University of Applied Sciences and Arts
of Southern Switzerland

SUPSI

Internet of Things

Master of Science in Engineering

Implementation of Modbus Master-Slave communication on ESP32 with E2002BL network analyzer

---

Students:

Axel Salaris
Erica Capocello
Matteo Metaldi
Roberto Vicario

Group 2

2023/2024 – Spring Semester

Contents

1 Introduction

As part of the Internet of Things (IoT) course, we developed an integrated system using an ESP32 board and an Esam E2002BL device [3], a network analyzer. The project involved using the ESP32 in two distinct operational modes: slave mode and master mode. In slave mode, the ESP32 received data requests from a software on the PC, such as Modscan, and responded with the requested information. In master mode, the ESP32 directly queried the E2002BL device to gather the necessary data. One of the main functionalities of the system was the ESP32's ability to create a Wi-Fi network, allowing connection to an existing network. Subsequently, the ESP32 acted as a web server, offering a configuration page that enabled users to view, add, delete, and modify registers in an interactive table. Additionally, the system included a real-time analytics page to monitor the values of a single register. For communication between devices, we used the Modbus RTU protocol, ensuring reliable and efficient data transmission between the system components.

2 Implementation details

2.1 Project Structure

The structure of the project is as follows:

- **./src**
  - **src.ino**
  - ModbusRTUMaster.h
  - ModbusRTUSlave.h
  - WebServerManager.h
  - Utils.h
  - DisplayOLed.h
  - ModbusRTUMaster.cpp
  - ModbusRTUSlave.cpp
  - WebServerManager.cpp
  - Utils.cpp
  - DisplayOLed.cpp

2.2 Pin setup

The ESP32's connected pins to the two RS485s and the display are (Fig. 1):

- **RS485 Slave** (Serial1)
  - RX: GPIO 32
  - TX: GPIO 33

  - DE: GPIO 27
- **RS485 Master** (Serial2)
  - RX: GPIO 16
  - TX: GPIO 17
  - DE: GPIO 15
- **Oled Display**
  - SCK: GPIO 22
  - SDA: GPIO 21

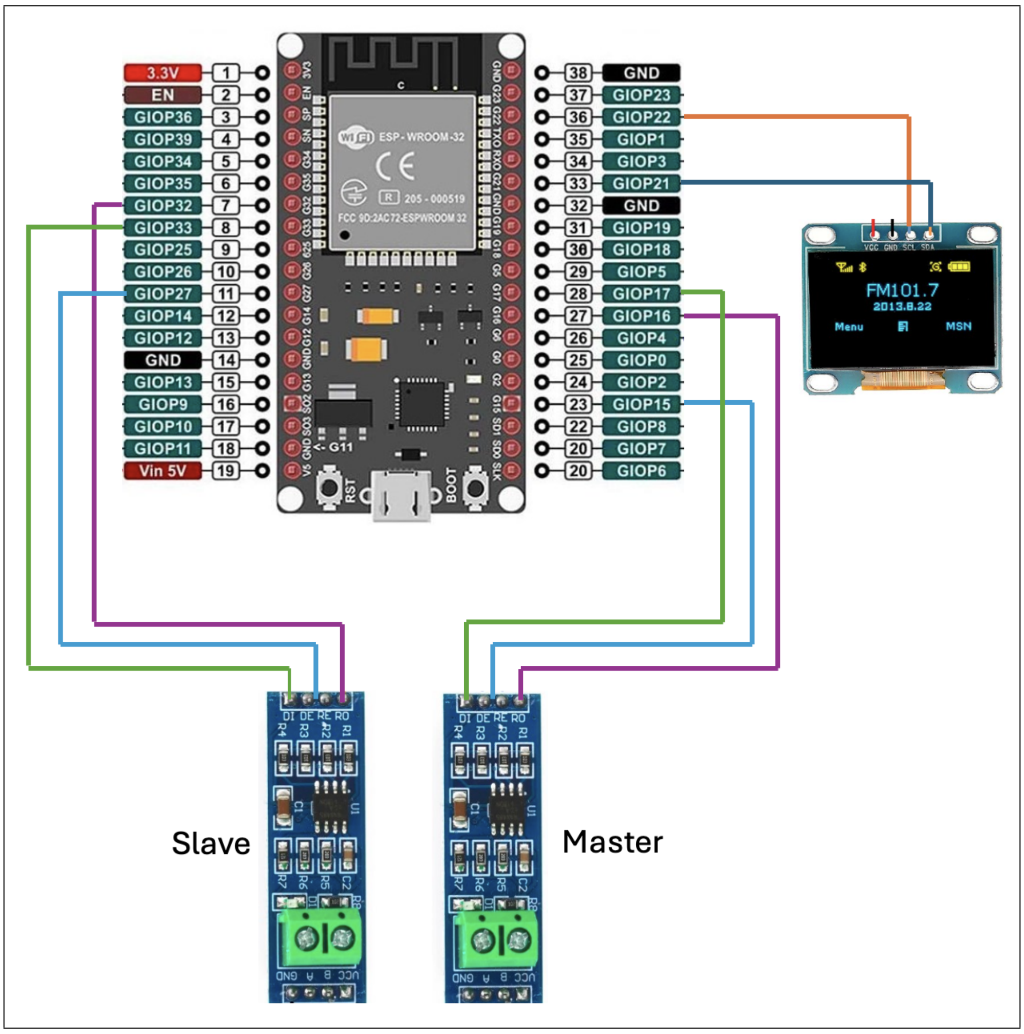


Figure 1: Schematic of the system.

2.3 Libraries and Functions used

To implement Modbus communication between ESP32, PC and E2002Bl and the web server, several libraries and different methods of these were used. Below are the details:

- **ModbusRTUMaster [1]:**
  - Initialisation:

    ```
    ModbusRTUMaster modbus(Serial2, dePin);
    ```

  - Start the communication:

    ```
    modbus.begin(9600, SERIAL_8N1, rxPin, txPin);
    ```

  - Reading values from registers:

    ```
    modbus.readHoldingRegisters(slaveId, i-1, registersArray, 2);
    ```

    Note that the function takes the value of the register on the datasheet minus one.

