Getting started

Development Kit

Contents

1 Document Info

This document provides technical information about the first steps with the iMCP HTNB32L-XXX. It is intended to contribute only the necessary information to run the basic examples available in its SDK. For more information, please go to our link tree.

This project was supported by the Ministery of Science, Tecnology and Inovations, with resources from Law number 8.248, Oct 23, 1991, within the scope of PNM-Design, coordinated by Softex and published microelectronics 2A, DOU 01245009534/2022-14.

2 Setup Environment

This section describes the setup environment that users should build in order to start developing and testing their application. It is strongly recommended to follow all instructions pointed out here to reproduce the expected results.

2.1 Operating System

Although the HTNB32L-XXX SDK could be easily adapted to Unix systems, the current version was developed to run only in the following OS:

- Windows 10: 32 or 64 bits.
- Windows 11: 32 or 64 bits.

2.2 iMCP HTNB32L-XXX SDK

You can access the iMCP HTNB32L-XXX SDK repository here:

- https://github.com/htmicron/HTNB32L-XXX-SDK

iMCP HTNB32L-XXX SDK is composed by the following resources:

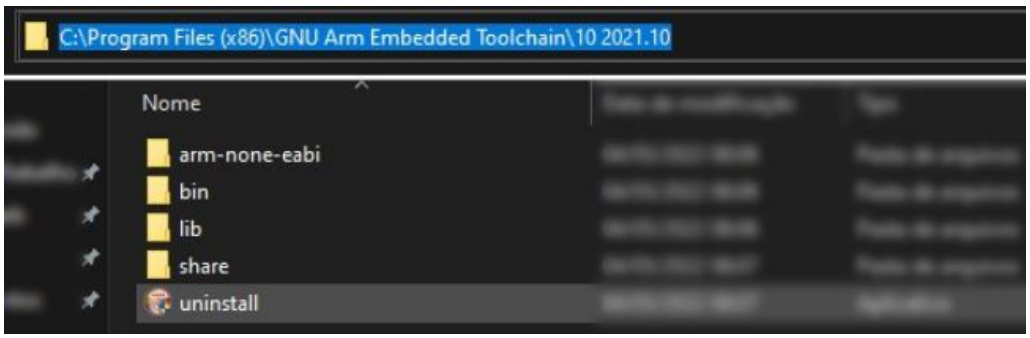
- Visual Studio Code Workspace (with debugging and flashing features).
- Firmware Examples.
- Software Applications.
- Flashing tool.
- Application notes and User Guides.

2.3 Toolchain Download and Installation

GNU ARM Embedded Toolchain, which includes GNU Compiler (GCC), is used to compile the HTNB32L-XXX firmware. The following steps show how the compiler should be installed:

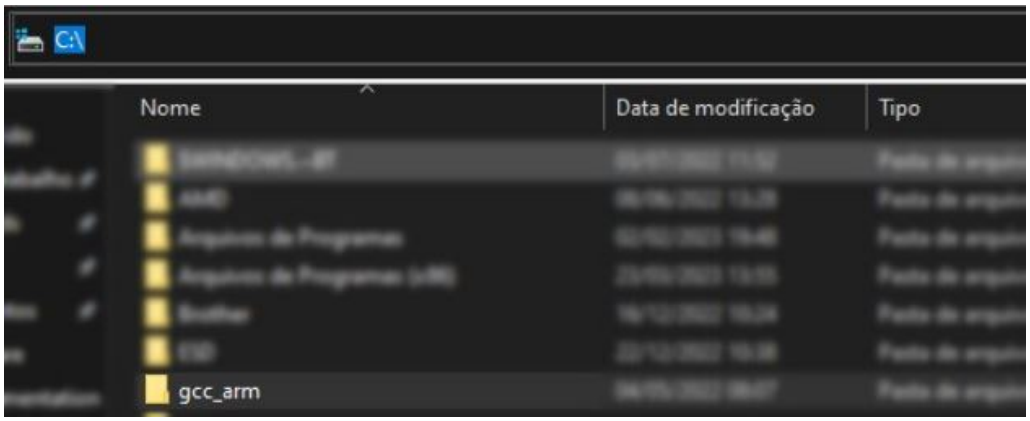
- Go to ARM Developer website and download and install the lasted version of GNU ARM Embedded Toolchain (gcc-arm-none-eabi-10.3-2021.10-win32.exe).
- Open the directory where you have installed it (usually "C:\Program Files (x86)\GNU Arm Embedded Toolchain"):

Figure 1: Toolchain path



- Copy the "10 2021.10" (it depends on the toolchain version you had installed) directory and paste it to your "C:\".
- Rename the directory to "gcc_arm":

Figure 2: Renaming compiler path to "gcc_arm"



2.4 Make download

The compilation procedure requires the Make tool installed on Windows. Following instructions indicates how to download and install Make on Windows using Chocolatey:

- Install Chocolatey: Here
- Open a PowerShell terminal and run the following command to install Make on Windows:

choco install make

2.5 Download and installation

You can access the iMCP HTNB32L-XXX SDK repository here:

- https://github.com/htmicron/HTNB32L-XXX-SDK

To clone the repository, use this command on git bash:

- git clone https://github.com/htmicron/HTNB32L-XXX-SDK.git

2.6 Visual Studio

Even though users can choose any other text editor of their preference, Visual Studio Code is the programming tool recommended to utilize with HTNB32L-XXX SDK. A complete and ready-to-use workspace, with debugging and flashing features, was built on VSCODE to support developers in their programming process.

