Software Development Environments for Microcontrollers

An **IDE (Integrated Development Environment)** is a computer program that brings together features and tools to support software development. In the embedded system context, the IDEs have a friendly interface and provide quick results for the development of applications focused on microcontrollers. The basic tools that compose an IDE are discussed below:

Image 01: Examples of IDE. Source: *The author.*

- **Code Editor:** Designed for writing and editing source code**.**
- **Compiler:** they convert source code, written in a programming language, into binary files that can be executed by a microcontroller.
- **Debugger:** Debuggers are used during testing and can help developers find errors during code execution, one of the most used features is the line-by-line code execution, which enables the developer to view the result of expressions, functions and even values stored in a certain memory location on the device in use.
- **Compilation Automation Tool:** Automates the task of compiling code and generates the binary files needed to update the device.
- **Code write tool:** Together with the compilation tool, it updates the device's firmware.
- **Code versioning tool:** Displays code changes, compares it to previous versions, automates commit, push, and pull tasks [1].

The code editor of modern IDEs have a number of features that assist the developer during the writing of code, among these features are: different font coloring for variables, methods and functions of the programming language, auto-complete functionality, syntax error indication and others. These features make the developer much more productive during development. Some examples of IDEs for microcontroller programming are: Wise-Studio, MPLAB X, Keil, STM32Cube, Visual Studio Code (using the PlatformIO extension) and others. Figure 2 shows the interface of Wise Studio, used to develop code for some microcontrollers from the manufacturer STMicroelectronics.

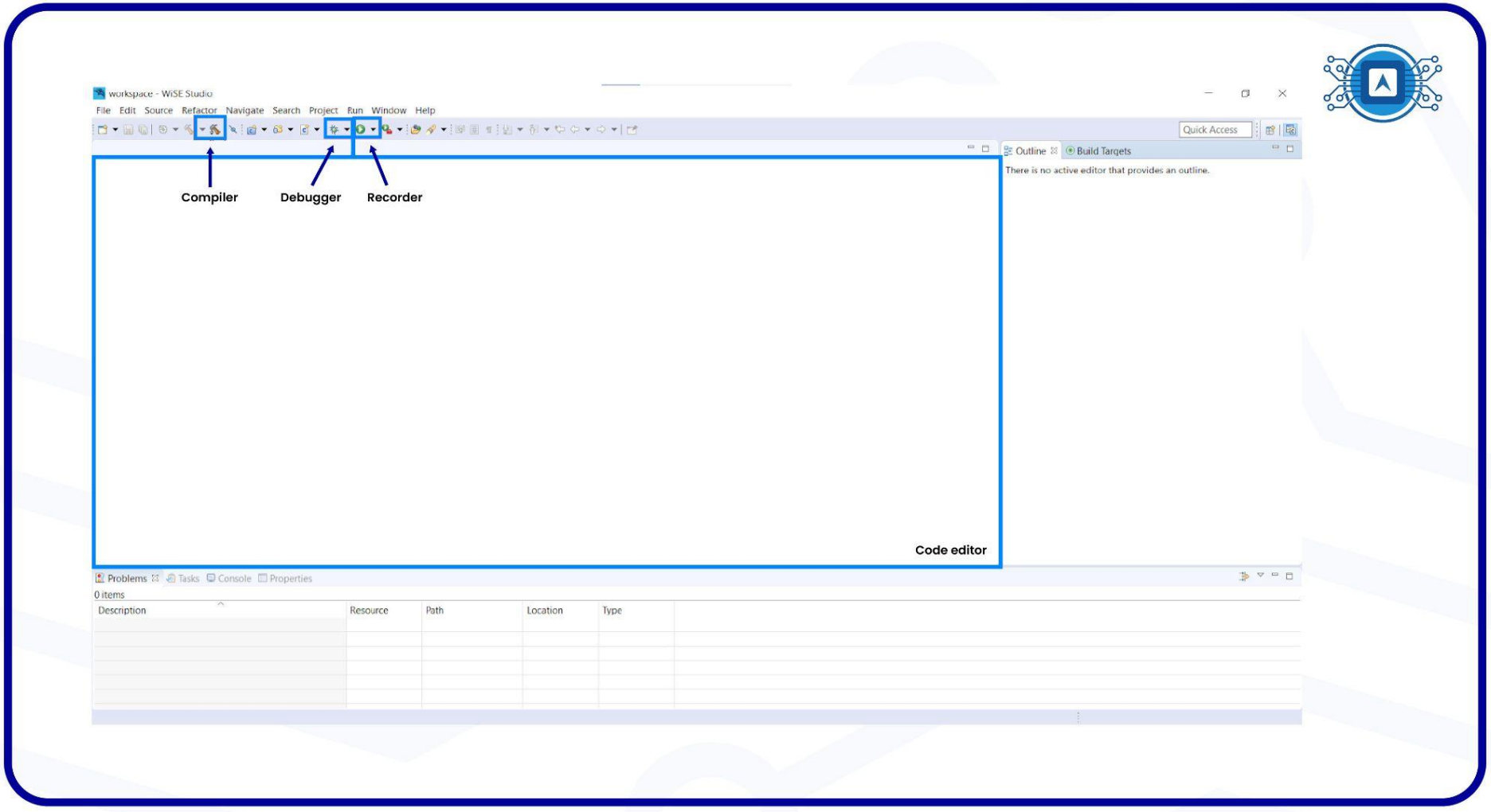

Image 02: Wise Studio IDE Home Screen. Source: *the author*.

Writing the Firmware

The term “firmware” was created by Ascher Opler in 1967. At that time, the name belonged to a specialized part of memory that housed a small piece of code to implement a set of computer instructions. Today, the word “firmware” refers to any computer program that is tightly coupled to the hardware [2].

The firmware of a device can be reprogrammed through an ISP (In-system programming) or through an IAP (In-application programming). Devices that have an ISP are reprogrammed through serial interfaces using mostly proprietary protocols and need a secondary device (programmer) to do the interface of computers with the device. The ISPs besides allowing the recording of the devices also allow the execution of integrity tests and code debugging in runtime. Some examples of recorders are: st-link, pickit and avrisp. The Devices that have IAPs can be reprogrammed through a UART and use the RS232 protocol for communication with computers. This is possible because the IAPs, also known as bootloaders, are small firmwares, previously recorded in the device, responsible for reading the user's program and storing a specific section of flash memory. It is also responsible for executing the user's program. When starting the device the first program to be executed is the bootloader. Therefore, only a USB-to-serial converter is needed to reprogram the device and an application on the computer to send program data.