- **ModbusRTUSlave [2]:**
  - Initialisation:

    ```
    ModbusRTUSlave modbusSlave(Serial1, dePin);
    ```

  - Start the communication:

    ```
    modbusSlave.begin(slaveId, 9600, SERIAL_8N1, rxPin, txPin);
    ```

  - Handling of requests within the loop:

    ```
    modbusSlave.poll();
    ```

- **WebServer:**
  - Web server initialisation:

    ```
    WebServerManager webServer("CONFIGURE_ME", "", master, swap);
    ```

  - Listening on the port 80:

    ```
    server.handleClient();
    ```

- **DNSServer:**
  - Initialising the DNS server to activate the captive portal:

```
dnsServer.start(53, "*", WiFi.softAPIP());
```

- **Adafruit_GFX and Adafruit_SSD1306** (optional to use the Oled display):
  - Initialising the display:

    ```
    Adafruit_SSD1306 display(128, 64);
    ```

  - Start the communication via I2C:

    ```
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C)
    ```

  - Write text on the display:

    ```
    display.println(text)
    display.display();
    ```

2.4 Endpoints explanation

Endpoint	Method	Function in code	Description
/setWifi	POST	handleWifiConfig	This endpoint is used to set up the new Wi-Fi configuration. The Wi-Fi SSID and password credentials, entered by the user, are retrieved and used to establish the new connection.
/config	GET	handleDataPage	This endpoint is the main one and is responsible for rendering the configuration page, which includes the log table.
/readRegister	GET	handleReadRegister	This endpoint is used to make AJAX calls from the frontend in order to read data from the network analyser's holding registers.
/addData	POST	handleAddData	This endpoint handles an AJAX call from the frontend. In this context, a new register is added to the existing registers. The register is added either to the register storage array or directly to the table on the frontend.
/deleteData	DELETE	handleDeleteData	This endpoint is used to delete a register from both the local register and the table on the page. Like the other operations, it is also performed via an AJAX call.
/updateData	PUT	handleUpdateData	This endpoint is used as an AJAX call from the frontend to update a register within the local array and in the table on the page.
/toggleMaster	POST	handleToggleMaster	With this endpoint, using the toggle on the frontend page, it is possible to switch from configuring the ESP32 as a master to configuring it as a slave and vice versa.
/toggleSwap	POST	handleToggleSwap	With this endpoint, using the toggle on the frontend page, it is possible to swap the reading of the registers.
*	-	handleRoot	The DNS server is activated on port 53. When the required endpoint is not found, it redirects to the handleRoot function, whose responsibility it is to show the captive portal. This is where the Wi-Fi configuration can be carried out.

Table 1: Description of the all endpoints

2.5 Considerations

In the case of the ESP32 is connected to a WiFi network but this network does not have Internet access, the logic of the user interface (UI) and analytics chart would not work as expected since they depend on Bootstrap and Chart.js, libraries usually loaded from online CDNs. An effective solution is to load the CSS and JS files into the ESP32's SPIFFS memory, making the resources available locally and independent of the Internet connection. However, it was not possible to implement this logic within the SPIFFS in the time available and as it was not a priority objective of the project. Another consideration concerns the role of the master in the project architecture. Although in the current context the master does not have a specific task, it is possible to implement logic that allows the ESP32, when configured as master, to upload read data in real time to a cloud or time series database. Finally, with regard to the OLED display, the implementation of logic to write on it is optional, but especially useful for debugging, offering an additional resource to monitor the operation of the system directly and immediately.

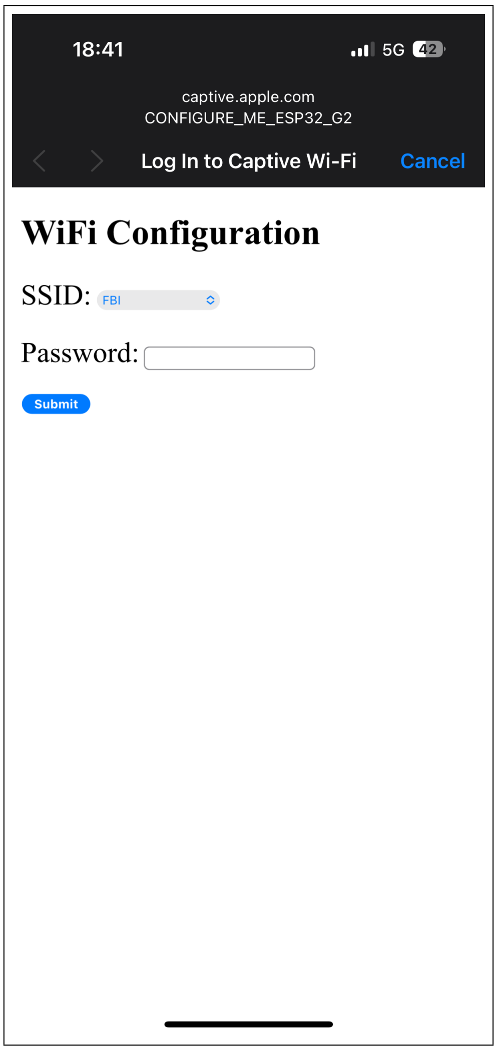
3 User guide

3.1 Hardware and software required

- ESP32 W-ROOM 32
- E20002BL device
- 2 RS485
- Cables
- Oled display (optional)

3.2 Wifi configuration

The ESP32 creates a WiFi network named '*CONFIGURE_ME_ESP32_G*'. Once you connect to this network, a captive portal automatically appears to guide you through the configuration process (Fig. 2). In this portal, you can enter the SSID and password of the new WiFi network to which the ESP32 is to connect.



