

XPLR-HPG software

Required tools for XPLR-HPG kits

Application note



Abstract

This document provides instructions and guidelines for installing all software tools required for use with the XPLR-HPG kits with public u-blox software. It also describes how to clone the `xplr-hpg-sw` repository.

Document information

Title	XPLR-HPG software	
Subtitle	Required tools for XPLR-HPG kits	
Document type	Application note	
Document number	UBX- 23008217	
Revision and date	R01	26-Oct-2023
Disclosure restriction	C1-Public	

This document applies to the following products:

Product name

XPLR-HPG-1

XPLR-HPG-2

u-blox or third parties may hold intellectual property rights in the products, names, logos, and designs included in this document. Copying, reproduction, or modification of this document or any part thereof is only permitted with the express written permission of u-blox. Disclosure to third parties is permitted for clearly public documents only.

The information contained herein is provided "as is" and u-blox assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability, and fitness for a particular purpose of the information. This document may be revised by u-blox at any time without notice. For the most recent documents, visit www.u-blox.com.

Copyright © u-blox AG.

Contents

Document information	2
Contents	3
1 Introduction	4
1.1.1 XPLR-HPG-1 with pluggable L-band, GNSS, LTE, and radio	4
1.2 XPLR-HPG-2 with embedded L-band, GNSS, LTE, and radio	4
1.3 Required software tools.....	5
2 Installing the ESP-IDF core.....	6
3 Cloning the XPLR-HPG repository	8
3.1 Opening the ESP-IDF tools.....	8
3.2 Cloning the XPLR-HPG software repository	9
4 Installing Visual Studio Code and extensions	10
4.1 Installing VS Code	10
4.2 Installing Espressif IDF extension	10
4.3 Setting up the ESP-IDF extension	11
4.4 ESP-IDF extension toolbar	13
5 Building and flashing the captive portal	14
5.1 Connecting the board on your computer	15
5.2 Building and flashing the code	16
Appendix	19
A Glossary	19
Related documentation	20
Revision history	20
Contact.....	20

1 Introduction

XPLR-HPG kits offer a complete platform for evaluating and prototyping, which uses u-blox's high-precision GNSS solution with PointPerfect GNSS augmentation service as well as generic, socket based, Networked Transport of RTCM via Internet Protocol (NTRIP) services. These kits [1][2] incorporate GNSS (Global Navigation Satellite System) and communication modules necessary for processing correction data from a satellite broadcast, using an L-band satellite GNSS receiver or via IP (Internet Protocol) connectivity over Wi-Fi or LTE (Long-Term Evolution) networks [1][2]. Correction data is provided by PointPerfect, the u-blox GNSS augmentation service, and delivered via the Thingstream IoT service delivery platform over MQTT. When using NTRIP as the source of correction data source, a simple socket connection is established with the NTRIP provider of your choice.

XPLR-HPG kits are supplied as two different models:

- XPLR-HPG-1 high-precision GNSS kit [1]
- XPLR-HPG-2 high-precision GNSS kit [2]

The source code provided for the XPLR-HPG kits includes tested and well-documented code snippets and examples to help developers write their own applications [8]. Instructions and guidelines for installing all software tools required for use with publicly available u-blox the XPLR-HPG software are described in this document.

1.1.1 XPLR-HPG-1 with pluggable L-band, GNSS, LTE, and radio

XPLR-HPG-1 provides a flexible evaluation kit with an embedded NORA-W10 standalone, multiradio, Wi-Fi and Bluetooth module soldered directly to the C213 baseboard. The baseboard is equipped with three microBUS™ [3] connectors for plugging the MIKROE Click Boards™ [4] delivered in the kit:

- GNSS RTK 2 Click board [5] featuring the ZED-F9R multi-band GNSS module [17]
- 4G LTE 2 Click board [7] featuring the LARA-R6 LTE Cat 1 cellular module [19]
- LBAND RTK 2 Click board [6] featuring the NEO-D9S satellite receiver for L-band correction [18]

With over 1400 other Click Boards to choose from, developers can explore the endless possibilities for combining u-blox modules with the functionality offered by other technology suppliers – in the same kit. Simply plug the modules you want to try into an available mikroBUS connector.

1.2 XPLR-HPG-2 with embedded L-band, GNSS, LTE, and radio

XPLR-HPG-2 kit provides is a compact kit that includes embedded multiradio, GNSS, L-band, and cellular modules on a single C214 baseboard:

- NINA-W106 standalone, multiradio, Wi-Fi and Bluetooth module [20]
- ZED-F9R multi-band, high-precision, GNSS module [17]
- LARA-R6001D LTE Cat 1, multi-region, multiband cellular module [19]
- NEO-D9S satellite receiver module for L-band correction [18]

1.3 Required software tools

You can program the XPLR-HPG kits using the examples and libraries available on the XPLR-HPG software repository on GitHub [\[8\]](#) or develop your own. The kit is designed for use with (but not limited to) the Windows operating system and requires the following additional software:

- ESP-IDF core, version 4.4.2 [\[10\]](#)
- Microsoft VS Code [\[11\]](#)
- ESP-IDF extension for VS Code [\[12\]](#)
- u-blox host library, ubxlib [\[16\]](#)

To start developing your own software with the XPLR-HPG kits, check that your computer is using the latest version of the Windows and then clone XPLR-HPG repository [\[8\]](#).

 Some software tools are built-in in the ESP-IDF core installation, such as Python and Git. It is not necessary to install these tools if you always intend to work in the ESP-IDF environment, using the ESP-IDF command prompt or the ESP-IDF PowerShell.

2 Installing the ESP-IDF core

XPLR-HPG software was developed using the Espressif IoT Development Framework (ESP-IDF), version 4.4.2. You must install and use this version to implement your project to comply with the u-blox software. To install the ESP-IDF core, you should follow the steps:

1. Access the ESP-IDF Windows installer download site: <https://dl.espressif.com/dl/esp-idf/>
2. Scroll down to the section **Download links to available releases and mirrors**
3. In this section, scroll down and look for the **Offline Installer 4.4.2**

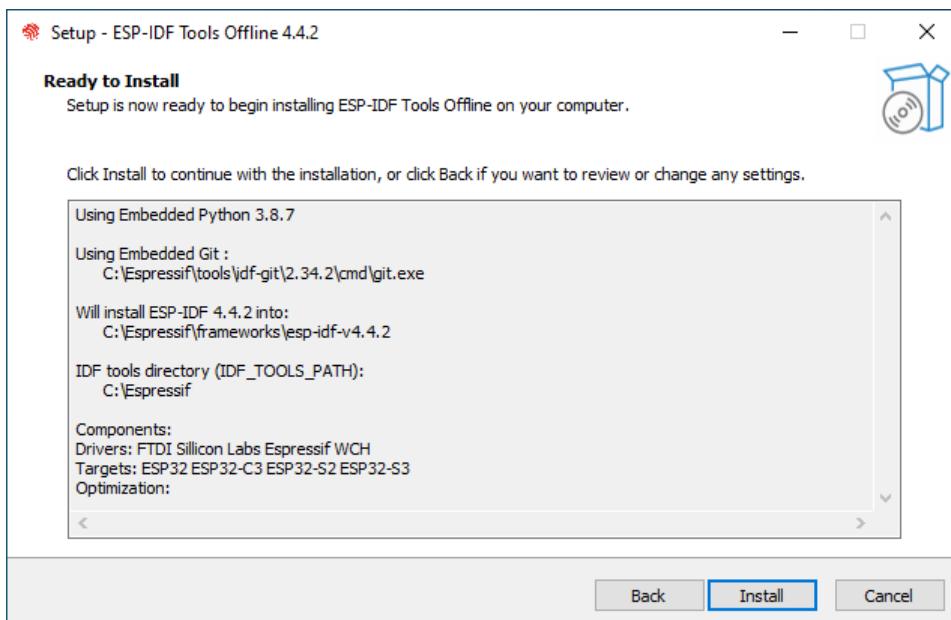
Download links to available releases and mirrors.

