

Esta aplicación se ha desarrollado para facilitar el acceso por modbus al Inversor SAJ H1 S2.

¿Cómo funciona? El esp32 se conecta por bluetooth al Dongle del inversor utilizando la librería blufi de espressif. A su vez, el esp32 levanta un servicio tcp que permite recibir llamadas con protocolo modbus tcp. Para poder levantar el servicio tcp, es necesario configurar el ssid y password para poder conectarse a la red local.

Pasos para cargar el código fuente en el esp32:

1º Instalar Arduino IDE

2º Abre el fichero sah_h1_s2_modbus_esp32.ino con Arduino IDE.

3º Conecta el esp32 al equipo. Deberás configurar tu tipo de placa. Si tu esp32 es de 4Mb, debes modificar el esquema de particiones como "Huge APP" para tener espacio suficiente.

4º Debes instalar la librería "ESP32 BLE for Arduino" (BLEDevice). Puedes hacerlo desde "Herramientas" / "Administrar bibliotecas" -> "BLEDevice"

5º Configura el ssid y password de tu wifi:

```
const char* ssid = "TU_SSID";
```

```
const char* password = "TU_PASSWORD";
```

Este trabajo ha sido gracias a la colaboración del grupo de Telegram https://t.me/saj_nooficialoriginal. No es una aplicación oficial y podría dejar de funcionar en cualquier momento.


Úsala bajo tu propia responsabilidad. Desconocemos si afecta en algún sentido al rendimiento y fiabilidad de su instalación fotovoltaica

Arduino Web Editor


Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

CODE ONLINE

GETTING STARTED



Downloads

 **Arduino IDE 2.0.4**

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows

Win 10 and newer, 64 bits

Windows

MSI installer

Windows

ZIP file

Linux

AppImage 64 bits (D86-64)

Linux

ZIP file 64 bits (D86-64)

macOS

Intel, 10.14: "Mojave" or newer, 64 bits

macOS

Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

Nightly Builds

Download a **preview of the incoming release** with the most updated features and bugfixes.

Windows Version 10.14: "Mojave" or newer, 64 bits
macOS AppImage 64 bits (D86-64)
Linux ZIP file 64 bits (D86-64)

[Help](#)

Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **70,728,092** times — impressive! Help its development with a donation.

