Development of code to write to the microcontroller

Currently microcontrollers are programmed in high-level languages, such as C/C++ or Java. However, in embedded *firmware* programming, the C language is widely used. The reason for this is that C is a language that talks directly to the *hardware*. While the Assembly language is closer to the *hardware*, because its sequence of commands are like machine code that generates hexadecimal instructions, the C language has structures that are mapped directly into machine instructions.

During the firmware development process, we write a sequence of instructions into the microcontroller's memory through an Integrated Development Environment (IDE) in an embedded language, such as C, compiling and linking the code modules with libraries.

1. Cloning a repository

You can review the process of cloning a repository in the **Software Development Environments for Microcontrollers** content. In this tutorial we will clone the example project "**PushButton_LoRaWAN**" available in the HT Micron GitHub repository through the branch "**git clone --single-branch --branch SDK https://github.com/htmicron/htlrbl32l.git**". As seen in the image below.

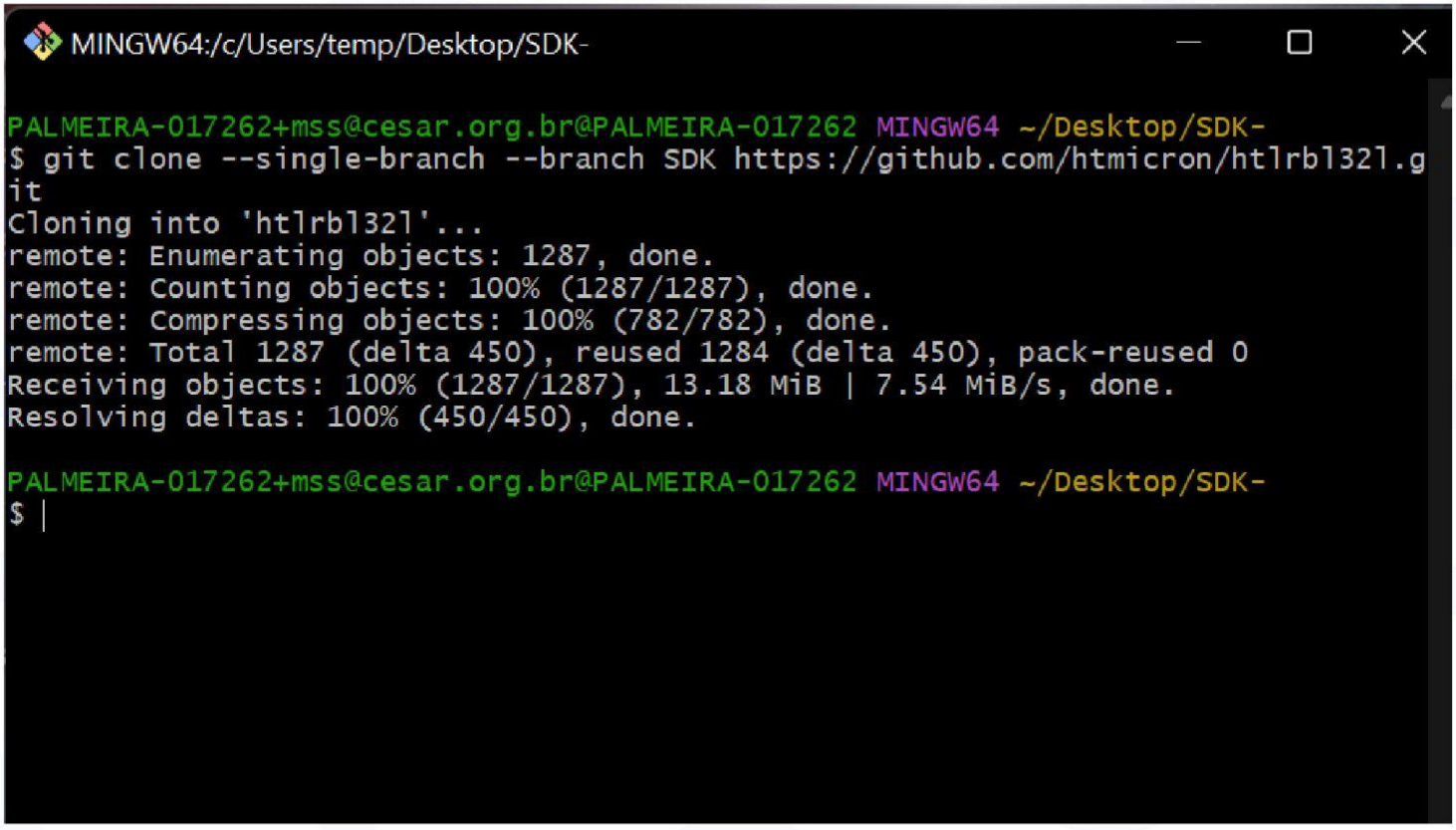

Image 1: Git Clone. Source: *The author*.

To view the cloned branch files, type the “LS” command.