Release version	Release date	Release notes
Offline Installer v4.2.4	2022-10-07	Download / Mirror - 376 MB
Online Installer v2.16	2022-08-03	Download / Mirror - 4 MB
Offline Installer v4.4.2	2022-08-18	Download / Mirror - 600 MB
Offline Installer v4.3.3	2022-06-13	Download / Mirror - 585 MB
Offline Installer v4.2.3	2022-03-14	Download / Mirror - 376 MB

4. Double-click on the downloaded file to install the ESP-IDF core on your computer.
5. On the installation window, select the **Setup Language** of your preference and click **OK**.
6. On **License Agreement**, select the “**I accept the agreement**” radio button and click **Next**.
7. On the **Pre-installation system check**, select the **Apply Fixes** button (if necessary) and click **Next**.
8. On **Select Destination Location**, keep the suggested folder (C:\Espressif), and then click **Next**.

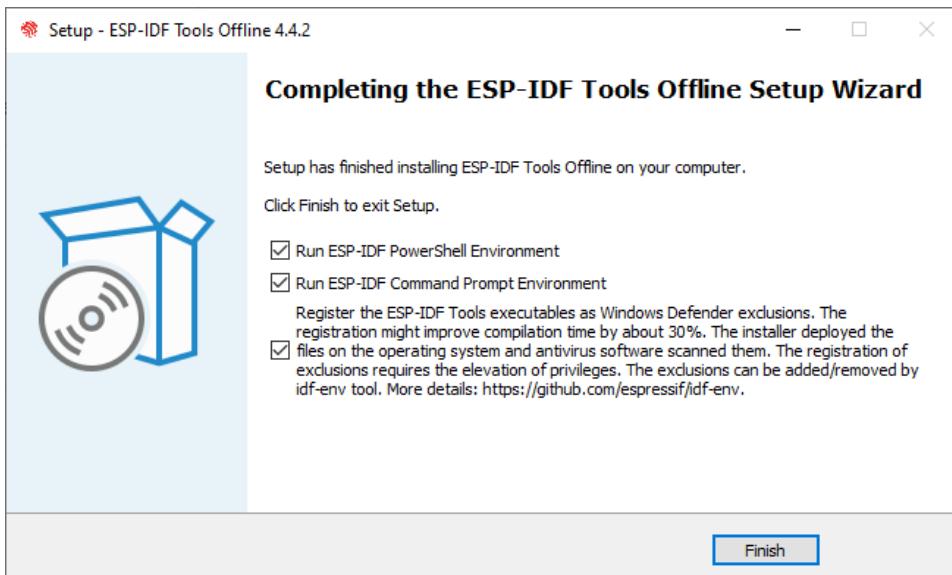
 If you change the default folder, be sure that the absolute path of the new folder does include a space or special Unicode characters. Including any these characters in the ESP-IDF installation path can cause issues during the building process **[9]**.

9. On **Select Components**, select **Full Installation** from the dropdown menu, and click **Next**.
10. Check the installation summary and click **Install**.



11. During the installation, you must permit the setup to configure the ESP-IDF environment. This step prepares and configures the ESP-IDF Power Shell and ESP-IDF Command Prompt environments.

12. When the setup has finished installing the ESP-IDF tools, keep all checkboxes marked and click **Finish**.



13. Allow the ESP-IDF environment to make changes on your computer.

Having followed all installation steps correctly, you now have a working ESP-IDF environment installed in your computer. Two shortcuts should also be included on your desktop: one for running ESP-IDF on Command Prompt and another for running on PowerShell, as shown in [Figure 1](#).



Figure 1. Shortcuts of the ESP-IDF environment

3 Cloning the XPLR-HPG repository

The cloning the XPLR-HPG repository step requires the Git software included in the ESP-IDF environment. The following procedures describe how to clone the XPLR-HPG repository, including its dependencies, using the built-in Git software included in the ESP-IDF environment.

3.1 Opening the ESP-IDF tools

1. Create a folder to save all cloned repositories, for example: C:\git_repos, and open it in a File Explorer window.

 You can choose a different path/folder to save your cloned repositories but be aware that you might experience building errors if the path contains spaces or special Unicode characters.

2. On the Address Bar of the File Explorer windows, type the command below to open the current directory in Command Prompt with the ESP-IDF tools set up, as shown in [Figure 2](#).

```
cmd /k ""C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe"
```

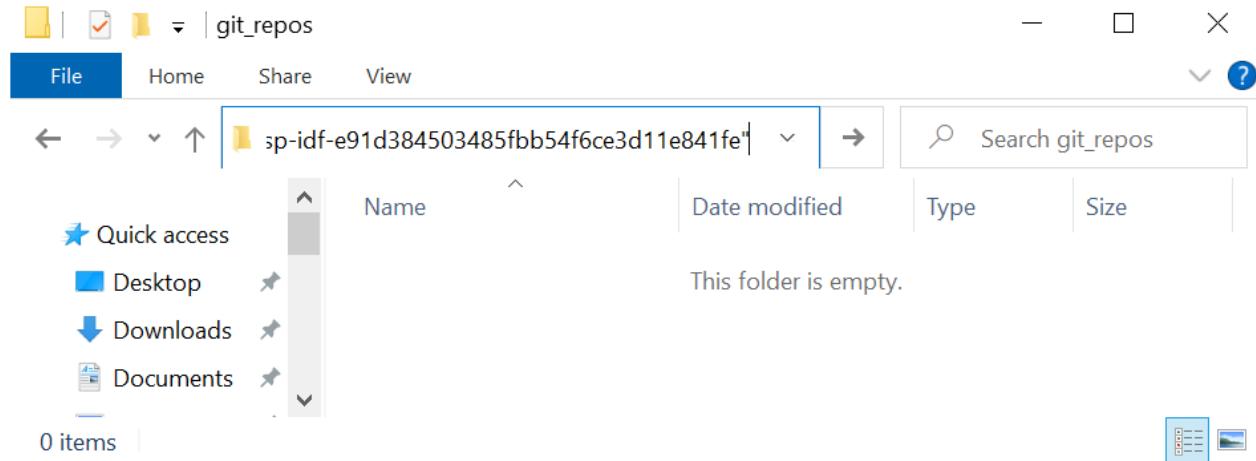
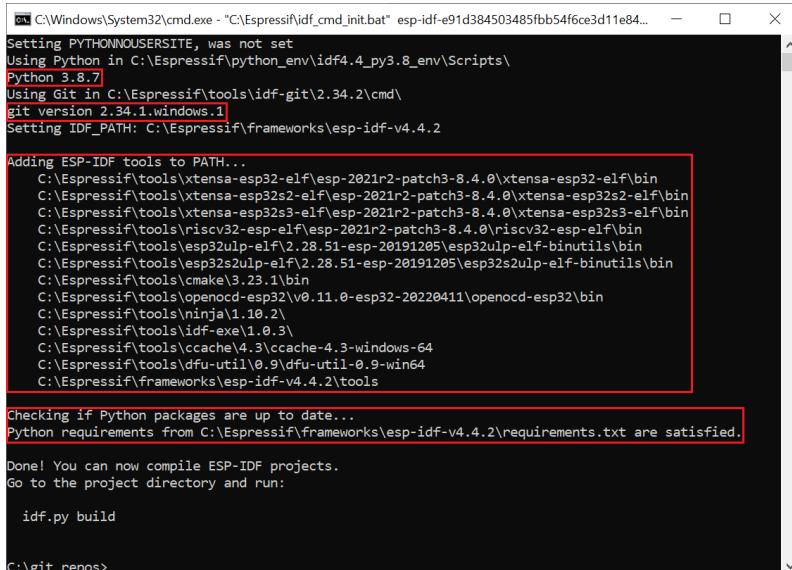


Figure 2: Running the command to open the ESP-IDF tools

 If the open Command Prompt shows an error message or fails to load the required tools correctly, try using the ESP-IDF CMD shortcut shown in [Figure 1](#) or follow the instructions on the ESP-IDF programming website [\[10\]](#).