Down below we will exemplify how to configure a writer and a USB-to-serial converter to write firmware to a microcontroller. To do this, we will use the RF-Flasher application from STMicroelectronics that does the writing to the BlueNRG microcontrollers.

RF-Flasher

Installing RF-Flasher

Not all IDEs come with a firmware writer, for some applications it is necessary to get parallel and simple writers, such as the RF-Flasher Utility, a computer application provided for free, which aims to write the firmware inside the flash memory of the microcontroller. It also allows the BlueNRG-1, BlueNRG-2, BlueNRG-LP and BlueNRG-LPS Bluetooth low power flash to be written, erased and read by your terminal. However, download it from this website to start the program installation process. After the installation you will see the screenshot below.

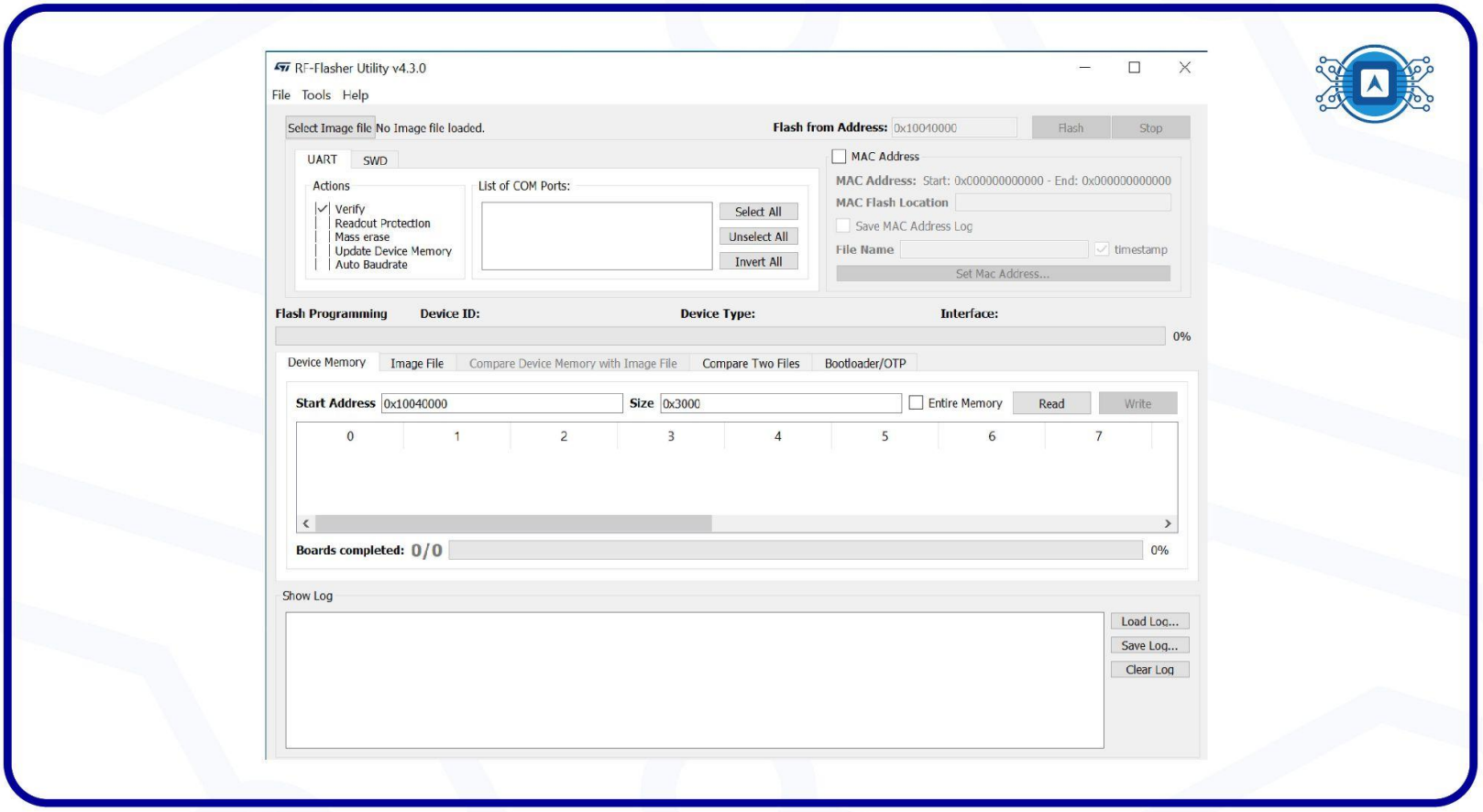

Image 03: RF-Flasher console. Source: *the author*.

- You can use the following operations in the toolbar:
  - ➔ Load an existing .bin or .hex file (extended intel) (in File: open file option);
  - ➔ Save the current memory image to a .bin file (in File: option save file as);
  - ➔ Set the ST link frequency (in Tools: Settings options);

➔ Enable or disable the creation of log files in UART/SWD mode (in Tools: Settings. options);
➔ Check the Flash content (in Tools: check the Flash content option).

Setting up FTDI in RF-Flasher

Installing the FTDI Module

The acronym FTDI refers to the Scottish company Future Technology Devices International. Their modules have become very popular worldwide, being called just “FTDI” for short. The FTDI module is based on the FT232RL chip that works as a USB to TTL serial converter. Therefore easy to use in projects involving microcontrollers that have a serial communication port (UART). When we want to connect the FTDI converter to the computer's USB port, usually the installation of the drivers is done automatically, with the computer recognizing the device. However, to get more technical information about the FTDI Module we can check the datasheet of the board.

