



INDUSTRIAL SHIELDS

OPEN MOTE B

Open Mote B User Guide:



Open Mote B User Guide

Revision August 2019

Preface

This User Guide is been implemented by Boot & Work, S.L. working under the name Industrial Shields.

Purpose of the manual

The information contained in this manual can be used as a reference to operating, to functions, and to the technical data of the signal modules, power supply modules and interface modules.

Intended Audience

This User Guide is intended for the following audience:

- Persons in charge of introducing automation devices.
- Persons who design automation systems.
- Persons who install or connect automation devices.
- Persons who manage working automation installation.



Warnings:

- Unused pins should not be connected. Ignoring the directive may damage the controller.
- Improper use of this product may severely damage the controller.
- Refer to the controller's User Guide regarding wiring considerations.
- Before using this product, it is the responsibility of the user to read the product's User Guide and all accompanying documentation.



Advertisements:

- Les broches non utilisées ne doivent pas être connectées. Ignorer la directive peut endommager le contrôleur.
- Une utilisation incorrecte de ce produit peut endommager gravement le contrôleur.
- Reportez-vous au Guide de l'utilisateur du contrôleur pour les considérations de câblage.
- Avant d'utiliser ce produit, il incombe à l'utilisateur de lire le Guide de l'utilisateur du produit et la documentation qui l'accompagne.

Application Considerations and Warranty

Read and Understand this Manual

Please read and understand this manual before using the product. Please consult your comments or questions to Industrial Shields before using the product.

Application Consideration

THE PRODUCTS CONTAINED IN THIS DOCUMENT ARE NOT SAFETY RATED. THEY SHOULD NOT BE RELIED UPON AS A SAFETY COMPONENT OR PROTECTIVE DEVICE FOR ENSURING SAFETY OF PERSONS, AS THEY ARE NOT RATED OR DESIGNED FOR SUCH PURPOSES.

Please know and observe all prohibitions of use applicable to the products.

FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, NEVER USE THE INDUSTRIAL SHIELDS PRODUCTS.

NEVER USE THE INDUSTRIAL SHIELDS PRODUCTS BEFORE THEY ARE PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Industrial Shields shall not be responsible for conformity with any codes, regulations or standards that apply to the combination of products in the customer's application or use of the product.

The following are some examples of applications for which particular attention must be given. This is not intended to be an exhaustive list of all possible uses of the products, nor is it intended to imply that the uses may be suitable for the products:

- Systems, machines, and equipment that could present a risk to life or property.
- Nuclear energy control systems, combustion systems, railroad systems, aviation systems, medical equipment, amusement machines, vehicles, safety equipment, and installation subject to separate industry or government regulations.
- Outdoor use, uses involving potential chemical contamination or electrical interference, or conditions or uses not described in this document.

At the customer's request, INDUSTRIAL SHIELDS will provide applicable third party certification documents identifying ratings and limitations of use that apply to the products. This information by itself is not sufficient for a complete determination of the suitability of the products in combination with the system, machine, end product, or other application or use.

Disclaimers

Weights and Dimensions

Dimensions and weights are nominal and they are not used for manufacturing purposes, even when tolerances are shown.

Performance Data

The performance data given in this manual is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of INDUSTRIAL SHIELDS's test conditions, and the users must correlate it to actual application requirements. Actual performance is subject to the INDUSTRIAL SHIELDS Warranty and Limitations of Liability.

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons.

It is our practice to change model numbers when features are changed, or published ratings or when significant construction changes are made. However, some specifications of the products may be changed without any notice. When in doubt, special numbers may be assigned to fix or establish key specifications for your application on your request. Please consult with your INDUSTRIAL SHIELDS representative at any time to confirm actual specifications of purchased products.

Errors and Omissions

The information in this document has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

Warranty and Limitations of Liability

Warranty

Industrial Shields's exclusive warranty is that the products are free from defects in materials and workmanship for a period of one year (or other period if specified) from date of sale by Industrial Shields.

INDUSTRIAL SHIELDS MAKES NO REPRESENTATION OR WARRANTY, EXPRESSED OR IMPLIED, REGARDING MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR PARTICULAR PURPOSE OF THE PRODUCTS. ANY BUYER OR USER ACKNOWLEDGES THAT THE BUYER OR USER ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE. INDUSTRIAL SHIELDS DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED

Limitations of Liability

INDUSTRIAL SHIELDS SHALL NOT BE RESPONSIBLE FOR SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR COMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED ON CONTRACT, WARRANTY, NEGLIGENCE, OR STRICT LIABILITY.

IN NO EVENT SHALL INDUSTRIAL SHIELDS BE RESPONISBLE FOR WARRANTY, REPAIR OR OTHER CLAIMS REGARDING THE PRODUCTS UNLESS INDUSTRIAL SHIELDS'S ANALYSIS CONFIRMS THAT THE PRODUCTS WERE PROPERLY HANDLED, STORED, INSTALLED, AND MAINTAINED AND NOT SUBJECT TO CONTAMINATION, ABUSE, MISUSE, OR INAPPROPRIATE MODIFICATION OR REPAIR.

Table of Contents

1	General Description Open Mote B product	9
1.1	Industrial Internet of Things development platform	9
2	Technical Specifications	9
2.1	Micro-Controller (Texas Instruments, CC2538)	9
2.2	Transceiver 1 (Texas Instruments, CC2538)	10
2.3	Transceiver 2 (ATMEL, AT86RF215)	10
3	Platform Characteristics	10
4	System overview	11
4.1	Texas Instruments CC2538	11
5	Dimensions	14
6	Software	15
6.1	Preelimitaries.....	Error! Bookmark not defined.
6.2	Windows 10.....	15
6.2.1	Python 3.x.....	15
6.2.2	GCC Compiler for ARM Embedded Processors	19
6.2.3	GCC Compiler for ARM Embedded Processors	23
6.3	GNU/Linux	29
6.4	OpenMote	30
6.4.1	Windows 10.....	30
6.4.2	GNU/Linux	33
6.4.3	Blinking LED example	34
7	Revision Table	37



INDUSTRIAL SHIELDS



1 General Description Open Mote B product

1.1 Industrial Internet of Things development platform

The OpenMote B is the ultimate hardware development and prototyping platform for the Industrial Internet of Things (IIoT), specifically to researchers and developers working towards next-generation long-range and low-power wireless field area networks based on the IPv6 stack. It is built around the well-supported Texas Instruments CC2538 ARM-Cortex-M3 micro-controller, and it features simultaneous multi-band operation in the 2.4 GHz and 868/915MHz ISM bands with complete support for the latest IEEE 802.15.4 standards, including the MR-OFDM modulations of IEEE 802.15.4g-2012.

2 Technical Specifications

2.1 Micro-Controller (Texas Instruments, CC2538)

- **ARM Cortex-M3 with code pre-fetch**
 - Running at 16 MHz or 32 MHz
 - 32 Kbytes RAM
 - 512 Kbytes FLASH
- **On-chip peripherals:**
 - 4x general purpose, 1x sleep timer
 - 1x 12 bit ADC with 8 channels
 - 2x SPI, 2x UART, 1x I2C
- **Security hardware acceleration:**
 - AES-128/256/SHA2 encryption
 - ECC-128/256 secure key exchange
- **Low-power operation:**
 - Active mode: 7/13mA (16/32 MHz)
 - LPM1: 600uA (full retention, 4us wake-up)
 - LPM2: 1.3 uA (16 Kbyte RAM retention, 128us wake-up, wake-up from RTC)
 - LPM3: 0.4 uA (16 Kbyte RAM retention, 128us wake-up, wake-up from GPIO)

2.2 Transceiver 1 (Texas Instruments, CC2538)

- **Operates in the 2.4 GHz ISM band with support for IEEE 802.15.4-2006**
 - Modulation: OQPSK with DSSS
 - Data rate: 250 kbps
 - Receiver sensitivity: -97 dBm
 - Transmit power: 7 dBm
 - Transmit current: 24 mA at 0 dBm
 - Receive current: 20 mA

2.3 Transceiver 2 (ATMEL, AT86RF215)

- **Operates in the 868/915MHz and 2.4 GHz ISM bands with support for IEEE 802.15.4g-2012**
 - Modulation: MR-FSK/OFDM/O-QPSK
 - Data rate: 6.25 kbps to 2400 kbps
 - Receiver sensitivity: -123 dBm
 - Transmit power: 14.5dBm
 - Transmit current: 62 mA at 14 dBm
 - Receive current: 28 mA

3 Platform Characteristics

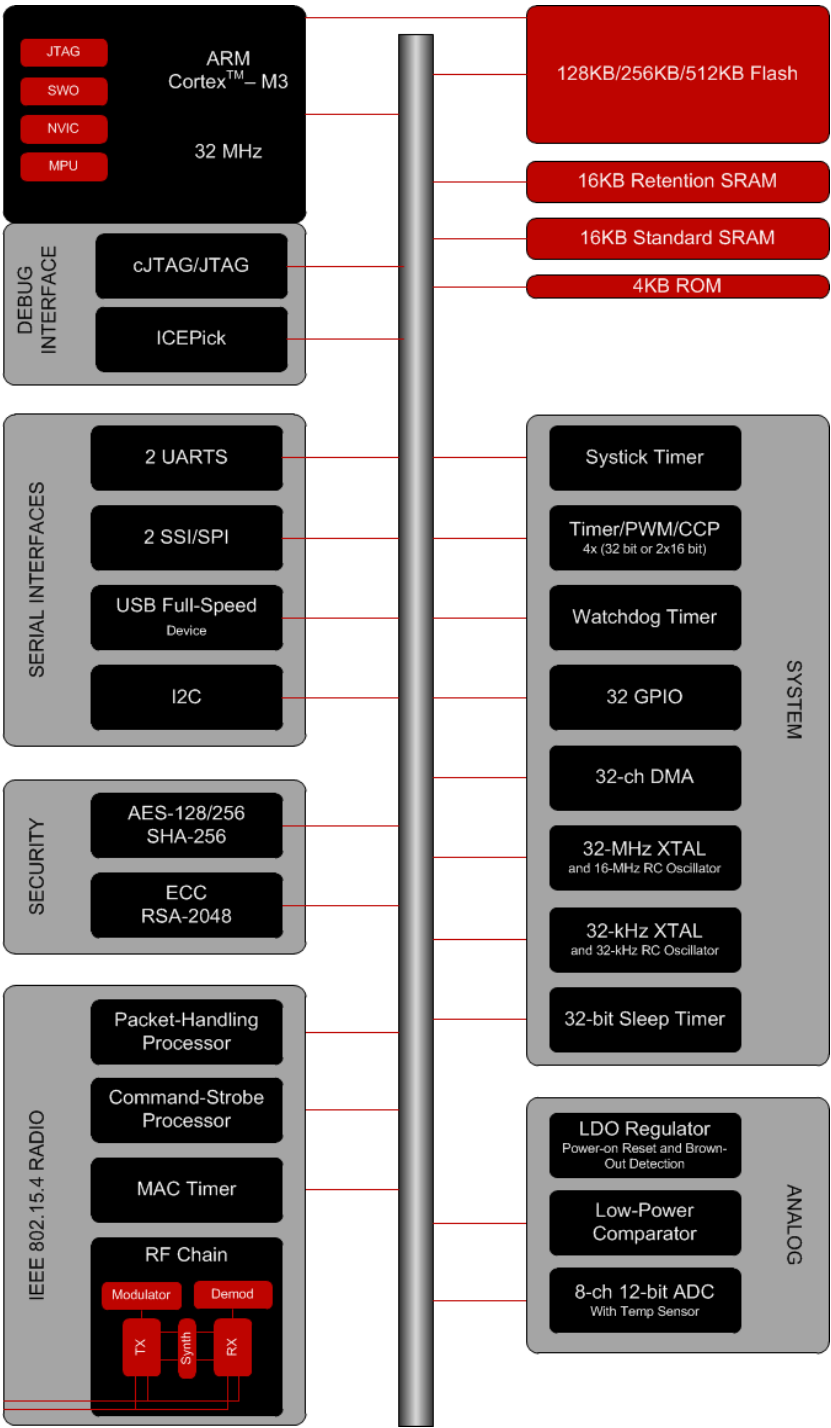
- **Human interfacing:** The main board includes 4 LEDs (Green, Yellow, Orange and Red) and a user button which are intended for debugging purposes. In addition, the main board also includes a reset button that allows to perform a hardware reset.
- **Serial communications:** The main board can communicate with a computer using a UART port on the CC2538. The solution is based on the FTDI FT2232H chip, a Serial-to-USB converter that allows to communicate with a computer using a standard UART port. In addition, the FTDI chip allows to program the CC2538 directly using the internal bootloader and the cc2538-bsl Python script.
- **Board expansion:** The main OpenMote-B board includes an expansion port (8 pins with 2.54 mm spacing) that can be used for debugging or to connect daughter boards, i.e., the OpenMote-B sensors board. The expansion board includes a VCC (2.5V) and a GND pin, as well as six configurable pins.
- **Extended security:** The main OpenMote-B board includes hardware-accelerated support for cryptographic functions using SHA2, AES-128/256, ECC-128/256 and RSA algorithms.