3. The file opened in the Command Prompt window now includes all of the necessary software for using the ESP-IDF tools, shown in [Figure 3](#). The loaded and configured tools include:

- Python software
- Git software
- Path of the ESP-IDF tool
- Required python package



```

C:\Windows\System32\cmd.exe - "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fb54f6ce3d11e84...
Setting PYTHONNOUSER SITE, was not set
Using Python in C:\Espressif\python\env\idf4.4_py3.8_env\Scripts\Python 3.8.7
Using Git in C:\Espressif\tools\idf-git\2.34.2\cmd\git version 2.34.1.windows.1
Setting IDF_PATH: C:\Espressif\frameworks\esp-idf-v4.4.2

Adding ESP-IDF tools to PATH...
  C:\Espressif\tools\xtensa-esp32-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32-elf\bin
  C:\Espressif\tools\xtensa-esp32s2-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s2-elf\bin
  C:\Espressif\tools\xtensa-esp32s3-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s3-elf\bin
  C:\Espressif\tools\riscv32-esp-elf\esp-2021r2-patch3-8.4.0\riscv32-esp-elf\bin
  C:\Espressif\tools\esp32ulp-elf\2.28.51-esp-20191205\esp32ulp-elf-binutils\bin
  C:\Espressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
  C:\Espressif\tools\cmake\3.23.1\bin
  C:\Espressif\tools\openocd-esp32\v0.11.0-esp32-20220411\openocd-esp32\bin
  C:\Espressif\tools\ninja\1.10.2\bin
  C:\Espressif\tools\idf-exe\1.0.3\bin
  C:\Espressif\tools\ccache\4.3\ccache-4.3-windows-64
  C:\Espressif\tools\dfu-util\0.9\dfu-util\0.9-win64
  C:\Espressif\frameworks\esp-idf-v4.4.2\tools

Checking if Python packages are up to date...
Python requirements from C:\Espressif\frameworks\esp-idf-v4.4.2\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build

C:\git_repos>

```

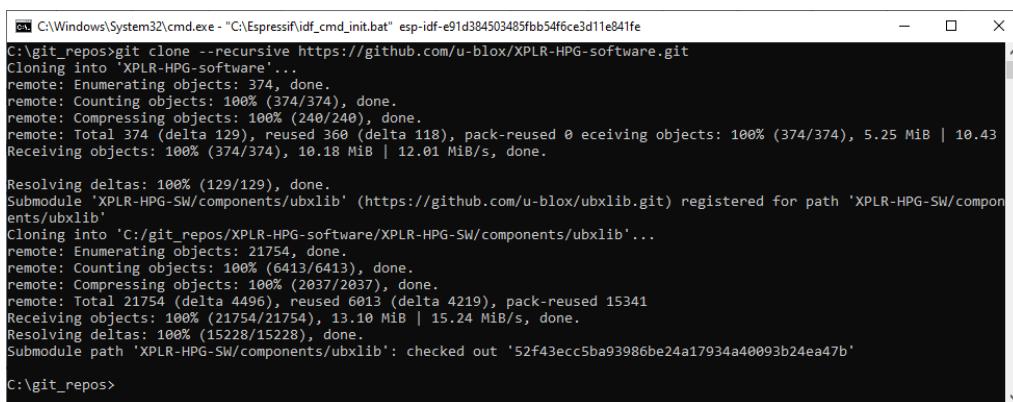
Figure 3: Required tools on the Command Prompt

3.2 Cloning the XPLR-HPG software repository

- Having created and opened the target folder for your git repositories, clone the XPLR-HPG software to your local computer using the following command:

```
git clone --recursive https://github.com/u-blox/XPLR-HPG-software.git
```

 This command clones the XPLR-HPG repository and fetches all sub-module repository dependencies, such as the u-blox library, ubxlib, shown in [Figure 4](#).



```

C:\Windows\System32\cmd.exe - "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fb54f6ce3d11e84...
C:\git_repos>git clone --recursive https://github.com/u-blox/XPLR-HPG-software.git
Cloning into 'XPLR-HPG-software'...
remote: Enumerating objects: 374, done.
remote: Counting objects: 100% (374/374), done.
remote: Compressing objects: 100% (240/240), done.
remote: Total 374 (delta 129), reused 360 (delta 118), pack-reused 0 receiving objects: 100% (374/374), 5.25 MiB | 10.43
Receiving objects: 100% (374/374), 10.18 MiB | 12.01 MiB/s, done.

Resolving deltas: 100% (129/129), done.
Submodule 'XPLR-HPG-SW/components/ubxlib' (https://github.com/u-blox/ubxlib.git) registered for path 'XPLR-HPG-SW/components/ubxlib'
Cloning into 'C:/git_repos/XPLR-HPG-software/XPLR-HPG-SW/components/ubxlib'...
remote: Enumerating objects: 21754, done.
remote: Counting objects: 100% (6413/6413), done.
remote: Compressing objects: 100% (2037/2037), done.
remote: Total 21754 (delta 4496), reused 6013 (delta 4219), pack-reused 15341
Receiving objects: 100% (21754/21754), 13.10 MiB | 15.24 MiB/s, done.
Resolving deltas: 100% (15228/15228), done.
Submodule path 'XPLR-HPG-SW/components/ubxlib': checked out '52f43ecc5ba93986be24a17934a40093b24ea47b'

C:\git_repos>

```

Figure 4: Cloning the XPLR-HPH repository

- After cloning the repository, close the Command Prompt.

4 Installing Visual Studio Code and extensions

Visual Studio (VS) Code is the recommended integrated development environment (IDE) for implementing applications using the XPLR-HPG software. It is a versatile tool with several extensions that allow you to add new languages, themes, debugging tools and connect to additional services [11].

To use XPLR-HPG software, you must install VS Code with the Espressif and C++ extensions.

4.1 Installing VS Code

If you have not already, install the VS Code on your computer, go to the VS Code website [11]. Download a stable build installation for the Windows operating system and install it using the default instructions provided with the software installer.

4.2 Installing Espressif IDF extension

This Espressif IDF extension helps you develop, build, flash, monitor, and debug your applications for the XPLR-HPG kits, using Espressif IoT Development Framework (ESP-IDF) [12].

To install the Espressif IDF extension in your computer (Figure 5):

1. Open the VS Code IDE and click the **Extension** button in the **Activity Bar** [13].
2. In the Search bar, type **espressif**
3. Select the correct extension and select the install button, as shown in Figure 5.