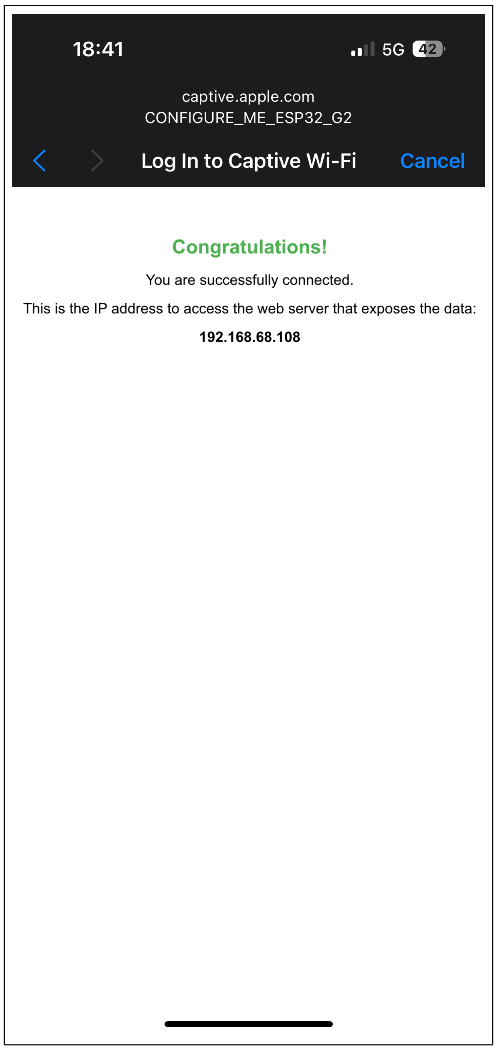
WiFi Configuration

SSID: FBI

Password:

Figure 2: Wifi captive portal

After providing the correct data and confirming the configuration, the ESP32 connects to the specified network. Next, a confirmation page is displayed containing the IP address assigned to the ESP32 by the router (Fig. 3). This IP address allows access to the built-in webserver functionality, facilitating interaction and control of the device via a convenient web interface.



Congratulations!

You are successfully connected.

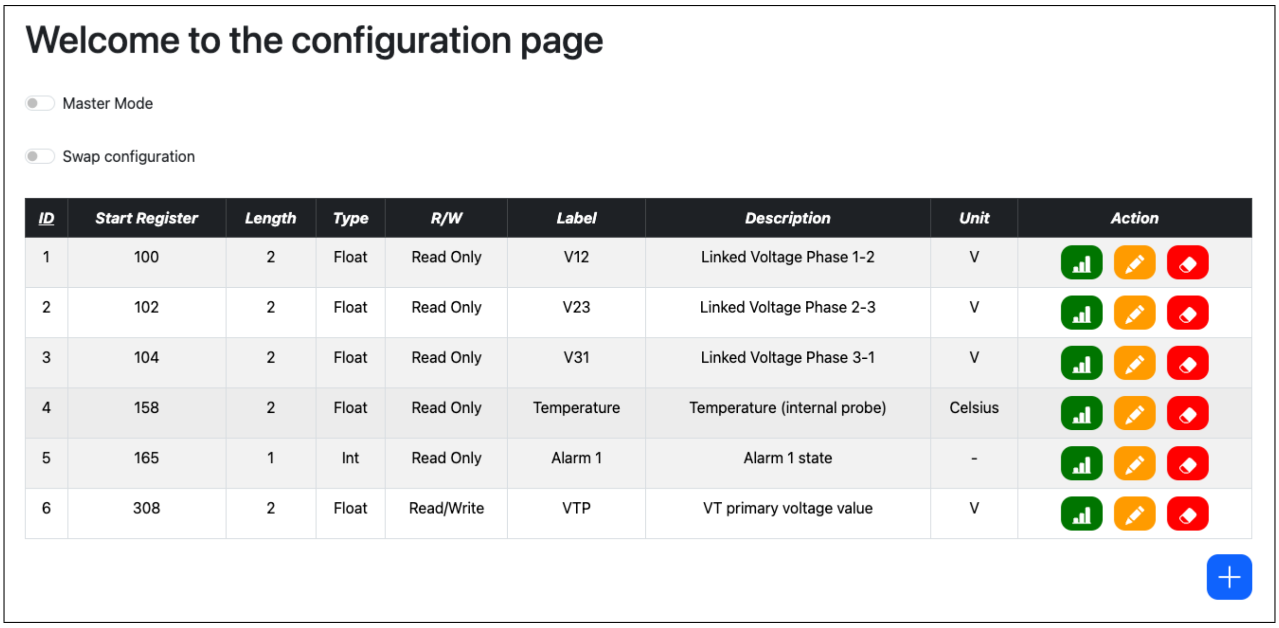
This is the IP address to access the web server that exposes the data:

192.168.68.108

Figure 3: Successfully connected

3.3 Configuration and Analytics page

After noting the IP address provided by the ESP32, it is necessary to connect to the same network to which the ESP32 is connected using a smartphone or computer. Open the browser and type in the IP address followed by */config*. A detailed page will be loaded presenting a table with six predefined registers (Fig. 4).


Welcome to the configuration page

Master Mode

Swap configuration

ID	Start Register	Length	Type	R/W	Label	Description	Unit	Action
1	100	2	Float	Read Only	V12	Linked Voltage Phase 1-2	V	
2	102	2	Float	Read Only	V23	Linked Voltage Phase 2-3	V	
3	104	2	Float	Read Only	V31	Linked Voltage Phase 3-1	V	
4	158	2	Float	Read Only	Temperature	Temperature (internal probe)	Celsius	
5	165	1	Int	Read Only	Alarm 1	Alarm 1 state	-	
6	308	2	Float	Read/Write	VTP	VT primary voltage value	V	