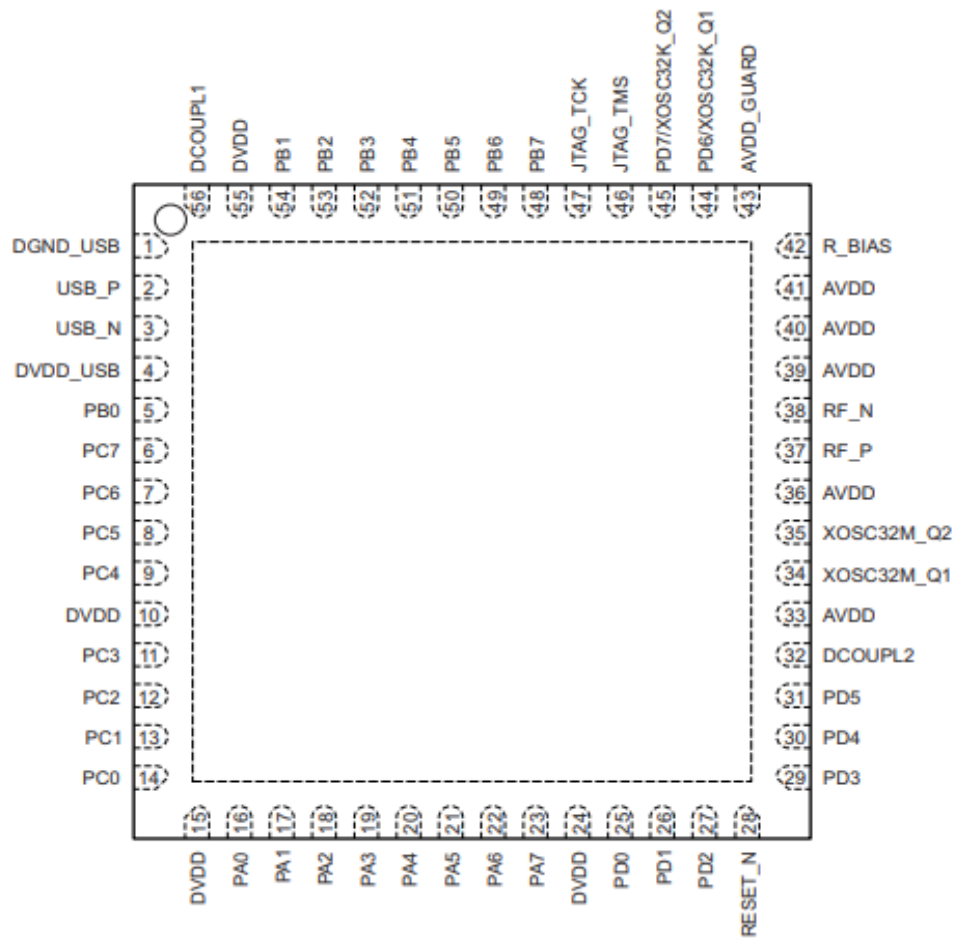
- **Antenna connectors:** The main board includes two SMA antenna connectors for the Sub-GHz and 2.4 GHz antennas. The Sub-GHz antenna connector is directly connected to the Sub-GHz radio on the AT86RF215. The 2.4 GHz antenna is multiplexed using an RF switch to the CC2558 and the AT86RF215 radio transceivers. The direction of the RF switch can be controlled using two CC2538 pins.
- **Power:** The main board can be powered from a USB port (5V) through an A male plug connector, or from two AA batteries (3V) located on the back of the board. The board features an auto-switching mechanism that selects the USB source whenever it is available and seamlessly transitions to the AA batteries when disconnected. Finally, the board includes an on/off button to disconnect the two AA batteries when not used.
- **Current sensing:** The main board includes two ports (3 pins with 2.54 mm spacing) to measure current consumption of the whole system during operation. The first port measures the current consumption of the CC2538 chip, whereas the second port measures the current consumption of the AT86RF215 chip.
- **JTAG port:** The main board includes a 10-pin ARM connector that allows to load and debug code using an external JTAG probe. The interface is compatible with the main toolchains: Code Composer Studio, IAR Embedded Workbench and ARM.

4 System overview

4.1 Texas Instruments CC2538

The Texas Instruments CC2538 is a wireless micro-controller SoC (System on Chip) targeted at high-performance applications. It combines a powerful ARM Cortex-M3 running at 32 MHz, with 32 Kbytes of RAM and 512 Kbytes of Flash, with a robust 2.4 GHz radio transceiver compatible with the IEEE 802.15.4 standard. In addition, the CC2538 includes various peripherals to interface the processor with other systems: 2x SPI, 2x UART, 1x I2C.





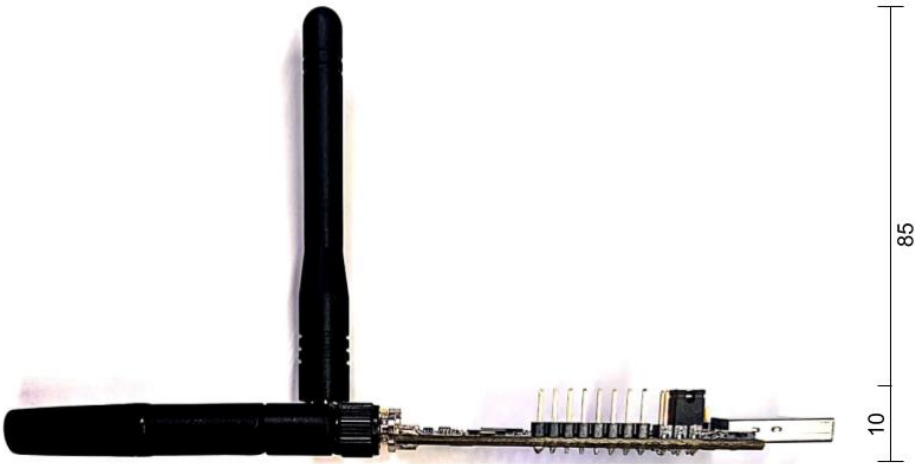
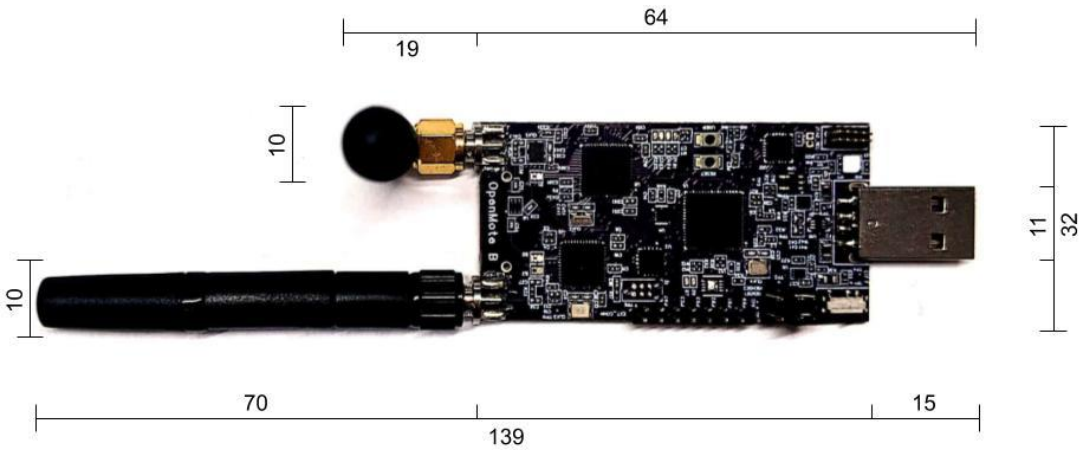
CC2538 Datasheet (<http://www.ti.com/lit/gpn/cc2538>)

CC2538 User Manual (<http://www.ti.com/lit/pdf/swru319>)

CC2538 Errata (<http://www.ti.com/lit/pdf/swrz045>)

5 Dimensions

In mm



6 Software

This section is aimed at providing a starting point to run the main open-source software projects on the OpenMote-B hardware.

6.1 Preliminaries

To program the OpenMote-B boards with the various open source software projects it is necessary to have a working environment to build, upload, and debug the code for the ARM Cortex-M3 processor. It is possible to create a working environment with either Windows or Linux, as described next. Macintosh is known to work, but the former operating systems are preferred as they have been thoroughly tested.

Hence, the following steps show how to install the tools required to compile, run and debug embedded software on the OpenMote-B boards using either Windows 10 or Ubuntu 18.04 LTS. The process assumes that the user has a working installation of either operating system, either natively on their computer or in a virtual environment (i.e. VMWare Workstation or VirtualBox).