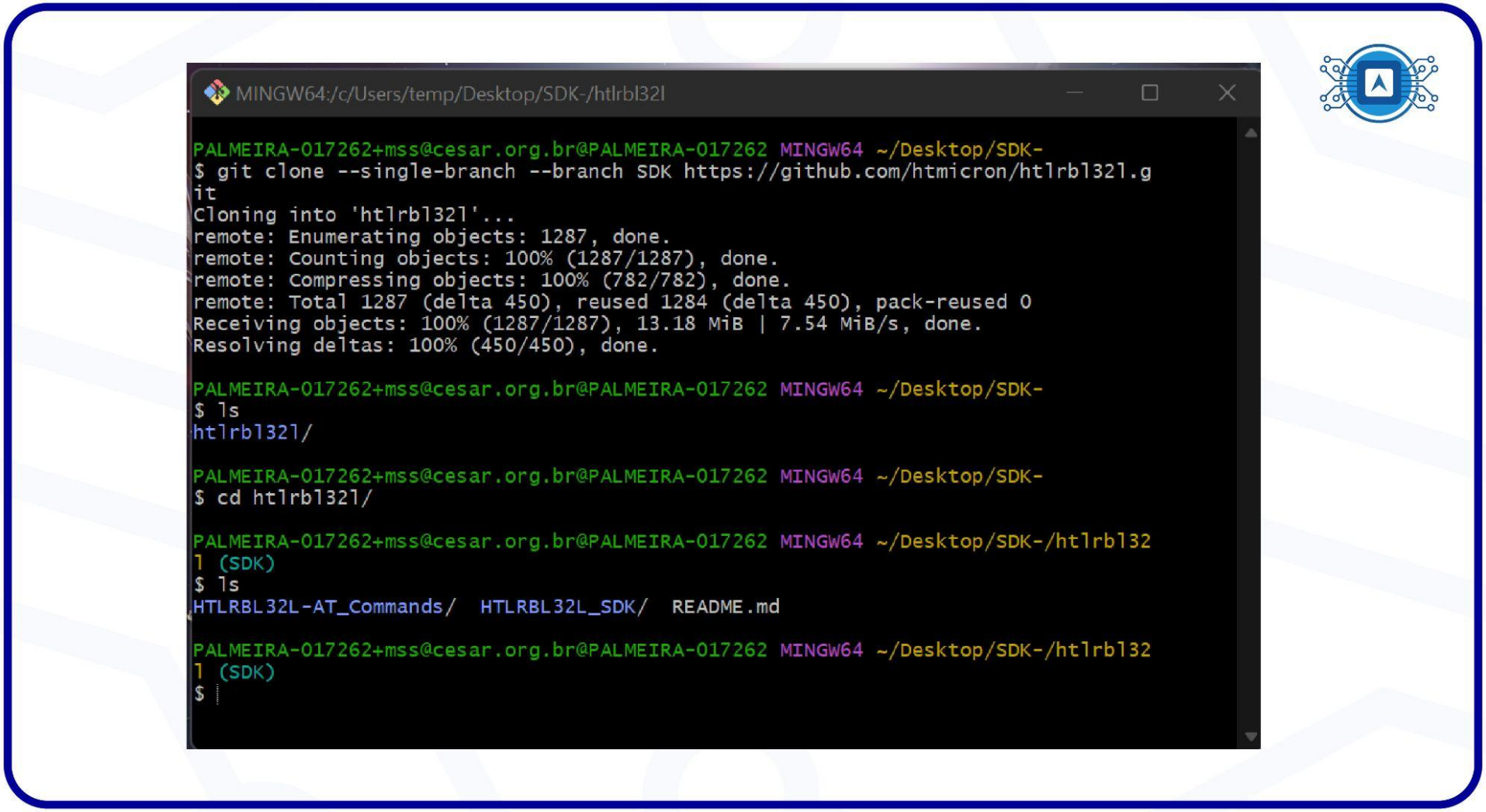

Image 2: Viewing Files in Windows PowerShell. Source: *The author*.

2. Project Code

After the code cloning process, use the Wise Studio IDE software to run it. To perform this procedure, open the "**File**" tab in Wise Studio, **click on "Open projects from file System"**. Next click "**Directory**", select the downloaded or cloned folder, and then click **Finish**.

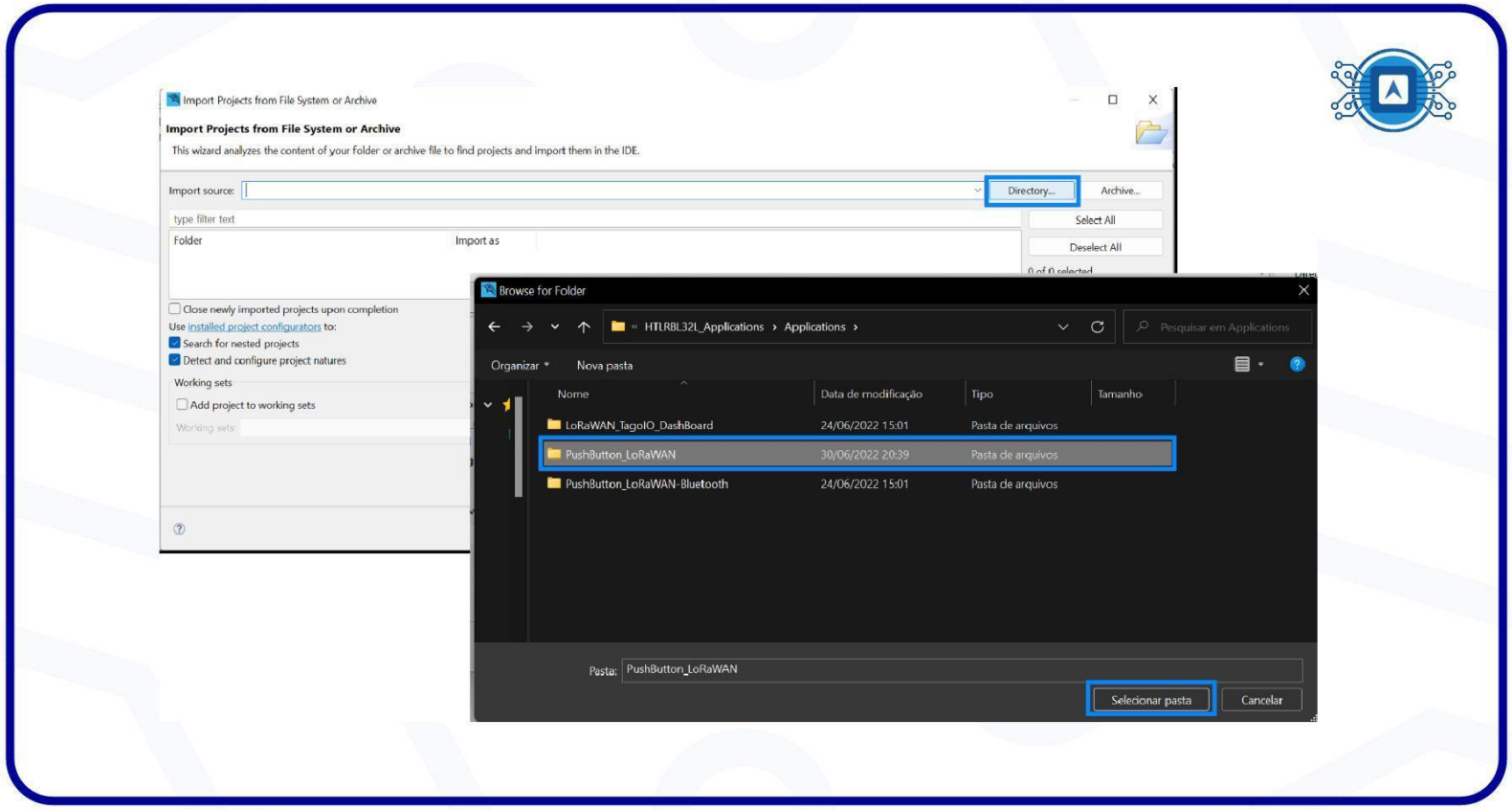

Image 3: Selection of the folder with the cloned project code. Source: *The author*.