\$3

\$5

\$10


\$25

\$50

Other

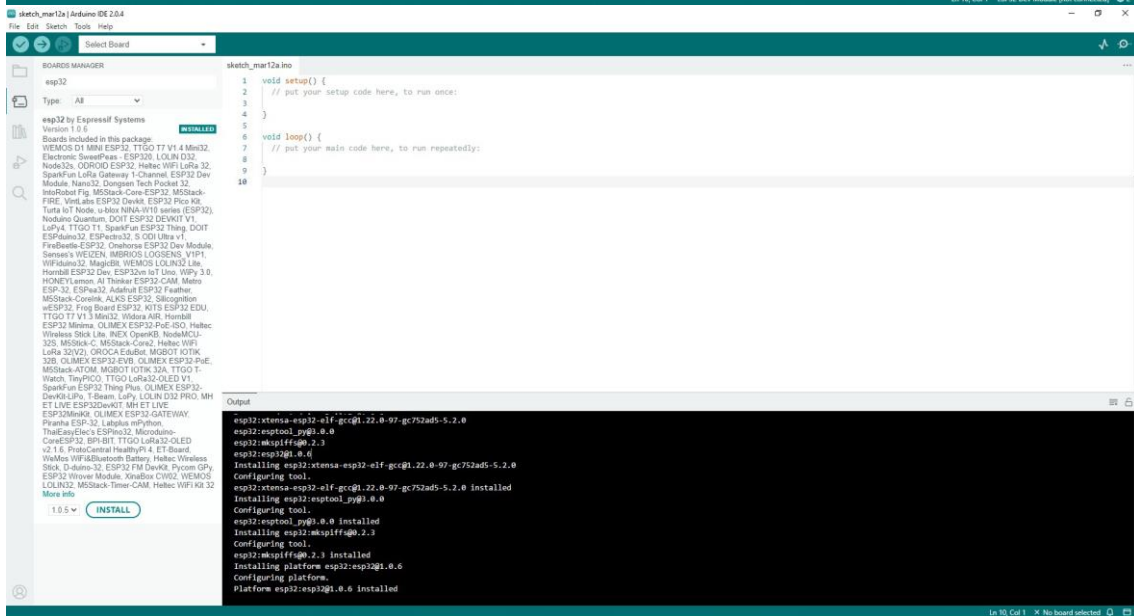
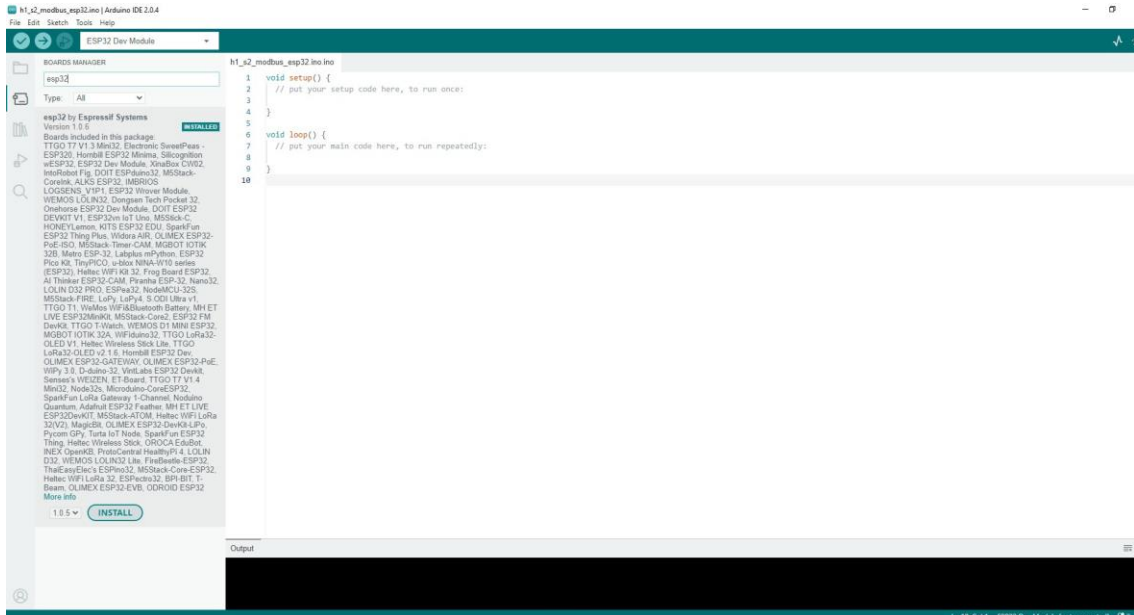
JUST DOWNLOAD

CONTRIBUTE & DOWNLOAD



[Learn more about donating to Arduino.](#)

[Help](#)



sketch_mar12a | Arduino IDE 2.0.4

File Edit Sketch Tools Help

ESP32 Dev Module

sketch_mar


Select other board and port...

1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8
9 }
10

Output

Ln 10, Col 1 No board selected

ESP-IDF Programming Guide



ESP32
master (latest)

Search docs

Get Started

Introduction
What You Need

Installation

IDE

Manual Installation

Windows Installer
Linux and macOS

Build Your First Project

API Reference
Hardware Reference
API Guides
Migration Guides
Libraries and Frameworks
Contributions Guide
ESP-IDF Versions
Resources
Copyrights and Licenses
About
Switch Between Languages

chips. The prerequisite tools include Python, Git, cross-compilers, CMake and Ninja build tools.

For this Getting Started we're going to use the Command Prompt, but after ESP-IDF is installed you can use Eclipse Plugin or another graphical IDE with CMake support instead.


Note

Limitations: - The installation path of ESP-IDF and ESP-IDF Tools must not be longer than 90 characters. Too long installation paths might result in a failed build. - The installation path of Python or ESP-IDF must not contain white spaces or parentheses. - The installation path of Python or ESP-IDF should not contain special characters (non-ASCII) unless the operating system is configured with "Unicode UTF-8" support.

System Administrator can enable the support via Control Panel - Change date, time, or number formats - Administrative tab - Change system locale - check the option "Beta: Use Unicode UTF-8 for worldwide language support" - Ok and reboot the computer.

ESP-IDF Tools Installer

The easiest way to install ESP-IDF's prerequisites is to download one of ESP-IDF Tools installers.