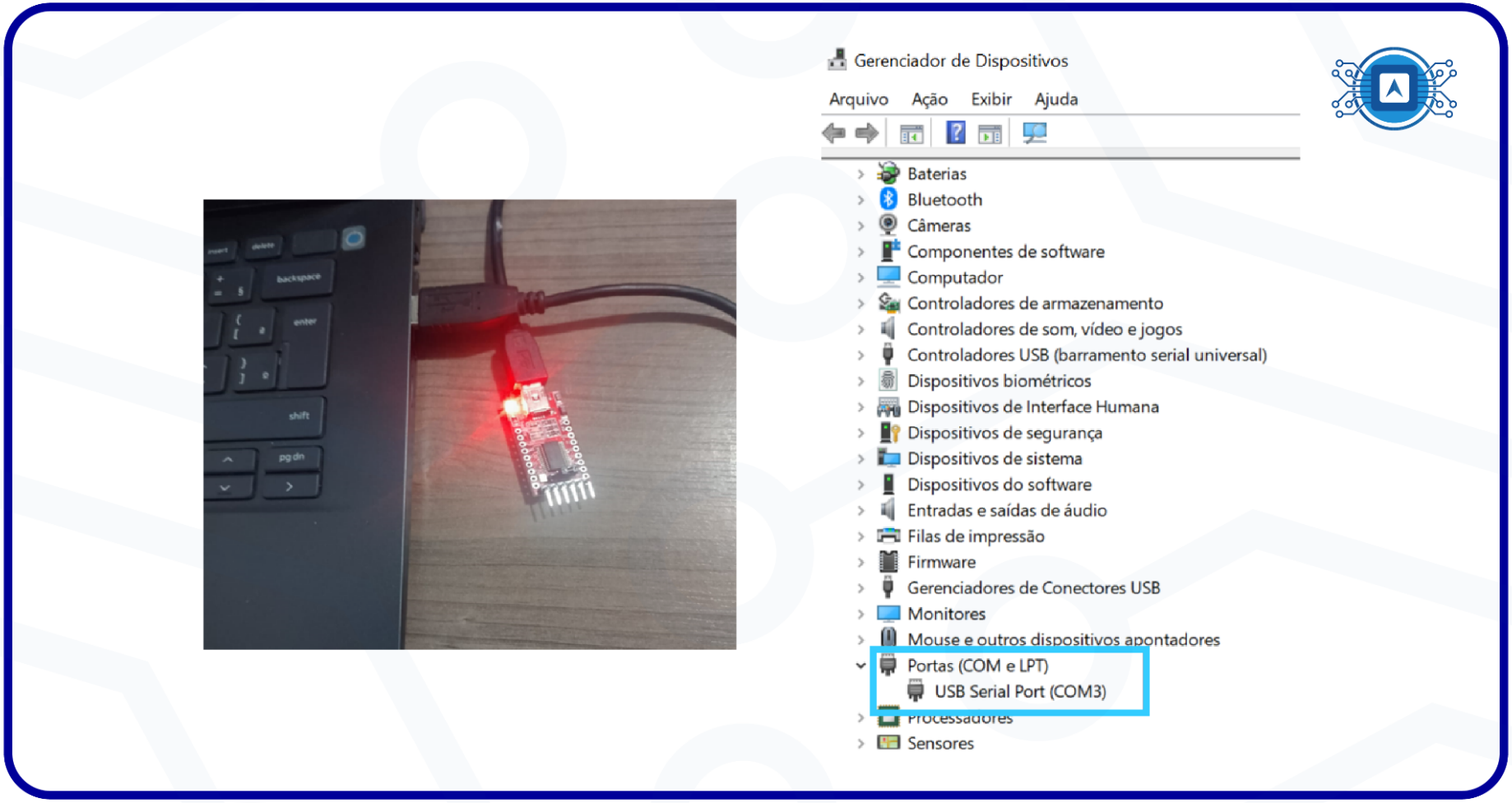

Image 04: FTDI module Serial Port. Source: *The author*.

We can observe that the COM3 serial port was detected. Iif the installation of the drivers did not occur automatically, download the drivers from this **website**, then put the device back in the USB port to continue the procedure. When we open the RF-Flasher program in conjunction with the FTDI USB, the software will recognize the COM3 serial port automatically, as seen in image 05.

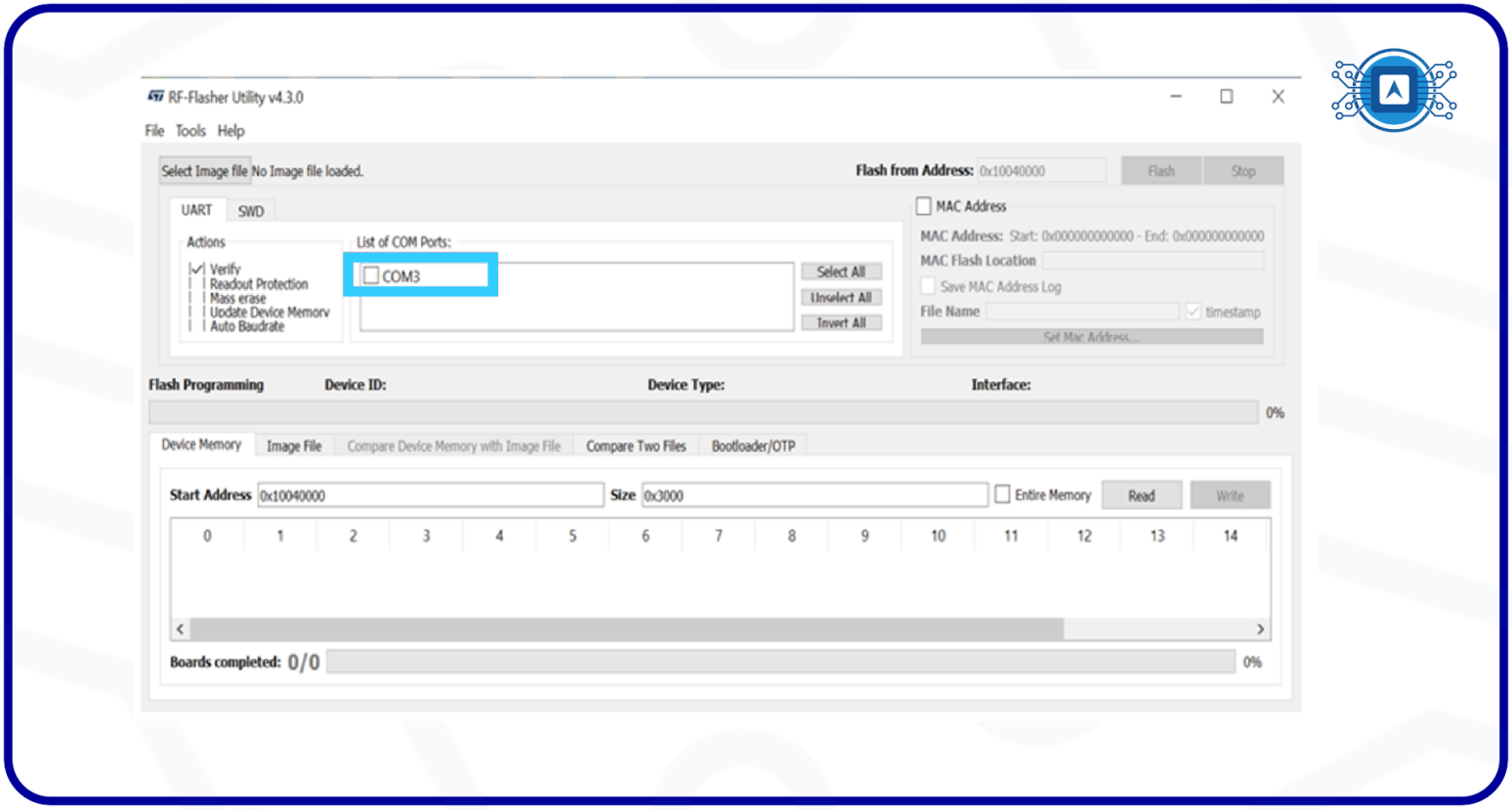

Image 05: CM03 port found. Source: *The author*.