The file will be available in **Wise Studio's** "**project explorer**" tab. Then open the main structure of the code in the "**Application**" folder.

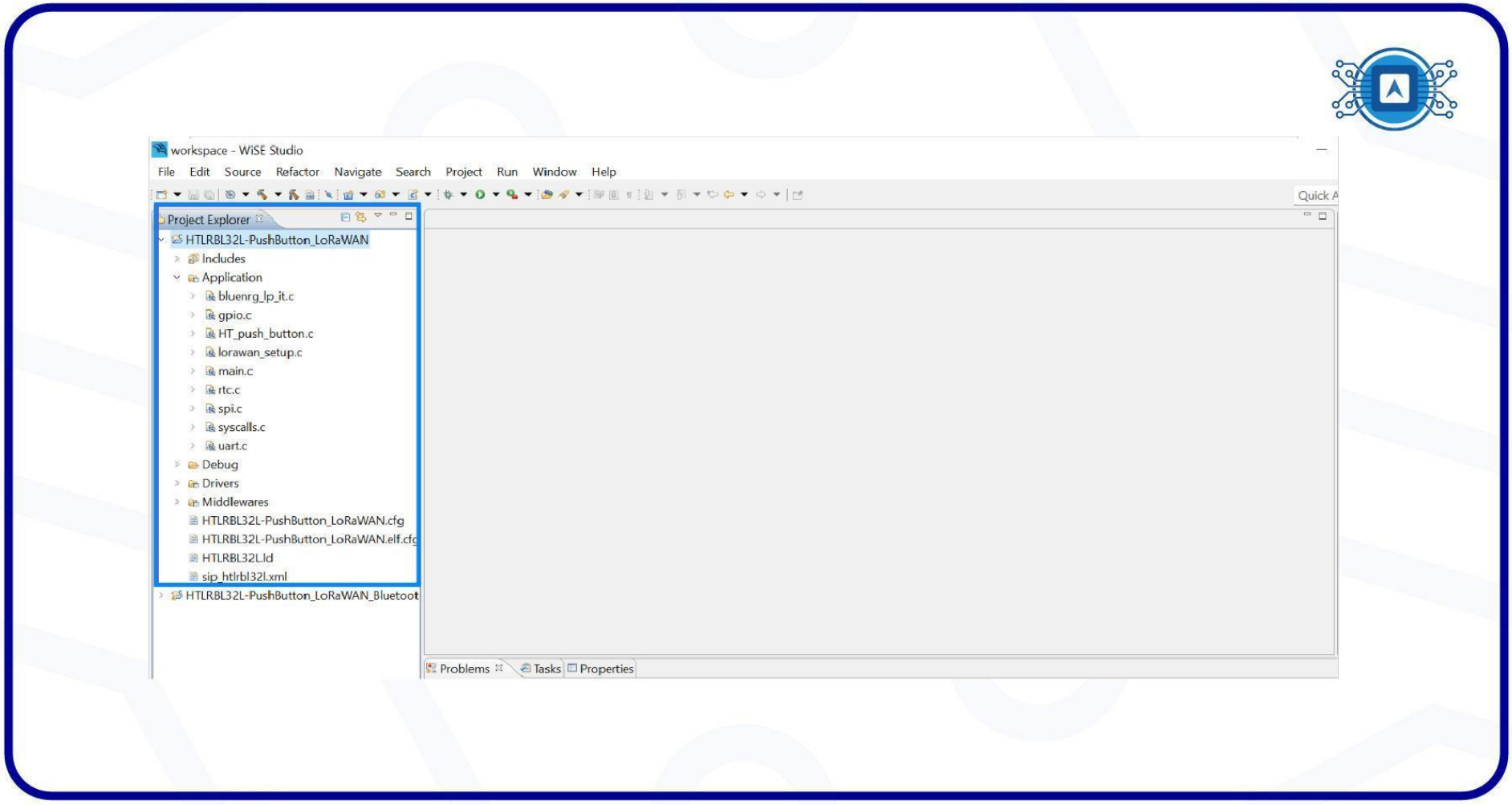

Image 4: Wise Studio Project Explorer. Source: *The author*.

2.1 How the code works.

The main functionality of the code is to activate the LoraWan communication protocol using a push button. When the push button is pressed, an interrupt “wakes up” the device and sends a payload in data frequency. This process is very useful in the IoT environment, as most devices need to save power and battery. However, we will show the main files that deal directly with the **LoraWan** protocol interrupt through the PushButton. The source code “main.c” initializes all system functions, we can see in the image below:

A) GPIO initialization
B) Initialization of the UART, it enables the serial communication that will be necessary for the exchange of information between the FTDi and the computer.
C) Initializes the LoRa Radio.