The tools that are required are:

- Python 3 with the SCons, PySerial and IntelHex packages
- GCC compiler for ARM Embedded Processors

6.2 Windows 10

6.2.1 Python 3.x

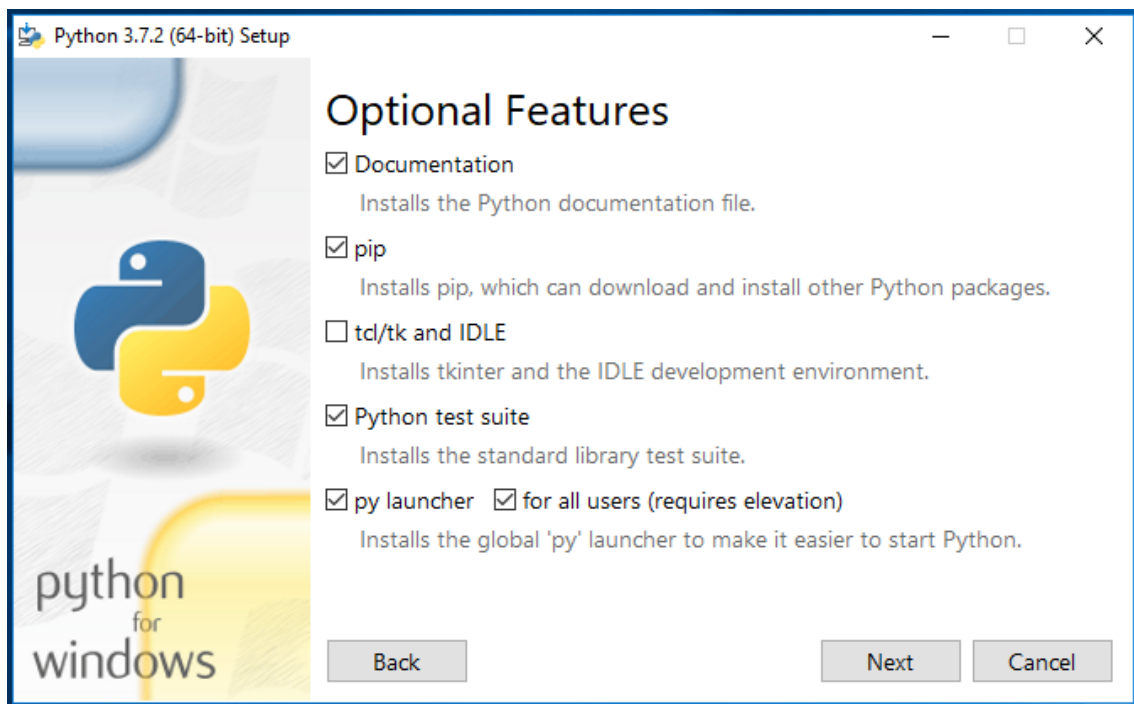
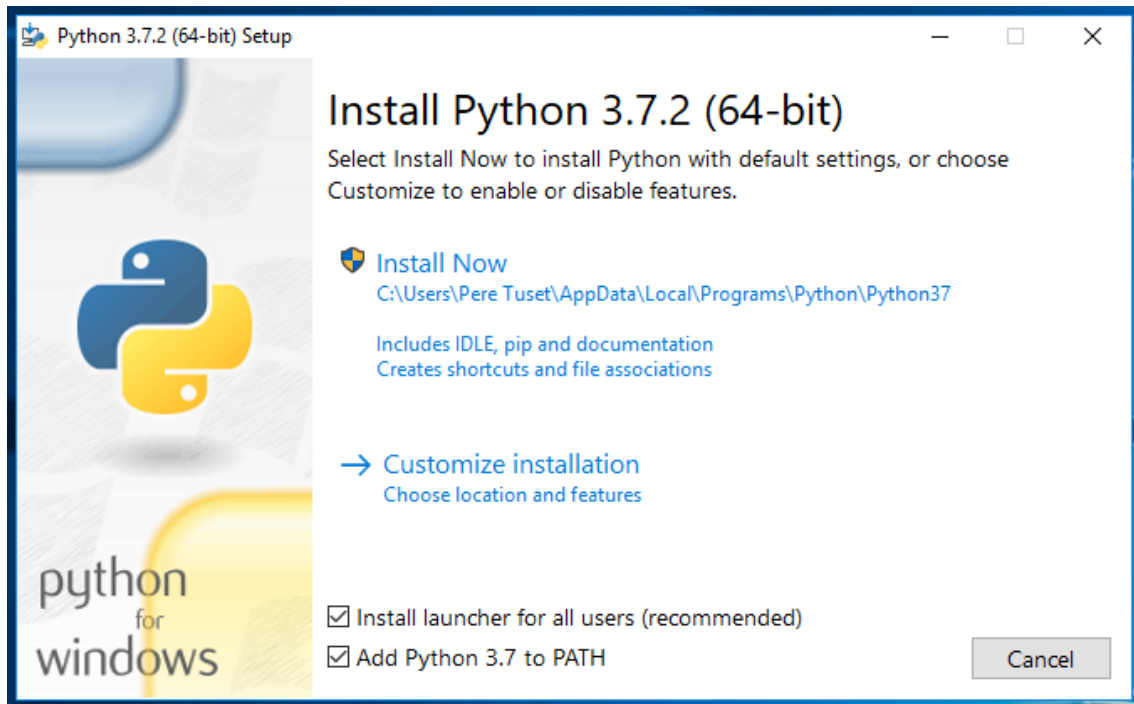
To install Python 3.x go to the official website and download the latest version¹.

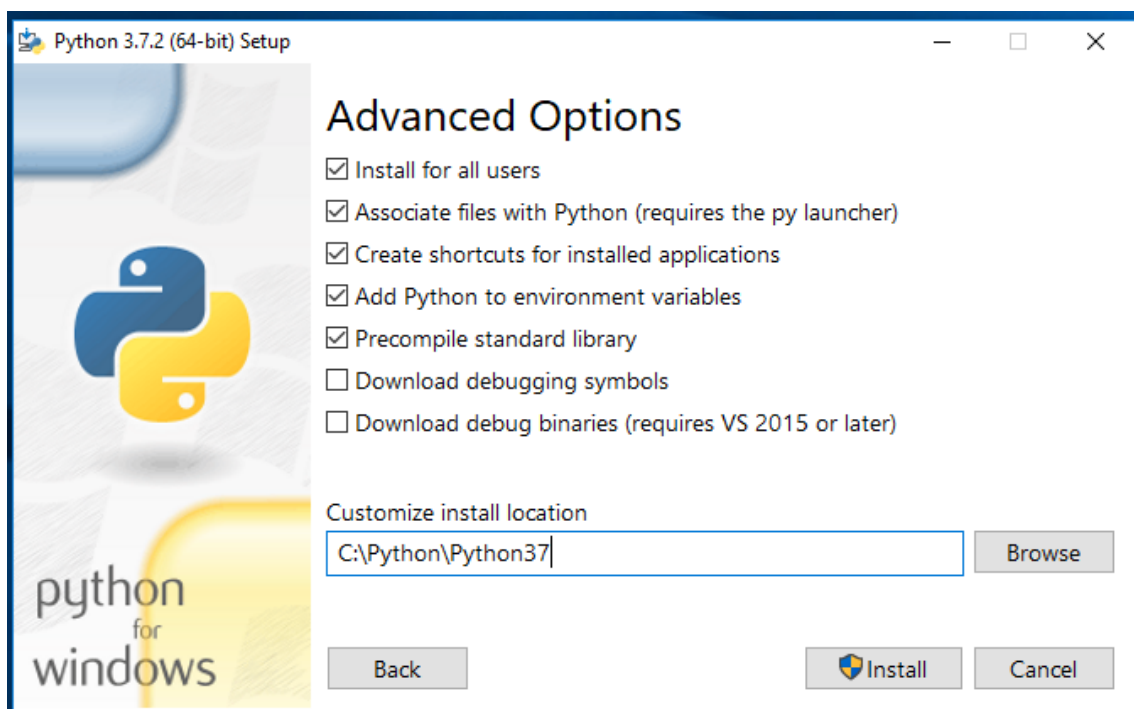
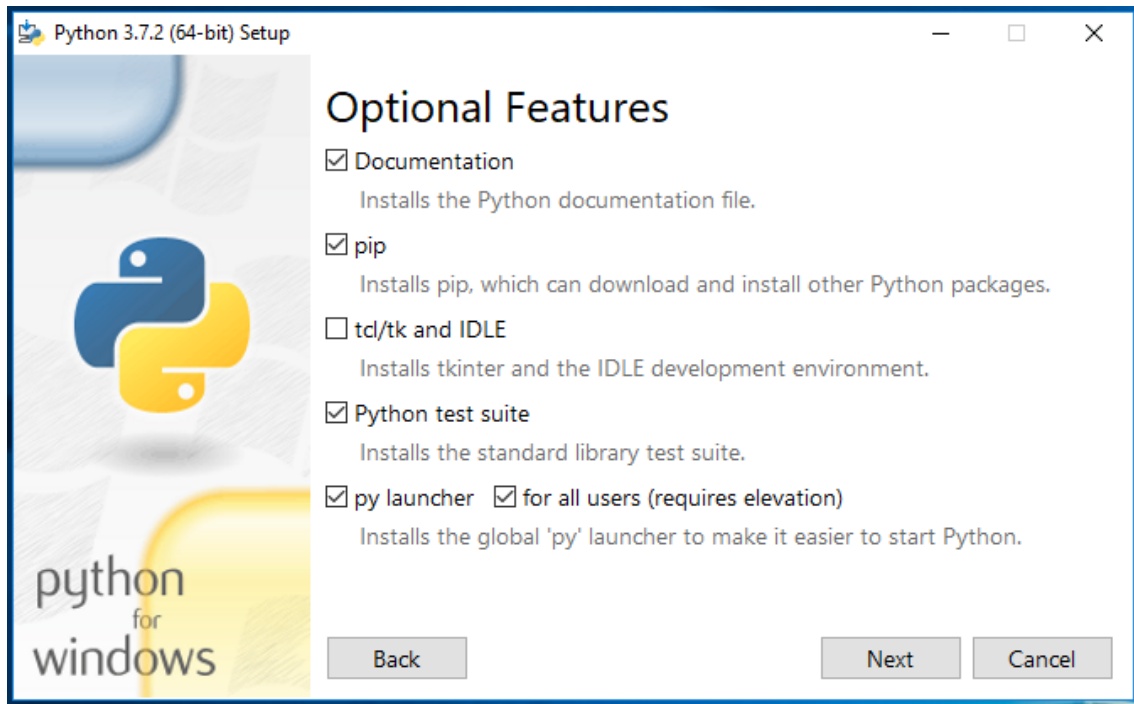
Python Releases for Windows

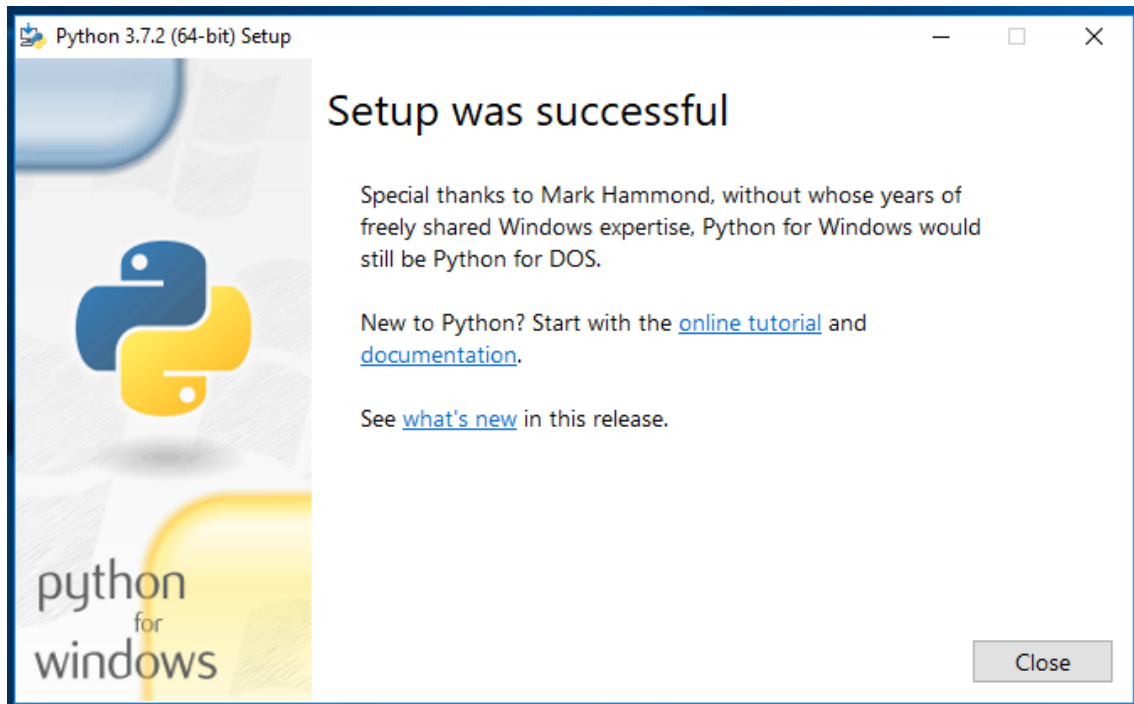
- [Latest Python 3 Release - Python 3.7.2](#)
- [Latest Python 2 Release - Python 2.7.15](#)
- [Python 3.7.2 - 2018-12-24](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86-64 web-based installer](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows help file](#)