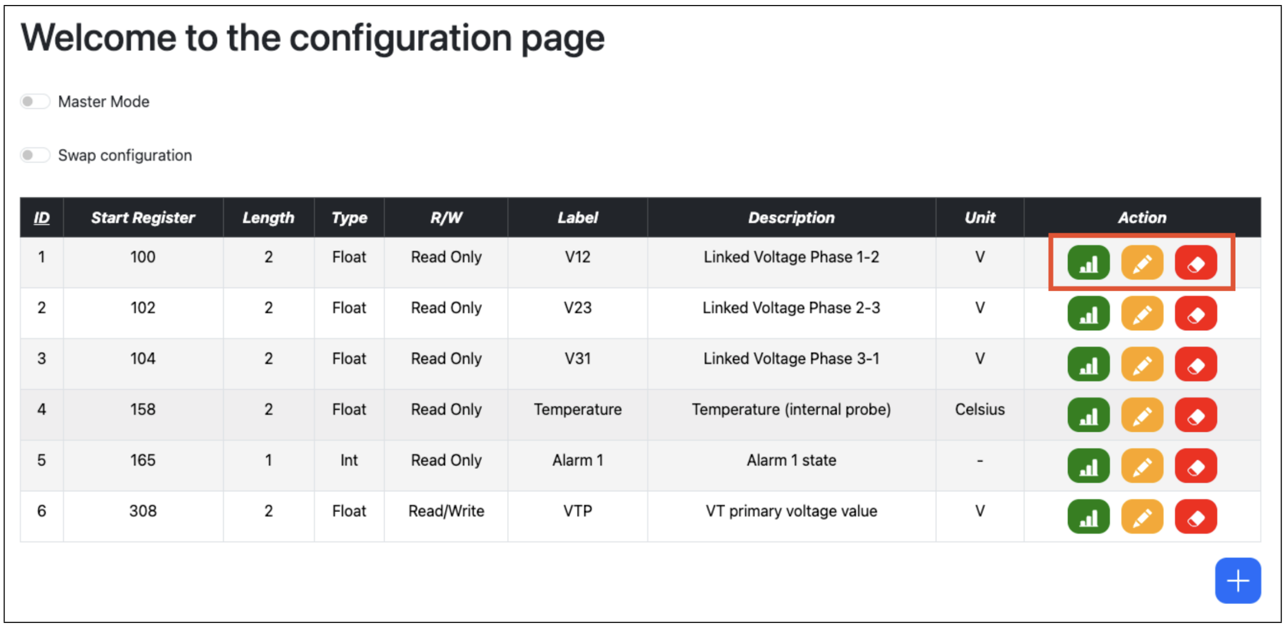
Figure 4: Configuration and Analytics page.

The table includes the following columns:

- **ID**: unique identifier of the register in the table
- **Register number**: identifies the specific register
- **Length**: for example, 2 if the type is float
- **Type**: may be float, int, or long
- **Access**: indicates whether the register is read-only or read/write
- **Label**: an identification of the register
- **Description**: more in-depth details about the register
- **Unit**: the unit in which the register value is expressed

Each log offers three main actions (Fig. 5):

1. View Analyses: allows you to see the real-time values of the register.
2. Modify the register: change the data of the register.
3. Delete register: removes the register from the table.



Welcome to the configuration page

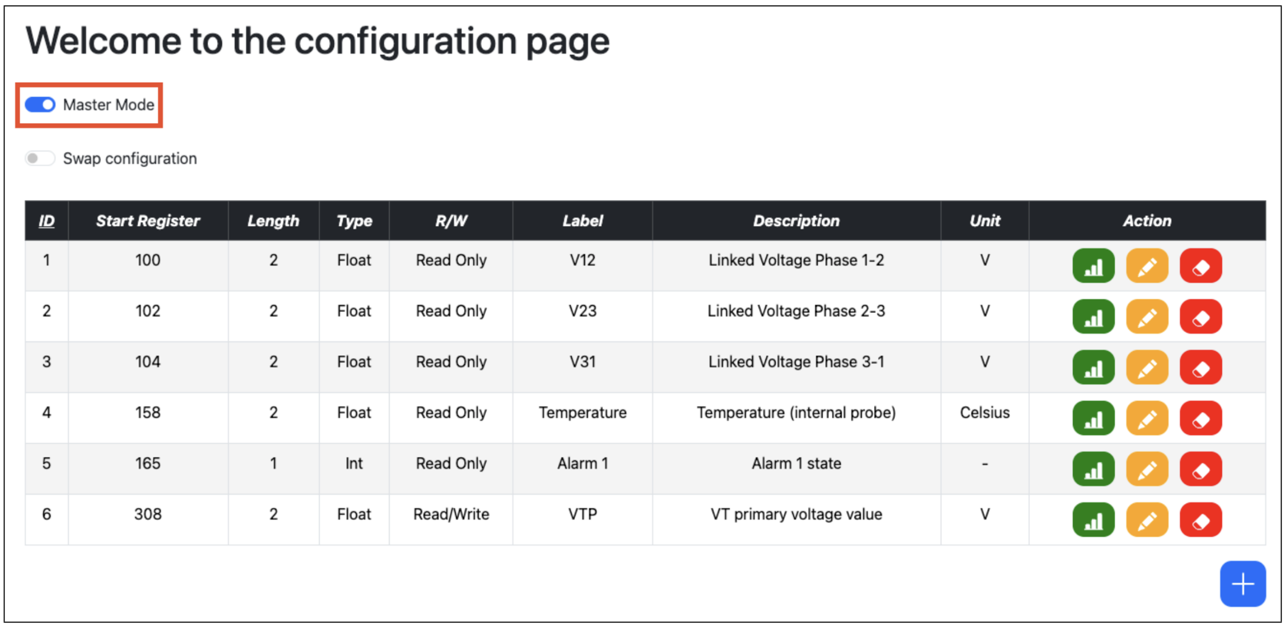
Master Mode

Swap configuration

ID	Start Register	Length	Type	R/W	Label	Description	Unit	Action
1	100	2	Float	Read Only	V12	Linked Voltage Phase 1-2	V	
2	102	2	Float	Read Only	V23	Linked Voltage Phase 2-3	V	
3	104	2	Float	Read Only	V31	Linked Voltage Phase 3-1	V	
4	158	2	Float	Read Only	Temperature	Temperature (internal probe)	Celsius	
5	165	1	Int	Read Only	Alarm 1	Alarm 1 state	-	
6	308	2	Float	Read/Write	VTP	VT primary voltage value	V	

Figure 5: The three actions: analytics, edit and delete.

There is also a toggle allows the ESP32 to be set to master or slave mode (Fig. 6). When the ESP32 is in slave mode, it can be connected to a PC via a USB and RS485 cable.