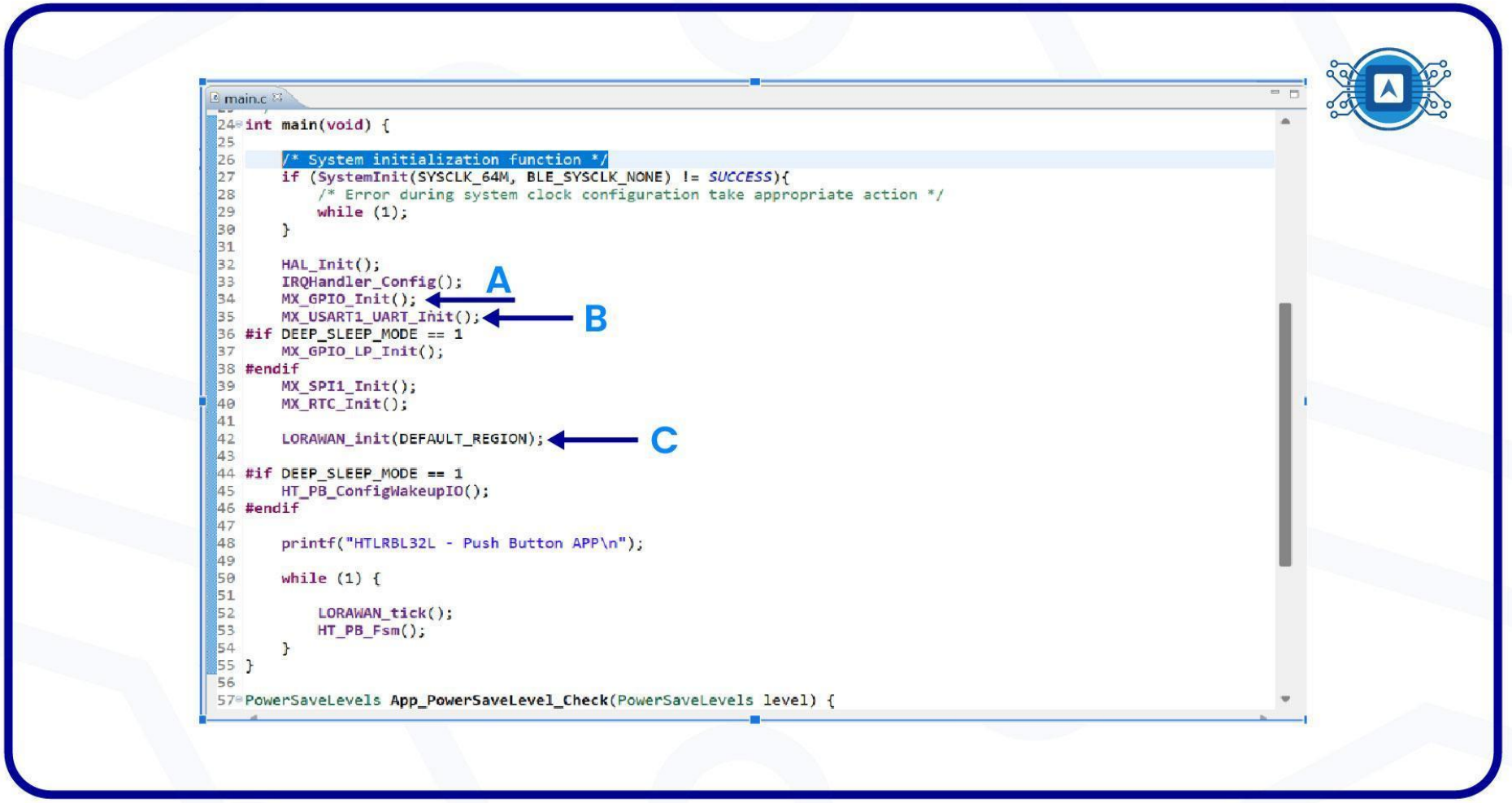

Image 5: LoraWan source code. Source: *the author*.

2.2 Channels, regions and classes

In the source code "**lorawan_setup.c**" we can configure the **channels, regions and classes** according to the needs of the microcontroller and the region where it is located. Each region of the planet or country has an operation frequency homologated for LoRa. In Brazil, the standard frequency is **902Mhz - 928Mhz**. In the image below the regions are highlighted.

Image 6 - Source code for channels and regions. Source: *The author.*

2.3 Device classes

Also in the same source code file “**lorawan_setup.c**”, we can see (image 7) where the “classes” configuration function is. The LoRa specification works with three types of **classes**: Class A, Class B and Class C. Normally, all LoRaWAN devices must implement **Class A**, while Class B and Class C are extensions of the specification for **Class A** devices.

- **Class A -** Bi-directional communication between devices such as sensors and microcontroller with the server. Fully optimized to reduce power consumption. One Rx1 window is opened when a message is sent.
- **Class B -** Multiple Rx1 receive windows open. However, at set times.
- **Class C -** Only two receiver windows are opened.

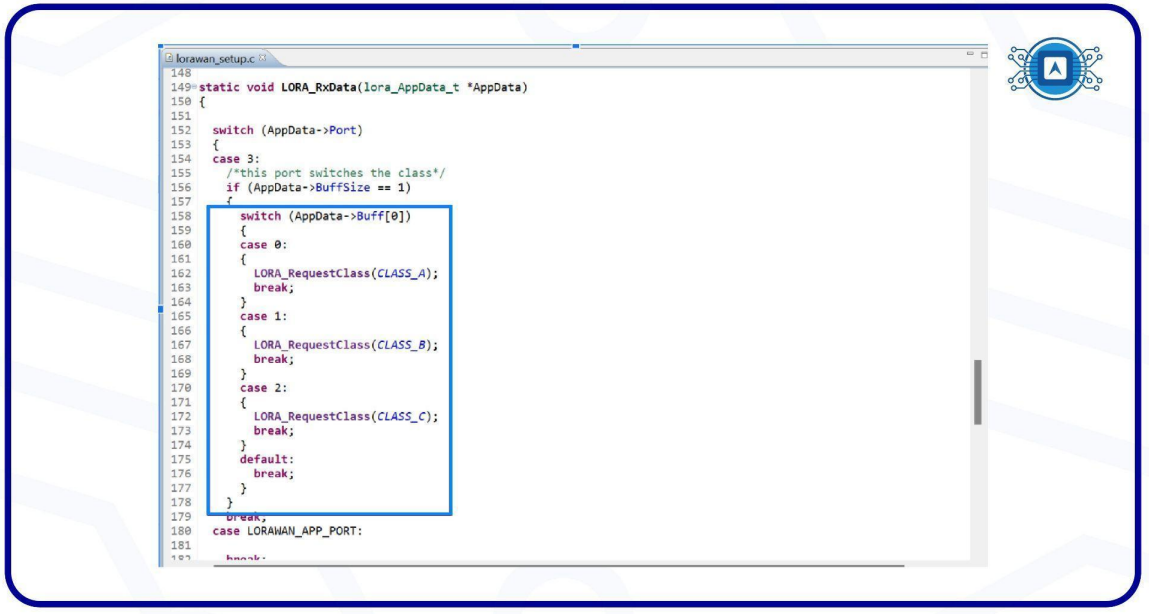

Image 7: Class source code. Source: *The author*.

2.3 Application Session Key

The library "**lorawandefines.h**", which is located inside the "includes" folder, is intended to implement the cryptographic element that supports **LoRaWAN** versions **1.0.x** and **1.1.x**. The code uses version 1.1.x.

