenarios y requisitos de la aplicación.

REQUERIMIENTOS NO FUNCIONALES

1. Usabilidad

- La interfaz debe ser intuitiva y fácil de usar, permitiendo la operación sin capacitación extensa
- Los botones deben tener etiquetas claras y ser suficientemente grandes para una fácil interacción
- La información crítica debe ser claramente distinguible mediante uso de colores contrastantes
- Las alertas deben ser visualmente prominentes e identificables inmediatamente

UNIVERSIDAD NACIONAL DE COLOMBIA



2. Rendimiento

- El sistema debe actualizar los datos visualizados con una frecuencia preestablecida
- El tiempo de respuesta para el control de la bomba debe ser inferior a 5 segundos
- El sistema guardará constantemente la información captada por los sensores

```
Datos Registrados: 1Temp=24
 /// Flujo=1
 /// Nivel=9.639175
 /// Estado=Apagado

Datos Registrados: 2Temp=24
 /// Flujo=1
 /// Nivel=136.0653
 /// Estado=Apagado

Datos Registrados: 3Temp=24
 /// Flujo=1
 /// Nivel=136.5292
 /// Estado=Apagado

Datos Registrados: 4Temp=24
 /// Flujo=1
 /// Nivel=136.0997
```

3. Confiabilidad

- El sistema será capaz de establecer un contacto con el sistema SCADA bajo una misma red local
- El sistema debe mantener los datos históricos incluso después de cierres o reinicios

4. Seguridad

- El acceso al sistema debe estar protegido mediante autenticación de usuario
- Las contraseñas de usuario deben almacenarse de forma segura en Firebase
- Solo usuarios autenticados pueden acceder a los controles del sistema y datos históricos

5. Compatibilidad

- El sistema debe funcionar en computadoras con sistema operativo Windows
- La aplicación requiere acceso a internet para autenticación en Firebase
- El sistema debe estar conectado a la misma red local que el ESP32

MANUAL DE USUARIO

Inicio de la aplicación

1. Ejecute el archivo ejecucion.py para iniciar el sistema.
2. Se abrirá la ventana principal del sistema con las siguientes opciones:
    - Registrarse
    - Inicio de Sesión
    - Salir

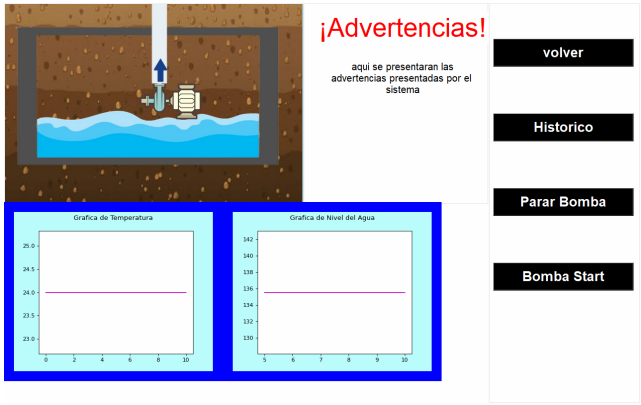
Registro de nuevo usuario

1. Haga clic en el botón "Registrarse".
2. En la ventana de registro, ingrese un nombre de usuario en el campo "usuario".
3. Ingrese una contraseña en el campo "contraseña".
4. Haga clic en el botón "registrarse".
5. Si el usuario ya existe, se mostrará un mensaje indicándolo.
6. Si el registro es exitoso, se mostrará un mensaje de bienvenida y se abrirá automáticamente la interfaz SCADA.

Inicio de sesión

1. Haga clic en el botón "Inicio de Sesión".
2. En la ventana de inicio de sesión, ingrese su nombre de usuario en el campo "usuario".
3. Ingrese su contraseña en el campo "contraseña".
4. Haga clic en el botón "iniciar sesión".
5. Si las credenciales son correctas, se mostrará un mensaje de bienvenida y se abrirá la interfaz SCADA.
6. Si las credenciales son incorrectas, se mostrará un mensaje de error.

Uso de la interfaz SCADA



La interfaz SCADA está dividida en cuatro secciones principales:

Sección de animación (superior izquierda)

- Muestra una representación visual del sistema de alcantarillado.
- Incluye una animación de flujo de agua cuando la bomba está en funcionamiento.

Sección de advertencias (superior derecha)

- Muestra alertas cuando:
  - La temperatura supera los 25°C.
  - El nivel de agua es bajo (menos de 10 unidades) y no hay flujo.

Sección de gráficas (inferior)

- Muestra dos gráficas en tiempo real:
  - Gráfica de temperatura.
  - Gráfica de nivel de agua.
- Las gráficas se actualizan automáticamente cada segundo.
- Se muestran datos de los últimos 10 segundos.

Panel de control (derecha)

- Contiene los siguientes botones:
  - "Volver": Cierra la interfaz SCADA y regresa a la ventana principal.
  - "Histórico": Abre la ventana de datos históricos.
  - "Parar Bomba": Detiene la bomba de agua.

UNIVERSIDAD NACIONAL DE COLOMBIA

- "Bomba Start": Inicia la bomba de agua.

Control de la bomba

1. Para iniciar la bomba, haga clic en el botón "Bomba Start".
2. Para detener la bomba, haga clic en el botón "Parar Bomba".
3. El estado actual de la bomba se refleja en la animación del flujo de agua.