Windows Installer Download

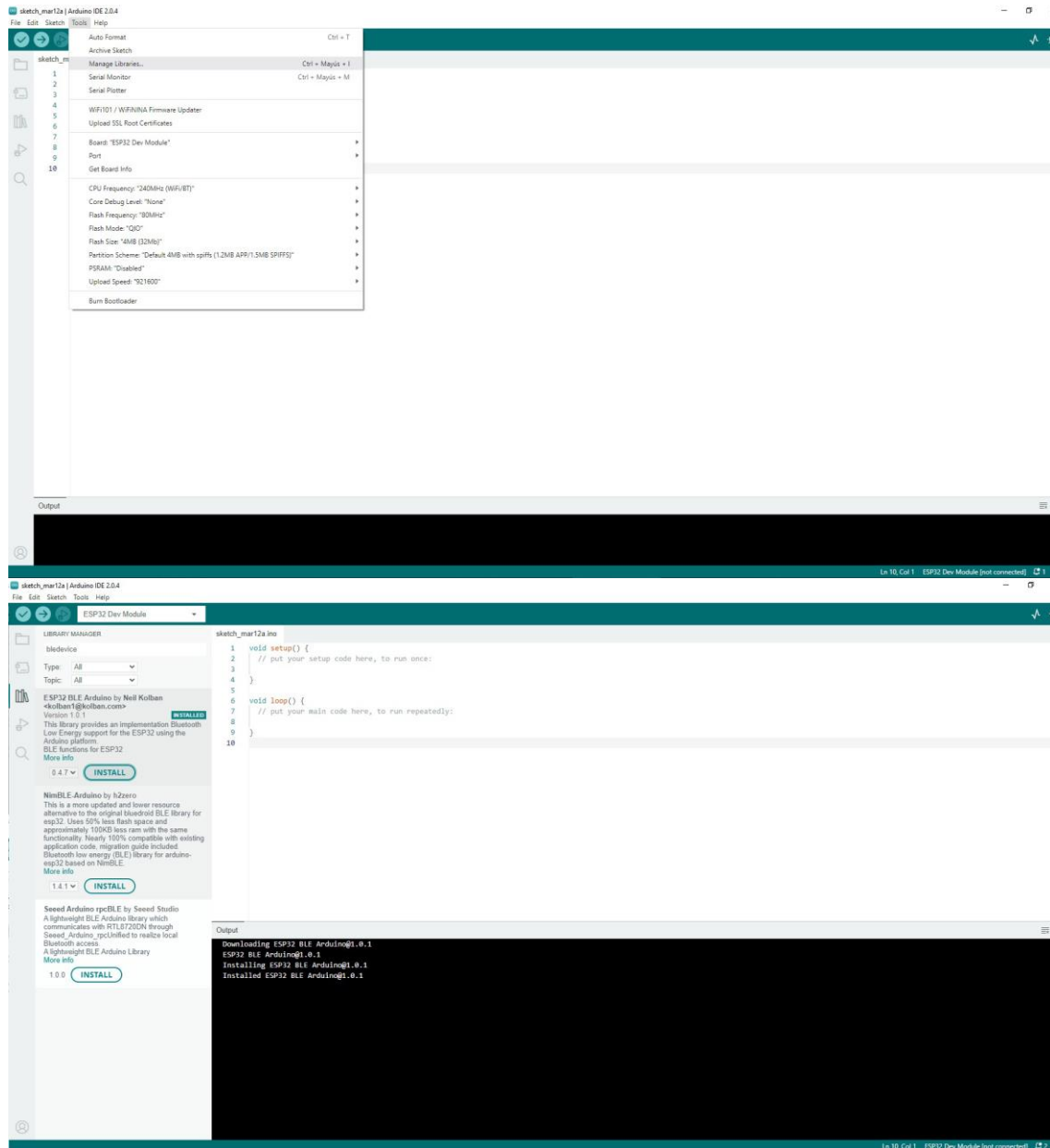
What is the usecase for Online and Offline Installer