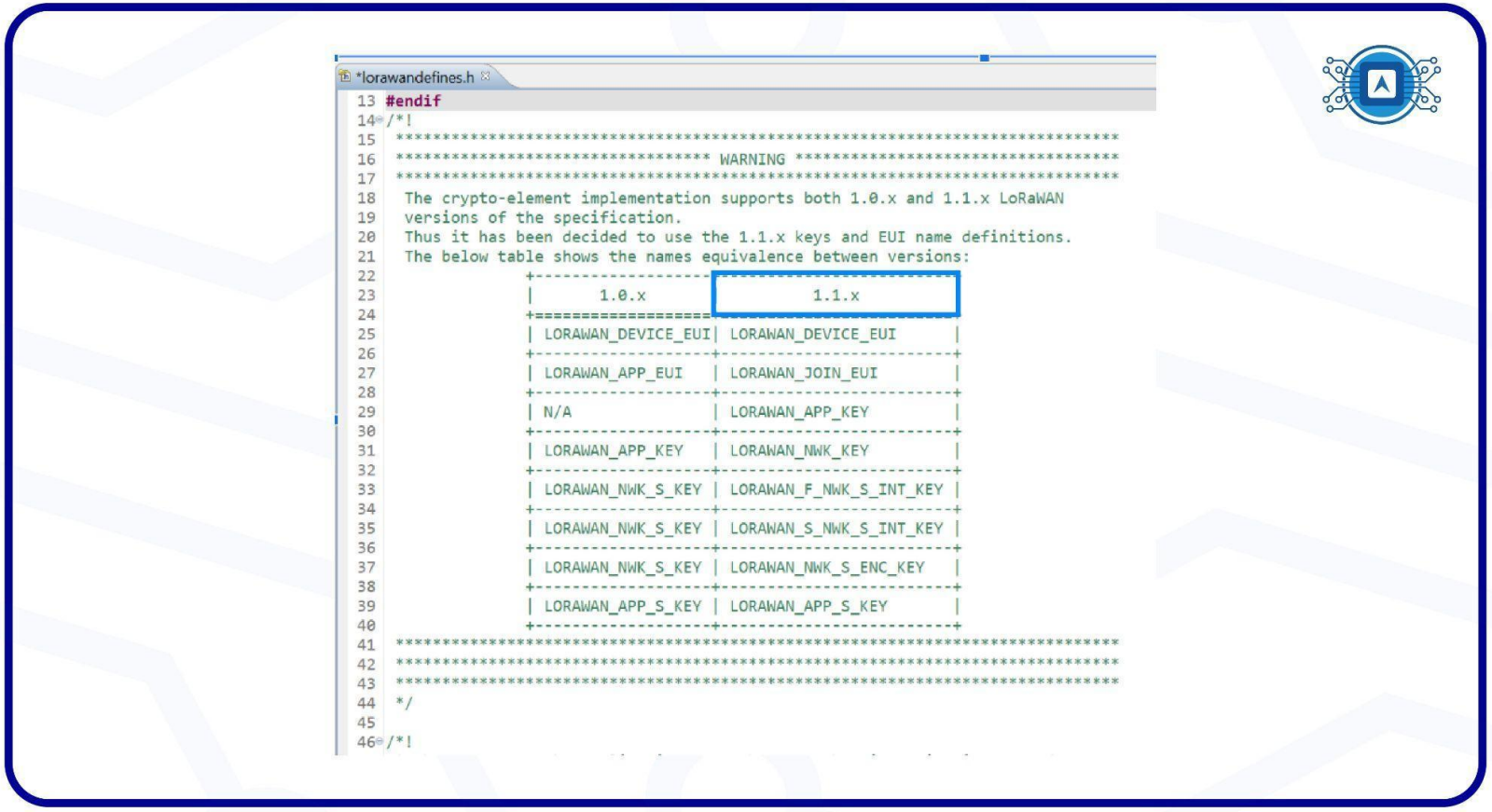

Image 8: LoraWan library source code. Source: *The author*.

The LoRaWan protocol, for security, transmits data (payload) encrypted using the 128-bit AES algorithm with a key known as the "**Application Session Key**". Take a look at image 9:

A) Represents the device (microcontroller) address to the network, works like a "MAC Address" of the device.
B) Used for encryption and decryption of a payload.
C) Normally used to interact between the device and the Network Server.

