



**unl**

Universidad  
Nacional  
de Loja

## Universidad Nacional de Loja

### Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Telecomunicaciones

Diseño e implementación de un prototipo portátil de monitoreo de variables  
eléctricas para las edificaciones de la Universidad Nacional de Loja

#### MANUAL DEL PROGRAMADOR

**AUTOR:**

Tito Xavier Zhanay Avila

**DIRECTOR:**

Ing. Luis Eduardo Rodríguez Montoya. Mg. Sc.

*Loja – Ecuador*

2025

## Tabla de Contenido

1. Introducción .....	3
2. Requisitos <i>del sistema</i> .....	4
2.1. Requisitos de hardware .....	4
2.2. Requisitos de software .....	4
3. Instalación del entorno de desarrollo.....	5
3.1. Instalación y configuración en el computador (ESP32).....	5
3.1.1. <i>Instalación del Arduino IDE</i> .....	5
3.1.2. <i>Configurar soporte para ESP32</i> .....	5
3.1.3. <i>Instalar las librerías necesarias</i> .....	7
3.1.4. <i>Configurar placa y puerto</i> .....	8
3.2. Configuración del servidor local (Raspberry Pi 5) .....	8
3.2.1. <i>Instalación del sistema operativo</i> .....	8
3.2.2. <i>Instalación de InfluxDB</i> .....	10
3.2.3. <i>Instalación de Grafana</i> .....	13
3.2.4. <i>Acceso a Grafana</i> .....	13
3.2.5. <i>Vinculación de Grafana con InfluxDB</i> .....	14
4. Descripción del código fuente del ESP32.....	16
4.1. Inclusión de librerías y declaración de objetos .....	16
4.2. Calibración y definición de sensores.....	17
4.3. Conexión Wi-Fi .....	17
4.4. Pantalla OLED .....	18
4.5. Cálculo de variables eléctricas .....	18
4.6. Envío de datos a InfluxDB .....	18
4.7. Control de tiempos .....	19
5. Flujo general del programa.....	20
5.1. Etapas del flujo de ejecución .....	20
5.2. Diagrama de flujo del sistema .....	20
5.3. Consideraciones importantes .....	22
6. Recomendaciones para mantenimiento, escalabilidad y personalización .....	22
6.1. Buenas prácticas de mantenimiento .....	22
6.2. Escalabilidad del sistema .....	22
6.3. Seguridad y respaldo .....	23

## **I. Introducción**

Este manual ha sido creado con el objetivo de servir como una guía técnica clara y completa para la instalación, configuración, operación y mantenimiento del sistema de monitoreo de variables eléctricas desarrollado dentro del proyecto de tesis titulado: **“Diseño e implementación de un prototipo portátil de monitoreo de variables eléctricas para las edificaciones de la Universidad Nacional de Loja”**.

El contenido está dirigido principalmente a programadores, técnicos, investigadores y futuros desarrolladores que requieran conocer en detalle el funcionamiento interno del sistema embebido, realizar ajustes en su programación o replicar su estructura para nuevas aplicaciones. A lo largo del documento se describe, de manera ordenada, el entorno de desarrollo utilizado, las herramientas empleadas, la estructura y lógica del código implementado en el microcontrolador ESP32, así como la configuración de los servicios backend responsables de la adquisición, almacenamiento y visualización de los datos registrados.

El sistema está diseñado para funcionar de manera autónoma y confiable en condiciones reales, empleando sensores no invasivos y comunicación inalámbrica para enviar la información a una plataforma de análisis energético. Su arquitectura modular permite adaptarlo sin dificultad a diferentes tipos de redes eléctricas (monofásicas, bifásicas o trifásicas), lo que asegura su flexibilidad y capacidad de expansión.

En definitiva, este manual constituye una pieza clave del proyecto, ya que no solo facilita la comprensión técnica del desarrollo, sino que también respalda su correcta implementación, mantenimiento y posible evolución a futuro.

## 2. Requisitos del sistema

En esta sección se especifican los elementos de hardware y software necesarios para instalar, configurar y garantizar el correcto funcionamiento del sistema de monitoreo trifásico desarrollado bajo un enfoque de arquitectura IoT.

### 2.1. Requisitos de hardware

Cantidad	Componente	Descripción técnica
1	<b>ESP32 DevKit V1</b>	Microcontrolador principal con conectividad Wi-Fi integrada, encargado del procesamiento y envío de datos.
3	<b>Sensor SCT013-100A</b>	Sensor de corriente no invasivo tipo “clamp”, con capacidad de medición hasta 100 A. Se utiliza uno por cada fase.
3	<b>Sensor ZMPT101B</b>	Sensor de voltaje de precisión para corriente alterna. Se instala uno por cada fase.
1	<b>Pantalla OLED SH1106</b>	Pantalla de 128x64 píxeles con interfaz I2C, utilizada para mostrar en tiempo real las variables medidas.
1	<b>Raspberry Pi 5</b>	Servidor local para el almacenamiento y visualización de los datos.
1	<b>Cargador Infinitix 70 W</b>	Fuente de alimentación estable para la Raspberry Pi, asegurando su funcionamiento continuo.

### 2.2. Requisitos de software

Herramienta / Librería	Descripción y propósito
<b>Arduino IDE (v2.0 o superior)</b>	Entorno de programación para el ESP32.
<b>ESP32 Board Package</b>	Paquete de soporte para placas ESP32, instalable desde el Gestor de placas del IDE.
<b>Librería EmonLib</b>	Permite la medición de corriente y voltaje en corriente alterna (CA).
<b>Librería Wire</b>	Librería base para la comunicación I2C, utilizada por la pantalla OLED.
<b>Adafruit_GFX + SH1106</b>	Conjunto de librerías para la gestión gráfica de la pantalla OLED.
<b>InfluxDB</b>	Base de datos de series temporales instalada en la Raspberry Pi para almacenar mediciones.
<b>Grafana</b>	Plataforma de visualización conectada a InfluxDB para el análisis y monitoreo en tiempo real.
<b>Raspberry Pi OS</b>	Sistema operativo (en versión Lite o Desktop) que sirve de base al servidor local.
<b>Cliente HTTP (ESP32)</b>	Módulo interno del ESP32 para el envío de datos mediante el protocolo HTTP.

**Nota:** El sistema está diseñado para trabajar de forma completamente local dentro de una red LAN. La instalación de los servicios en la Raspberry Pi se realizó sin el uso de contenedores Docker, lo que facilita la administración directa del sistema operativo y el mantenimiento general.