RF-Flasher writer software configuration:

For the test, we will use the HTmicron **HTLRBL32L-LoRa/BLE** microcontroller whose physical connection configuration to the FTDI module can be seen in image 06. We note that the RX of the FTDI will be connected to the TX of the microcontroller, then TX to RX, VCC to VCC and GND to GND.

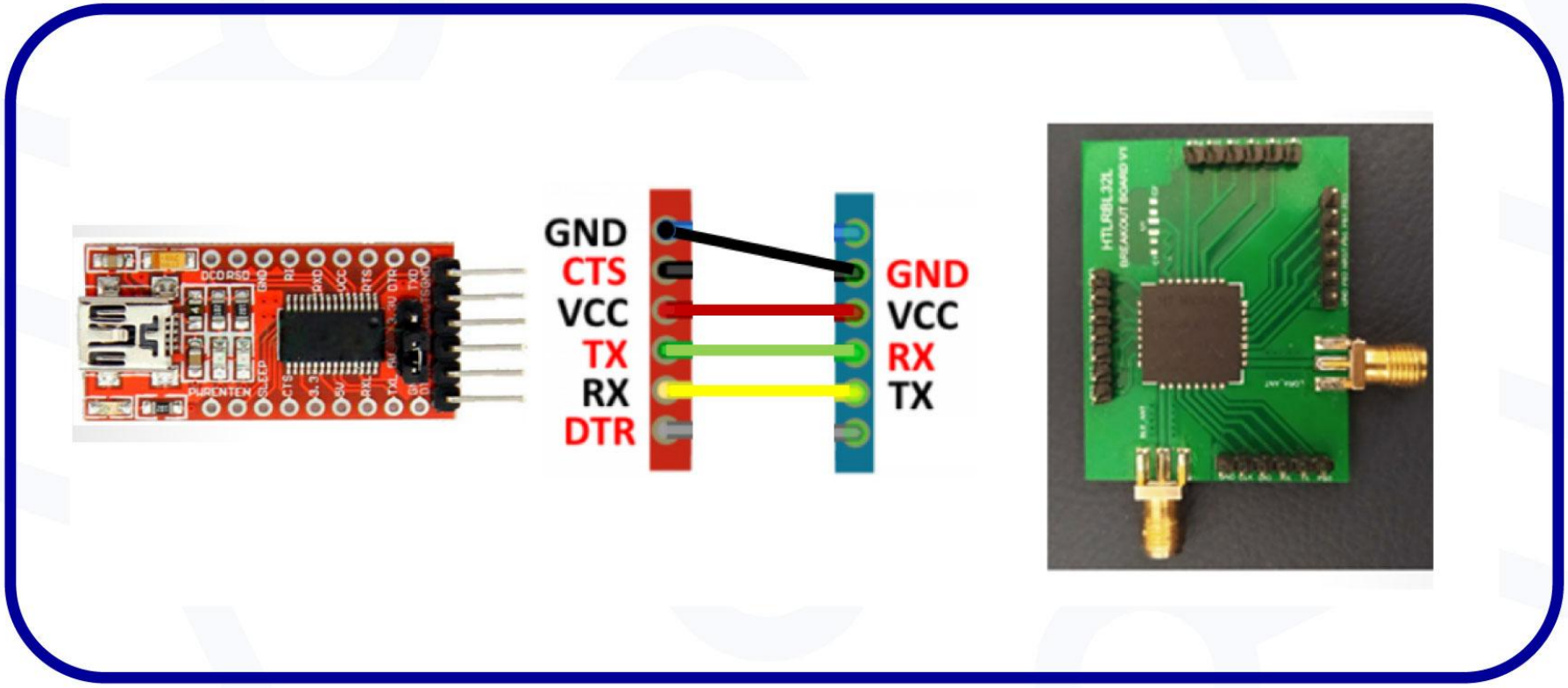

Image 06: Connection settings of the HTLRBL32L microcontroller to FTDI. Source: *The author*.

It is mandatory to put the output voltage selection jumper of the FTDI module in the correct position before powering up the device. The FTDI board can provide either 3.3V or 5V to power up the module. However, the HTLRBL32L operates under 3.3V, which requires the user to select the FTDI output voltage accordingly.