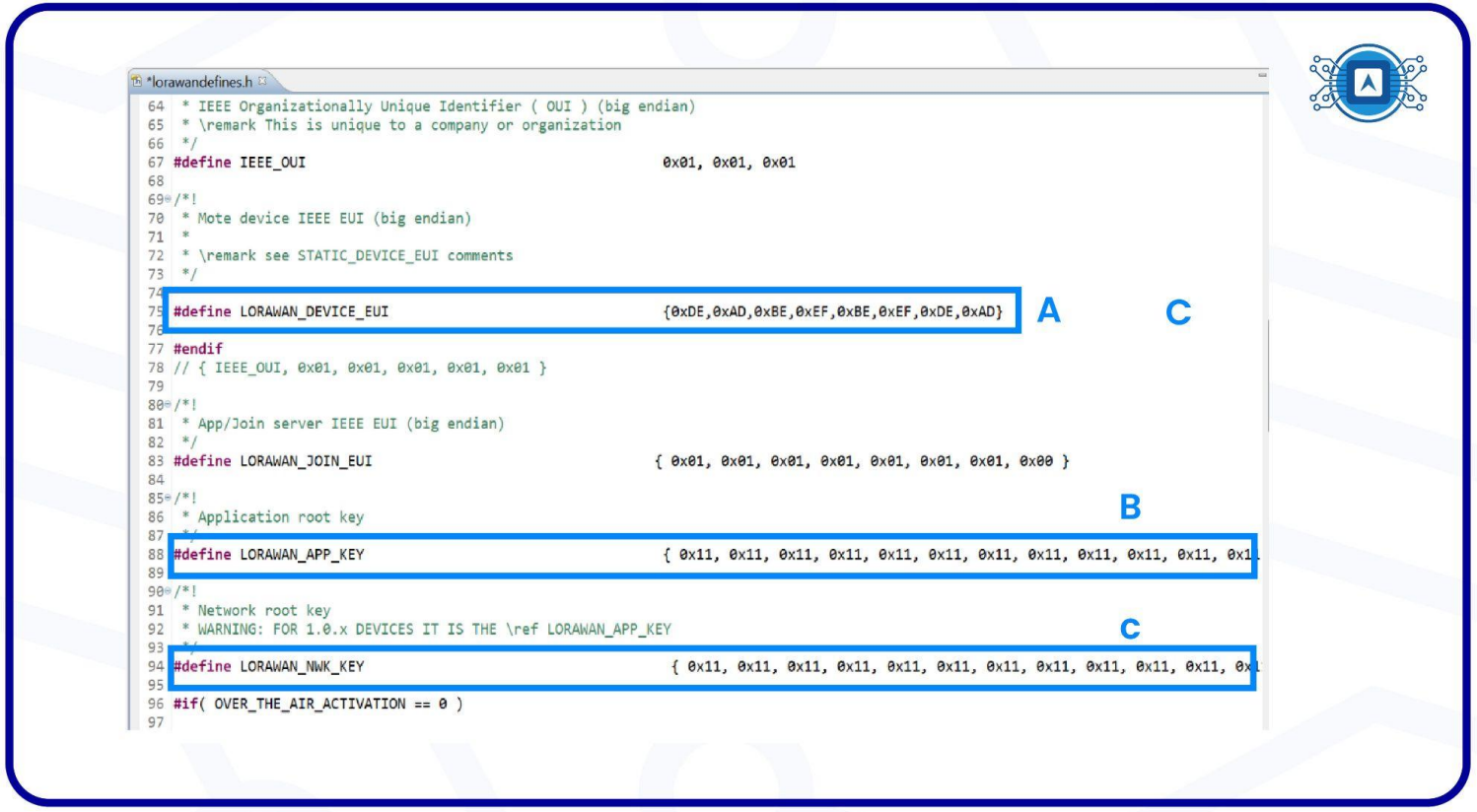

Image 9: Session keys source code. Source: *The author*.

Finally, still in the same source code of the "**lorawandefines.h**" library, (**A**) we set the data transmission cycle "15000 ms" and (**B**) we set the **Class** type, as visualized in image 10 is **class A**.

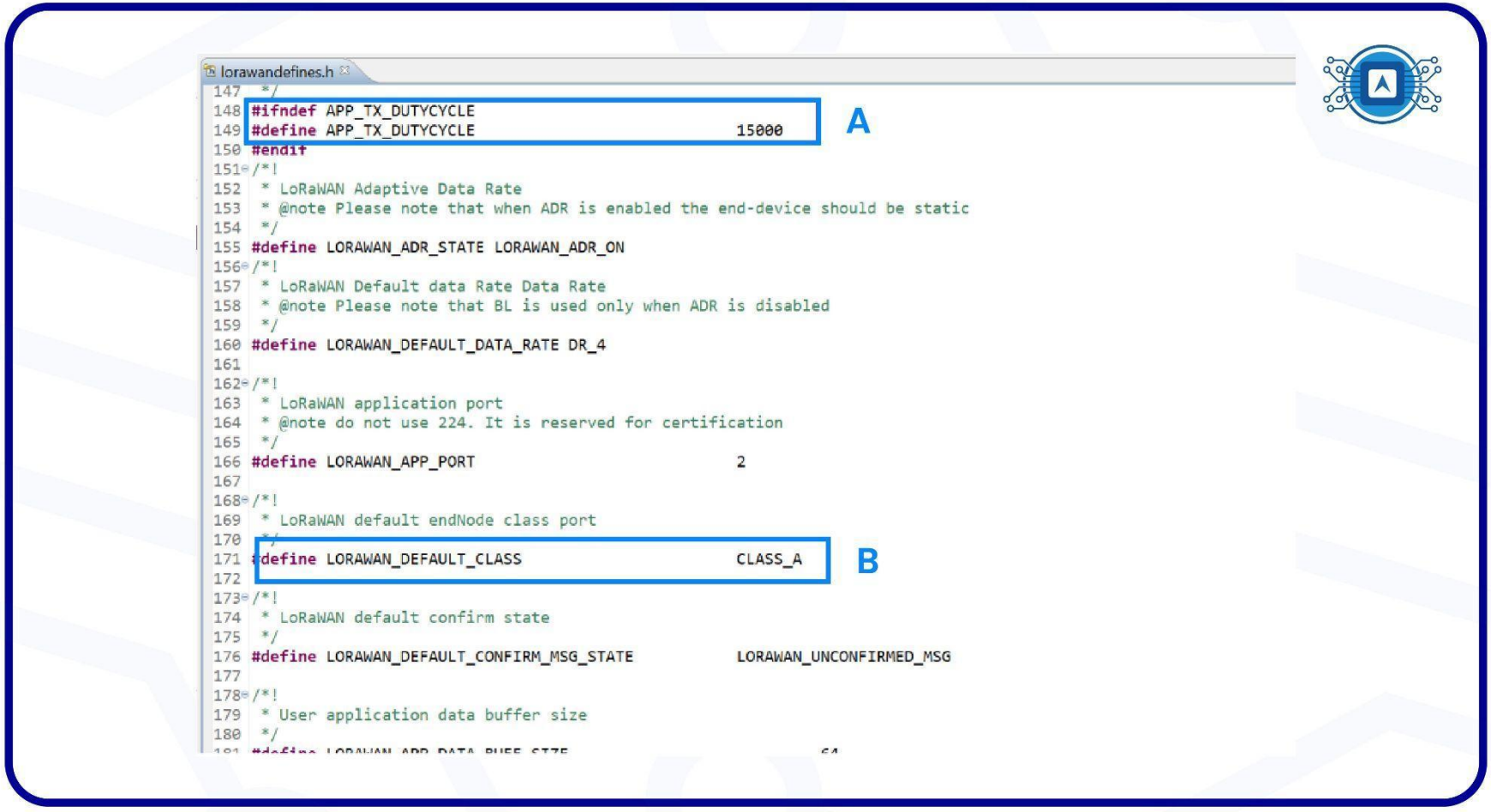

Image 10: Class and data transmission cycle definition. Source: *The author*.

In the source code “**gpio.c**” we can analyze which **GPIO** is set to receive the *push button*. In this case, it is **GPIO_4**.