### 3. Instalación del entorno de desarrollo

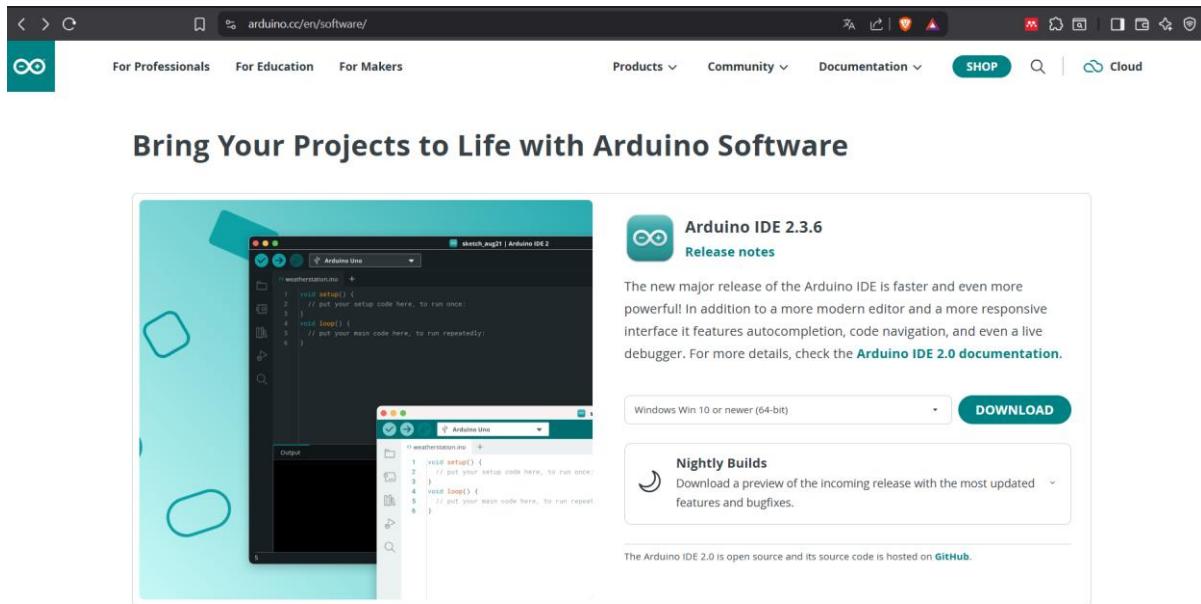
En esta sección se describen los pasos necesarios para instalar y configurar el entorno de desarrollo tanto en el computador (para programar el ESP32) como en la Raspberry Pi 5 (para actuar como servidor local). Este entorno permite la adquisición, transmisión, almacenamiento y visualización de datos de variables eléctricas en tiempo real.

#### 3.1. Instalación y configuración en el computador (ESP32)

##### 3.1.1. Instalación del Arduino IDE

1. Descargar la versión más reciente desde:

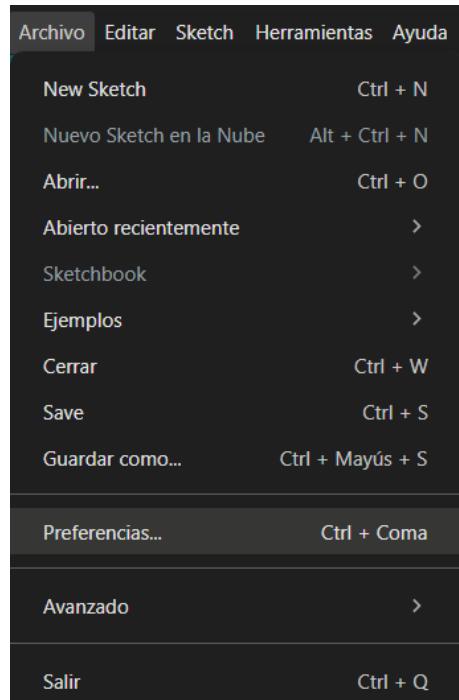
<https://www.arduino.cc/en/software>



2. Instalar como administrador siguiendo las instrucciones del instalador.

##### 3.1.2. Configurar soporte para ESP32

1. Abrir el IDE de Arduino.
2. Ir a **Archivo → Preferencias**.



3. En el campo **Gestor de URLs Adicionales de Tarjetas**, añadir:

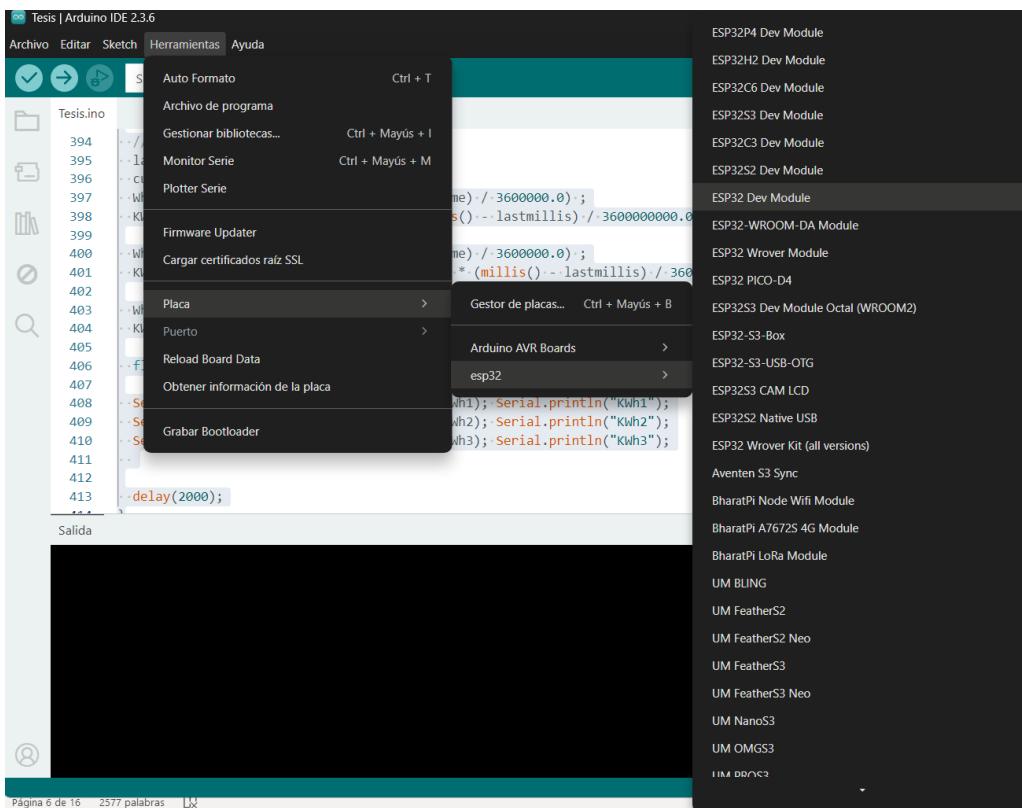
[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)

A screenshot of the Arduino IDE's 'Preferences' dialog box. The dialog has two tabs at the top: 'Configuración' (selected) and 'Red'. The 'Configuración' tab contains several configuration options:

- Ruta del Sketchbook: c:\Users\Usuario\Documents\Arduino
- Ver los ficheros dentro de los bocetos
- Tamaño de letra del editor: 14
- Escala de la interfaz:  Automático 100 %
- Tema de color: Light
- Lenguaje del editor: español (Reload required)
- Mostrar salida verbose durante alertas de compilación:  Compliar  Carga
- Verificar el código después de cargarlo
- Autoguardado
- Sugerencias rápidas del editor

At the bottom of the configuration section, there is a field for 'URLs adicionales de gestor de placas:' with the value 'https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json' and a 'RELOAD' button next to it. At the bottom right of the dialog are 'CANCELAR' and 'ACEPTAR' buttons.

4. Ir a **Herramientas** → **Placa** → **Gestor de tarjetas**.
5. Buscar **esp32** e instalar el paquete oficial de Espressif.