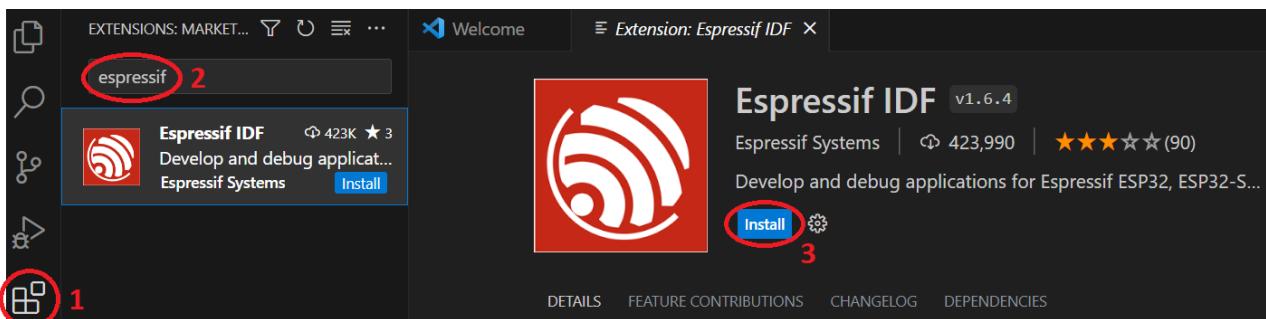


Figure 5: How to install the ESP-IDF extension

The ESP-IDF extension also installs the C/C++ extension, as shown in Figure 6. The extension software is required for building c-based projects.

☞ If the automatic installation of the C/C++ extension fails to work, install it manually.

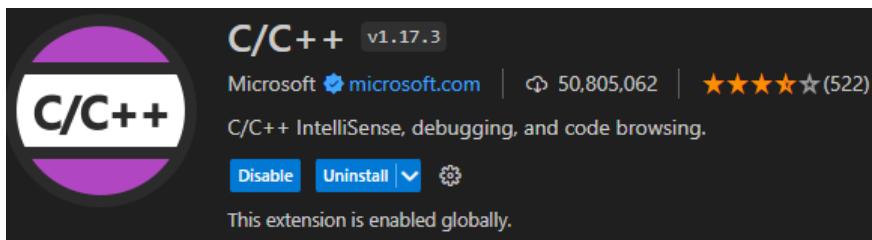


Figure 6: C/C++ extension

4.3 Setting up the ESP-IDF extension

After adding the Espressif extension in VSC, configure the extension:

1. Press F1 or CTRL+SHIFT+P and type **ESP-IDF: Configure ESP-IDF extension**, as shown in [Figure 7](#).

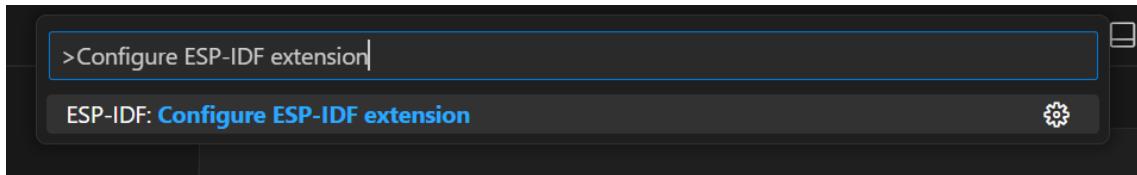


Figure 7: VS Code search bar

2. On the ESP-Setup tab, choose the **Global** option.
3. From **Select where to save these settings**, select the **USE EXISTING SETUP** option shown in [Figure 8](#).

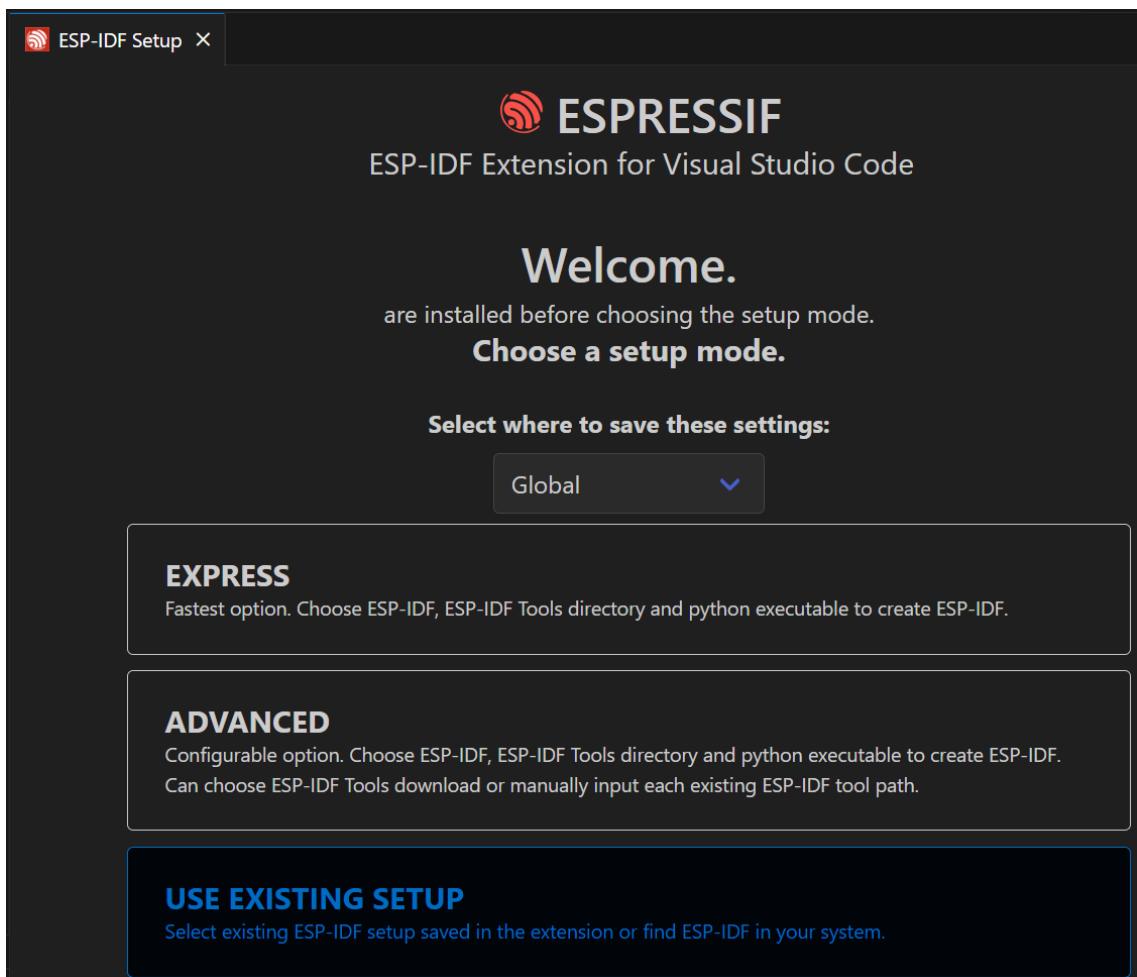


Figure 8: ESP-IDF extension setup

4. On following window, select `esp-idf-v4.4.2`. This is the same installation that you chose in [Installing the ESP-IDF core](#).

