



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.

	e Search Project Run Window	/ Help					– o ×	
1 1 • 11 10 5 • 5 • 5 • 16	• 63 • 67 • 10 • 0 • 0 •	🦢 🛷 🔹 🕼 🗉 🛪	19 • 6 • 0 • •	⇔ • 🖪			Quick Access	
T T	-/ 1					Coutline Outline Outline	ets 😐 🗆	
Compilor	Debugger Deser	dor				There is no active editor that	provides an outline.	
Compiler	Debugger Record	aer						
					Cor	de editor		
Problems ⅔ @ Tasks	Properties				Cor	de editor	⇒ v = □	
또 Problems 12 중 Tasks © Console Otems Description ^	Properties Resource	Path	Location	Туре	Cod	de editor	* ~ □	
0 items		Path	Location	Туре	Cor	de editor	\$ ∀ ₽	
0 items		Path	Location	Туре	Cod	de editor	≫ ∝ ≂ □	
0 items		Path	Location	Туре	Cod	de editor	* ∝ ≂	

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 <u>website</u> to start the program installation process. After the installation you will see the screenshot below.

File Tools Help			
Select Image file No Image file loaded	L II	Flash from Address: 0x10040000 Hash Sto	р
UART SWD		MAC Address	
Actions I Verify Readout Protection Mass erase Update Device Memory Auto Baudrate	List of COM Ports: Select Al Unselect J Invert A	All Save MAC Address Log	
Flash Programming Device ID:	Device Type:	Interface:	
			0%
Device Memory Image File Co	ompare Device Memory with Image File Compare Two	Files Bootloader/OTP	
Start Address 0x10040000	Size 0x3000	Entire Memory Read Write	
0 1	2 3	4 5 6 7	>
Boards completed: 0/0		0	1%
Show Log		1	
		Load Lo Save Li	
		Clear	

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);
 - \rightarrow 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. If the installation of the drivers did not occur automatically, download the drivers from this <u>website</u>, 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.

₩ RF-Flasher Utility v4.3.0 File Tools Help		- 🗆 X
Select Image file No Image file loaded.	fi	ash from Address: 0x10040000 Flash Stop
UART SWD Actions List of COM Ports: Verify COM3 Mass erase Update Device Memory Auto Baudrate	Select All Unselect All Invert All	Save MAC Address Log
Flash Programming Device ID:	Device Type:	Interface: 0%
Device Memory Image File Compare Device Memory with	Image File Compare Two Files Bootloader/OTP	670 C
Start Address 0x10040000	Size 0x3000	Entire Memory Read Write
0 1 2 3	4 5 6 7 8 9	10 11 12 13 14
<		>
Boards completed: 0/0		0%

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.

Select Image file No Image file loaded.		Flash from Address: 0x1004000	0 Flash Stop	
UART SWD Lis	st of COM Ports; ☑ COM3	Select All MAC Flash Location Unselect All Save MAC Address File Name	0x00000000000 - End: 0x0000000000000	
Flash Programming Device I Device Memory Image File Com Start Address 0x10040000 0 1 2 < Boards completed: 0/0	You have selected COM3. Please, put the device in bo	>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>	e Memory Read Write 12 13 14 0%	
Show Log	L		Load Log Save Log Clear Log	

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.

	Help																			o or
Select Ima	ge file N	o Image f	ile loaded.												Flas	h from /	Address: 0x10040000	Flash	Stop	
UART	SWD																MAC Address			
Actio				ist of COM	Ports:												MAC Address: Start: 0x000000000 MAC Flash Location	0000 - End: 0x0	0000000000	
	/erity teadout P Mass eras	rotertion vice Mem lizite	[🗸 сом	3										OFT 511		Save MAC Address Log			
	Indate De	wice Mem Irate	iorv											INA	ат ли	F	File Name Sot Mac Arter		 timestamp 	
Flash Mem	an Boar	lina	Device II	D	0	омз			Dou	rice Typ	e1	pl	INDC.I	D (may	lach add	leaves : Ou	d007ffff) Interface:	UART		
riasii mein	JIY REAL	mig	Device I	<i>.</i>	U	CIMC			Dev	nce ryp	e.	DI	Jenno-L	P (max		aress. ux	(1007mil) internate:	UART	100%	
Device Me	mory CO	M3 Ir	nage File	Comp	are Devi	ce Memo	ry with I	nage Fi	le O	ompare '	Two File	es E	lootloade	er/OTP	1					
Start A	dress	0x100400	00						Siz	e 0x300	0						Entire Memor	Read	Write	
		0	1 2	3		5	6	7	8	9	10	11	12	13	14	15		TROOT	A	
0~100/	10000			20	99	85	04	10	D1	69	04	10	5D	39	04	10	Ňi]9			
	10010				45	55	4C	42	00	00	00	00	00	00	00	00	*UU*EULB			
		ed: 1/		-	49	22	40	42	00	00	00	00	00	00	00	00	-00-0018		100%	
			-																	
Show Log																				
10:31:1	1.696:	Device	COM3 ->	Exit B	tead M	emory:	Except	ion I	D1: au	to bau	drate	funct	ionali	ty fa	led			^	Load Log Save Log	
*****R	EAD OPE	RATION																	Clear Log	
Read of	n.1 de	vice c	onnected	i by UAI	T: CO	жз														
10:34:3			COM3 ->						G-LP											
10:34:3																		~		

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 <u>website</u> 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.

	ello_World.bin					Flash from Address: 0x10040000 Flash Stop
UART SWD	-					MAC Address
Actions	List of COM Ports:					
Verify	☐ сомз ✓ сом13 (ST DK)				Select All	MAC Address: Start: 0x0000000000 - End: 0x0000000000
Readout Protection					Unselect Al	MAC Flash Location
Mass erase Update Device Memor					Invert All	Save MAC Address Log
Auto Baudrate	Y					File Name
						Set Mac Address
Flash Memory Reading	Device ID: COM	13 (ST DK)	Device Type:	BlueN	RG-LP (max flash address: 0x	(1007ffff) Interface: UART
0.10010000 00 00	01 20 2D 16 04	10 71 15 04	10 7D	15 04 10	~ q }	
	SS AA 4S SS 4C 00 00 00 00 00 00	42 00 00 00 00 00 00 00 00 77 00 00		00 00 00 15 04 10	-00-2018	v 100%
0x10040010 AA 55 0x10040020 00 00	SS AA 4S SS 4C 00 00 00 00 00 00				-00-2018	100%
0x10040010 AA 55 0x10040020 00 00 Boards completed: 1/1 Show Log	SS AA 4S SS 4C 00 00 00 00 00 00				-00-2018	100%
0x10040010 AA 55 0x10040020 00 00 Boards completed: 1/1 Show Log	SS AA 4S SS 4C 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00			-00-2018	

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: < <u>https://www.veracode.com/security/integrated-development-environment</u> >. Accessed on june 08th 2022.

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

Additional reading

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

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