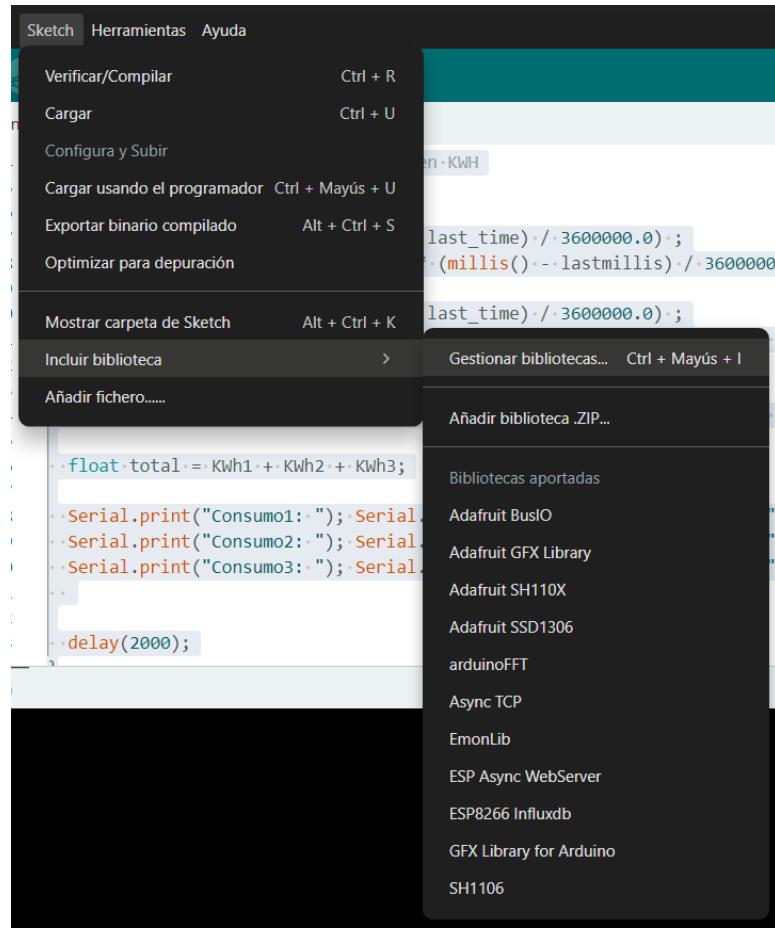


### 3.1.3. Instalar las librerías necesarias

Ir a **Programa** → **Incluir Librería** → **Administrar bibliotecas...**

Buscar e instalar las siguientes:

Librería	Uso principal
EmonLib	Lectura de corriente y voltaje
Wire	Comunicación I2C
Adafruit_GFX	Soporte gráfico OLED
Adafruit_SH1106	Controlador específico de pantalla
WiFi y HTTPClient	Comunicación por red Wi-Fi



Es recomendable reiniciar el IDE luego de instalar todas las librerías.

### **3.1.4. Configurar placa y puerto**

1. Conectar el ESP32 mediante cable USB.
2. Ir a **Herramientas → Placa → ESP32 Dev Module**.
3. Seleccionar el **Puerto COM** correspondiente (se detecta automáticamente).
4. Opcional: ajustar velocidad de carga (por defecto 115200 baudios).

## **3.2. Configuración del servidor local (Raspberry Pi 5)**

La Raspberry Pi debe tener acceso a la misma red local que el ESP32.

### **3.2.1. Instalación del sistema operativo**

1. Descargar **Raspberry Pi Imager**: <https://www.raspberrypi.com/software/>

## Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install **Raspberry Pi OS** and other operating systems to a microSD card, ready to use with your Raspberry Pi.

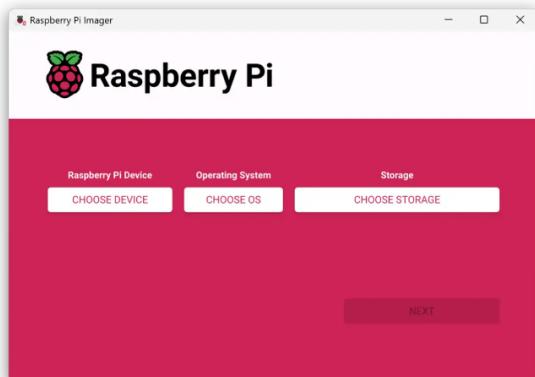
Download and install Raspberry Pi Imager on a computer with an SD card reader. Insert the microSD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Windows](#)

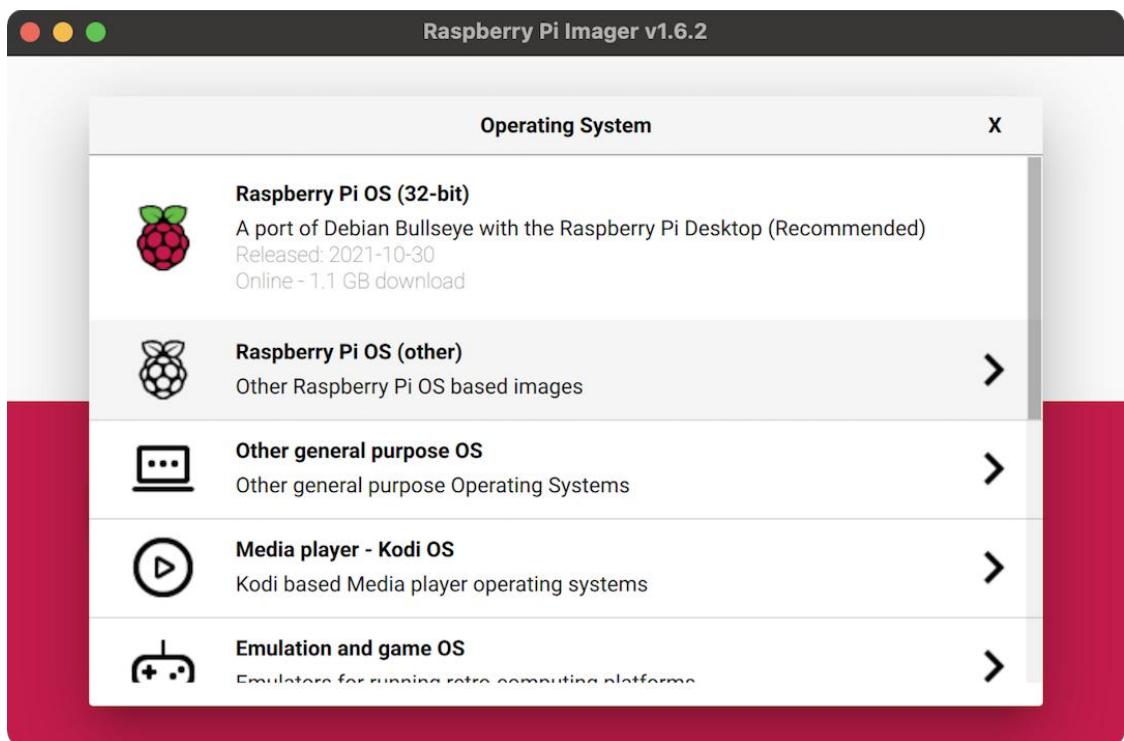
[Download for macOS](#)