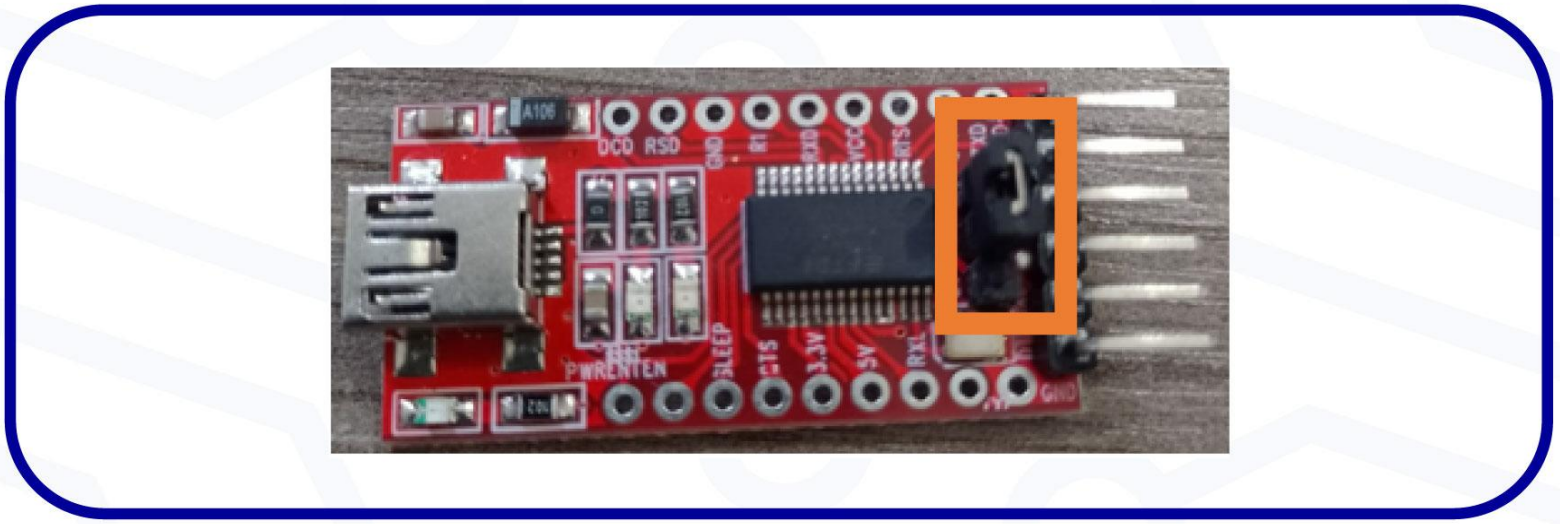

Image 07: FDTI with 3.3V. Source: *The author*.

After this, we set up the microcontroller to go into record mode. This configuration occurs in a physical way using some wires. Then, we should connect the **GPIO-10** of the **HTLRBL32L** internal microcontroller to the **3.3V** to leave the pin at high level. Next, we should connect the reset that is marked on the board by the acronym **NRST** to the GND pin. We can observe the complete assembly in image 08 with the use of a breakout board for the HTLRBL32L.

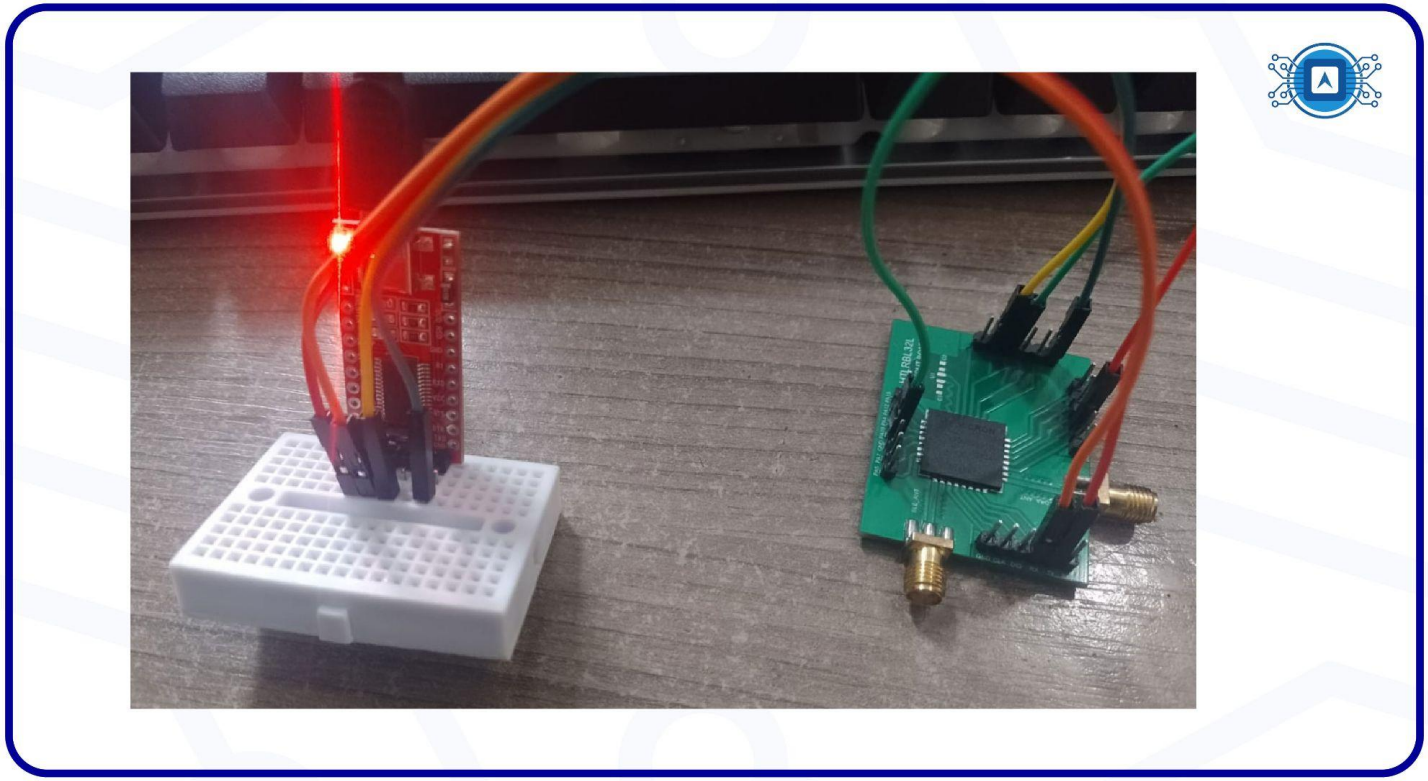

Image 08: Physical setup of the microcontroller with the FTDI. Source: *The author*.

After the setup process, we connect the FTDI cable to the computer for recognition of the **COM3** serial port in the **ST-Flasher** software. Then, when we select the COM3 port, the application provides the user with the instruction dialog for connection between the microcontroller and the FTDI, in which we have to select the value of the Baud Rate according to the board's settings. In our case, we selected the default value 115200. As shown in image 09.