VSCODE Extensions

To benefit from all features available in this workspace, the following list of VSCODE Extension must be installed:

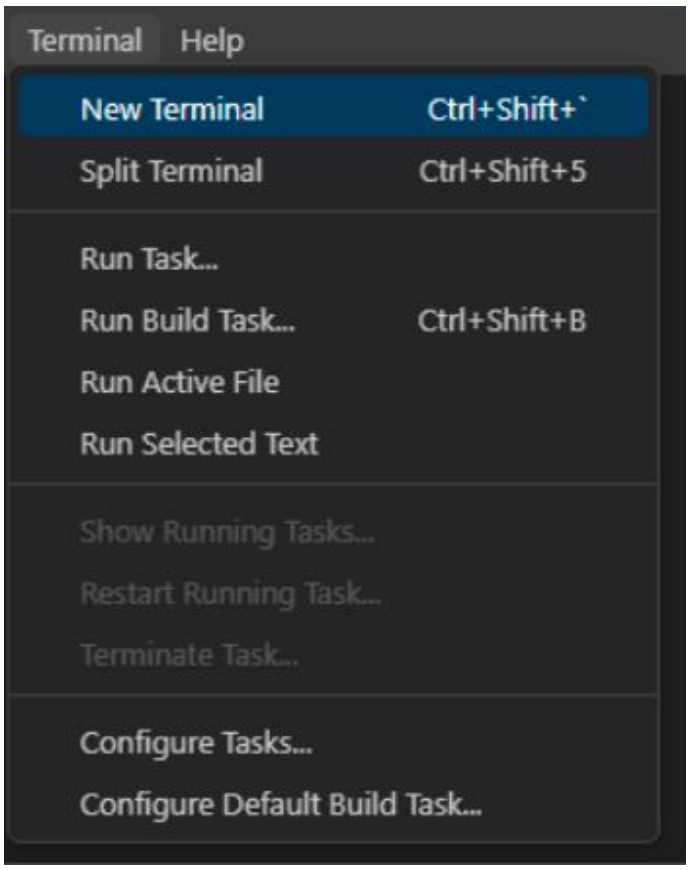
- C/C++
- Cortex-Debug.
- Embedded Tools.
- Makefile Tools.
- RTOS Views.
- Serial Monitor.
- ARM Assembly.
- Python.

VSCODE Default Terminal Profile

Due to some specific characteristics of Windows CMD, it is strongly recommended to change the VSCODE default terminal to the Git Bash terminal. Instructions below show how to do it:

- Open your VSCODE.
- Click on “Terminal”, “New Terminal”, as it is shown in Figure:

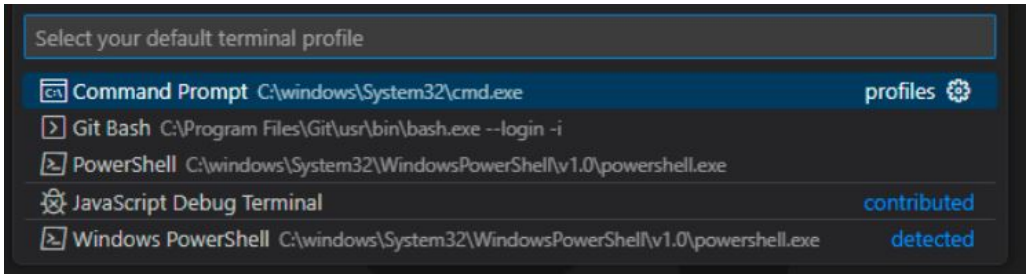
Figure 3: Opening a new terminal on VSCODE.



- Go to "Select Default Profile" and change the default terminal to "Git Bash". Figure shows where to find the "Select Default Profile" option and the other Figure shows the CMD terminal being replaced to Git Bash as default terminal profile.

Figure 4: "Select Default Profile" location.

Figure 5: Replacing default terminal to Git Bash.