[Download for Debian or Ubuntu \(x86\\_64\)](#)

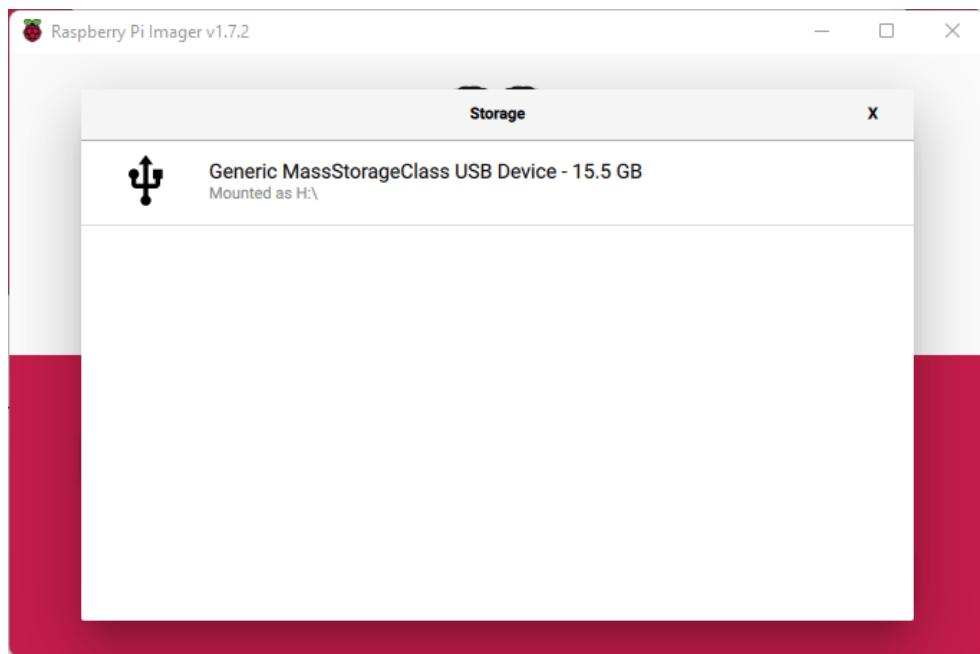
To install on **Raspberry Pi OS**, type  
sudo apt install rpi-imager  
into a terminal window



2. Seleccionar **Raspberry Pi OS Lite** (o Desktop si se desea interfaz gráfica).



3. Grabar la imagen en una microSD (mínimo 16 GB).



4. Insertar en la Raspberry Pi e iniciar el sistema.
5. Configurar idioma, red, y contraseña de usuario.

### 3.2.2. Instalación de InfluxDB

Con el sistema operativo de la Raspberry Pi instalado y en funcionamiento, se debe proceder a la instalación del servicio **InfluxDB** utilizando la terminal local o una conexión remota mediante herramientas como **PuTTY**.

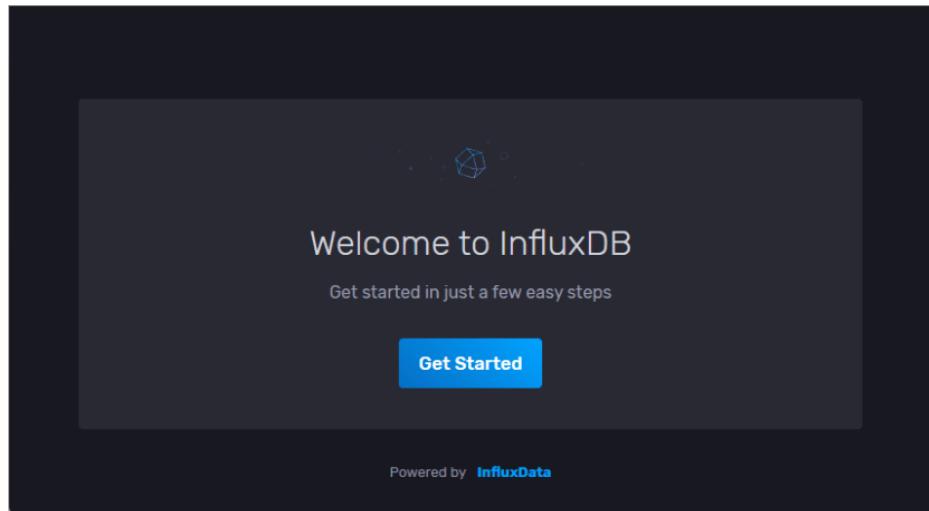
Los pasos a seguir son los siguientes:

```
sudo apt update
sudo apt upgrade -y
wget -qO- https://repos.influxdata.com/influxdb.key | gpg --dearmor | sudo tee /usr/share/keyrings/influx-archive-keyring.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/influx-archive-keyring.gpg]
https://repos.influxdata.com/debian \$ (lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
sudo apt update && sudo apt install -y influxdb
sudo systemctl unmask influxdb.service
sudo systemctl start influxdb
sudo systemctl enable influxdb.service
```

Con este procedimiento, el servicio **InfluxDB** queda instalado y configurado para iniciarse automáticamente cada vez que el sistema se encienda. Para acceder a la interfaz gráfica de administración de InfluxDB, se debe abrir un navegador web y dirigirse a:

```
http://<IP-de-la-RaspberryPi>:8086
```

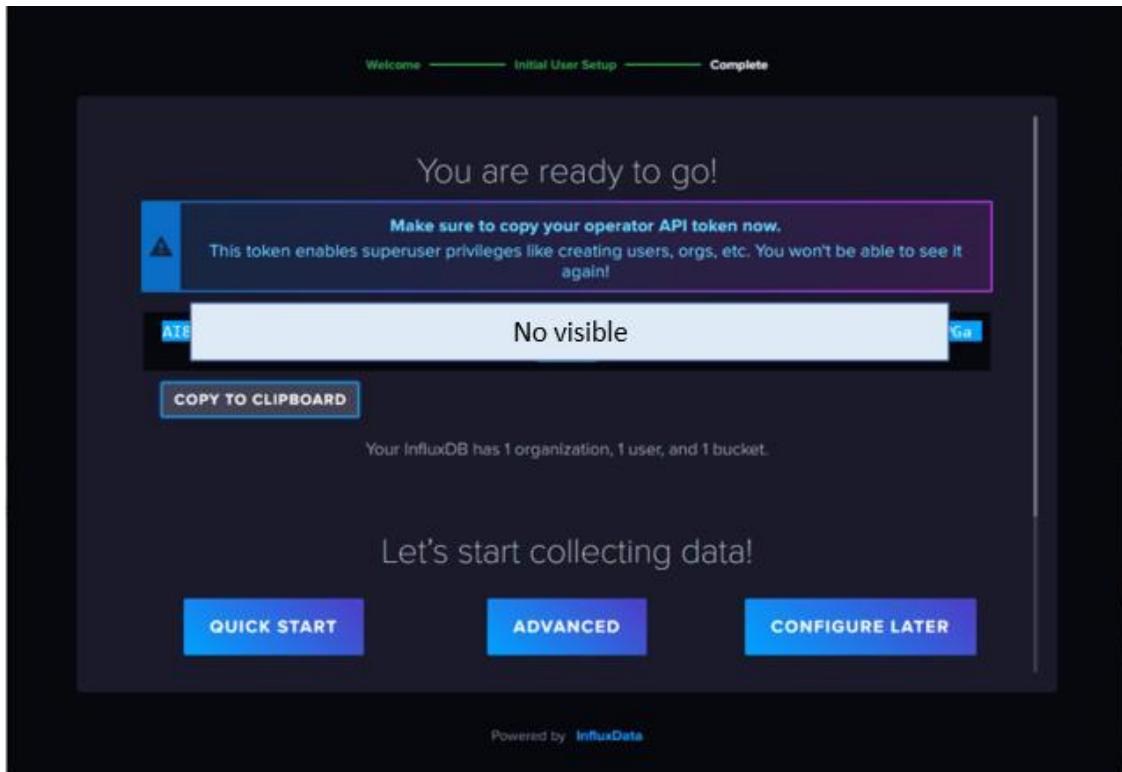
donde <IP-de-la-RaspberryPi> corresponde a la dirección asignada por el router en la red local.