¹ <https://www.python.org/downloads/windows/>

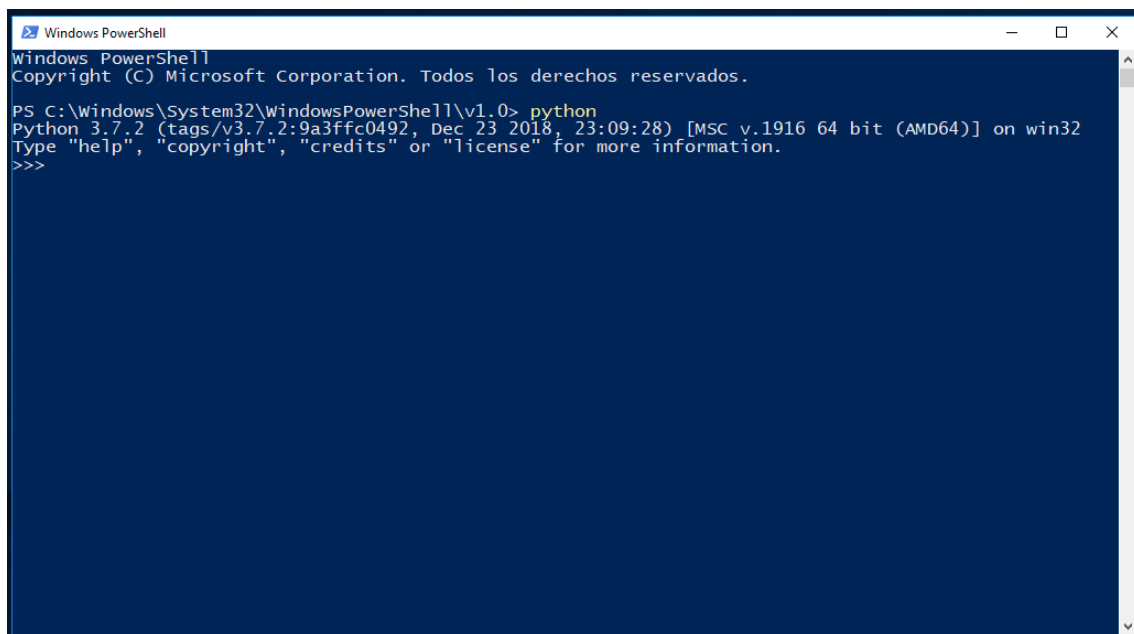
Once downloaded execute the installer and follow the steps depicted in the next Figures. It is important that you install Python for all users and also that you add the directory to the system PATH.





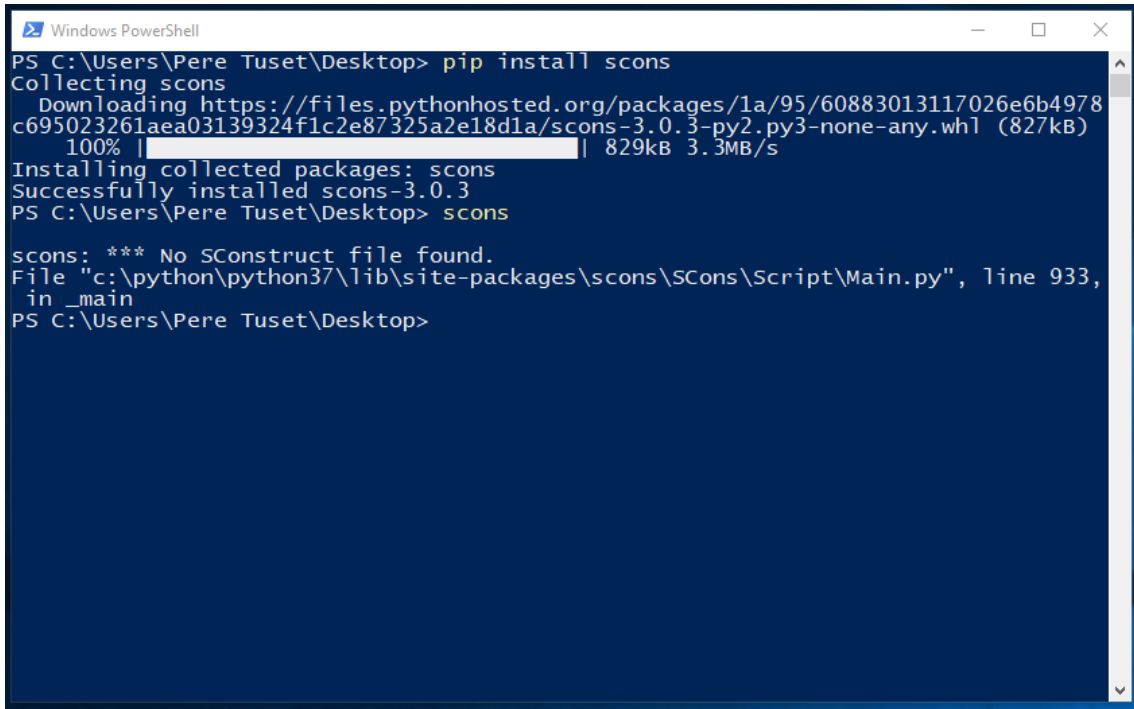


Once Python 3.x is installed in your system you should be able to execute it by running the python command from a command line, i.e. Windows PowerShell. If the installation was successful you should see a terminal like the next image.



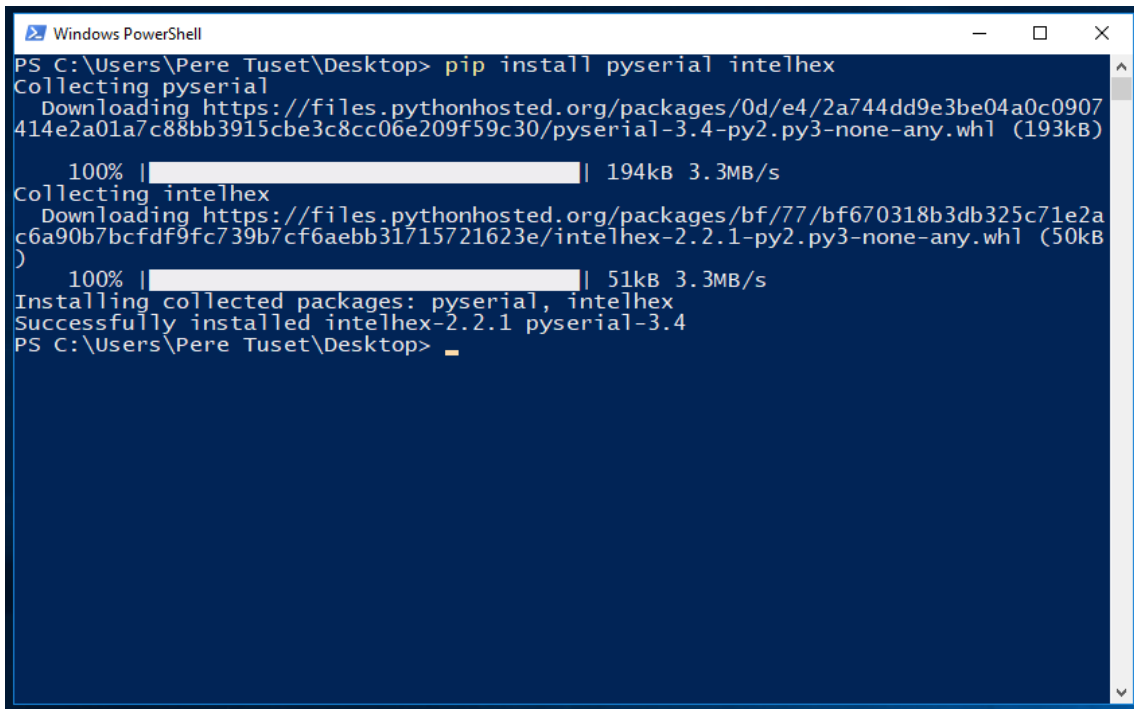
Once Python 3.x is installed you need to install the additional packages that are required to compile, load and debug code. As mentioned earlier, these packages are SCons, PySerial and IntelHex and they can be installed using pip, as shown in the next images.

- Pip install scons
- Pip install pyserial
- Pip install intelhex



```
Windows PowerShell
PS C:\Users\Pere Tuset\Desktop> pip install scons
Collecting scons
  Downloading https://files.pythonhosted.org/packages/1a/95/60883013117026e6b4978c695023261aea03139324f1c2e87325a2e18d1a/scons-3.0.3-py2.py3-none-any.whl (827kB)
    100% |#####| 829kB 3.3MB/s
Installing collected packages: scons
Successfully installed scons-3.0.3
PS C:\Users\Pere Tuset\Desktop> scons

scons: *** No SConstruct file found.
File "c:\python\python37\lib\site-packages\scons\SCons\Script\Main.py", line 933,
in _main
PS C:\Users\Pere Tuset\Desktop>
```



```
Windows PowerShell
PS C:\Users\Pere Tuset\Desktop> pip install pyserial intelhex
Collecting pyserial
  Downloading https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb3915cbe3c8cc06e209f59c30/pyserial-3.4-py2.py3-none-any.whl (193kB)
    100% |#####| 194kB 3.3MB/s
Collecting intelhex
  Downloading https://files.pythonhosted.org/packages/bf/77/bf670318b3db325c71e2ac6a90b7bcfdf9fc739b7cf6aebb31715721623e/intelhex-2.2.1-py2.py3-none-any.whl (50kB)
    100% |#####| 51kB 3.3MB/s
Installing collected packages: pyserial, intelhex
Successfully installed intelhex-2.2.1 pyserial-3.4
PS C:\Users\Pere Tuset\Desktop>
```

6.2.2 GCC Compiler for ARM Embedded Processors

To install the GCC Compiler for ARM Embedded Processors go to the official website and download the latest version: 7-2017-q4-major².