Welcome to the configuration page

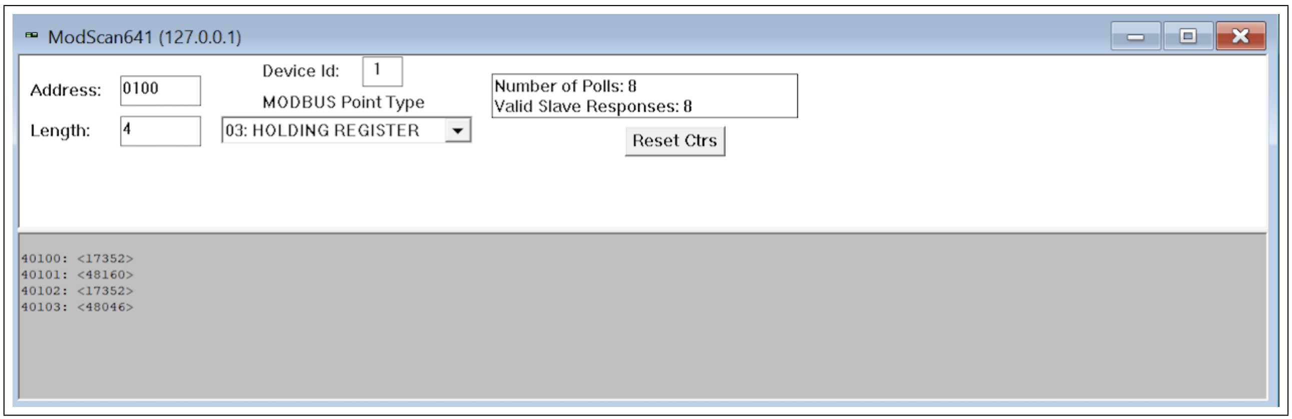
Master Mode

Swap configuration

ID	Start Register	Length	Type	R/W	Label	Description	Unit	Action
1	100	2	Float	Read Only	V12	Linked Voltage Phase 1-2	V	
2	102	2	Float	Read Only	V23	Linked Voltage Phase 2-3	V	
3	104	2	Float	Read Only	V31	Linked Voltage Phase 3-1	V	
4	158	2	Float	Read Only	Temperature	Temperature (internal probe)	Celsius	
5	165	1	Int	Read Only	Alarm 1	Alarm 1 state	-	
6	308	2	Float	Read/Write	VTP	VT primary voltage value	V	

Figure 6: Toggle for set the ESP32 as a master/slave.

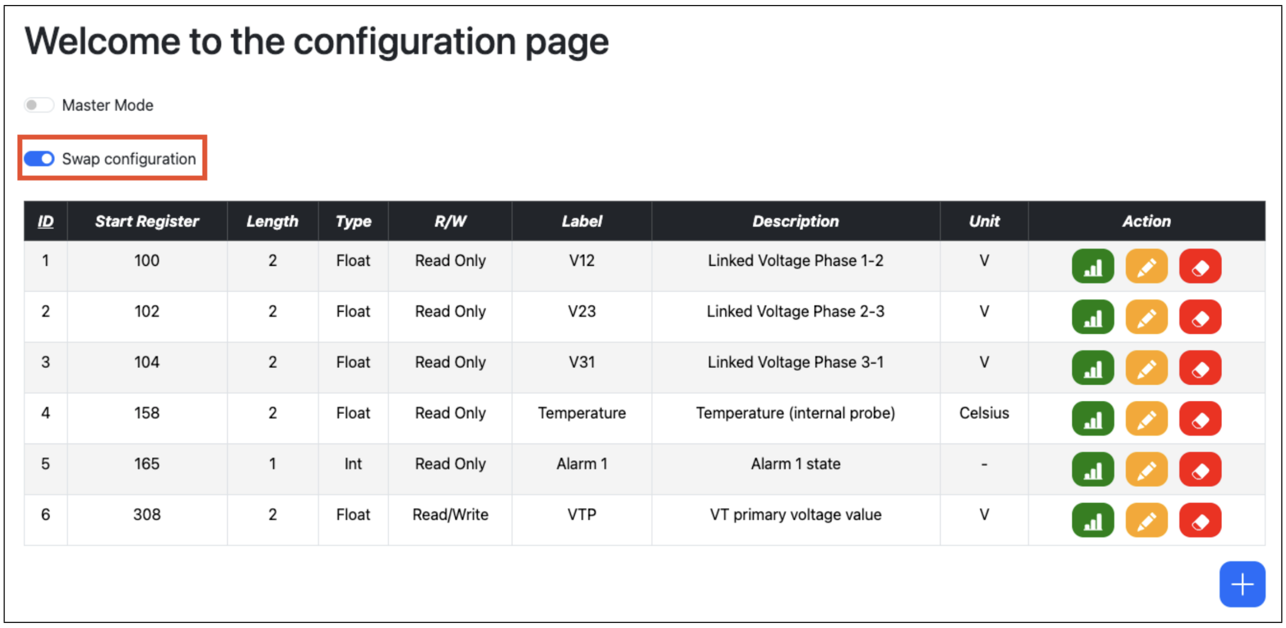
Using software such as Modscan, the register values of the E2002BL can be read (Fig. 7), allowing advanced management and monitoring of the device via the RS485 network.



```
40100: <17352>
40101: <48160>
40102: <17352>
40103: <48046>
```

Figure 7: Read first four registers from Modscan.

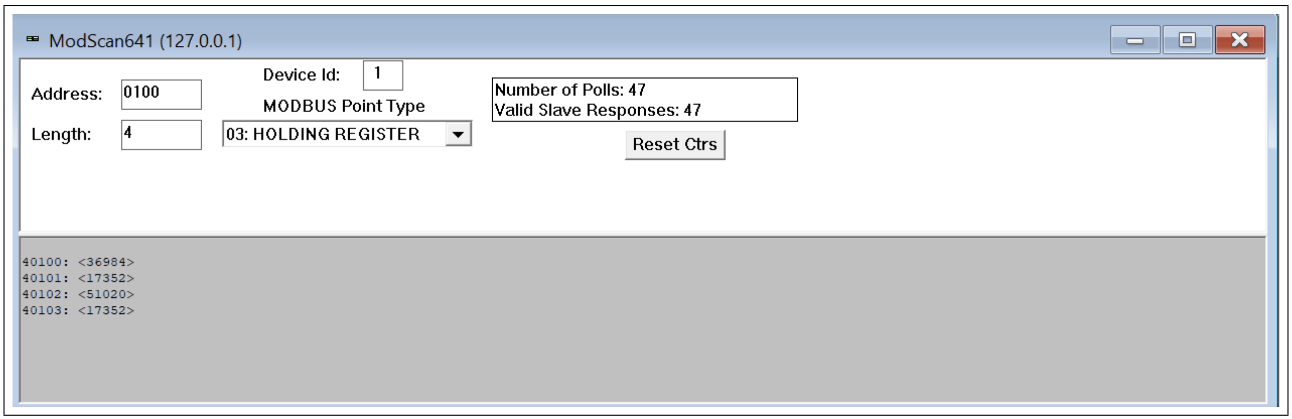
The 'swap configuration' toggle is used to swap registers, reversing the Most Significant Bit (MSB) with the Least Significant Bit (LSB) [4]. On this device (E2002BL), the default value is set to 'false', which can cause inconsistent or meaningless values to be read. When the toggle is activated, the data read from the registers may appear incorrect (Fig. 8).