Python packages

Scripts developed in Python are required to run the workspace built as programming tool for HTNB32L-XXX users. It is mandatory to download and install Python 3.9.8 or later. Table shows the list of Python packages that must be installed:

Package	Command
PyQt5	pip install PyQt5
PyQt5-Tools 5.15.9.3.3	pip install PyQt5-tools
PySerial	pip install pyserial

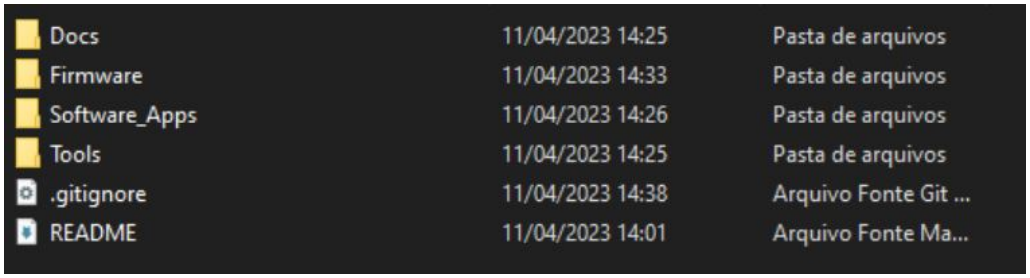
3 IMCP HTNB32L-XXX SDK

The Software Development Kit (SDK) built for HTNB32L-XXX is intended to provide a complete environment for programmers develop their own firmware application. It contains all libraries and examples needed for an OpenCPU solution, where users can embed their custom code. Following sections describe how HTNB32L-XXX SDK is assembled and how to configure and operate it.

3.1 SDK directories

This subsection describes the main directories available on HTNB32L-XXX SDK. The figure bellow shows the directories available in the SDK and the table describe them:

Figure 6: “SDK directories” location.



Name	Description
Docs	Directory where Application Notes and documents related to firmware and software are stored.
Firmware	Firmware workspace environment and examples. This directory also contains binary files that can be directly flashed to a HTNB32L-XXX device.
Software_Apps	Software developed as accessory for some specific firmware applications.
Tools	Development tools required.

The table bellow describes the Firmware directory, where the workspace environment for HTNB32L-XXX can be found.

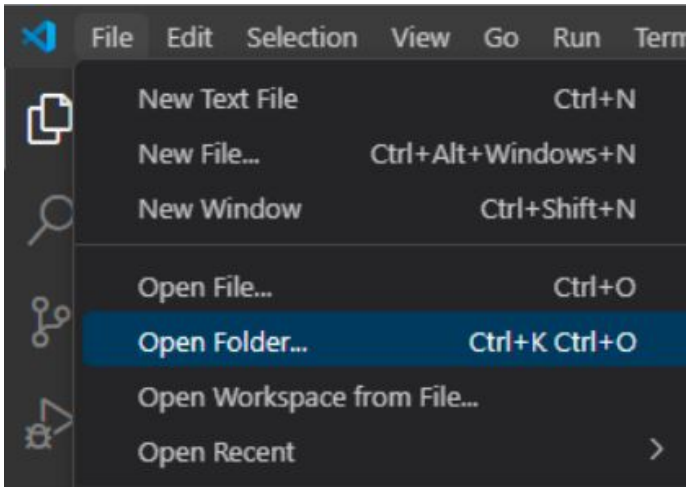
Name	Description
.vscode	Directory where all debugging and flashing environment are described (tasks, launch and settings etc).
Applications	Where the application examples for the HTNB32L-XXX are implemented. It is also where users should implement their own firmware.
Build	Directory where the binary and object files generated from an APP compilation will be stored. This directory is created after compiling an application available on SDK.
Debug	Directory that contains every script used in the flashing and debugging process.
SDK	Where driver libraries and their respective header files are stored on.
HT Prebuild	Static libraries pre-compiled for user applications.

3.2 Opening VSCODE Workspace

The following guidelines shows how to properly launch the VSCODE workspace:

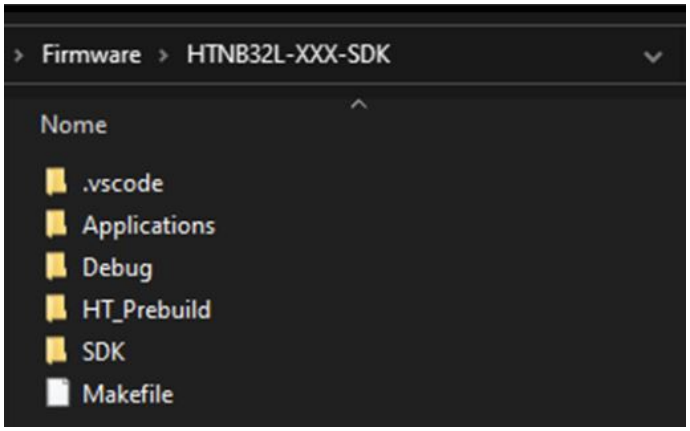
- Double check if the VSCODE environment was properly configured.
- Open VSCODE on your Windows OS.
- Click on “File”, “Open Folder...” as is shown in Figure bellow:

Figure 7: “Opening VSCODE environment.” location.