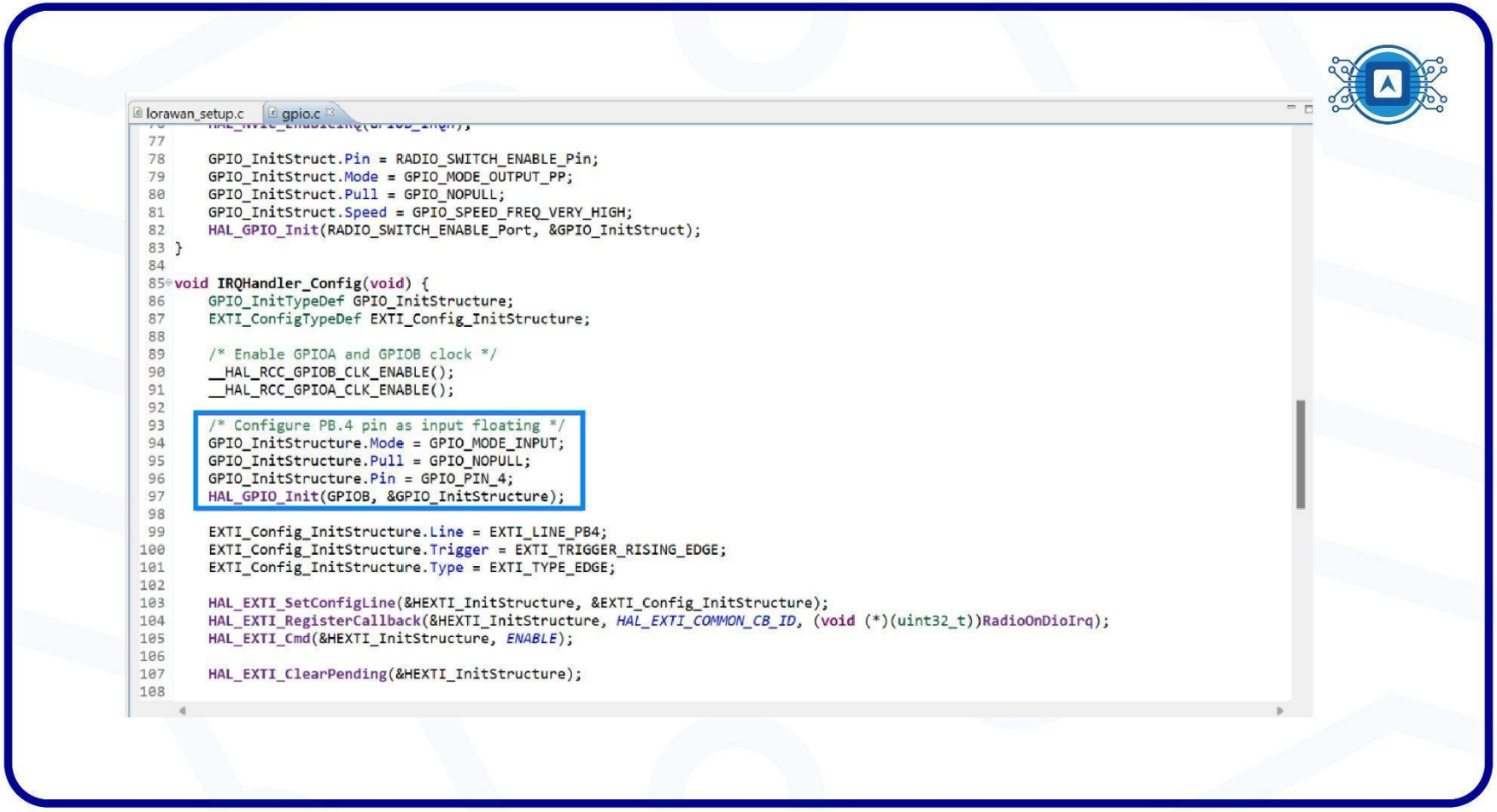

Image 11: GPIO source code. Source: *The author*.

2.2 Build Project

After analyzing the main structure that will be used in this application, right-click on the main project folder and then click “**Build Project**”. This operation will compile all the code and consequently will automatically build the binary of the code.