Visualización de datos históricos

1. Haga clic en el botón "Histórico" en el panel de control.
2. Se abrirá una ventana que muestra todos los registros de datos capturados por el sistema.
3. Los datos se presentan en formato texto, incluyendo:
   - Número de registro
   - Valor de temperatura
   - Valor de flujo
   - Valor de nivel
   - Estado de la bomba
4. Use la barra de desplazamiento para explorar todos los datos.
5. Para cerrar la ventana de datos históricos, haga clic en el botón "volver".

Cerrar el sistema

1. Para cerrar la interfaz SCADA, haga clic en el botón "volver" en el panel de control.
2. Para salir completamente de la aplicación, haga clic en el botón "Salir" en la ventana principal.

GUIA DE INSTALACIÓN Y USO

Para la correcta instalación y uso del programa debe seguir los siguientes pasos:

Programación del hardware

Programación del Hardware

Antes de iniciar, asegúrese de que el microcontrolador ESP32 tenga instalado MicroPython y verifique la disponibilidad de las siguientes bibliotecas:

- machine
- dht
- socket
- time

a) Instalación de MicroPython

El firmware de MicroPython puede descargarse desde MicroPython.org y debe ser flasheado en el ESP32 utilizando herramientas como esptool.py.

b) Verificación de Bibliotecas

Las bibliotecas estándar machine, socket y time generalmente vienen preinstaladas en MicroPython. Se recomienda verificar su disponibilidad antes de su uso.

La biblioteca dht puede requerir instalación manual. Para descargarla, acceda al siguiente enlace: https://github.com/micropython/micropython/tree/master/drivers/dht.

Compatibilidad de Hardware

Asegúrese de que la versión del ESP32 que está utilizando sea compatible con la lectura de sensores a través de los siguientes pines:

- Sensor de Flujo (Sich de Flujo) → Pin 25
- Sensor de Temperatura (DHT11) → Pin 19
- Sensor de Ultrasonido (HCSR04)
  - Pin Trig → Pin 5
  - Pin Echo → Pin 18
- Relé → Pin 27

Verifique que su ESP32 tenga suficientes pines GPIO disponibles y que no existan conflictos con otras funciones del microcontrolador que puedan interferir con la correcta lectura de los sensores.

Instalación de Archivos en el ESP32

Asegúrese de instalar los siguientes archivos en el ESP32. Estos se encuentran en la carpeta codigo_hard dentro del repositorio en GitHub:

UNIVERSIDAD NACIONAL DE COLOMBIA

- bomb_LTE4.py
- hcsr04.py
- webrepl_cfg.py
- wifi.py
- wifi_config.py

Configuración de Red WiFi

1. Abra el archivo wifi_config.py.

Modifique las siguientes líneas con los datos de su red WiFi:
python

SSID = "poner nombre de la red wifi a la que va a conectar su ESP32"
PSS = "poner la contraseña de dicha red"

2.

Configuración de Pines

- Abra el archivo bomb_LTE4.py.
- Luego del comentario # Configuración de sensores, encontrará la asignación de pines para cada sensor.
- Modifique los pines si es necesario para que coincidan con su hardware.

CONFIGURACIÓN DEL SOFTWARE

1. Introducción

Este documento describe el procedimiento para la instalación y configuración del software necesario para el correcto funcionamiento del sistema basado en ESP32 y su integración con Firebase y la interfaz SCADA.

2. Descarga de Archivos

1. Clonar o descargar el repositorio desde GitHub y almacenar todos los archivos en una misma carpeta.
2. Crear una base de datos en **Firebase Realtime Database**.
3. Descargar el archivo JSON de configuración de Firebase y guardarlo en la misma carpeta que los demás archivos del proyecto.

3. Configuración del Script interfaz.py

1. Abrir el archivo interfaz.py con un editor de texto o un IDE.
2. Modificar las credenciales de Firebase:
    - Especificar el nombre del archivo JSON descargado.
    - Ingresar la URL de la base de datos Firebase.

4. Instalación de Dependencias

Ejecutar los siguientes comandos en la terminal para instalar las bibliotecas requeridas:

pip install matplotlib
pip install firebase-admin
pip install pillow

5. Configuración del Script cliente.py

1. Abrir el archivo cliente.py.
2. Modificar la dirección IP del ESP32, reemplazandola por la correspondiente al dispositivo ESP32 utilizado.

6. Ejecución del Sistema

6.1. Verificación de Conexión de Red

Asegurar que tanto el ESP32 como la computadora estén conectados a la misma red WiFi para garantizar la comunicación.

6.2. Ejecución del Código en el ESP32

1. Subir y ejecutar el código bomb_LTE4 en el ESP32.
2. En la terminal del ESP32, se deberá visualizar el mensaje: **"Esperando conexión"**.

6.3. Ejecución del Sistema en la Computadora

1. Ejecutar ejecucion.py, lo que iniciará la interfaz SCADA.
2. Verificar en la interfaz que se haya establecido correctamente la conexión entre el ESP32 y la computadora.

UNIVERSIDAD NACIONAL DE COLOMBIA

3. En caso de interrupción de la conexión, ejecutar nuevamente cliente.py.
4. Una vez establecida la conexión, se podrá operar la interfaz SCADA sin inconvenientes.

Este documento proporciona una guía detallada para la correcta instalación y configuración del sistema. Para información adicional sobre el uso de la interfaz, se recomienda consultar el manual de usuario.

ESQUEMA ELÉCTRICO