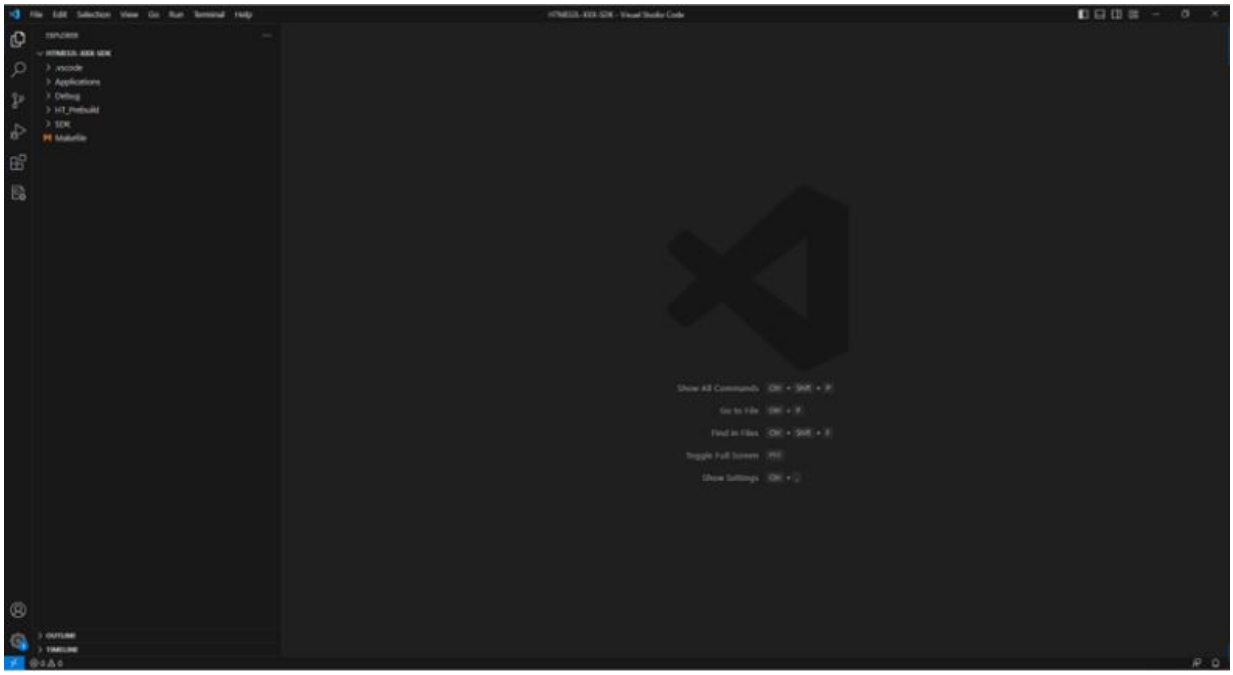
- Go to where the HTNB32L-XXX SDK repository was cloned and select the “Firmware/HTNB32L-XXX-SDK” directory:

Figure 8: “Firmware directory.” location.



- shows the VSCODE workspace properly launched:

Figure 9: “HTNB32L-XXX-SDK opened on VSCODE editor.” location.



4 Compiling Procedure

There are two different ways to compile a firmware example available on HTNB32L-XXX SDK. The most direct one uses the VSCODE environment to call the compilation task described on “.vscode” directory (Direct Compilation) and the second one executes the compiler by command line (CLI Compilation).

4.1 Compiler Path

Follow the instructions mentioned earlier to download and install GCC ARM toolchain. In order to change the compiler path, two different parameters must be replaced:

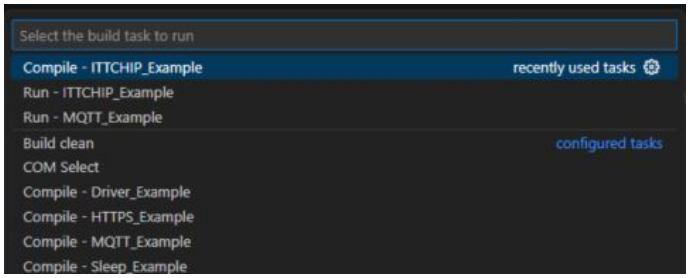
- “GCCLIB_PATH”, located at “Firmware/HTNB32L-XXX-SDK/SDK/PLATtools/scrips/Makefile.vars” file.
- “cortex-debug.gdbPath”, located at “Firmware/HTNB32L-XXX-SDK/ .vscode/settings.json” file.

4.2 Direct Compilation

The following steps show how to start a Direct Compilation, utilizing the VSCODE workspace built specially for HTNB32L-XXX SDK:

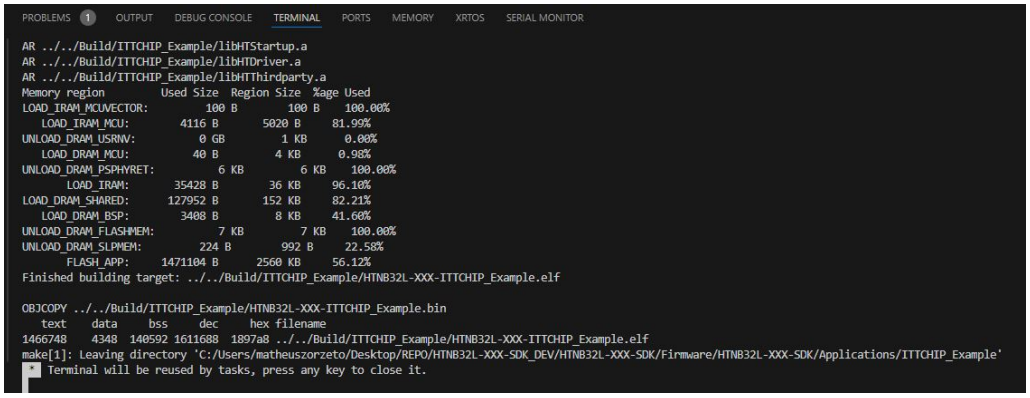
- Open the HTNB32L-XXX-SDK directory in your VSCODE editor.
- Press “CTRL+SHIFT+B” to open the task list with all tasks available in this workspace:

Figure 10: “HTNB32L-XXX SDK task list.” location.



- Select the example that is supposed to be compile and press “Enter”.
- A terminal should be opened showing the compilation logs. The figure bellow shows the logs generated from the example compilation:

Figure 11: “Compilation logs.” location.



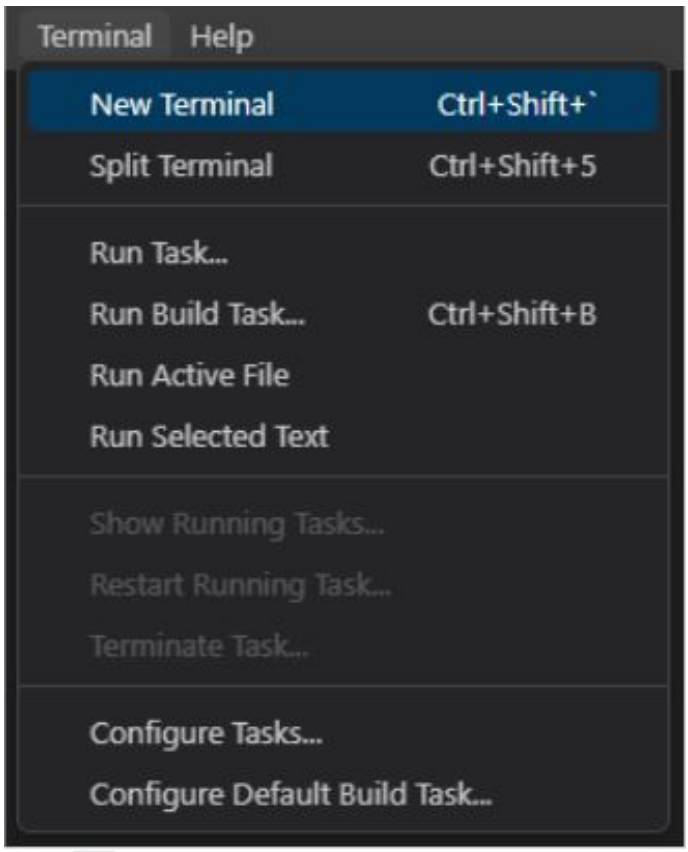
- The binary file will be available on “Build” directory.

4.3 CLI Compilation

Above steps show how to compile the firmware using command line:

- Open the HTNB32L-XXX-SDK directory in your VSCODE editor.
- Click on “Terminal” and select “New Terminal” option to open a new PowerShell (or bash, depending on your VSCODE settings) terminal:

Figure 12: “Opening a new PowerShell terminal.” location.



- Type the command related to the Application Example that is supposed to be compiled. The command structure is shown below:

  make -j4 gccall TARGET=qcx212_0h00 V=0 PROJECT=<APPLICATION_EXAMPLE>

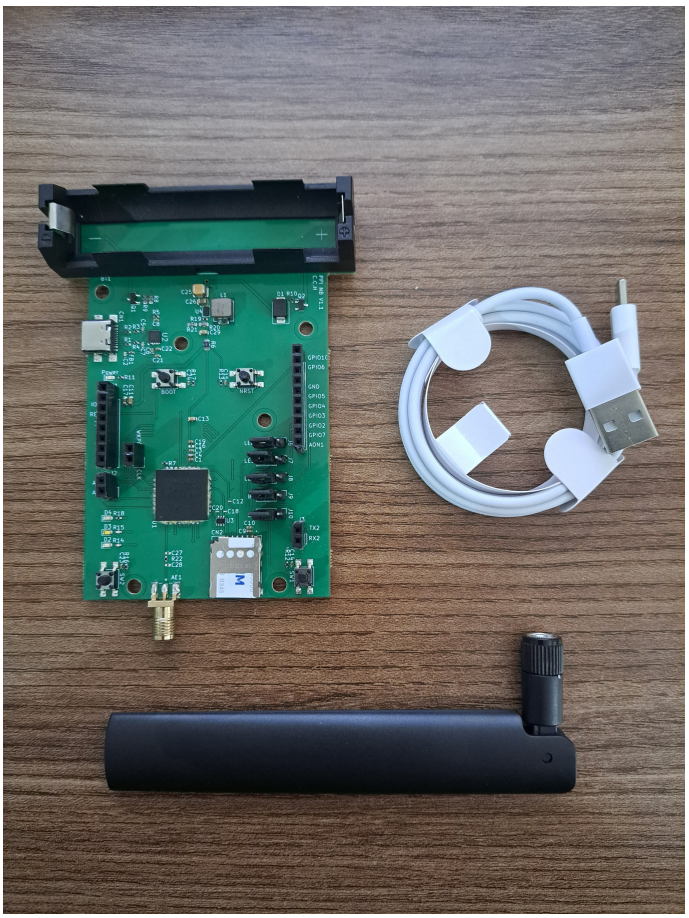
- Replace the “Application Example” field to the name of the respective application. The table bellow shows command examples to trigger a compilation process.