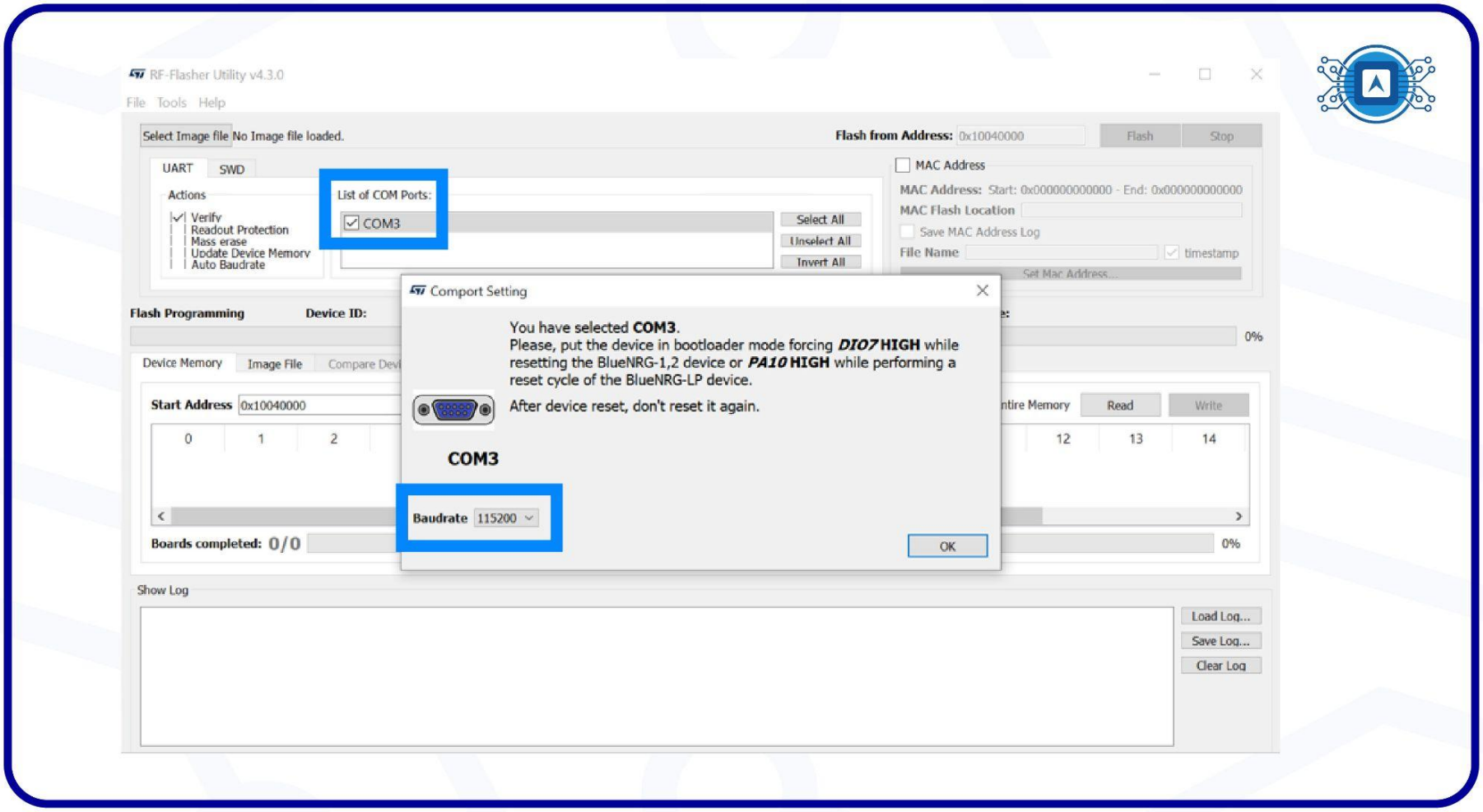

Image 09: Connection Settings. Source: *The author*.

To start recognizing the microcontroller with the **firmware** writer, we will disconnect the **NRST** from the **GND** pin on the board and click on the **Read** button of the RF-Flasher. The software will automatically load information on the board. As seen in image 10.

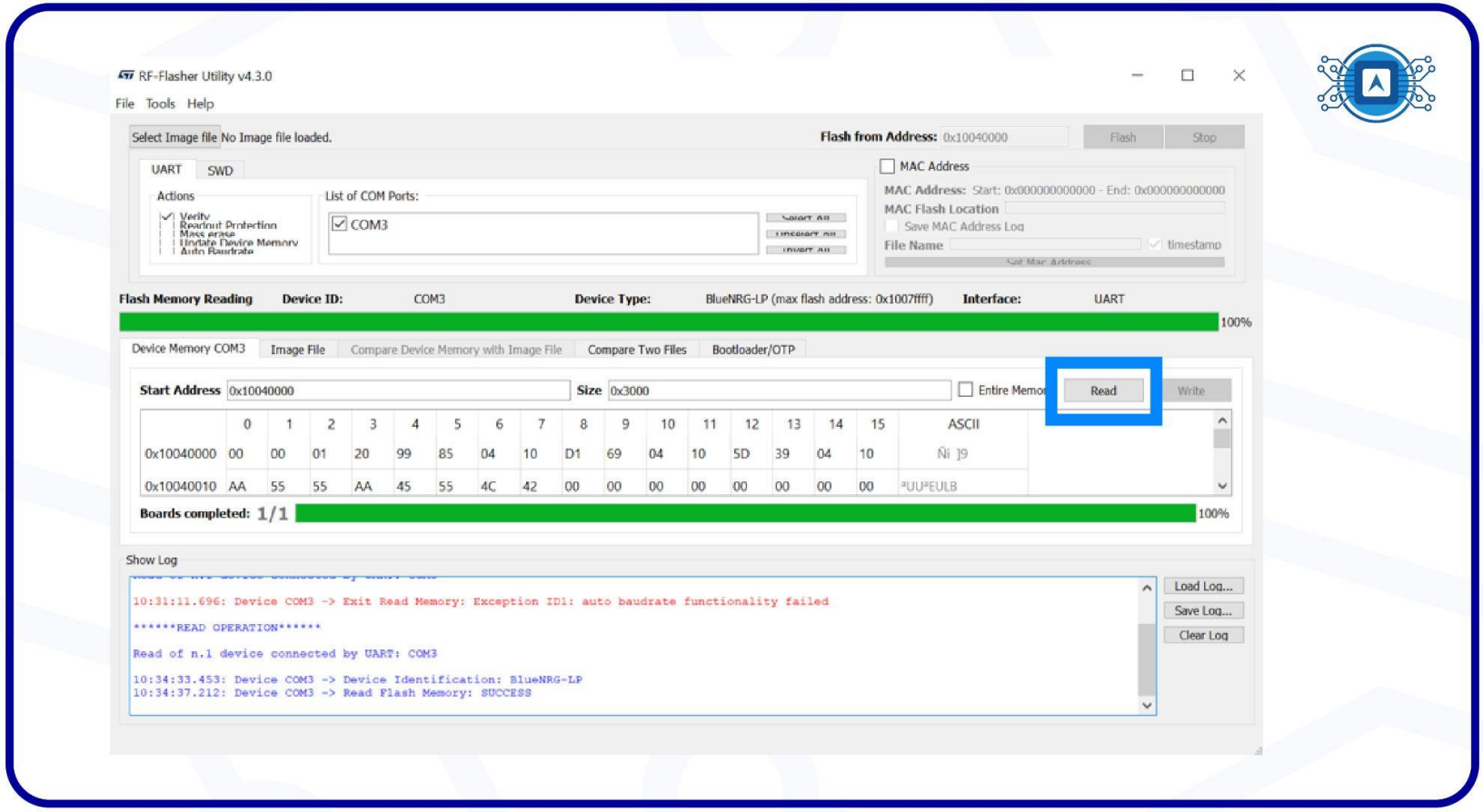

Image 10: Initial recognition of the microcontroller in the RF-Flasher. Source: *The author*.