² <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads/7-2017-q4-major-1-1>

What's new in 7-2017-q4-major

In this release

1 gcc-arm-none-eabi-7-2017-q4-major-win32-sha1.exe

Windows 32-bit Installer (Signed for Windows XP and Vista)
MD5: 66c48495d7eb7239acad0290cb318c6a

2 gcc-arm-none-eabi-7-2017-q4-major-win32-sha2.exe

Windows 32-bit Installer (Signed for Windows 7 and later)
MD5: 6d53dc8e301b78769e7d50b79811093f

Windows 32-bit

File: gcc-arm-none-eabi-7-2017-q4-major-win32-sha1.exe
(82.53 MB)

[Download](#)

Windows 32-bit

File: gcc-arm-none-eabi-7-2017-q4-major-win32-sha2.exe
(82.53 MB)

[Download](#)

Windows 32-bit

File: gcc-arm-none-eabi-7-2017-q4-major-win32.exe (82.53 MB)

[Download](#)

Windows ZIP

File: gcc-arm-none-eabi-7-2017-q4-major-win32.zip (123.71 MB)

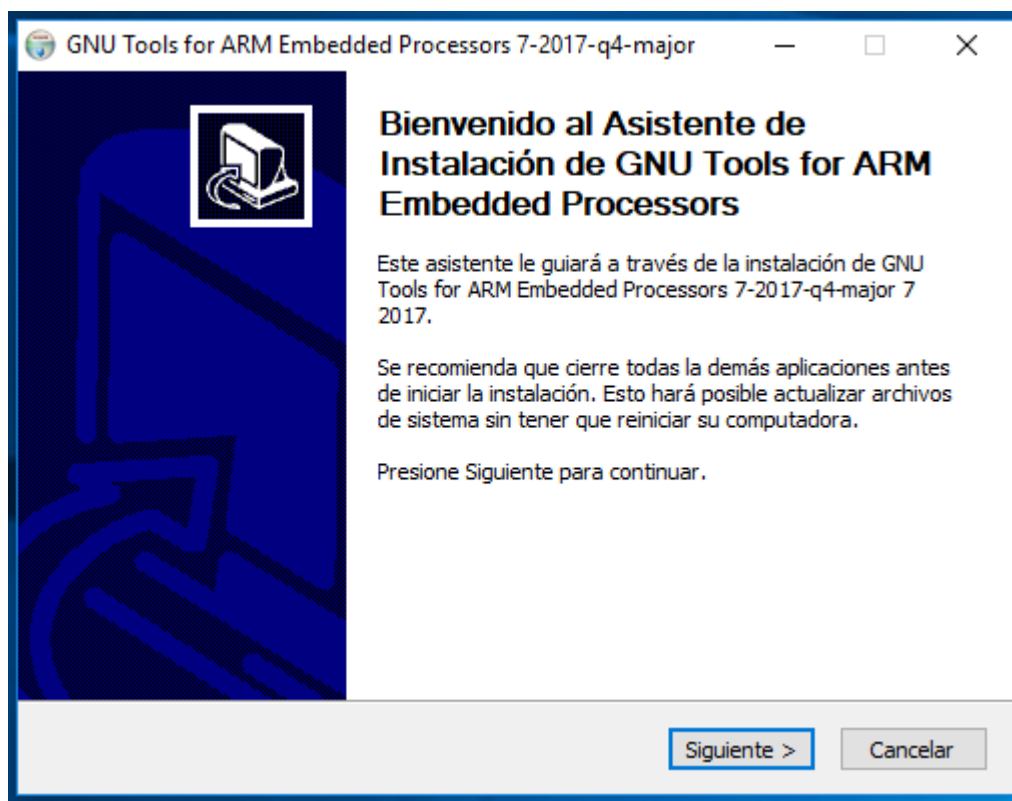
[Download](#)

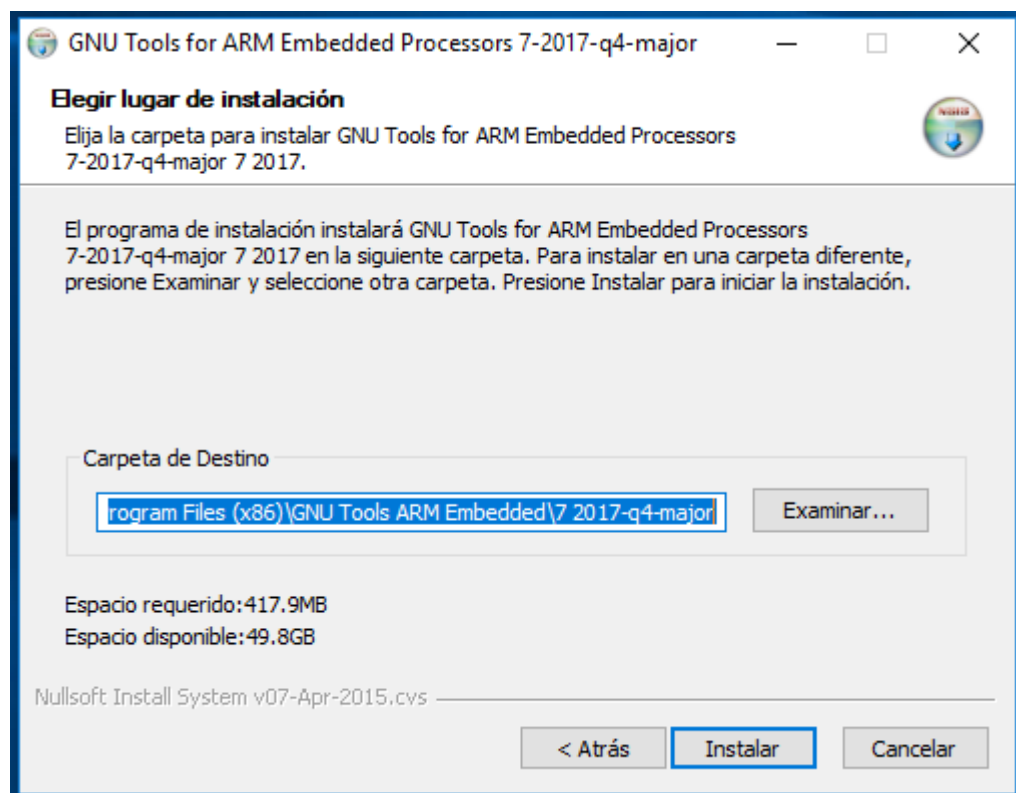
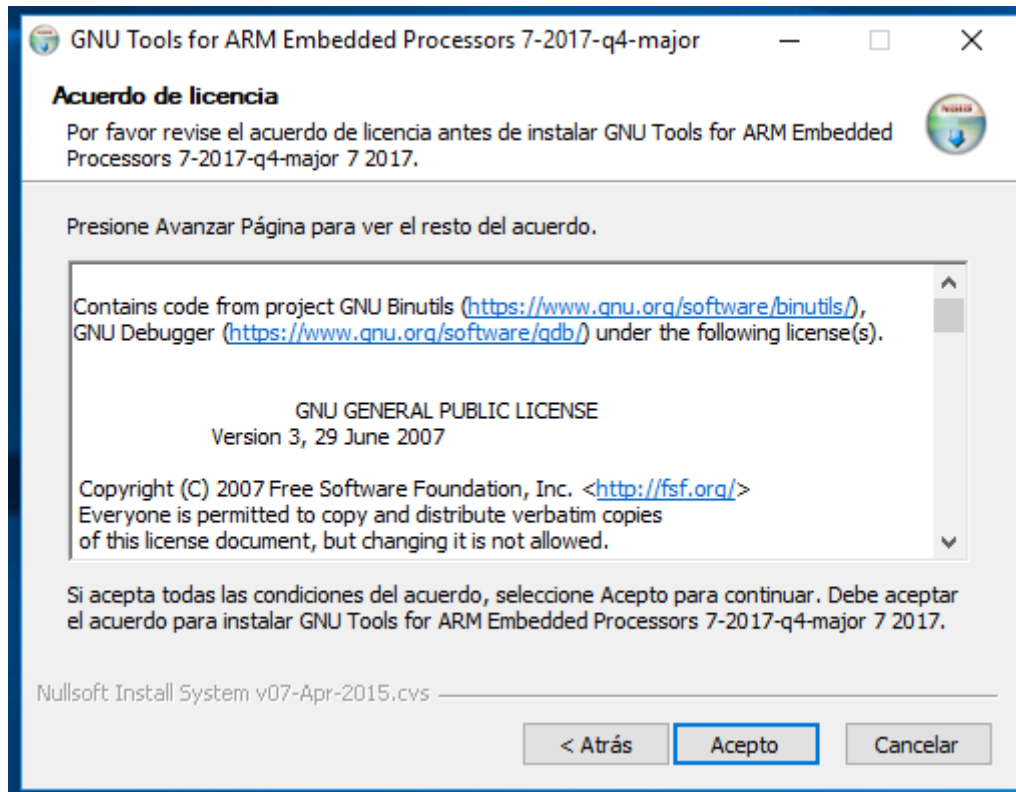
Linux 64-bit

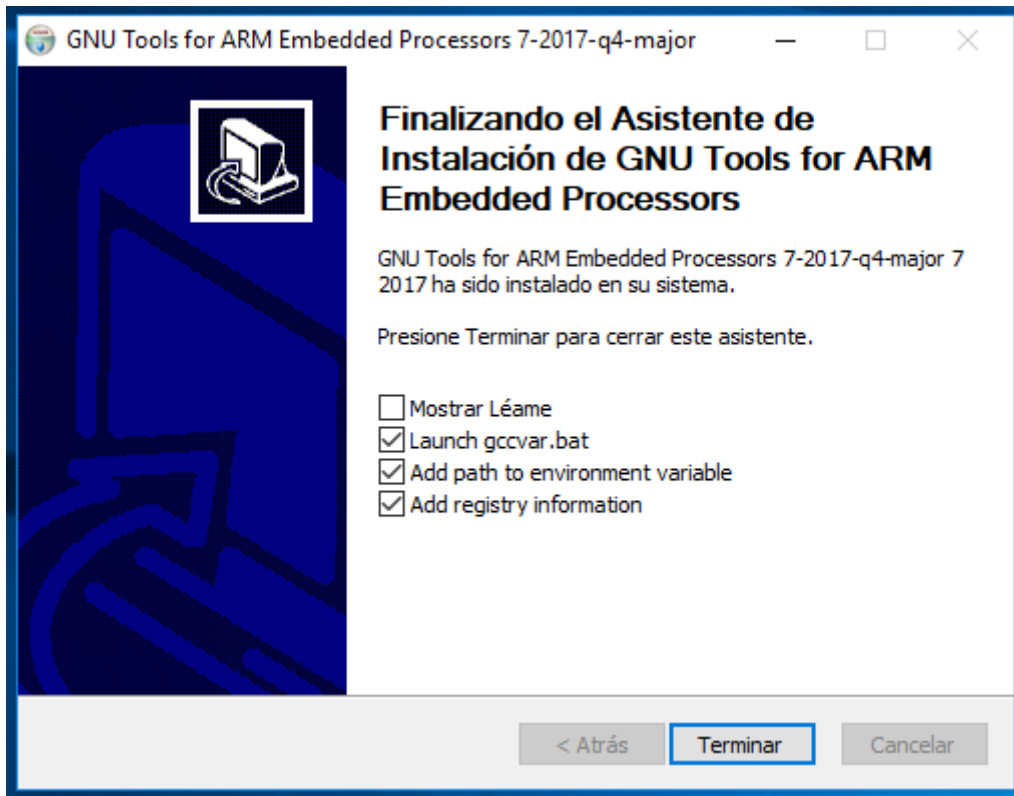
File: gcc-arm-none-eabi-7-2017-q4-major-linux.tar.bz2 (95.23 MB)

[Download](#)

Once downloaded execute the installer and follow the steps depicted in the next Figures. It is important that you add the install directory in the path environment.







Once it is installed in your system you should be able to execute it by running the `arm-none-eabi-gcc.exe --version` command from a command line, i.e. Windows PowerShell. If the installation was successful you should see a terminal like the next image.

The image shows a Windows PowerShell terminal window with a dark blue background and white text. The title bar says "Windows PowerShell". The command prompt shows the command `arm-none-eabi-gcc.exe --version` being executed. The output is as follows:

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> arm-none-eabi-gcc.exe --version
arm-none-eabi-gcc.exe (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 7
.2.1 20170904 (release) [ARM/embedded-7-branch revision 255204]
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

6.2.3 GCC Compiler for ARM Embedded Processors

<https://git-scm.com/download/win>

Downloading Git



Your download is starting...

You are downloading the latest (**2.20.1**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **about 1 month ago**, on 2018-12-15.

If your download hasn't started, [click here to download manually](#).