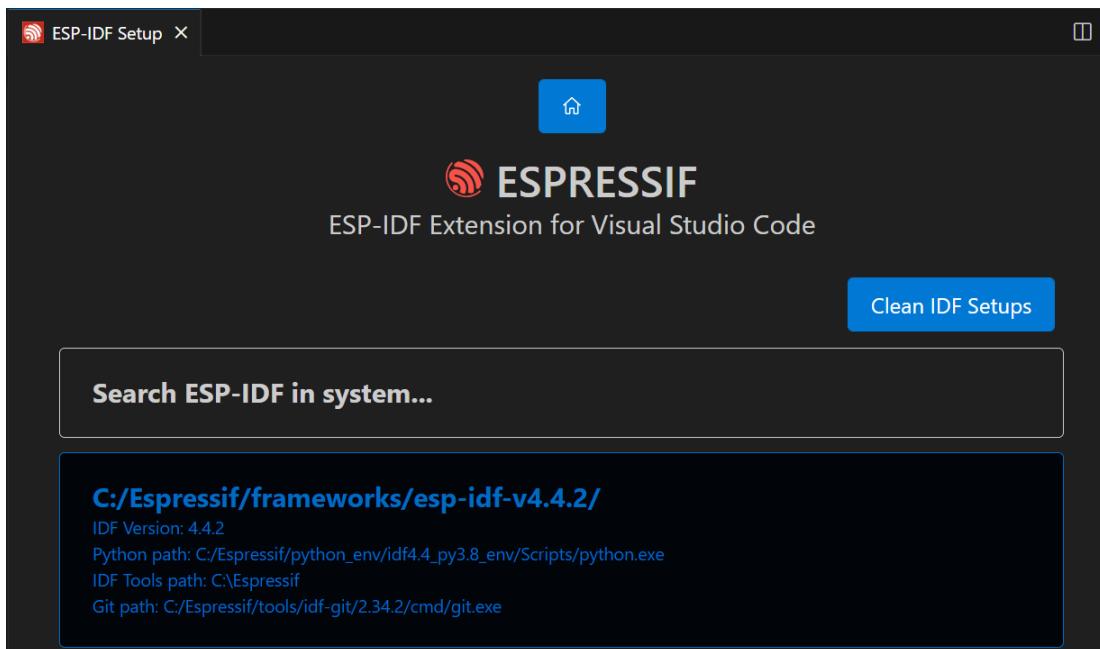


Figure 9: Selecting the existing setup for version 4.4.2.

A page showing the setup progress status is displayed, as shown in [Figure 10](#).

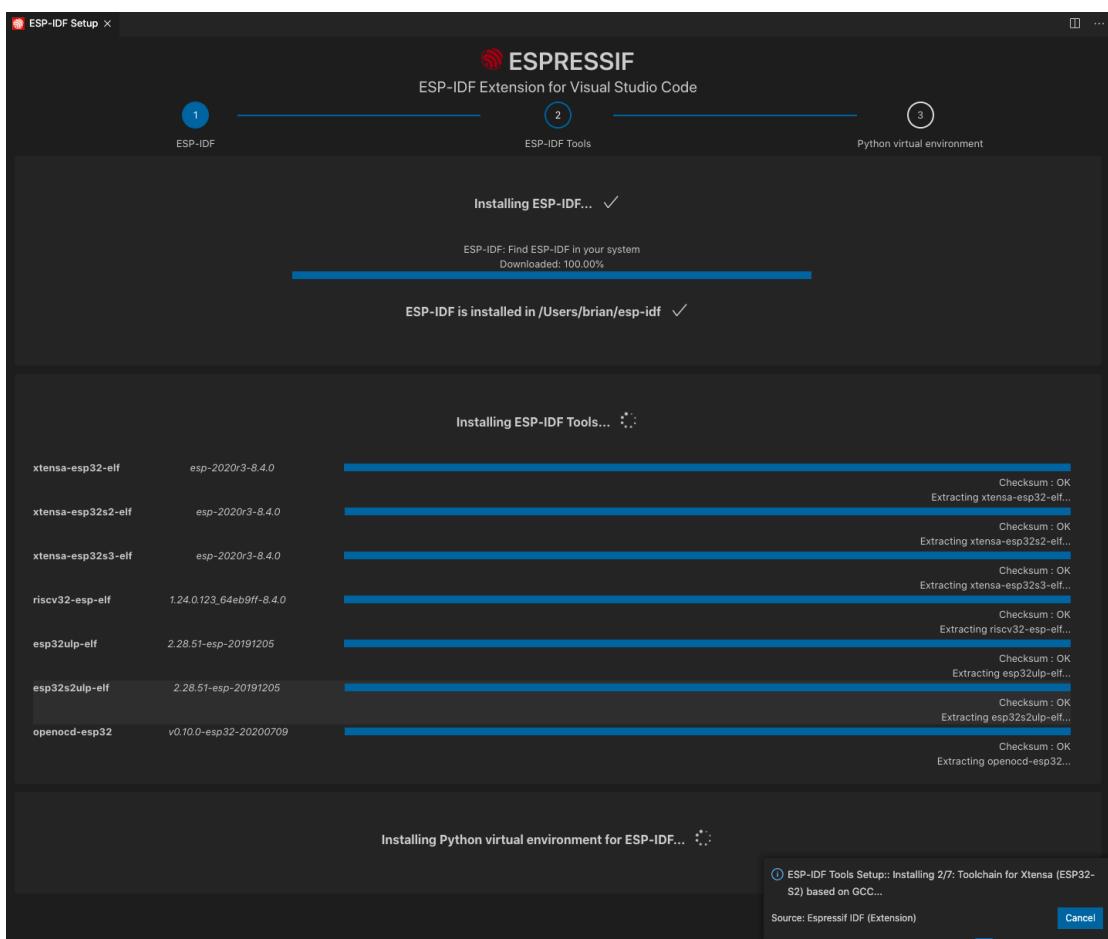
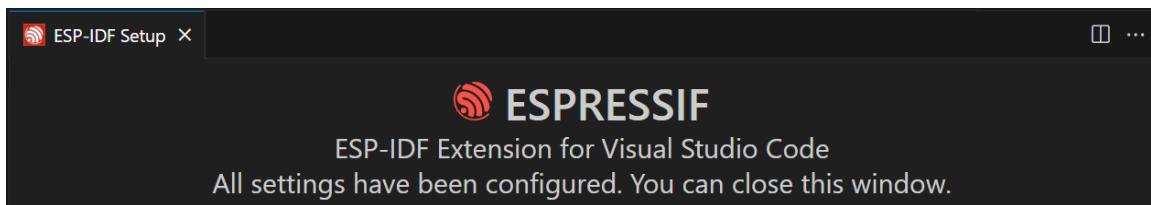


Figure 10: Extension setup progress status

5. Wait until the install is completed, as shown in [Figure 11](#).



[Figure 11: Extension setup is completed](#)

4.4 ESP-IDF extension toolbar

After installing the ESP extension software successfully, you can see the Espressif extension toolbar on the left bottom of the VS Code IDE. The toolbar has several buttons that allow you trigger different actions in your project. [Figure 12](#) shows the main buttons of the ESP-IDF toolbar.