ID	Start Register	Length	Type	R/W	Label	Description	Unit	Action
1	100	2	Float	Read Only	V12	Linked Voltage Phase 1-2	V	
2	102	2	Float	Read Only	V23	Linked Voltage Phase 2-3	V	
3	104	2	Float	Read Only	V31	Linked Voltage Phase 3-1	V	
4	158	2	Float	Read Only	Temperature	Temperature (internal probe)	Celsius	
5	165	1	Int	Read Only	Alarm 1	Alarm 1 state	-	
6	308	2	Float	Read/Write	VTP	VT primary voltage value	V	

Figure 8: Toggle to set swapped device registers.

Using Modscan, it can be observed that the previous four registers are reversed, confirming that bits have been swapped (Fig. 9). By deactivating the 'swap configuration' toggle, the order of the bits is corrected, allowing the recorded values to be read correctly.



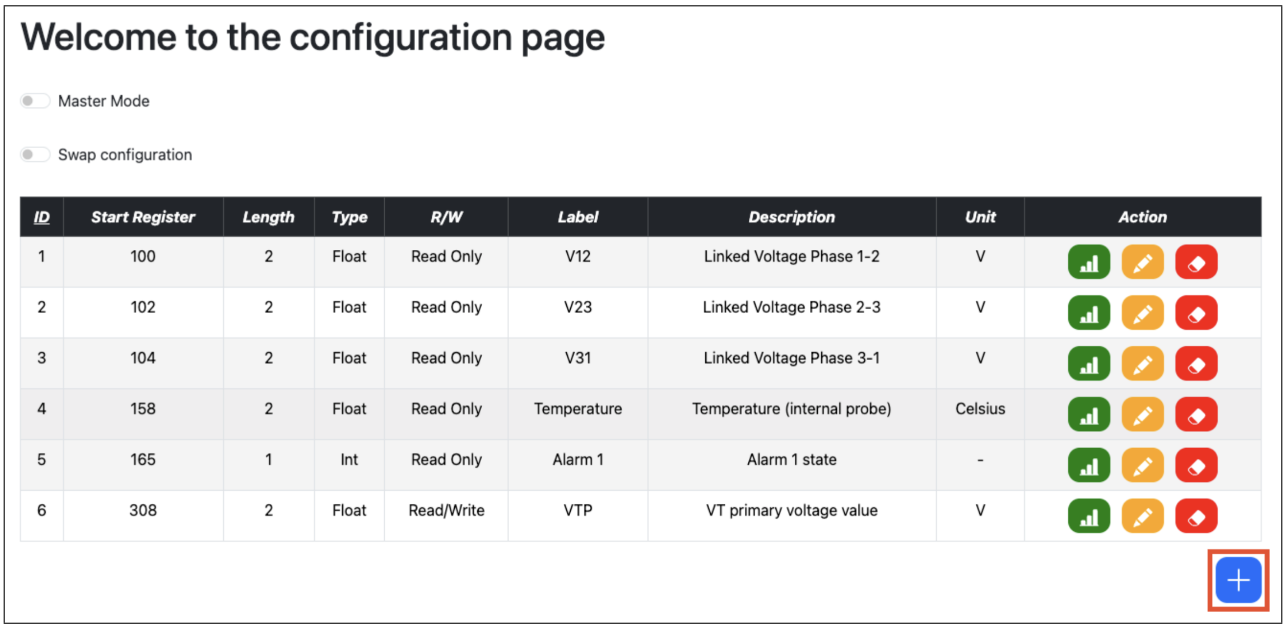
Address: 0100
Length: 4
Device Id: 1
MODBUS Point Type: 03: HOLDING REGISTER
Number of Polls: 47
Valid Slave Responses: 47

```
40100: <36984>
40101: <17352>
40102: <51020>
40103: <17352>
```

Figure 9: Toggle to set swapped device registers.

3.3.1 Add a new register

In addition, new registers can be added by clicking on the '+' button (Fig. 10). A modal will open in which the first two fields (register number and length) are mandatory (Fig. 11). Once this data has been entered, the new register will be added to the existing table.