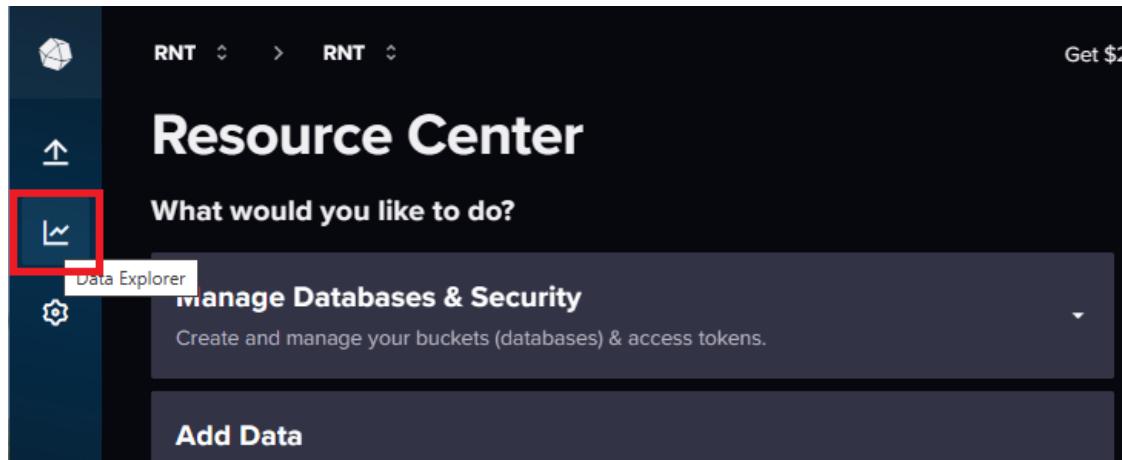
Nos pide crear un usuario y contraseña

The image shows the "Initial User Setup" step of the InfluxDB setup process. The top navigation bar indicates the current step: "Welcome" → "Initial User Setup" → "Complete". The main title "Setup Initial User" is centered above a descriptive text: "You will be able to create additional Users, Buckets and Organizations later". Below this, there are four input fields: "Username" (with a placeholder box), "Password" (with a placeholder box), "Confirm Password" (with a placeholder box), and "Initial Organization Name" (with a placeholder box). A tooltip for "Initial Organization Name" defines it as "An organization is a workspace for a group of users." Below these fields is another section for "Initial Bucket Name" (with a placeholder box) and a tooltip defining it as "A bucket is where your time series data is stored with a retention policy." At the bottom of the form is a blue "Continue" button.

Y nos crea la API key que usaremos en la esp32 para enviar los datos.



Luego vamos a resource center → data explorer



Y permite seleccionar los datos que llegan de la esp32

The screenshot shows the InfluxDB Data Explorer interface. At the top, there are buttons for '+ New Script', 'OPEN', and 'SAVE'. Below that, there's a 'Schema Browser' section with a 'Bucket' dropdown set to 'ESP32' and a 'Measurement' dropdown set to 'wifi\_status'. A search bar says 'Search fields and tag keys'. On the left, there are sections for 'Fields' (with 'rsssi' checked), 'Tag Keys' (with 'SSID' checked), and 'device' (with 'ESP32' checked). On the right, there's a code editor window displaying a SQL query:

```
1 SELECT *
2 FROM "wifi_status"
3 WHERE
4 time >= now()
5 AND
6 ("rsssi" IS NOT
7 AND
8 "SSID" IN ('HE...')
```

### 3.2.3. Instalación de Grafana

Grafana permite visualizar los datos registrados en InfluxDB mediante paneles interactivos. Su instalación se realiza con:

```
wget -O- https://packages.grafana.com/gpg.key | gpg --dearmor | sudo tee /usr/share/keyrings/grafana-archive-keyring.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/grafana-archive-keyring.gpg] https://packages.grafana.com/oss/deb stable main" | sudo tee /etc/apt/sources.list.d/grafana.list

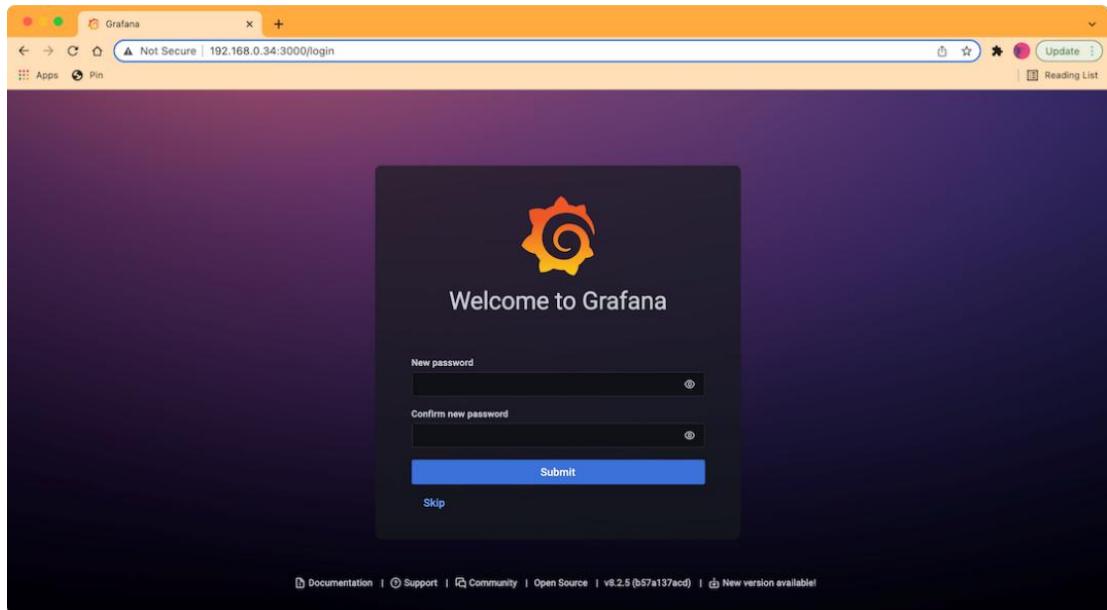
sudo apt update && sudo apt install -y grafana

sudo systemctl unmask grafana-server.service
sudo systemctl start grafana-server
sudo systemctl enable grafana-server.service
```