[Figure 12: ESP-IDF extension toolbar](#)

Each highlighted button shown on the toolbar has a specific function:

- **COMPORT**: selects the XPLR-HPG-2 communication port
- **Device target**: selects the CPU (for example, XPLR-HPG-1 -> esp32s3 and XPLR-HPG-2 -> esp32)
- **Current directory**: changes the Project directory
- **Configuration Editor**: opens **menuconfig** to set up the board/example configurations
- **Full Clean**: cleans up any existing build (deletes the build directory from the project)
- **Build Project**: builds the current project.
- **Select Flash Method**: chooses eJTAG, **UART** or DFU.
- **Flash Device**: flashes the binary files into the device
- **Monitor**: opens the monitoring terminal to print debug messages.
- **Build, Flash and Monitor**: executes three actions sequentially (build the project, flash the device and open the monitor terminal).
- **Open ESP-IDF terminal**: opens a Command Prompt with the ESP-IDF required tools

5 Building and flashing the captive portal

After installing all required applications, you can choose one of the code examples included in the XPLR-HPG software, build it, and then flash it to your XPLR-HPG kit. At this stage, you already have the required development environment to start your own project for the XPLR-HPG product.

The `hpg_wifi_mqtt_correction_captive_portal` example comes flashed from factory in the XPLR-HPG-1 and XPLR-HPG-2 kits.

To select, build, and flash the Captive Portal example:

1. Open the **XPLR-HPG-SW** folder on the VS Code IDE. You can open it by accessing the menu **File > Open Folder** and selecting the XPLR-HPG-SW folder, as shown in [Figure 13](#).

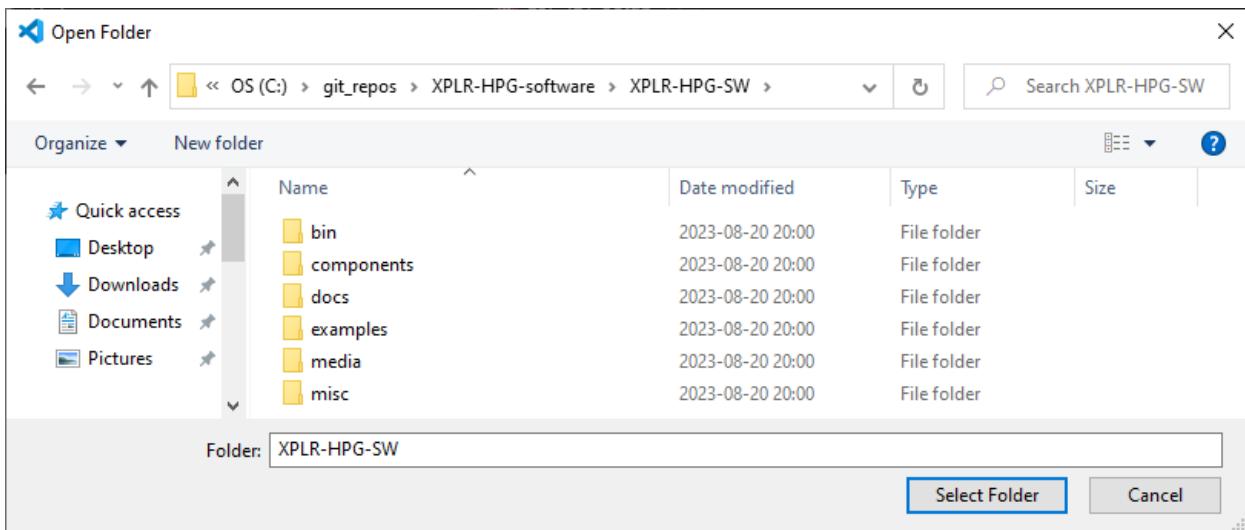


Figure 13: Selecting the XPLR-HPG-SW folder

⚠ Ensure the XPLR-HPG-SW folder is on the top of the project directory tree ([Figure 14](#)). Otherwise, several error messages are displayed during the building process.

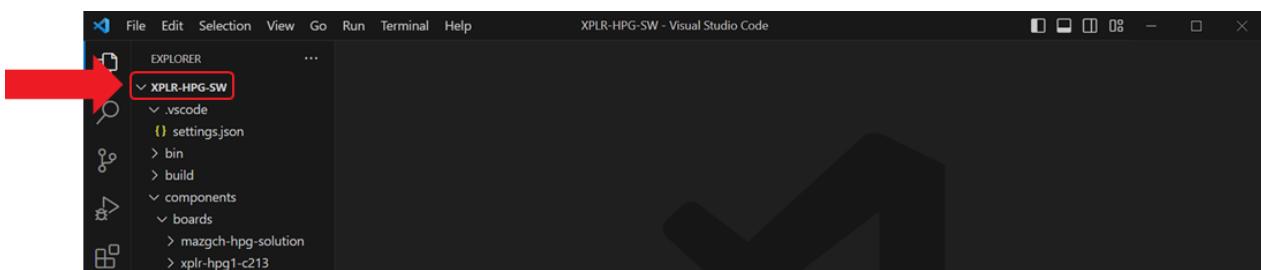


Figure 14: Project directory tree

- When opening the XPLR-HPG folder in VS Code for the first time, you must affirmatively "... trust the authors of the files in this folder" to enable all features in the XPLR-HPG software. To do that, click the checkbox and select "Yes, I trust the authors", as shown in [Figure 15](#).

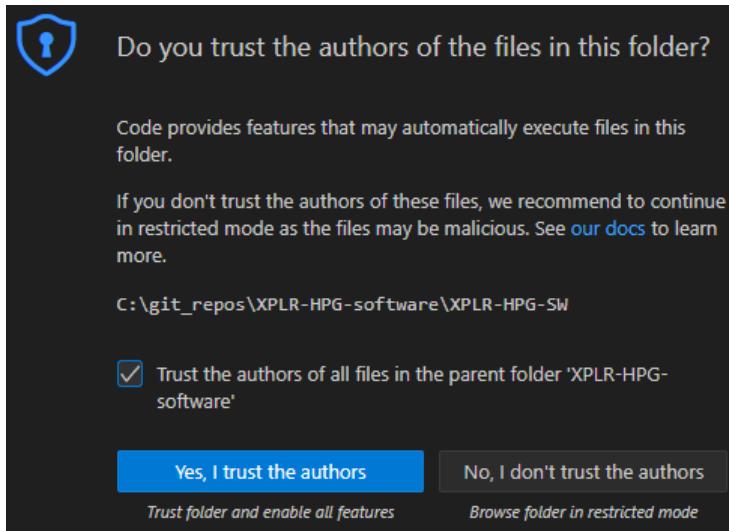


Figure 15: Enabling the features of the XPLR-HPG software

☞ Notification windows might be displayed on the bottom right of the VS Code IDE. It is not necessary to take any action. Just ignore and close them.

5.1 Connecting the board on your computer

- Using the provided USB cable, connect the XPLR-HPG kit to a USB port on your computer.
- Open the Device Manager and check the COMPORT addressed to the XPLR-HPG board. It will be the one named as Silicon Labs CP210x USB to UART Bridge ([Figure 16](#)).

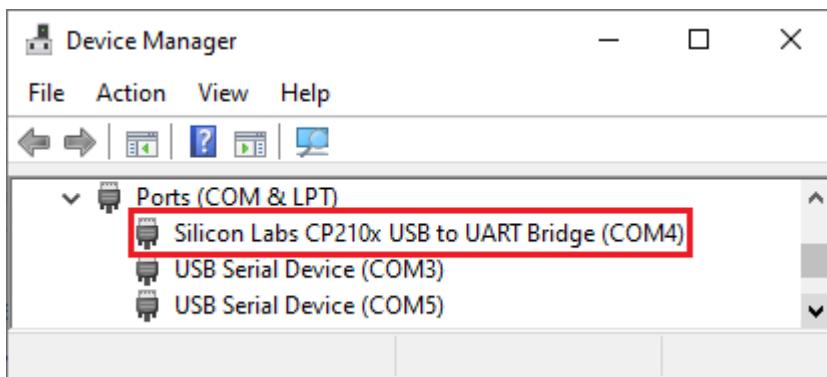


Figure 16: XPLR-HPG communication port

☞ If the correct communication port is not displayed, this might be attributable to a driver problem. In this case, you should access the CP210x USB to UART Bridge VCP Drivers website [\[14\]](#) and download the CP210x Universal Windows Driver.