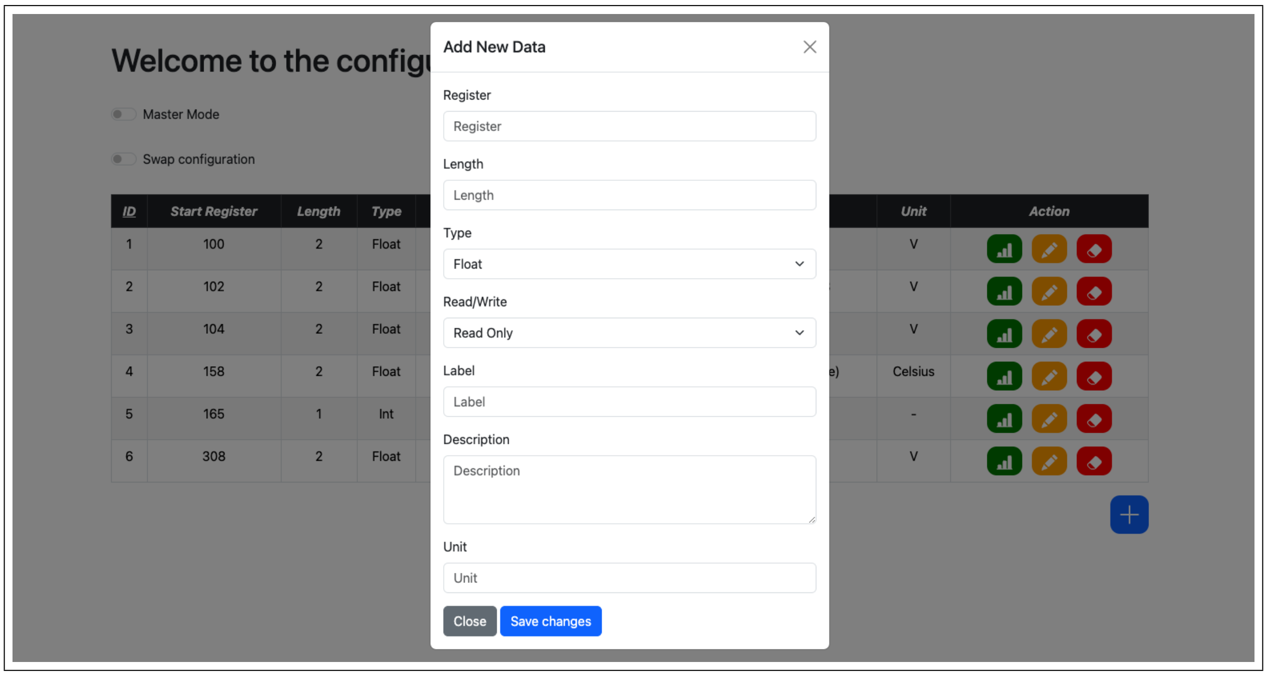
Welcome to the configuration page

Master Mode

Swap configuration

ID	Start Register	Length	Type	R/W	Label	Description	Unit	Action
1	100	2	Float	Read Only	V12	Linked Voltage Phase 1-2	V	
2	102	2	Float	Read Only	V23	Linked Voltage Phase 2-3	V	
3	104	2	Float	Read Only	V31	Linked Voltage Phase 3-1	V	
4	158	2	Float	Read Only	Temperature	Temperature (internal probe)	Celsius	
5	165	1	Int	Read Only	Alarm 1	Alarm 1 state	-	
6	308	2	Float	Read/Write	VTP	VT primary voltage value	V	

Figure 10: Button to add a new register.



Welcome to the config

Master Mode

Swap configuration

ID	Start Register	Length	Type	Unit	Action
1	100	2	Float	V	
2	102	2	Float	V	
3	104	2	Float	V	
4	158	2	Float	Celsius	
5	165	1	Int	-	
6	308	2	Float	V	

Add New Data

Register: Register

Length: Length

Type: Float

Read/Write: Read Only

Label: Label

Description: Description

Unit: Unit

Close | Save changes

Figure 11: Modal to add a new register.

3.3.2 Analytics

In the analytics section, an interactive graph is displayed which updates in real-time every 5 seconds (Fig. 14). This is updated by the ESP32 periodically reading the value contained in the E2002BL device register. The graph is presented within a modal, providing a clear and dynamic representation of the data.

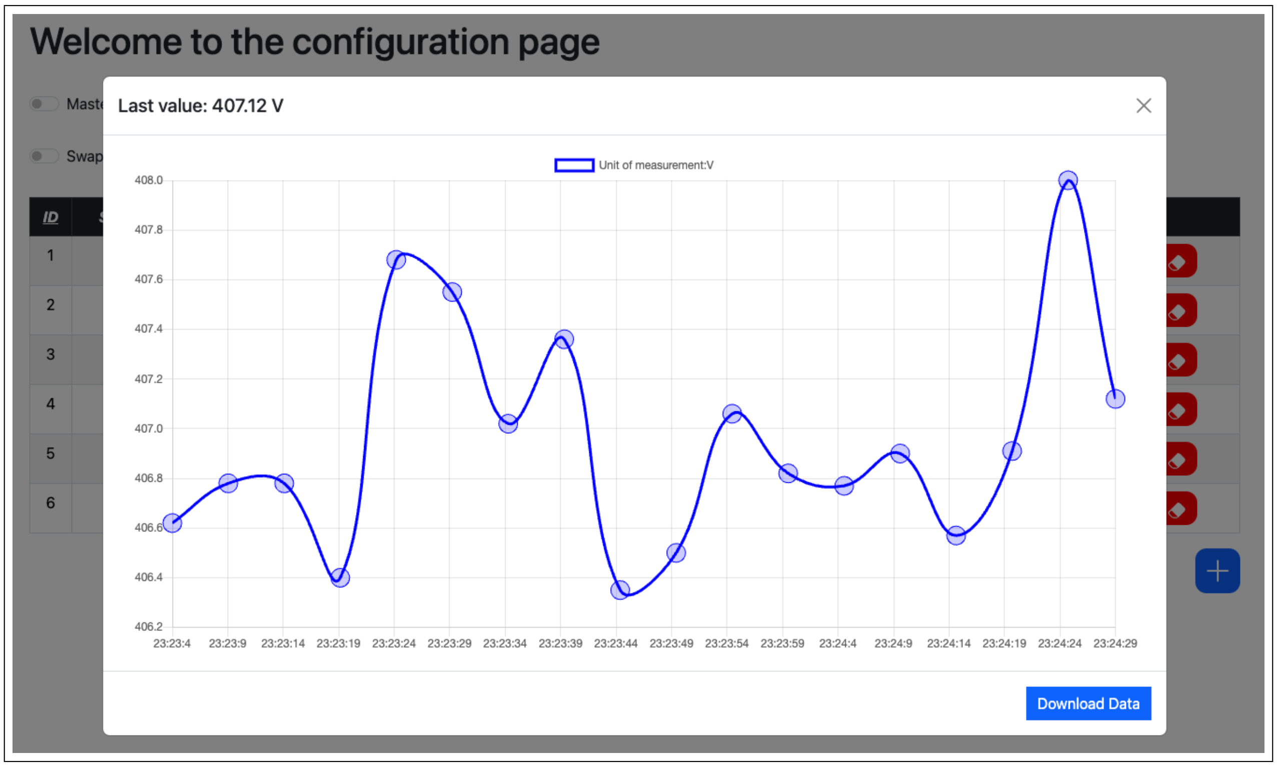
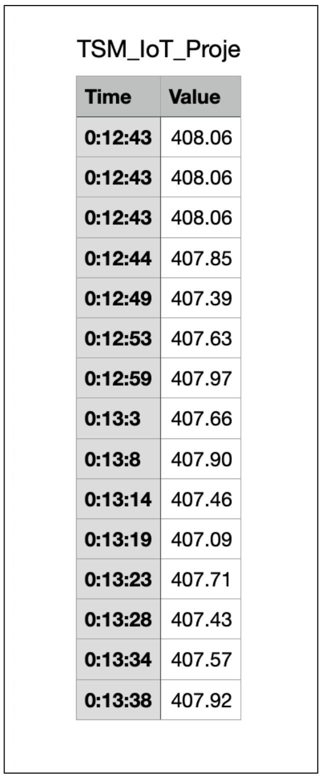


Figure 12: Real-time register value chart.