Application	**Command**
MQTT_Example	make -j4 gccall TARGET=qcx212_0h00 V=0 PROJECT=MQTT_Example
Driver_Example	make -j4 gccall TARGET=qcx212_0h00 V=0 PROJECT=Driver_Example
Sleep_Example	make -j4 gccall TARGET=qcx212_0h00 V=0 PROJECT=Sleep_Example

5 Hardware

For the ITTCHIP example, you are gonna need our educational development kit PCB, as in the image bellow. You also need an antenna, a c-type USB, and a NB-IoT enabled SIM card. In case you don’t have it, the PCB project is open in the documentation and free to replicate.

Figure 13: ITTCHIP hardware example



6 Running your first example

The HTNB32L-XXX SDK provide some basic firmware examples that can be used as start point of new OpenCPU projects. This section describes the first steps with the ITTCHIP Example.

6.1 ITTCHIP example

The ITTCHIP Example is an applications based on the MQTT protocol that transmits periodic SiP internal temperature measurements and blinks led. It uses a software, developed in Python, to publish and subscribe commands through a public MQTT broker.

6.2 MQTT broker and details

Since more people will be using this, the publishes and subscriptions of the same topics can be mixed up. We strongly suggest you create your own broker in HiveMQ or other MQTT platform, as other people will be using this one.

- Broker address: broker.hivemq.com (Please check HiveMQ) website for more details)

- Password: HTmicron
- TCP port: 1883

To change to your account in the application:

Figure 14: MQTT settings

6.3 Flashing

The HTNB32L-XXX is uploaded through UART. It uses an internal bootloader to receive the new firmware through UART1 RX pin.
Considering the same circuit for GPIO1 (BOOT pin) and NRST, in the figure bellow. Press and hold the BOOT button, then press the NRST button. Then release the NRST button and release the BOOT button.

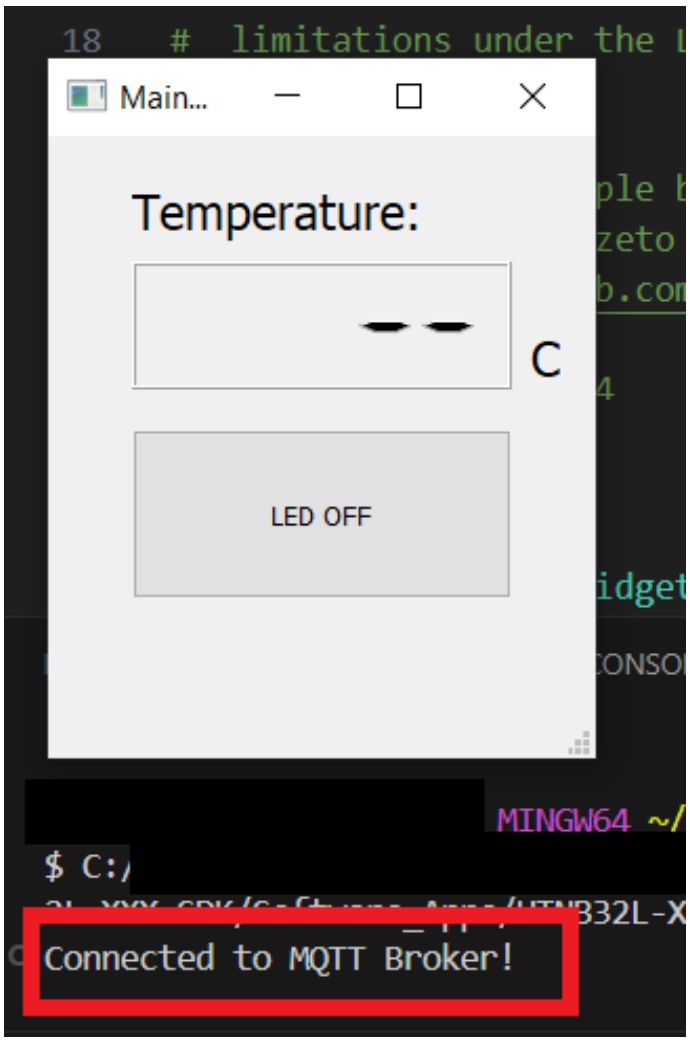
Figure 15: BOOT and NRST pins.