### 3.2.4. Acceso a Grafana

1. Abrir un navegador en un equipo conectado a la misma red local.
2. Ingresar la dirección:

`http://<IP-de-la-RaspberryPi>:3000`



### 3. Usar las credenciales por defecto:

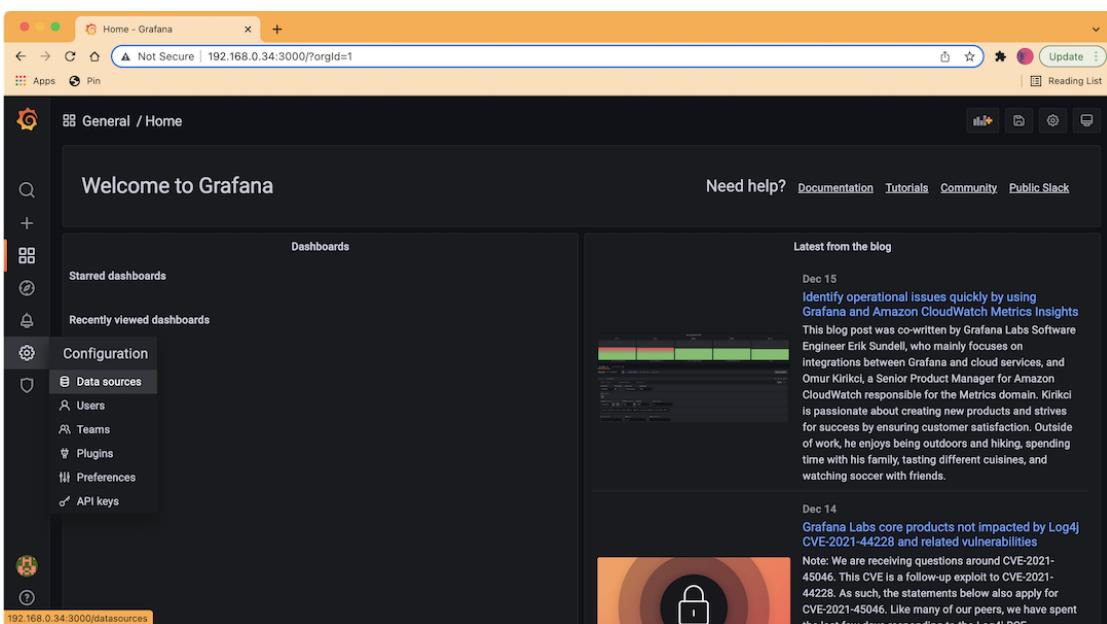
Usuario: admin

Contraseña: admin

#### 3.2.5. Vinculación de Grafana con InfluxDB

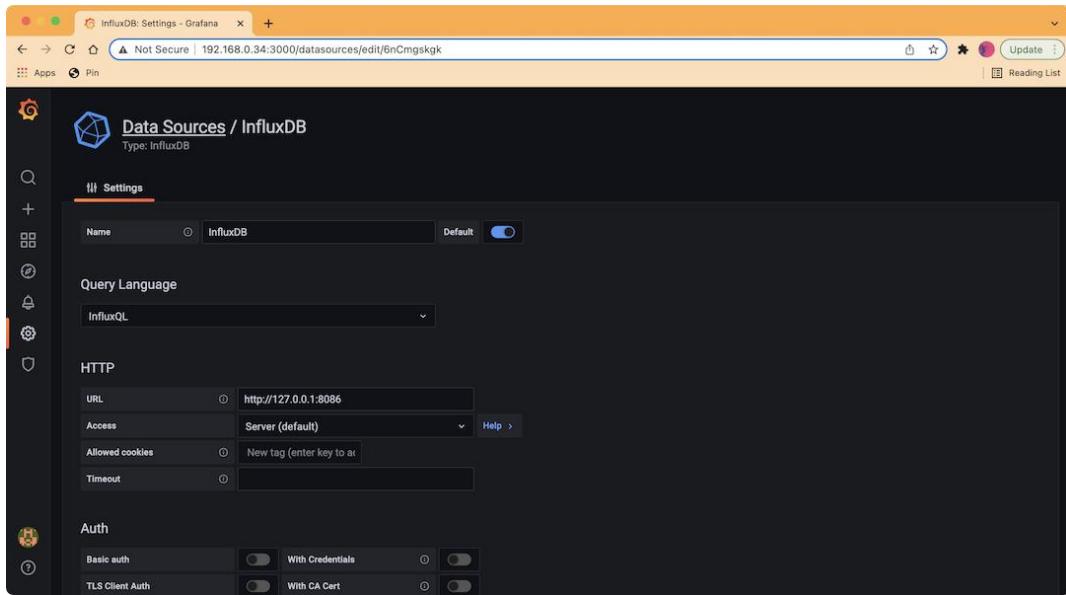
Dentro de Grafana, se debe configurar la conexión con la base de datos:

1. Ir a **Configuration** → **Data Sources** → **Add data source**.
2. Seleccionar **InfluxDB**.



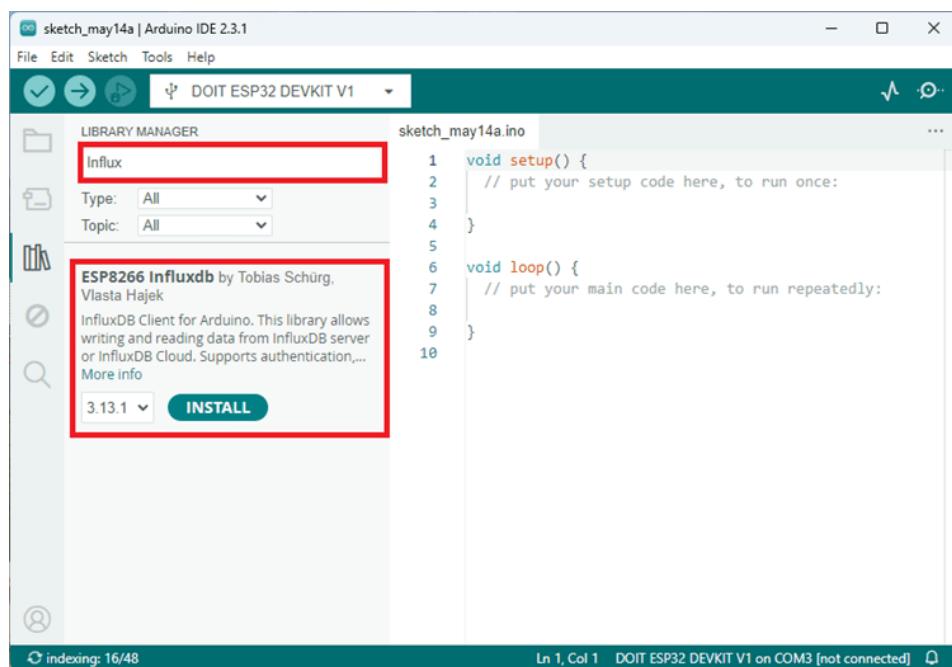
### 3. Configurar los parámetros:

- **URL:** `http://localhost:8086`
- **Database:** `mediciones`
- **HTTP Method:** `POST`



#### 4. Guardar y probar la conexión.

Para que la esp32 envíe los datos se tiene que usar la librería



Con este procedimiento, el entorno queda completamente preparado para recibir datos desde el ESP32, almacenarlos en InfluxDB y visualizarlos en tiempo real mediante Grafana.