3. Go to the VS Code IDE and, on the ESP-IDF toolbar, choose the COM option to select the board COMPORT, as shown in [Figure 17](#).

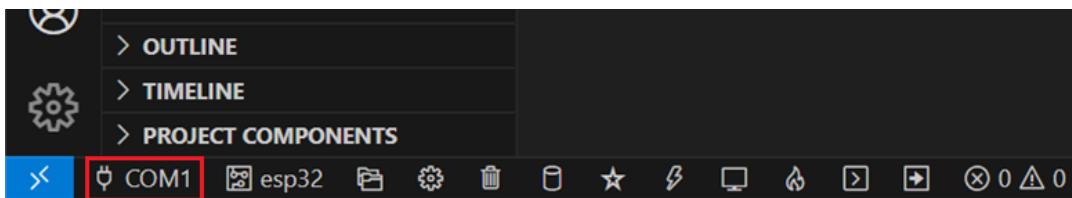


Figure 17: Select the board communication port

4. A drop-down menu is displayed in the middle of the VS Code IDE. Select the board COMPORT, as shown in [Figure 18](#). After choosing the COMPORT, select the XPLR-HPG-SW folder as a Workspace folder and apply the changes.

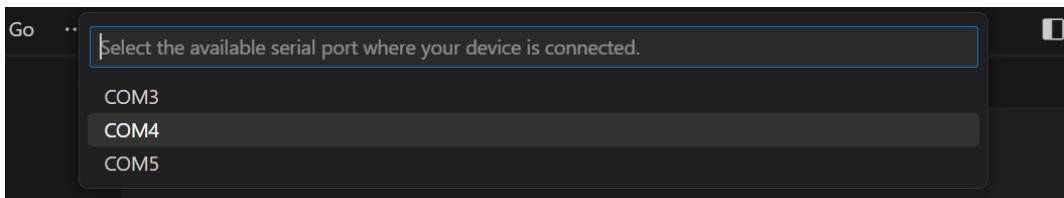


Figure 18: Select the board COMPORT

5. If necessary, select the device target. Click the target device option and choose XPLR-HPG-SW as the Workspace folder.
6. From the drop-down menu, select the correct **ESP** device¹ of your board, as shown in [Figure 19](#). Enter **OpenOCD configuration** and select Custom board.

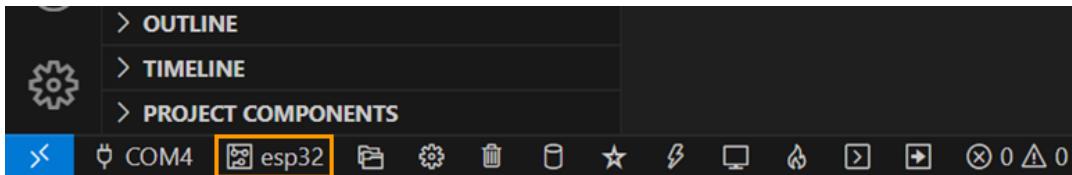


Figure 19: Target device option

5.2 Building and flashing the code

To build the `hpg_wifi_mqtt_correction_captive_portal` example:

1. In VS Code, open the **CMakeList.txt** file and “uncomment” the captive portal example, as shown in [Figure 20](#).

⚠ Ensure that there is only one uncommented project.

 A screenshot of the VS Code editor showing the 'CMakeLists.txt' file. The code is as follows:


```

17  # shortrange examples
18  #set(ENV{XPLR_HPG_PROJECT} "hpg_base")
19  #set(ENV{XPLR_HPG_PROJECT} "hpg_wifi_http_ztp")
20  #set(ENV{XPLR_HPG_PROJECT} "hpg_wifi_mqtt_correction_certs")
21  #set(ENV{XPLR_HPG_PROJECT} "hpg_wifi_mqtt_correction_ztp")
22  set(ENV{XPLR_HPG_PROJECT} "hpg_wifi_mqtt_correction_captive_portal")
  
```

 The line '22 set(ENV{XPLR_HPG_PROJECT} "hpg_wifi_mqtt_correction_captive_portal")' is highlighted with a red box.

Figure 20: Selecting the captive portal on the shortrange examples section.

¹ Select esp32s3 for XPLR HPG 1 kit and esp32 for XPLR-HPG-2.

2. Delete the `sdkconfig` file and the `build` folders if they are displayed inside the `XPLR-HPG-SW` directory, as shown in [Figure 21](#).

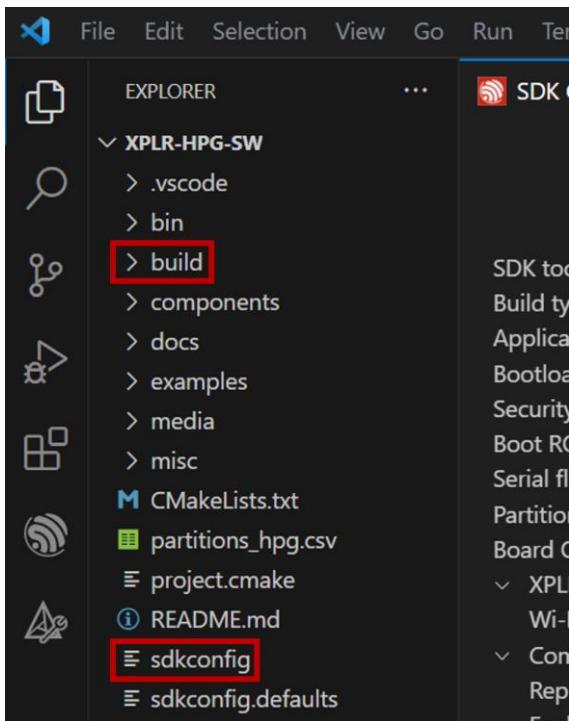


Figure 21: `sdkconfig` file and `build` folder

3. Click on the **cog/gear** button to set up the default configuration for chosen example, as shown in [Figure 22](#).

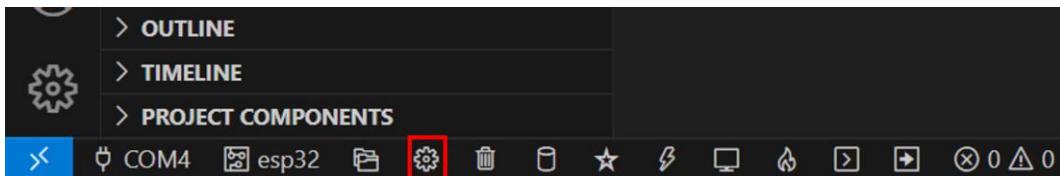


Figure 22: Setting up the default configuration

4. In the **SDK Configuration editor** window, select **Board options** and then **Choose Target Board**. Select one of the available board models and select **Save**, as shown in [Figure 23](#).