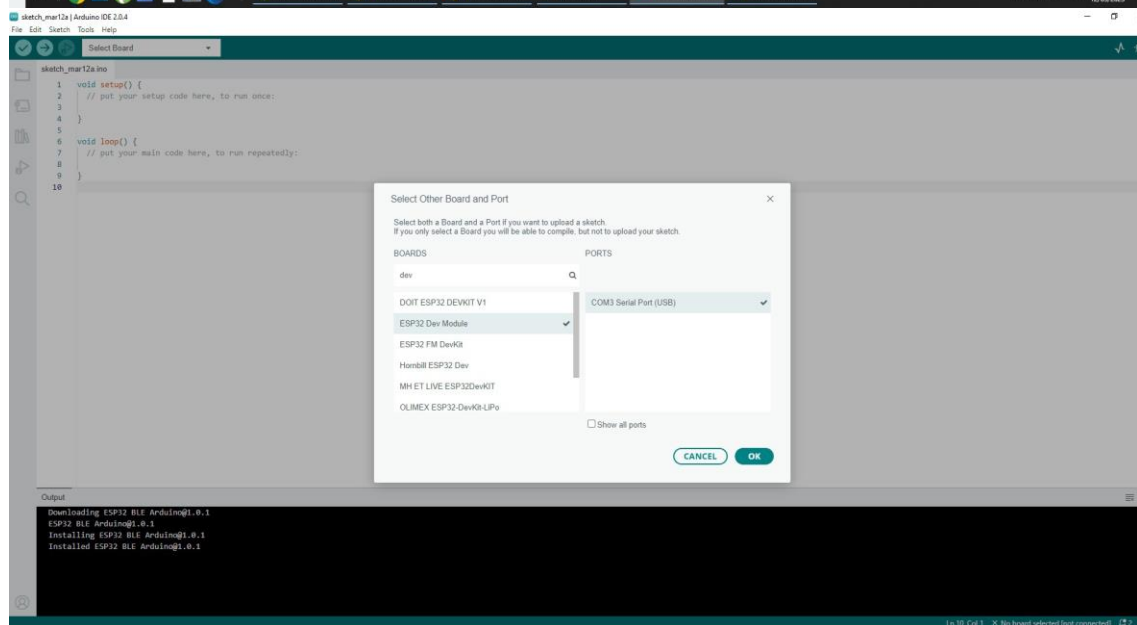
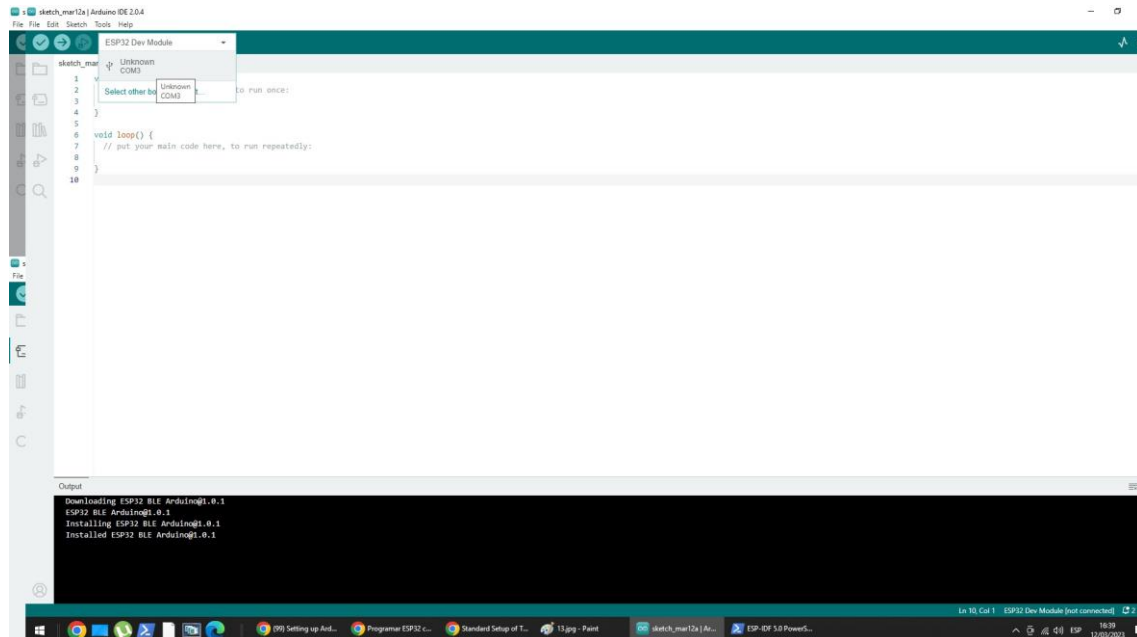
Online Installer is very small and allows the installation of all available releases of ESP-IDF. The installer will download only necessary dependencies including Git For Windows during the installation process. The installer stores downloaded files in the cache directory
[./userprof11sk\espressif](#)

Offline Installer does not require any network connection. The installer contains all required dependencies including Git For Windows .