## 4. Descripción del código fuente del ESP32

El firmware del sistema está contenido en un único archivo con extensión. ino, desarrollado en Arduino IDE. Este código se encarga de integrar todos los componentes del prototipo, desde la adquisición de datos eléctricos hasta la transmisión hacia la base de datos InfluxDB para su posterior visualización en Grafana.

La estructura general del código contempla:

1. Inclusión de librerías necesarias para cada módulo del sistema.
2. Declaración de objetos que representan sensores, pantalla y comunicación.
3. Calibración de cada canal de medición.
4. Conexión automática a la red Wi-Fi configurada.
5. Lectura y procesamiento de valores eléctricos.
6. Visualización en la pantalla OLED.
7. Envío de datos en formato Line Protocol mediante protocolo HTTP.
8. Control de intervalos de muestreo para optimizar recursos y evitar sobrecarga de red.

### 4.1. Inclusión de librerías y declaración de objetos

En las primeras líneas del archivo se incluyen las librerías que permiten habilitar funciones clave del sistema:

```
#include <WiFi.h>
#include <WiFiMulti.h>
#include <EmonLib.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH1106.h>
#include <HTTPClient.h>
```

- **WiFi.h**: gestiona la conexión básica del ESP32 a redes inalámbricas.
- **WiFiMulti.h**: permite configurar múltiples redes para que el sistema seleccione automáticamente la más disponible, garantizando redundancia.
- **EmonLib.h**: biblioteca especializada en cálculos eléctricos en corriente alterna; proporciona funciones para obtener valores de voltaje RMS, corriente RMS, potencia real, potencia aparente y factor de potencia.
- **Wire.h**: habilita la comunicación I2C entre el ESP32 y dispositivos externos, en este caso la pantalla OLED.
- **Adafruit\_GFX.h** y **Adafruit\_SH1106.h**: controlan la pantalla OLED SH1106, incluyendo posicionamiento de texto, gráficos y refresco de pantalla.

- **HTTPClient.h**: permite enviar datos a un servidor mediante solicitudes HTTP POST, utilizado para comunicarse con InfluxDB.

La correcta inclusión y orden de estas librerías es esencial para evitar conflictos y asegurar compatibilidad con la versión de Arduino Core para ESP32.

## 4.2. Calibración y definición de sensores

Para el monitoreo trifásico se definen **seis instancias** de la clase `EnergyMonitor`: tres para corriente y tres para voltaje.

```
EnergyMonitor emon1, emon2, emon3; // Corriente
EnergyMonitor volt1, volt2, volt3; // Voltaje
```

Cada sensor se calibra con valores obtenidos experimentalmente:

```
emon1.current(34, 0.15); // Fase 1 - Corriente
volt1.voltage(35, 45.5, 1.7); // Fase 1 - Voltaje
```

- **Primer parámetro**: número del pin ADC donde está conectado el sensor.
- **Segundo parámetro**: constante de calibración para convertir lecturas crudas en amperios o voltios.
- **Tercer parámetro (solo en voltaje)**: ángulo de fase en radianes, para corregir el desfase entre señal de voltaje y corriente.

La calibración fina de estos valores se realizó mediante mediciones comparativas con equipos patrón, asegurando que las lecturas tengan un margen de error mínimo.

## 4.3. Conexión Wi-Fi

Para garantizar la conectividad del sistema, se utiliza la clase `WiFiMulti`, que permite almacenar más de una red como respaldo:

```
WiFiMulti wifiMulti;
wifiMulti.addAP("NOMBRE_RED", "*****");
```

- Se oculta la contraseña en la documentación para evitar exposición de credenciales.
- Durante la ejecución, el ESP32 intenta conectarse a la primera red disponible.
- La función `wifiMulti.run()` se ejecuta dentro del `loop()` para mantener la conexión activa y reconectar en caso de pérdida.

Este esquema permite que el sistema continúe transmitiendo datos incluso si el punto de acceso principal deja de estar disponible.

#### 4.4. Pantalla OLED

El sistema utiliza una pantalla OLED SH1106 de 1,3 pulgadas, conectada por I2C. La inicialización se realiza así:

```
#define OLED_RESET -1  
Adafruit_SH1106 display(OLED_RESET);
```

- `OLED_RESET` se define como `-1` ya que el módulo no cuenta con un pin físico de reset.
- Durante el `setup()`, se muestra un mensaje de bienvenida o logotipo institucional, indicando que el sistema está iniciando.
- En el `loop()`, la pantalla se actualiza cada ciclo con las mediciones actuales de voltaje, corriente, potencia real, potencia aparente y factor de potencia por fase.

Ejemplo de impresión en pantalla:

```
display.setCursor(0, 0);  
display.print("V1: ");  
display.println(voltage1);
```

El uso de una pantalla OLED independiente permite una supervisión directa de los datos sin necesidad de acceder a Grafana.

#### 4.5. Cálculo de variables eléctricas

Las lecturas se realizan con la función `calcVI()`, que analiza un número determinado de semiciclos y calcula los valores RMS:

```
emon1.calcVI(20, 2000);  
float voltaje1 = emon1.Vrms;  
float corriente1 = emon1.Irms;  
float potencial = emon1.realPower;  
float aparentel = emon1.apparentPower;  
float fp1 = emon1.powerFactor;
```

Este proceso se repite para las tres fases. Además, se calcula la energía acumulada (en kWh) mediante:

```
energia1 += potencial * (intervalo / 3600000.0);
```

El cálculo considera el intervalo de medición para que el valor acumulado sea coherente con el tiempo real de funcionamiento.

#### 4.6. Envío de datos a InfluxDB

Los datos obtenidos se preparan en formato Line Protocol, que InfluxDB interpreta directamente:

```

String data = "mediciones,dispositivo=esp32 ";
data += "voltaje1=" + String(voltaje1) + ",";
data += "corriente1=" + String(corriente1) + ",";
...

```

Luego se envían al servidor InfluxDB mediante `HTTPClient`:

```

HTTPClient http;
http.begin("http://xxx.xxx.xxx.xxx:8086/write?db=mediciones");
http.addHeader("Content-Type", "text/plain");
int httpResponseCode = http.POST(data);
http.end();

```

- La IP real se reemplaza por `xxx.xxx.xxx.xxx` en la documentación para mantener la seguridad del sistema.
- Un código de respuesta HTTP 204 indica que la escritura fue exitosa.

## 4.7. Control de tiempos

Para evitar saturar la red y el procesador, se establece un intervalo fijo entre lecturas:

```

unsigned long tiempoAnterior = 0;
const unsigned long intervalo = 5000;

if (millis() - tiempoAnterior >= intervalo) {
    tiempoAnterior = millis();
    // Lectura y envío de datos
}

```

El uso de `millis()` en lugar de `delay()` permite que el microcontrolador ejecute otras tareas en paralelo, como mantener la conexión Wi-Fi o actualizar la pantalla, sin bloquear la ejecución.