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

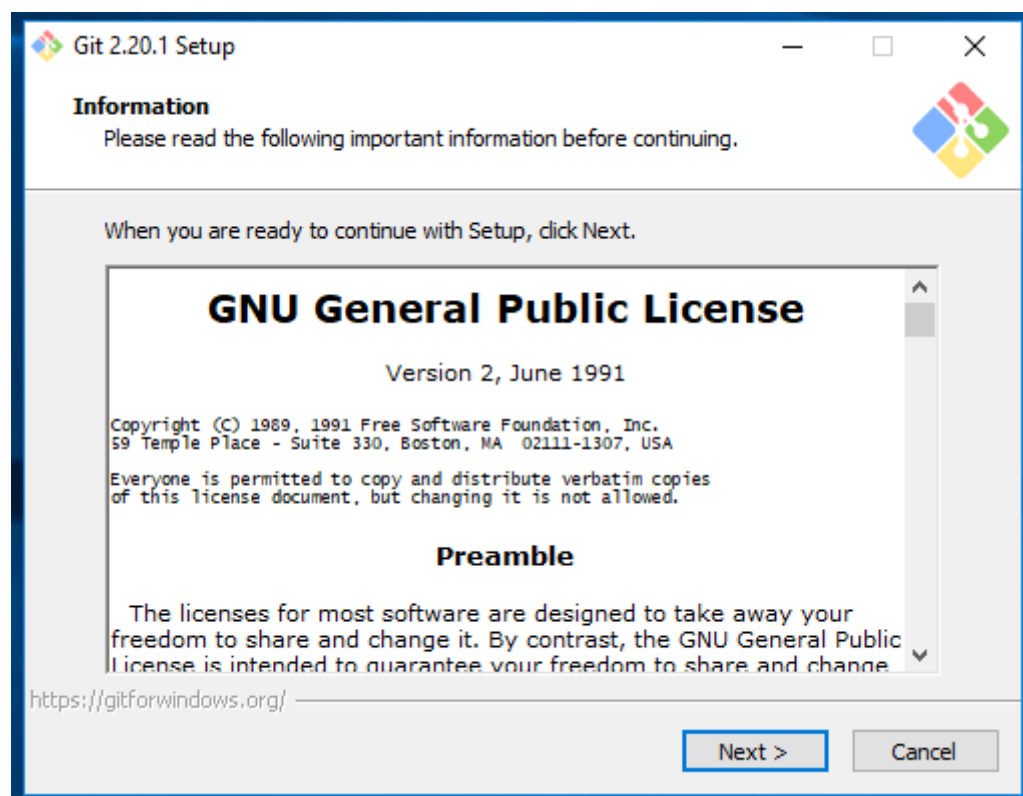
64-bit Git for Windows Setup.

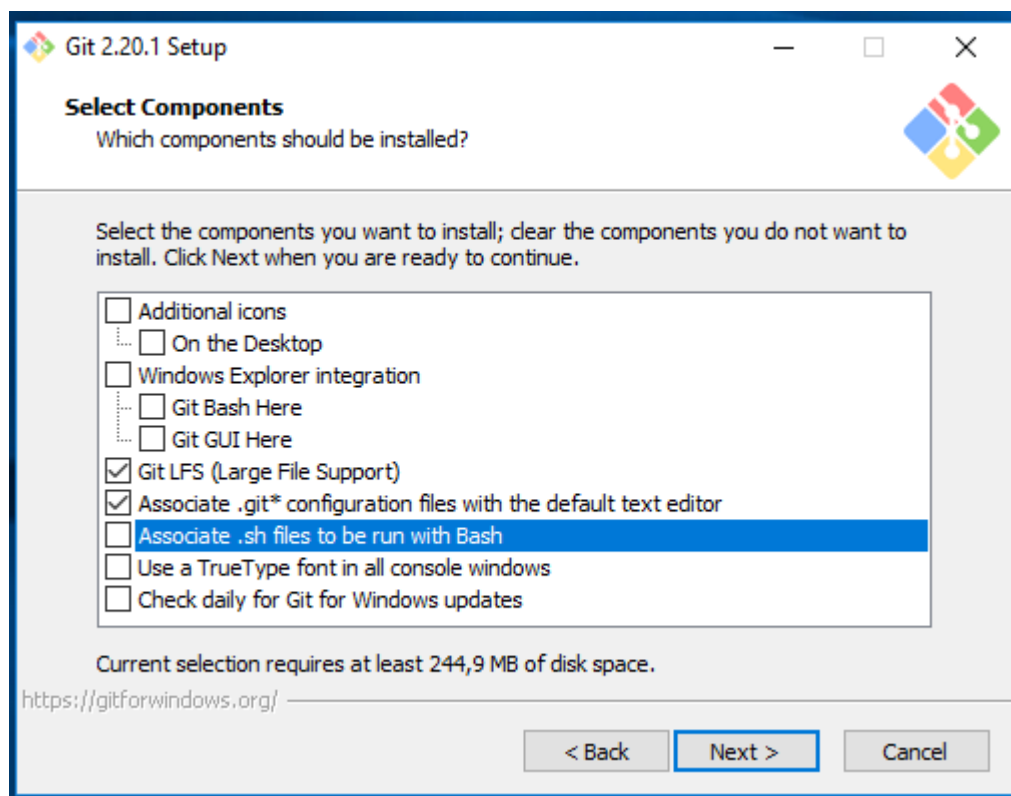
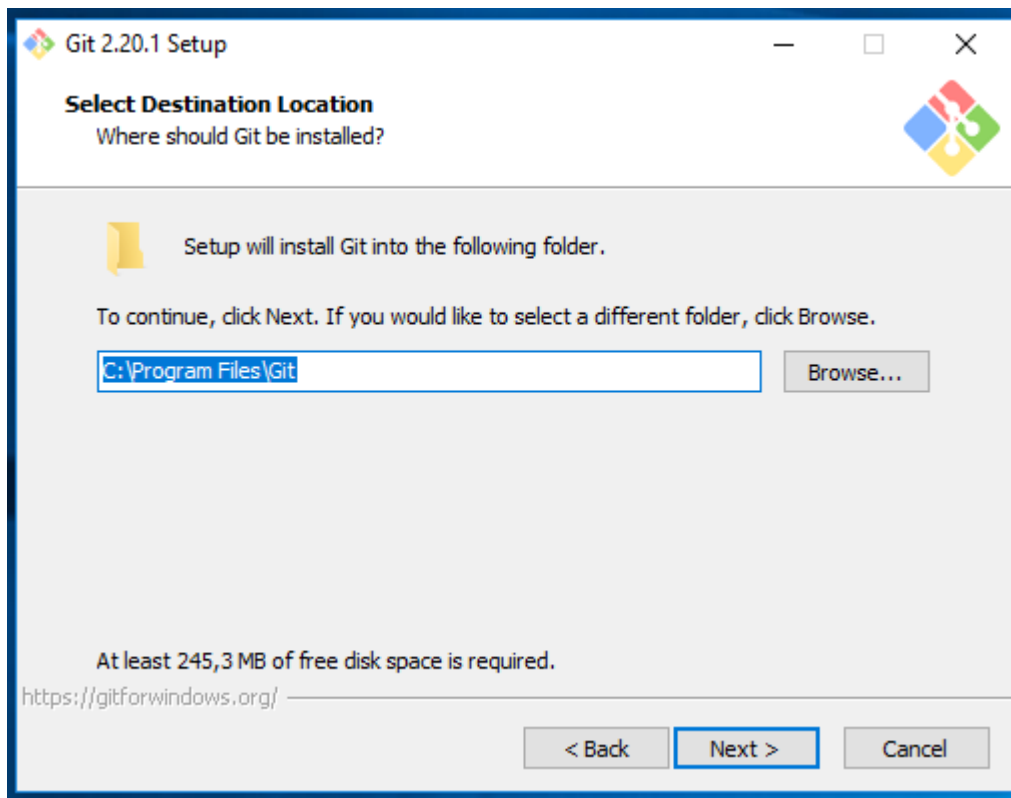
Git for Windows Portable ("thumbdrive edition")

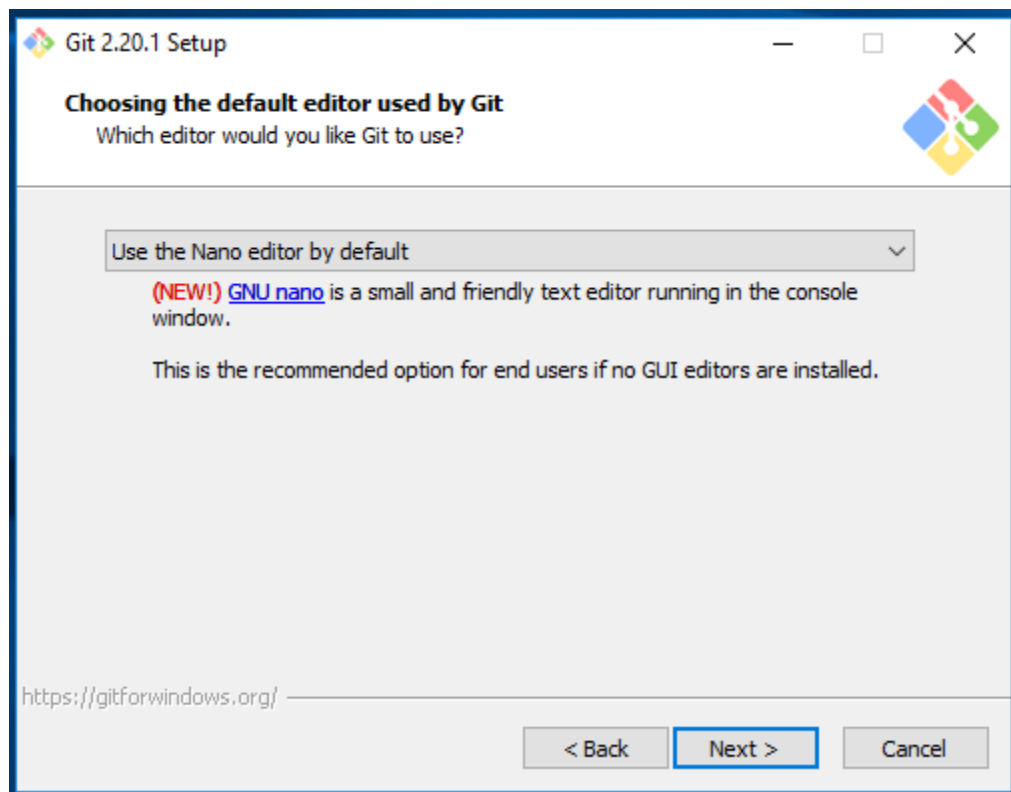
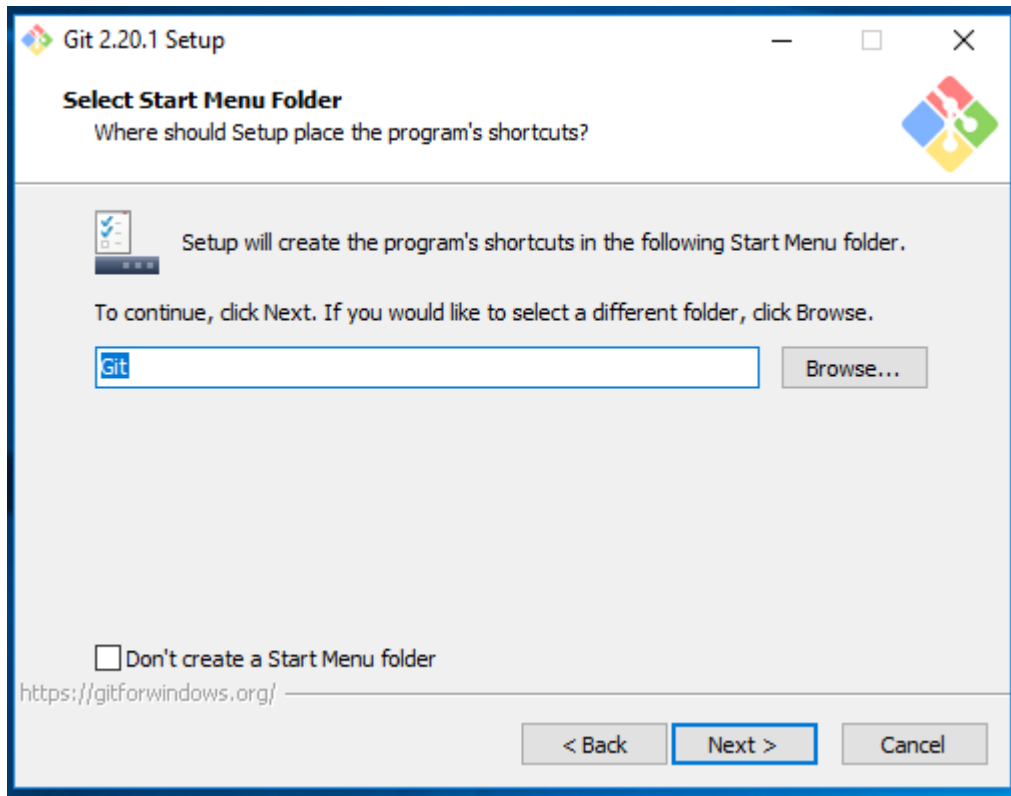
32-bit Git for Windows Portable.