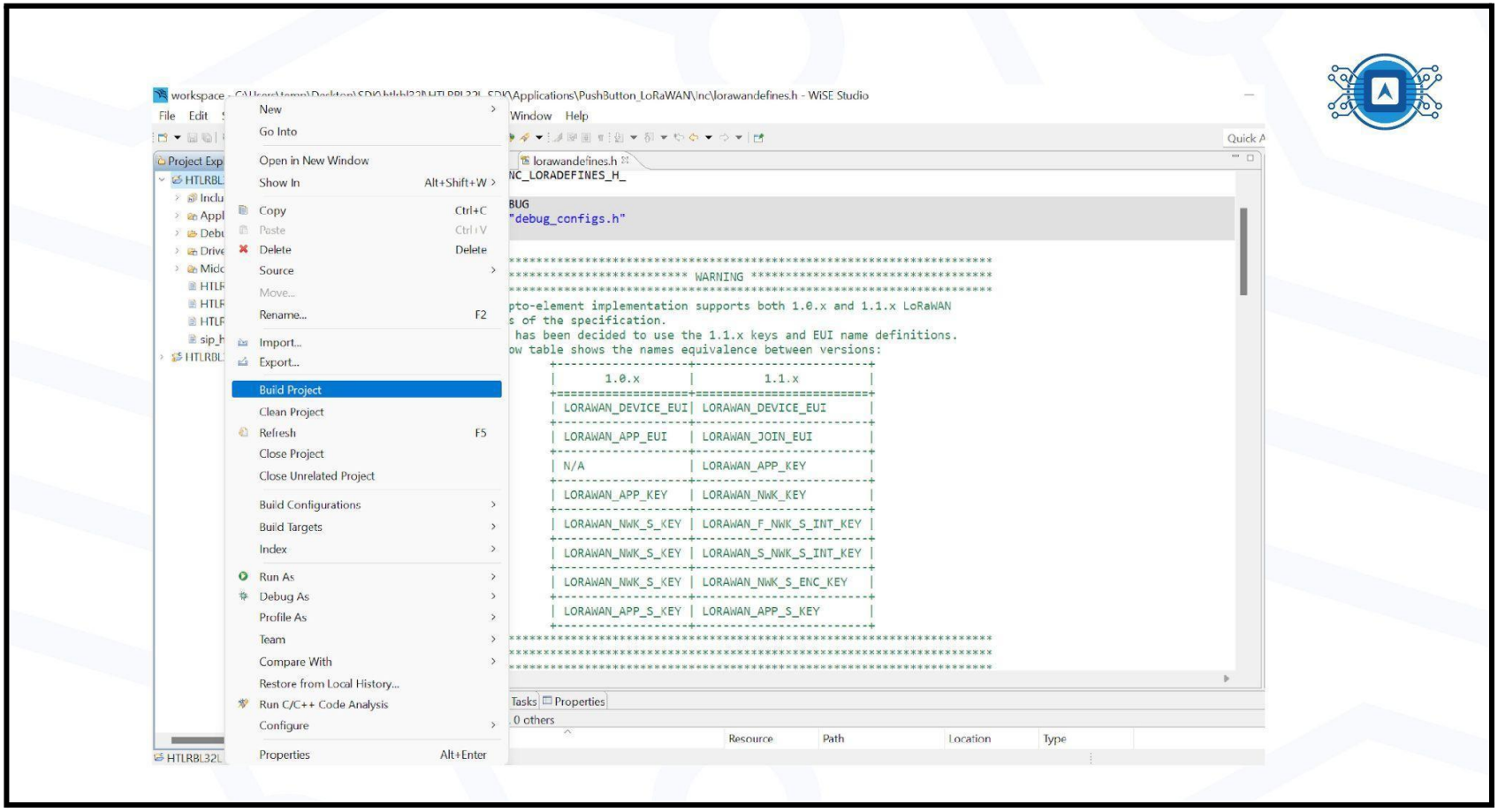

Image 12: Build Project. Source: *The author*.

The “Console” message shows that the Build was finished and that no errors occurred. We can also see that a new folder called “Binaries” has been created.

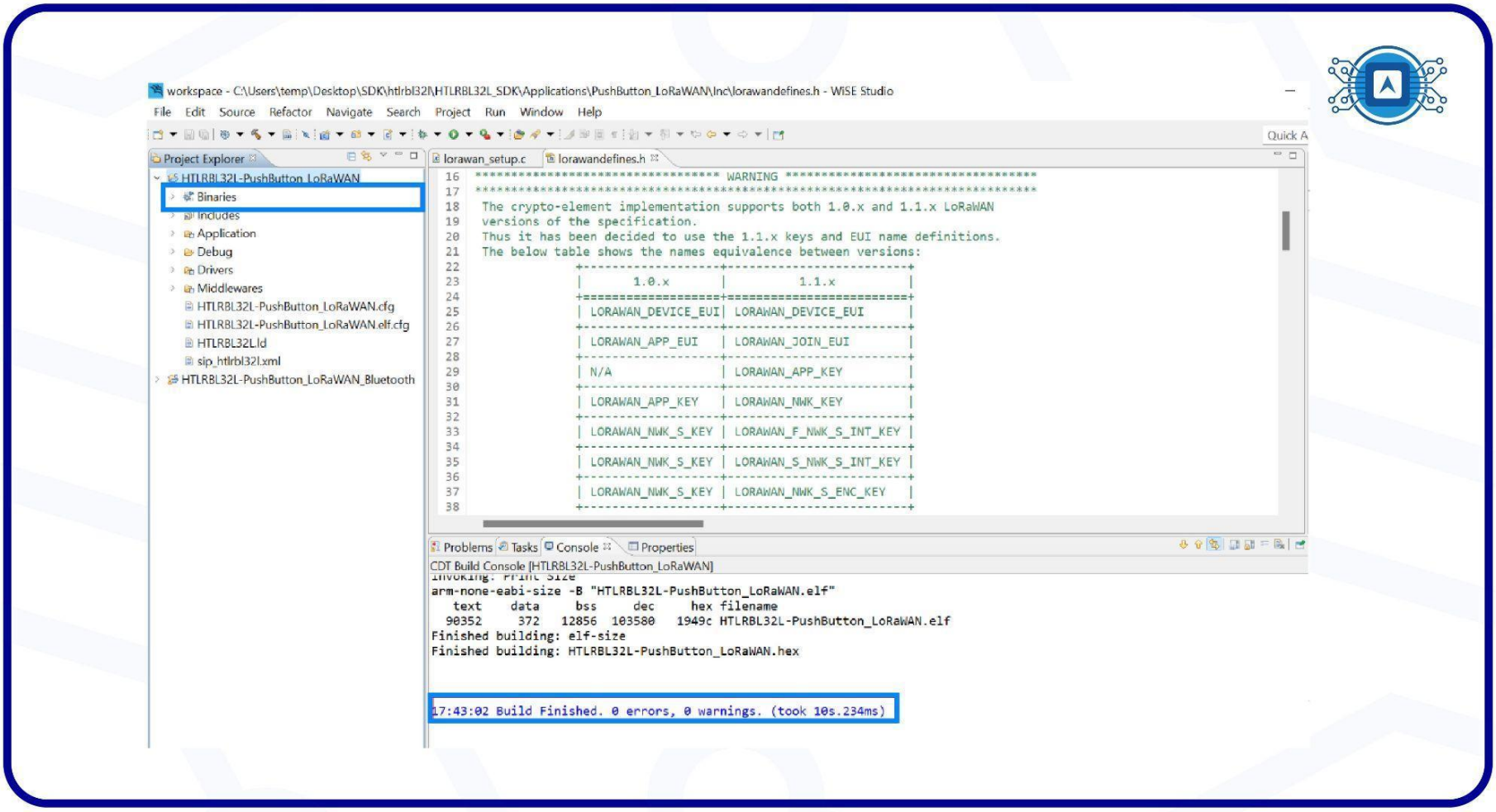

Imagem 13: Arquivo Binário criado. Source: the author.

Additional reading

GIT. **Git.** 2022. Available at: **<** https://git-scm.com/**>.** Accessed on july 28th 2022.

GIT-HUB. **Git guides.** 2022. Available at: **<** https://github.com/git-guides **>.** Accessed on july 28th 2022.

LORA-ALLIANCE. **About-lorawan.** Available at: **<** https://lora-alliance.org/about-lorawan/ **>.** Accessed on july 28th 2022.

PREDICTABLEDESIGNS. **Introduction to Embedded Firmware Development,2021.** Available at: **<** https://predictabledesigns.com/introduction-to-embedded-firmware-development/ **>.** Accessed on july 28th 2022.

THETHINGSNETWORK. **Security 2022.** Available at: **<** https://www.thethingsnetwork.org/docs/lorawan/security/ **>.** Accessed on july 28th 2022.