## 5. Flujo general del programa

El código fue diseñado con una lógica secuencial, estructurada y fácil de mantener. El sistema ejecuta las siguientes fases de forma cíclica cada 5 segundos, permitiendo una adquisición y transmisión de datos continua y eficiente.

### 5.1. Etapas del flujo de ejecución

#### 1. Inicialización

- Se cargan las librerías y se definen los objetos.
- Se establece la conexión con la red Wi-Fi.
- Se inicializa la pantalla OLED.
- Se calibra cada sensor de corriente y voltaje.

#### 2. Bucle principal (loop)

Cada 5 segundos:

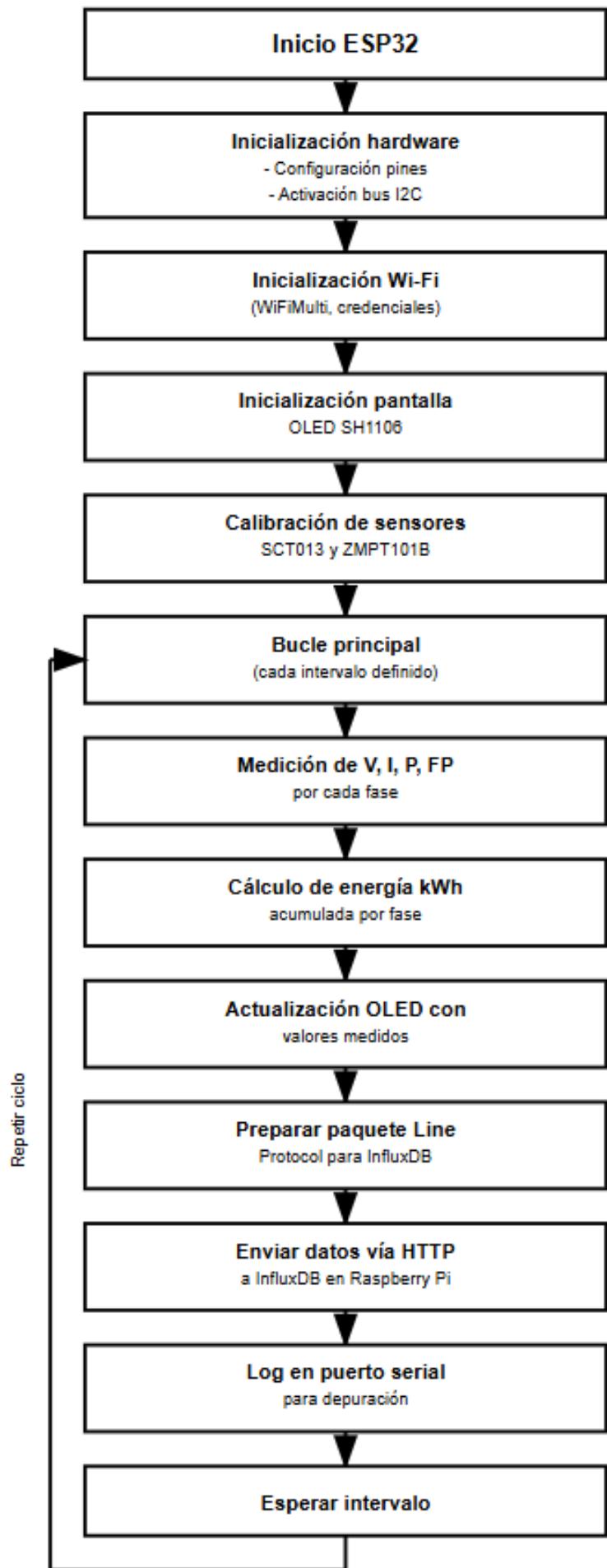
- Se calcula voltaje, corriente, potencia, factor de potencia y energía por fase.
- Se muestra la información en pantalla OLED.
- Se genera el paquete de datos en formato Line Protocol.
- Se envían los datos a InfluxDB por HTTP.
- Se imprime información de estado en el monitor serial.

#### 3. Manejo de errores

- Si falla la conexión Wi-Fi o el envío a InfluxDB, se imprime un mensaje de error.
- El sistema intenta reconectarse automáticamente.

### 5.2. Diagrama de flujo del sistema

El siguiente diagrama de flujo describe de forma lógica el comportamiento del firmware implementado en el ESP32, considerando las fases de inicialización, adquisición de datos, procesamiento, visualización y envío a la base de datos.



### 5.3. Consideraciones importantes

- **Tolerancia a fallos de red:** si la conexión Wi-Fi se interrumpe, el sistema ejecuta intentos automáticos de reconexión antes de proceder con nuevos envíos de datos.
- **Operación no bloqueante:** se evita el uso de `delay()` en favor de temporizadores basados en `millis()`, garantizando que el ESP32 pueda seguir ejecutando otras tareas concurrentes.
- **Bajo consumo de recursos:** el ciclo de lectura y transmisión se completa en menos de un segundo, reduciendo el tiempo activo de la CPU y el consumo energético.
- **Reinicio seguro:** en caso de fallo crítico o error de conexión prolongado, el ESP32 puede ser reiniciado por software para reestablecer el funcionamiento sin intervención manual.

## 6. Recomendaciones para mantenimiento, escalabilidad y personalización

### 6.1. Buenas prácticas de mantenimiento

Práctica	Descripción
Modularización del código	Segmentar el código en funciones y/o librerías propias para simplificar modificaciones futuras.
Depuración controlada	Activar mensajes por <code>Serial</code> únicamente en la fase de pruebas y desactivarlos en producción.
Control de versiones	Utilizar <code>Git</code> para registrar cambios, permitiendo revertir a versiones estables en caso de errores.
Recalibración periódica	Comprobar y ajustar cada sensor SCT013 y ZMPT101B para mantener la precisión en las mediciones.

### 6.2. Escalabilidad del sistema

El sistema permite su expansión y adaptación a distintas necesidades mediante varias estrategias: la incorporación de sensores adicionales SCT013 y ZMPT101B conectados a pines libres del ESP32, organizando sus lecturas mediante estructuras tipo *struct* para monitorear múltiples subcircuitos; la implementación multipunto, donde varios nodos ESP32 envían datos a la misma base InfluxDB con una etiqueta única (*tag*) que facilite su identificación en los paneles de Grafana; el aumento de la resolución temporal reduciendo el intervalo de muestreo de 5 s a 1 s, lo que incrementa el detalle de las mediciones pero también la carga y el tráfico en el servidor, por lo que requiere pruebas previas; y la incorporación de una visualización local embebida mediante un servidor web básico implementado con la librería AsyncWebServer en el propio ESP32, lo que permite consultar los datos en tiempo real sin depender de Grafana.

### **6.3. Seguridad y respaldo**

- Utilizar contraseñas seguras en Grafana e InfluxDB.
- Configurar respaldo automático de la base de datos con scripts `cron` en la Raspberry Pi.
- Habilitar cifrado WPA2 en la red Wi-Fi y, de ser posible, ubicar el sistema en una red VLAN separada.
- Cerrar puertos innecesarios en el firewall para minimizar superficies de ataque.