If the modal is closed, the acquired data is not automatically saved. However, to keep the data, you can use the '*Download Data*' button at the bottom right of the modal. By clicking on this button, the collected data are exported and saved in a CSV file, facilitating the analysis and storage of information (Fig. 13). This tool allows users to accurately monitor and record log values, providing essential support for diagnostics and analysis.



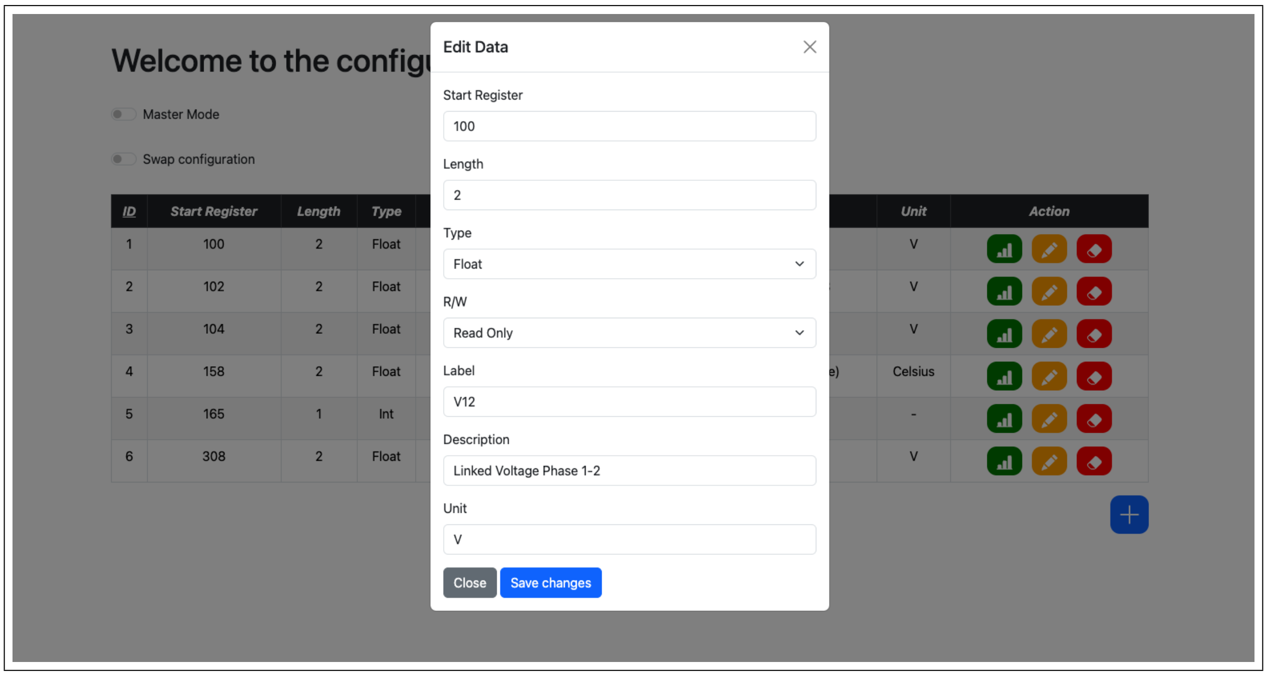
TSM_IoT_Proje

Time	Value
0:12:43	408.06
0:12:43	408.06
0:12:43	408.06
0:12:44	407.85
0:12:49	407.39
0:12:53	407.63
0:12:59	407.97
0:13:3	407.66
0:13:8	407.90
0:13:14	407.46
0:13:19	407.09
0:13:23	407.71
0:13:28	407.43
0:13:34	407.57
0:13:38	407.92

Figure 13: Real-time register value CSV file.

3.3.3 Edit register

To edit a register, a pre-filled modal is opened with the data extracted from the selected row in the table. This modal allows existing register information to be easily viewed and edited. As in the process of adding a new register, the fields 'Register number' and 'Length' are mandatory and must be completed to proceed.



Welcome to the config

Master Mode

Swap configuration

ID	Start Register	Length	Type		Unit	Action
1	100	2	Float		V	
2	102	2	Float		V	
3	104	2	Float		V	
4	158	2	Float	e)	Celsius	
5	165	1	Int		-	
6	308	2	Float		V	

Edit Data

Start Register: 100

Length: 2

Type: Float

R/W: Read Only

Label: V12

Description: Linked Voltage Phase 1-2

Unit: V

Close | Save changes

Figure 14: Model to edit a existing register.

4 Conclusion

The ESP32 offers an advanced and intuitive system for managing and configuring logs via a WiFi network. via the web interface, users can add, edit and delete logs with ease, ensuring that all information is always up-to-date. In addition, the real-time monitoring function via the interactive graph provides an immediate overview of the E2002BL network analyser's performance. The ability to export data in CSV format further facilitates the analysis and storage of information. Finally, the option to configure the ESP32 in master or slave mode extends its capabilities for integration and use in different application scenarios, making it a powerful and flexible tool for developers and advanced users. Our thanks go to ESAM for the opportunity to implement a system with a sensor that is still used in companies today.

References

[1] `https://github.com/CMB27/ModbusRTUMaster`

[2] `https://github.com/CMB27/ModbusRTUSlave`

[3] `https://esam.biz/prodotto/e2002-bl/?lang=en`

[4] `https://esam.biz/wp-content/uploads/2020/05/E2002BLMan.pdf`