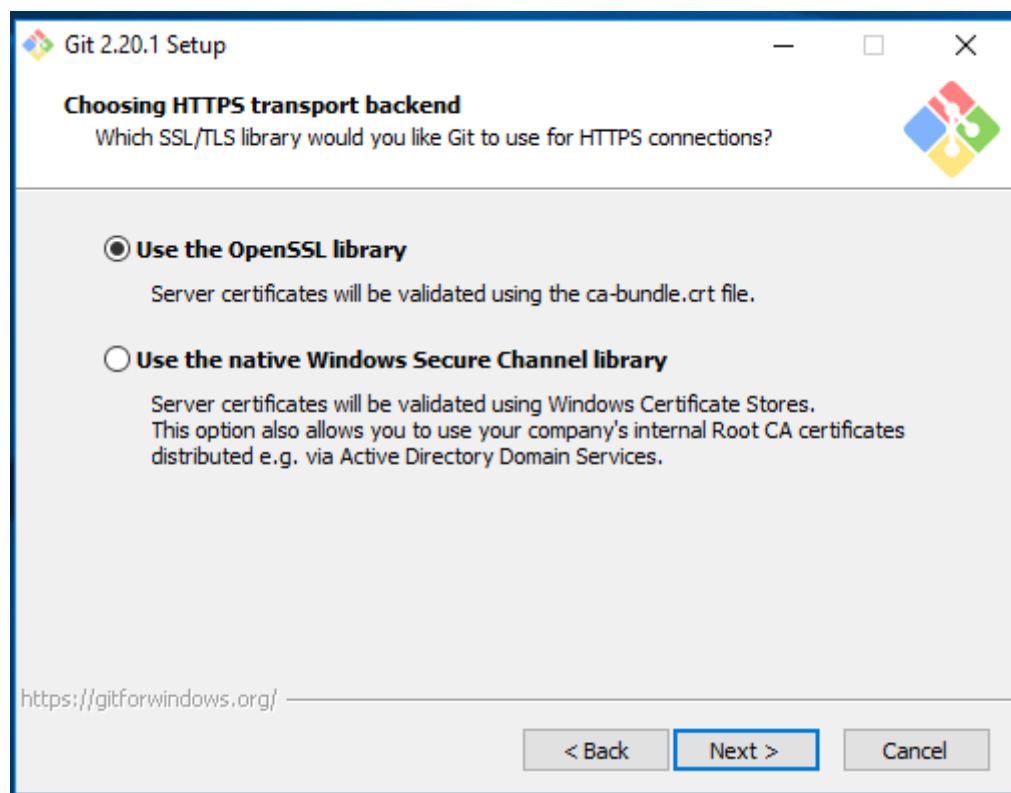
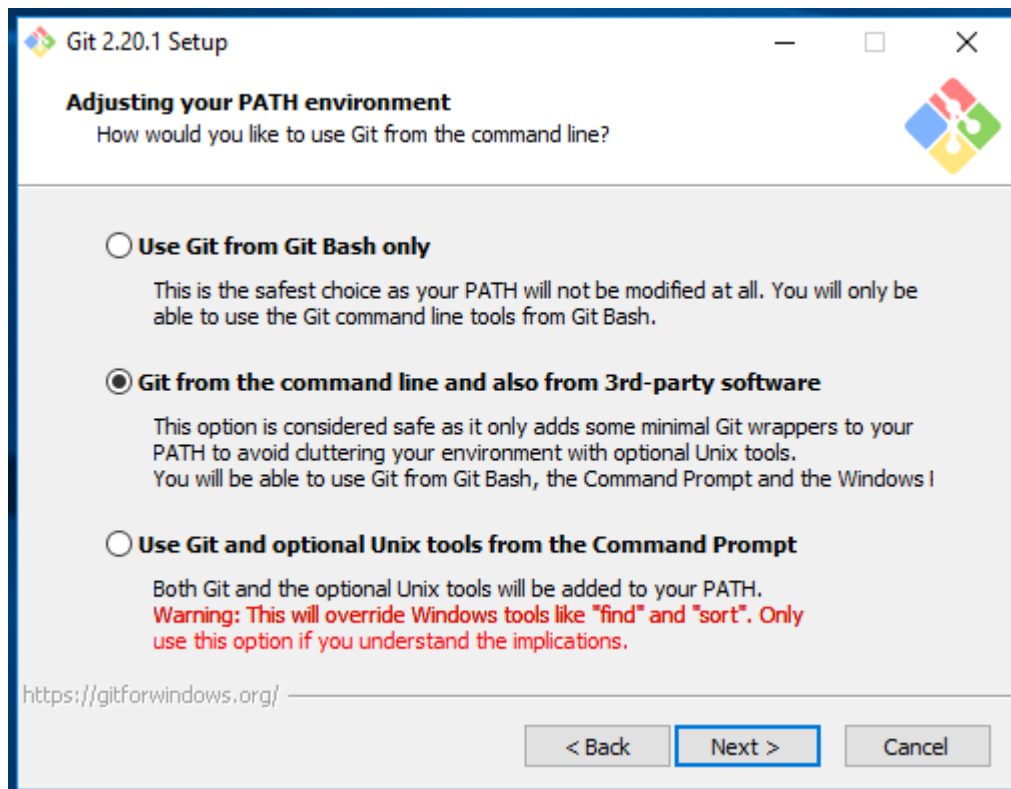
64-bit Git for Windows Portable.

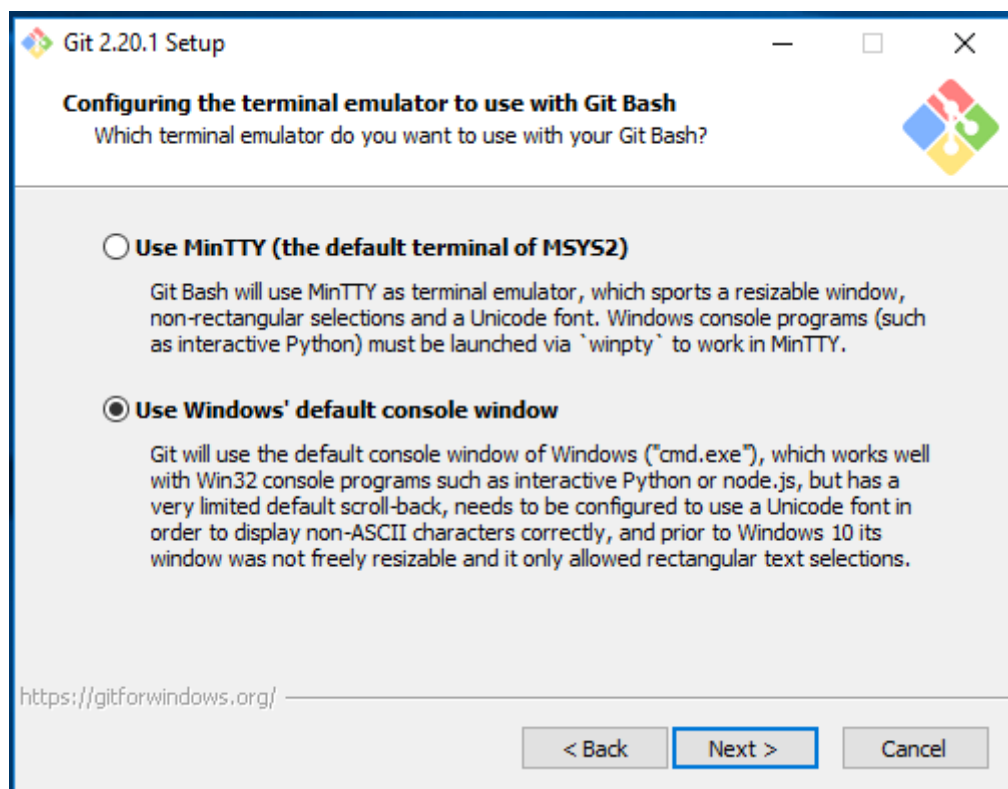
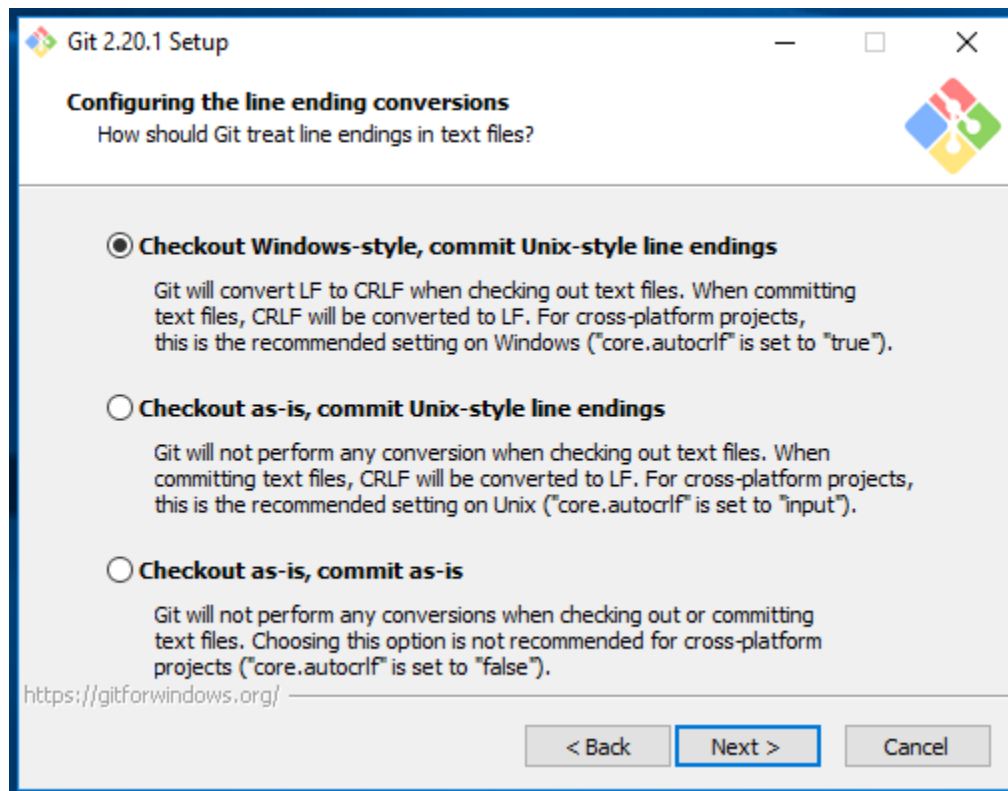
The current source code release is version **2.20.1**. If you want the newer version, you can build it from [the source code](#).