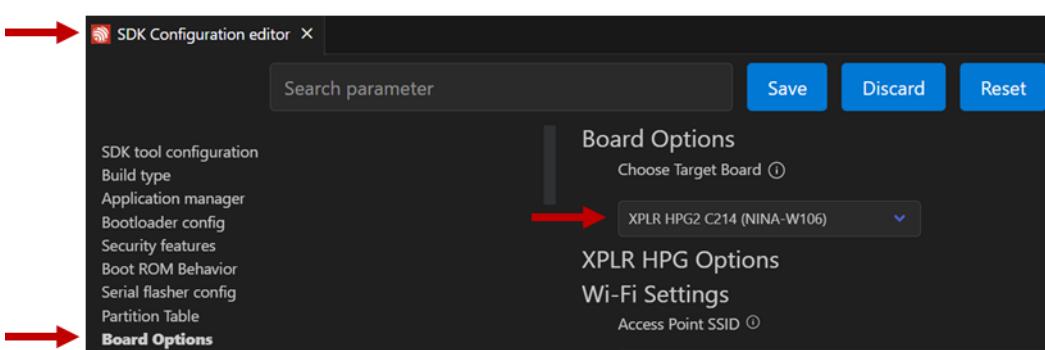
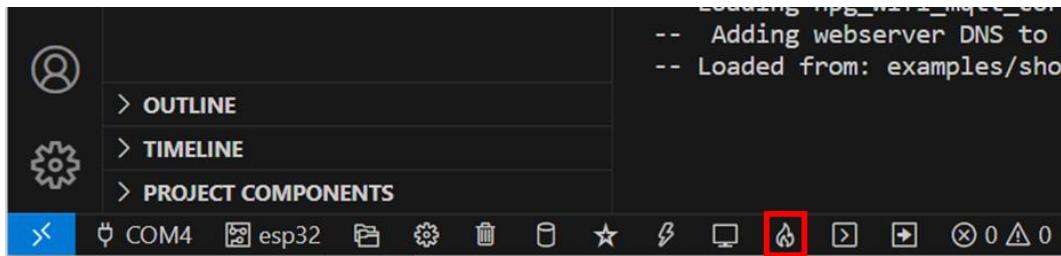


Figure 23: Selecting the board version

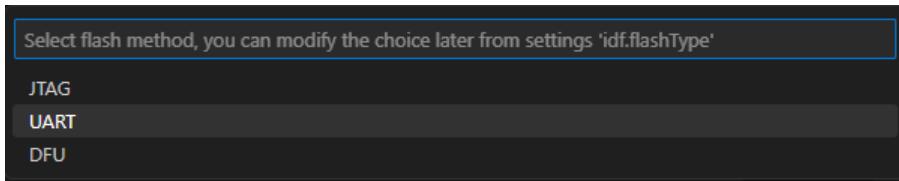
- Select one of three options: XPLR HPG2 C214 (NINA-W106), XPLR HPG1 C213, or MAZGCH HPG SOLUTION (NINA-W106). Select the model you are using.

5. From the ESP-IDF toolbar, select **Flame** to build and flash the project to the XPLR-HPG board, as shown in [Figure 24](#). This option also open the monitor on the terminal to print the log messages.



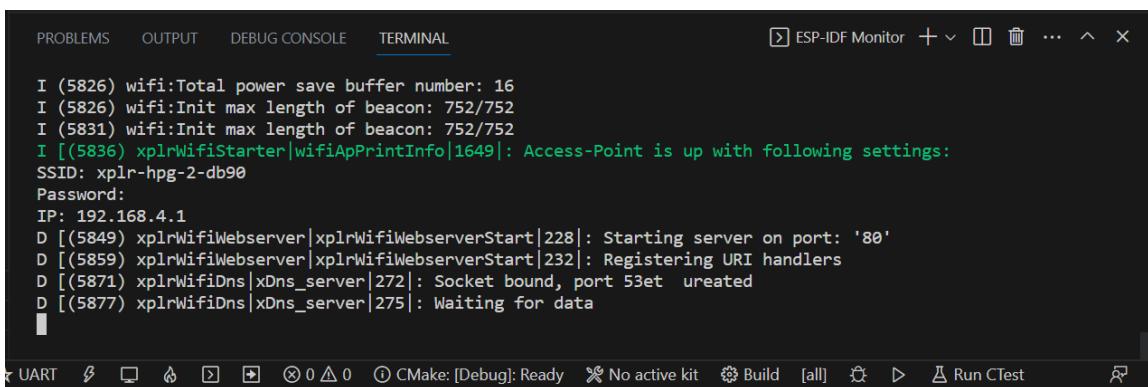
[Figure 24: Build, flash and monitor option](#)

6. After building the example, select the **flash method** to upload the application into the board, as shown in [Figure 25](#). Choose the **UART** method.



[Figure 25: Selecting the flash method](#)

7. After flashing the board, the printed messages displayed on the monitor console are opened in the VS Code terminal, as shown in [Figure 26](#).



[Figure 26: Monitor console.](#)

8. After successfully executing all described parts in this document, follow the instructions available for the ESP-IDF VS Code extension [\[15\]](#). Add your Wi-Fi and Thingstream credentials in the web interface before opening the Live tracking map to see your location.

Appendix

A Glossary

Abbreviation	Definition
ESP-IDF	Espressif IoT Development Framework
IDE	Integrated Development Environment

Table 1: Explanation of the abbreviations and terms used

Related documentation

- [1] XPLR-HPG-1 user guide, UBX- 23000692
- [2] XPLR-HPG-2 user guide, UBX- 23003719
- [3] MIKROBUS – Mikroelektronika Doo Beograd Zemun Trademark: [\[link\]](#)
- [4] Click boards™ standard: [\[link\]](#)
- [5] GNSS RTK 2 Click board with ZED-F9R multi-band GNSS module, [\[link\]](#)
- [6] LBAND RTK 2 Click board with NEO-D9S-00B satellite receiver for L-band correction, [\[link\]](#)
- [7] 4G LTE 2 Click board with the LARA-R6 LTE Cat 1 cellular module [\[link\]](#)
- [8] XPLR-HPG software on GitHub: [\[link\]](#)
- [9] ESP Forum: ninja: build stopped issue: [\[link\]](#)
- [10] Updating ESP-IDF tools on Windows – ESP32 – ESP-IDF Programming: [\[link\]](#)
- [11] Visual Studio Code: [\[link\]](#)
- [12] ESP-IDF VS Code extension – Installation: [\[link\]](#)
- [13] Visual Studio Code User Interface: [\[link\]](#)
- [14] CP210x USB to UART Bridge VCP Drivers: [\[link\]](#)
- [15] Live tracking map using a captive portal: [README.md](#)
- [16] u-blox host library, [ubxlib](#)
- [17] ZED-F9R data sheet, [UBX-22024085](#)
- [18] NEO-D9S data sheet, [UBX-19026111](#)
- [19] LARA-R6 data sheet, [UBX-21047783](#)
- [20] NORA-W10 data sheet, [UBX-21036702](#)
- [21] ESP-IDF core, version 4.4.2, [\[link\]](#)
- [22] Git download for Windows page: [\[link\]](#)
- [23] Python installer page: [\[link\]](#)
- [24] MSYS2 website: [\[link\]](#)

 For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Revision history

Revision	Date	Name	Comments
R01	26-Oct-2023	ftor, aang	Initial release

Contact

u-blox AG

Address: Zürcherstrasse 68
8800 Thalwil
Switzerland

For further support and contact information, visit us at www.u-blox.com/support.