- Open the HTNB32L-XXX-SDK directory in your VSCODE editor
- Press “CTRL+SHIFT+B” to open the task list.
- Select “Run - <Application Example>” to start uploading the new firmware.
- If needed, select the development kit COM port.

6.4 How to use

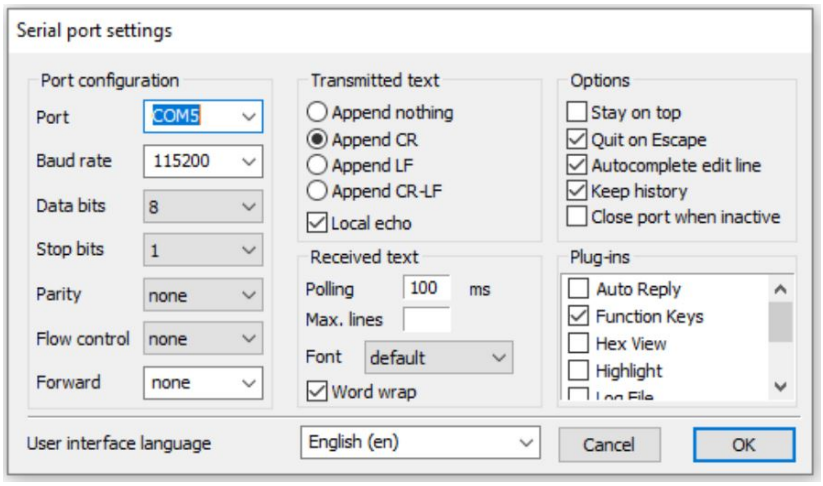
Open the HTNB32L-XXX-ITTCHIP-Demo-SW software, available on the Software_Apps directory. For that, open "ittapp.py" and "front.py" in VSCode. On VSCode, click to run the code on the top right corner. The software will appear as the image bellow and you can see if its connected to the server in the terminal.

Figure 16: ITT software.



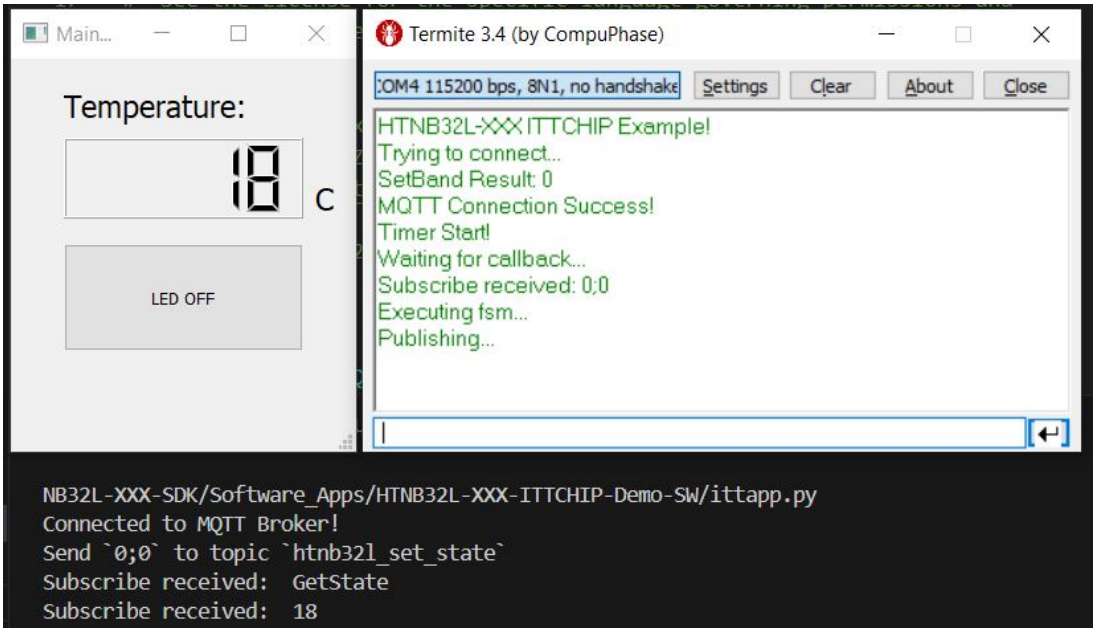
Since you already flash the firmware in the development kit, just click the NRST button to start the code. You can follow the firmware logs using a serial terminal. Here are the configurations used in the Termite terminal:

Figure 17: Termite configuration.



If its the first time you are connecting with this kit and SIM card, it will probably take a few minutes. When connected, a LED in the dev kit will start blinking. When connected, the device will transmit an internal temperature measurement and keep periodically transmiting it every 2 minutes.

Figure 18: Application.



You can also turn a LED on and off with the user button on the device or the button in the software application.

Figure 19: Application with LED on.