Finally, we can see the serial **UART COM3** connected to the software writer and the **HTLRBL32L** microcontroller has also been identified through the “**BlueNRG-LP**” directive. The software announces that the connection was successful.

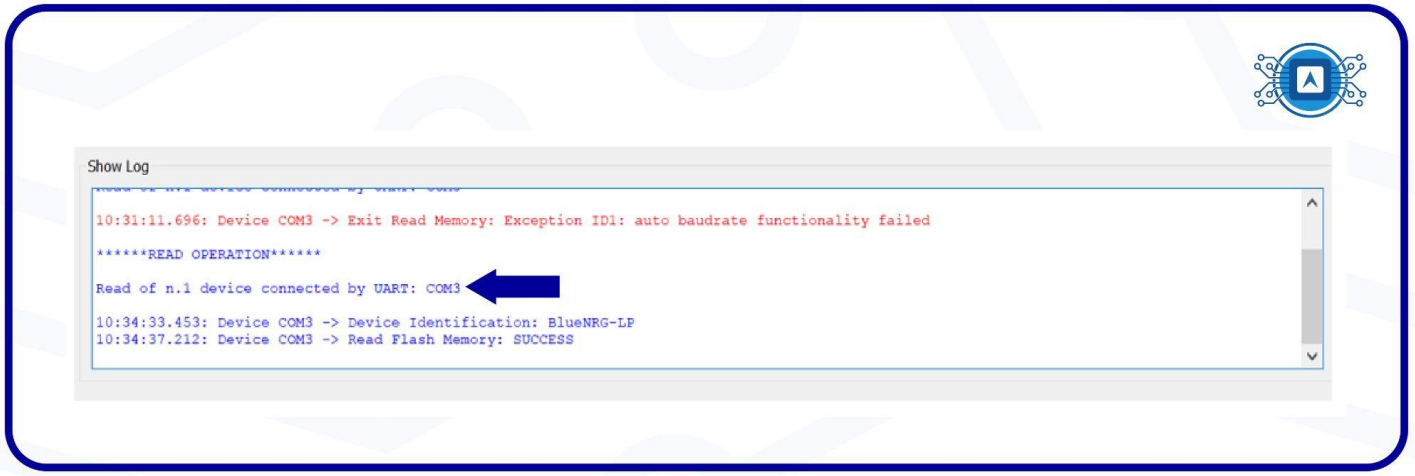

Image 11: Log from ST-Flasher. Source: *The author*.

Configuring ST-LINK in RF-Flasher

The **ST-LINK** is an open source toolkit for programming and debugging STM32 devices and boards developed by STMicroelectronics. It can support different debugging protocols, depending on the ST-LINK hardware version and firmware version. Its purpose is to perform the serial conversion from the microcontroller to the computer, as well as to transfer the written code to the microcontroller's memory using the firmware write program. Below are images of commercially available ST-LINK devices.

The blue colored device (Image 12) is the official ST-LINK/V2 manufactured by STMicroelectronics and can be purchased at any major electronics retailer. Since it is an official device it comes with its complete JTAG interface.

Image 12: ST-LINK/V2 (left) and JTAG interface (rigt).
Source: STMicroelectronics (2022) - *https://www.st.com/en/development-tools/st-link-v2.html*

Installing the ST-LINK driver

Depending on the ST-LINK version you need to install two USB drivers: one for the debugger itself and another for the serial communication terminal (ST-LINK/V2-1). The device drivers can be found at this website according to your operating system. After downloading, install the drive. You can see it in image 13.