Components of the installation

The installer deploys the following components:





```
h1_x2_modbus_esp32.ino
1 #include "BLEDevice.h"
2 #include <WiFi.h>
3
4 unsigned long previousMillis = 0;
5 unsigned long previousMillis2 = 0;
6
7 static BLEUUID serviceUUID("0000ffff-0000-1000-8000-00005f9b34fb");
8 static BLEUUID charUUIDh("0000ff01-0000-1000-8000-00005f9b34fb");
9 static BLEUUID charUUIDl("0000ff02-0000-1000-8000-00005f9b34fb");
10
11 const char* ssid = "TU_SSID";
12 const char* password = "TU_PASSWORD";
13
14 static boolean doConnect_ble = false;
15 static boolean connected_ble = false;
16 static boolean doScan_ble = false;
17 static BLERemoteCharacteristic* pRemoteCharacteristic;
18 static BLERemoteCharacteristic* pRemoteCharacteristic2;
19 static BLEAdvertisedDevice* myDevice;
20
21 BLEClient* pClient = BLEDevice::createClient();
22
23 bool error = false;
24 bool waitingMessage = false;
25
26 int soft = 0;
27 byte modbus_frame_response[8];
28 int total_registers;
29
30 #define UInt16 uint16_t
31
32 int ModbusTCP_port = 502;
33 MIFServer MIFServer(ModbusTCP_port);
34
35 //***** Required for Modbus TCP / IP // Requerido para Modbus TCP/IP //*****
36
37 #define PB_FC_NONE 0
38 #define PB_FC_READ_REGISTERS 3 //implemented
39 #define PB_FC_WRITE_REGISTER 6 //implemented
40
41
42 WiFiClient client;
43 byte modbus_request[200];
44
45 int errLen = 0;
46
47 void(* resetFunc)(void) = 0; //declare reset function @ address 0
```

h1_x2_modbus_esp32.ino | Arduino IDE 2.0.4

File Edit Sketch Tools Help

h1_x2_m

1
2
3
4
5
6
7 Board: "ESP32 Dev Module"
8 Port: "COM3"
9 Get Board Info
10
11 CPU Frequency: "240MHz (WiFi/BT)"
12 Core Debug Level: "None"
13 Flash Frequency: "80MHz"
14 Flash Mode: "QIO"
15 Flash Size: "4MB (32Mb)"
16 Partition Scheme: "Huge APP (DJB No OTA/1MB SPIFFS)"
17 PSRAM: "Disabled"
18 Upload Speed: "921600"
19 Burn Bootloader
20
21
22
23
24 bool waitingMessage = false;
25
26 int soft = 0;
27 byte modbus_frame_response[8];
28 int total_registers;

Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)
Default 4MB with Fat (1.2MB APP/1.5MB FATFS)
8M Flash (3MB APP/1.5MB FAT)
Minimal (1.5MB APP/700KB SPIFFS)
No OTA (2MB APP/2MB SPIFFS)
No OTA (1MB APP/3MB SPIFFS)
No OTA (1MB APP/2MB FATFS)
No OTA (1MB APP/3MB FATFS)
Huge APP (DJB No OTA/1MB SPIFFS)
Minimal SPIFFS (1.5MB APP with OTA/190KB SPIFFS)
16M Flash (2MB APP/12.5MB FAT)
16M Flash (2MB APP/9MB FATFS)

Sketch uses 1506694 bytes (114%) of program storage space. Maximum is 1310720 bytes.
Global variables use 55240 bytes (10%) of dynamic memory, leaving 444760 bytes free. Maximum is 444760 bytes.
Sketch too big: see <https://docs.arduino.cc/hw/esp/limitations/known-issues> for tips on reducing it.
text section exceeds available space in board
Compilation error: text section exceeds available space in board

Compilation error: text section exceeds available space in board

COPY ERROR MESSAGES

h1_x2_modbus_esp32.ino | Arduino IDE 2.0.4

File Edit Sketch Tools Help

h1_x2_m

1 #include "BLEDevice.h"
2 #include <WiFi.h>
3
4 unsigned long previousMillis = 0;
5 unsigned long previousMillis2 = 0;
6
7 static BLEUUID serviceUUID("0000ffff-0000-1000-8000-00005f9b34fb");
8 static BLEUUID charUUIDh("0000ff01-0000-1000-8000-00005f9b34fb");
9 static BLEUUID charUUIDl("0000ff02-0000-1000-8000-00005f9b34fb");
10
11 const char* ssid = "TU_SSID";
12 const char* password = "TU_PASSWORD";
13
14 static boolean doConnect_ble = false;
15 static boolean connected_ble = false;
16 static boolean doScan_ble = false;
17 static BLERemoteCharacteristic* pRemoteCharacteristic;
18 static BLERemoteCharacteristic* pRemoteCharacteristic2;
19 static BLEAdvertisedDevice* myDevice;
20
21 BLEClient* pClient = BLEDevice::createClient();
22
23 bool error = false;
24 bool waitingMessage = false;
25
26 int soft = 0;
27 byte modbus_frame_response[8];
28 int total_registers;

Output

Compiling sketch...