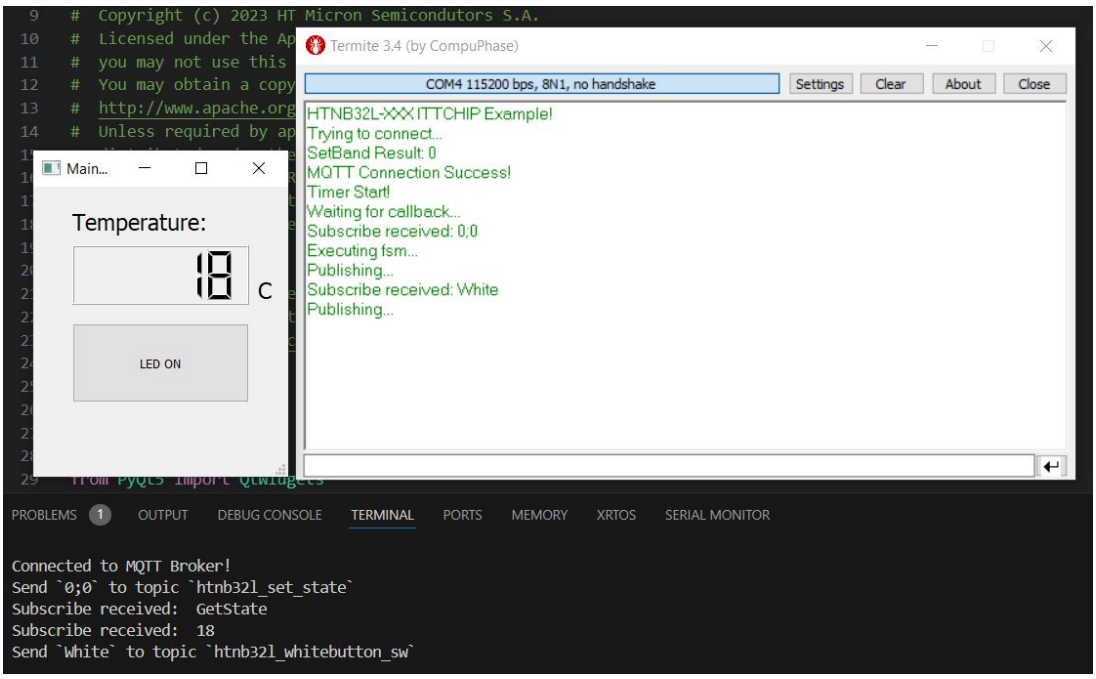
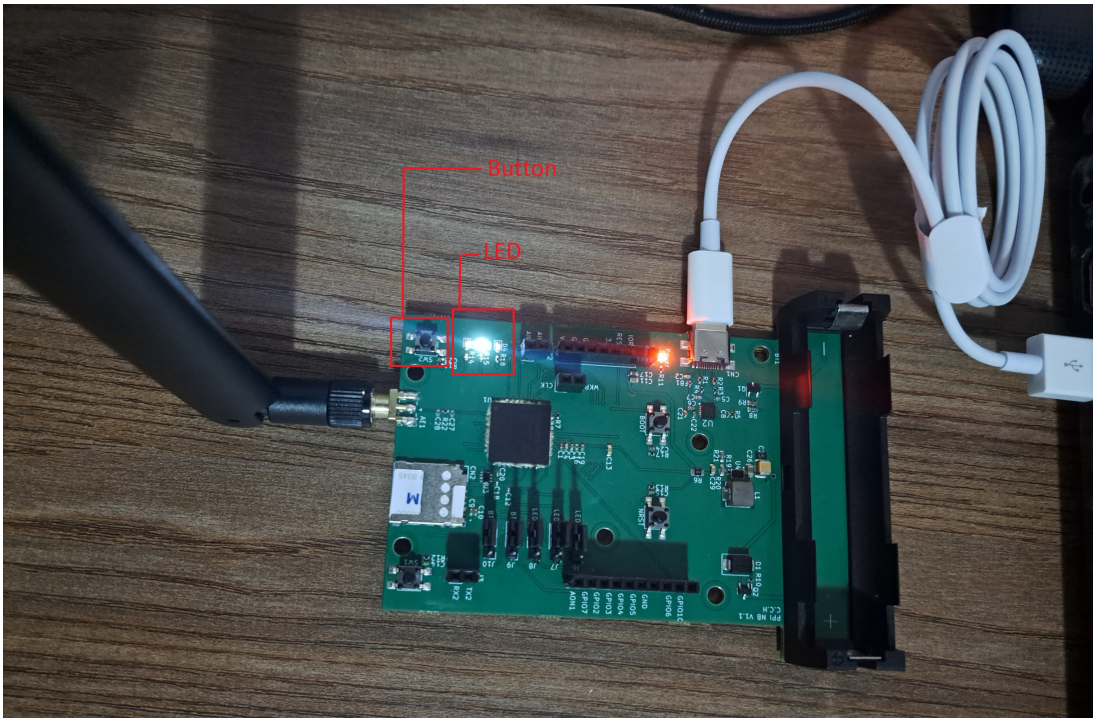


Figure 20: LED on device.



7 Document Version

Version	Date	Name
1.0	08/20/2024	Matheus Zorzeto
1.1	09/03/2024	Matheus Zorzeto