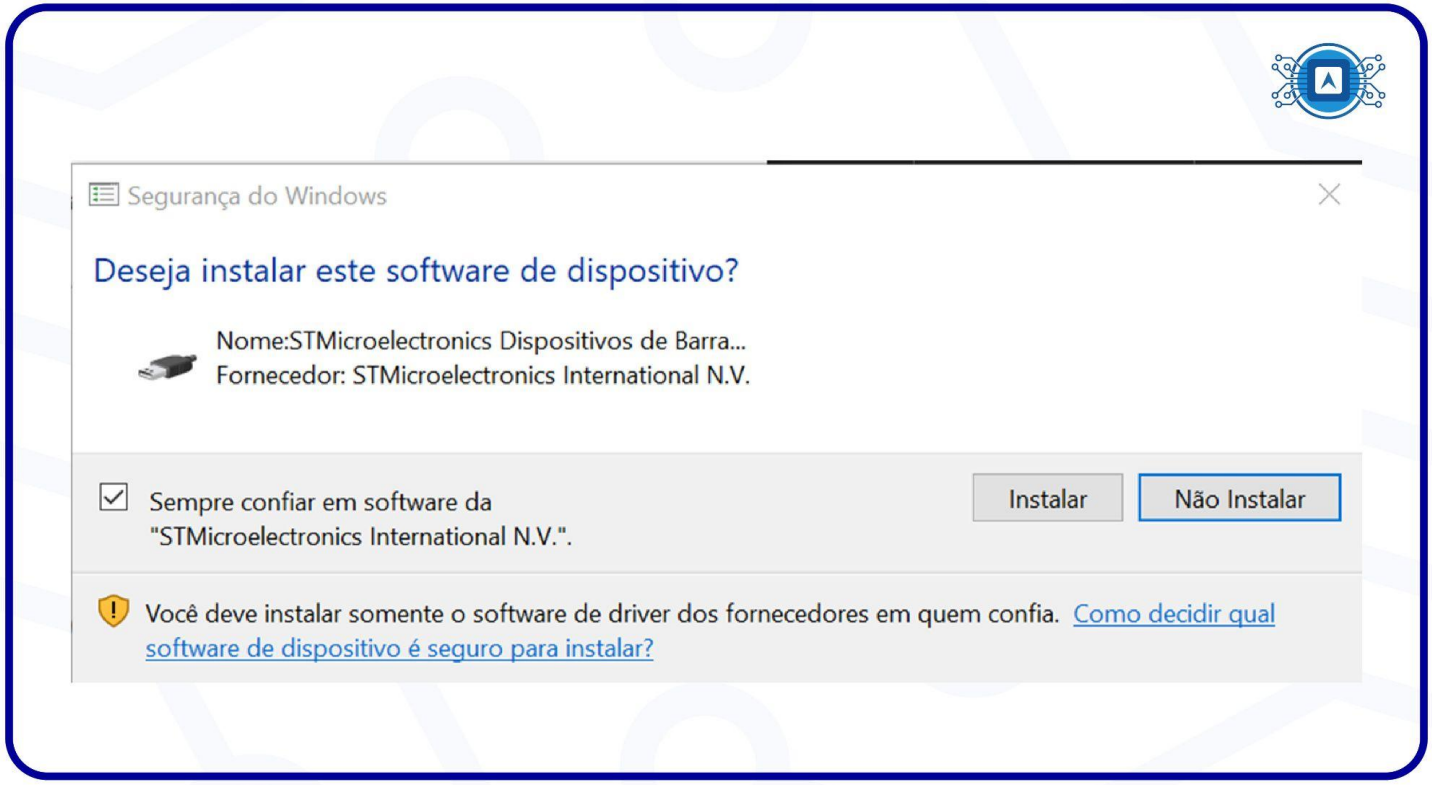

Image 13: STMicroelectronic driver installation. Source: *the author*.

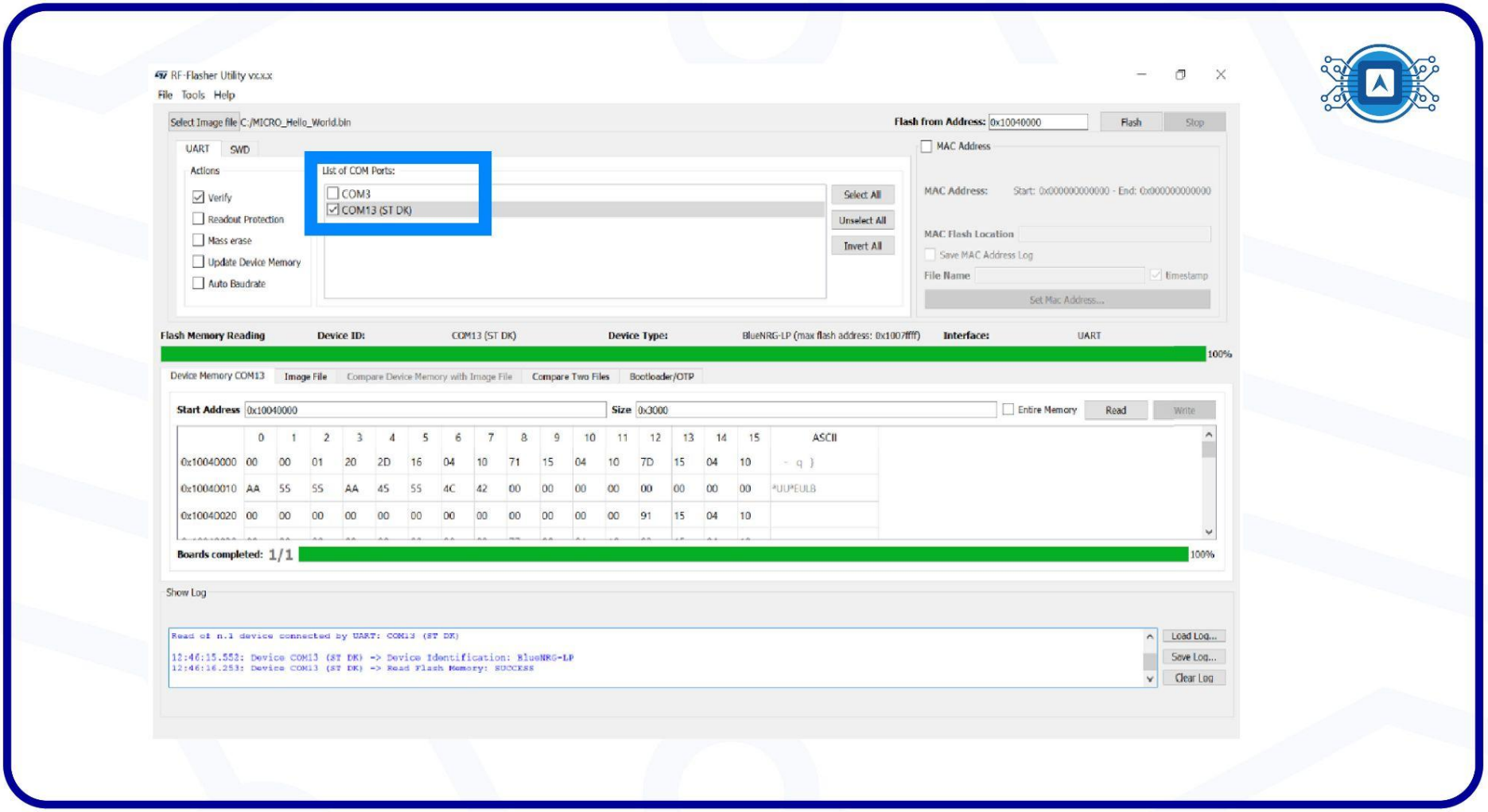

Image 14: STMicroelectronic device recognized by the computer displayed on RF Flasher console. Source: *the author.*

References

[1] VERACODE. **WHAT Is An Integrated Development Environment (IDE)?.** 2022**.** Available at: **<** https://www.veracode.com/security/integrated-development-environment **>.** Accessed on june 08th 2022.

[2] TECHSLANG. **What is Firmware?.2021.** Available at: **<** https://www.techslang.com/definition/what-is-firmware-in-computer/ **>** Accessed on june 08th 2022.

Additional reading

FTDICHIP. **DRIVERS.** Available at: **<** https://ftdichip.com/drivers/ **>** Accessed on june 08th 2022.

SMT32. **GUIDE: Connecting your debugger**. Available at: **<** https://stm32-base.org/guides/connecting-your-debugger.html **>** Accessed on june 08th 2022.