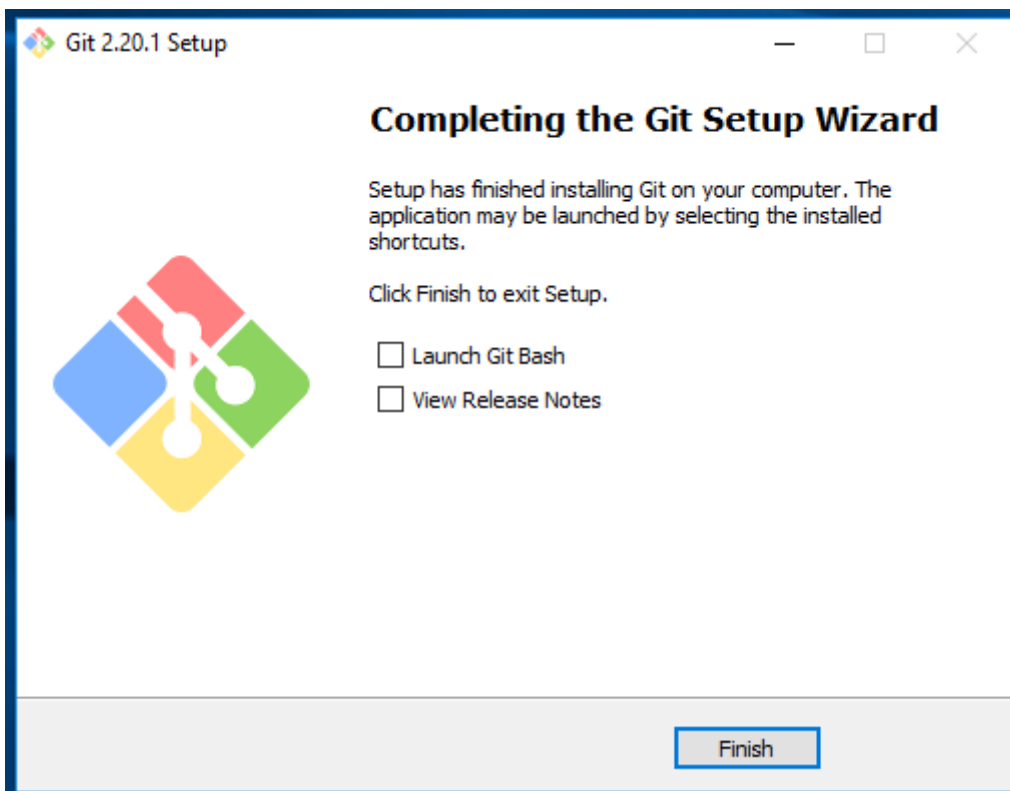
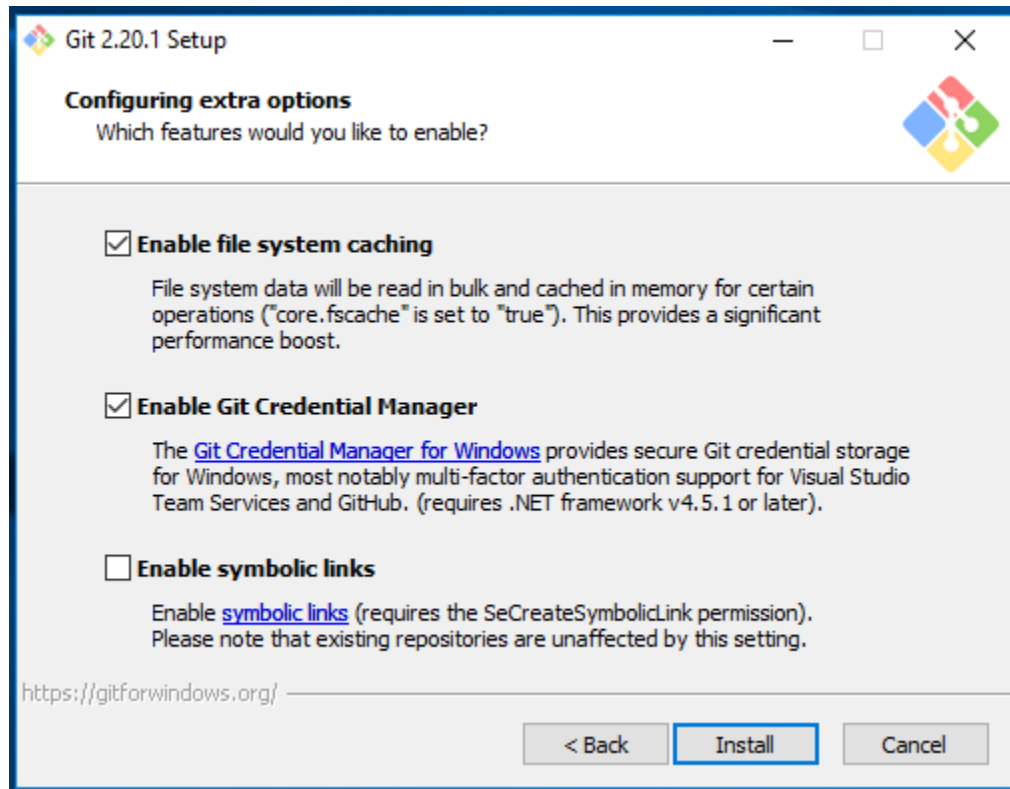


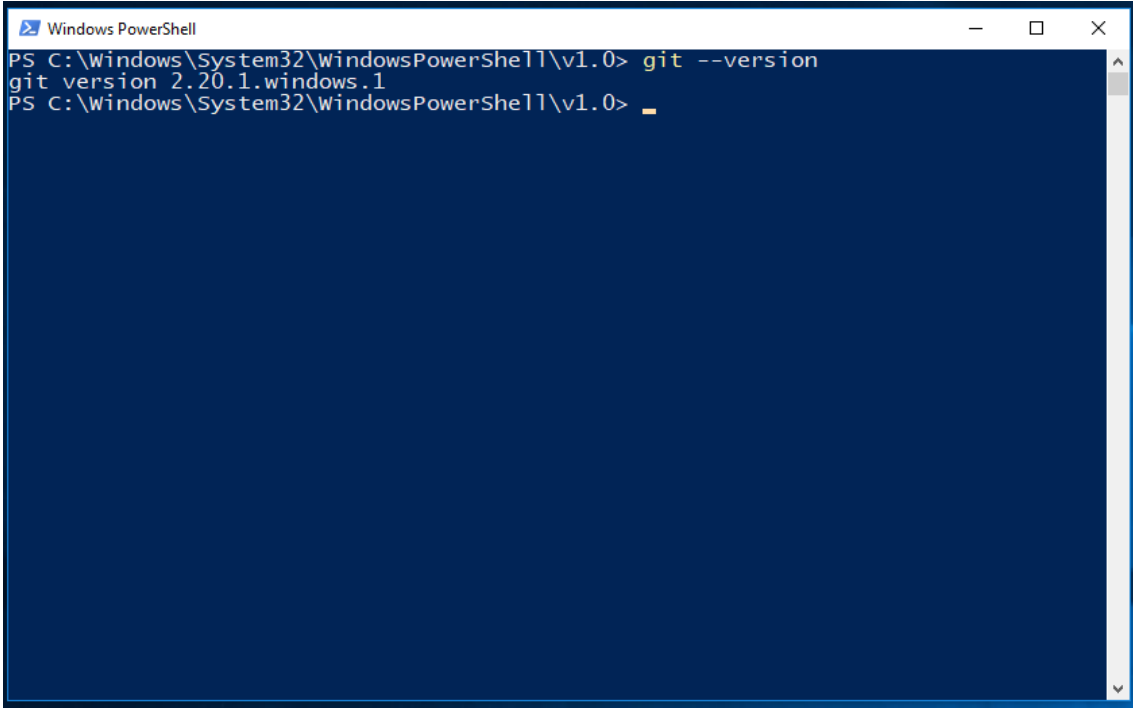












```
Windows PowerShell
PS C:\Windows\System32\WindowsPowerShell\v1.0> git --version
git version 2.20.1.windows.1
PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

6.3 GNU/Linux

The first step is installing the toolchain to compile C/C++ code for the ARM Cortex-M3 processor that is used in the Texas Instruments CC2538 micro-controller.

```
sudo add-apt-repository ppa:team-gcc-arm-embedded/ppa
```

```
sudo apt-get install gcc-arm-embedded
```

Once the toolchain is installed, we will also need some additional tools to be able to work with the source code.

```
sudo apt-get install build-essential git scons python-pip
```

Finally, we will also need to install some Python modules that will be needed

```
sudo pip install intelhex pyserial
```

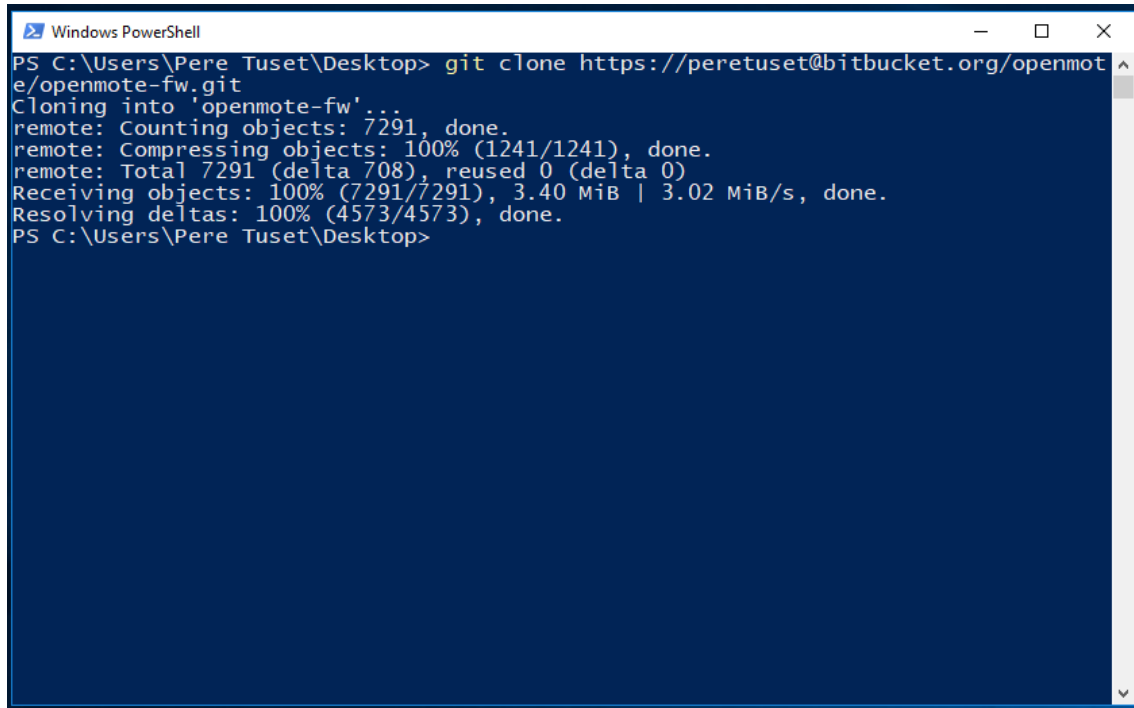
Once all the software has been properly installed in the system we can proceed to program the Open Mote B module.

6.4 OpenMote

The next sub-sections will show how to compile and execute the default OpenMote firmware projects step-by-step with Windows 10 and Ubuntu 18.04 LTS.

6.4.1 Windows 10

The first step is to clone the OpenMote repository using git. To do so, type the following command:

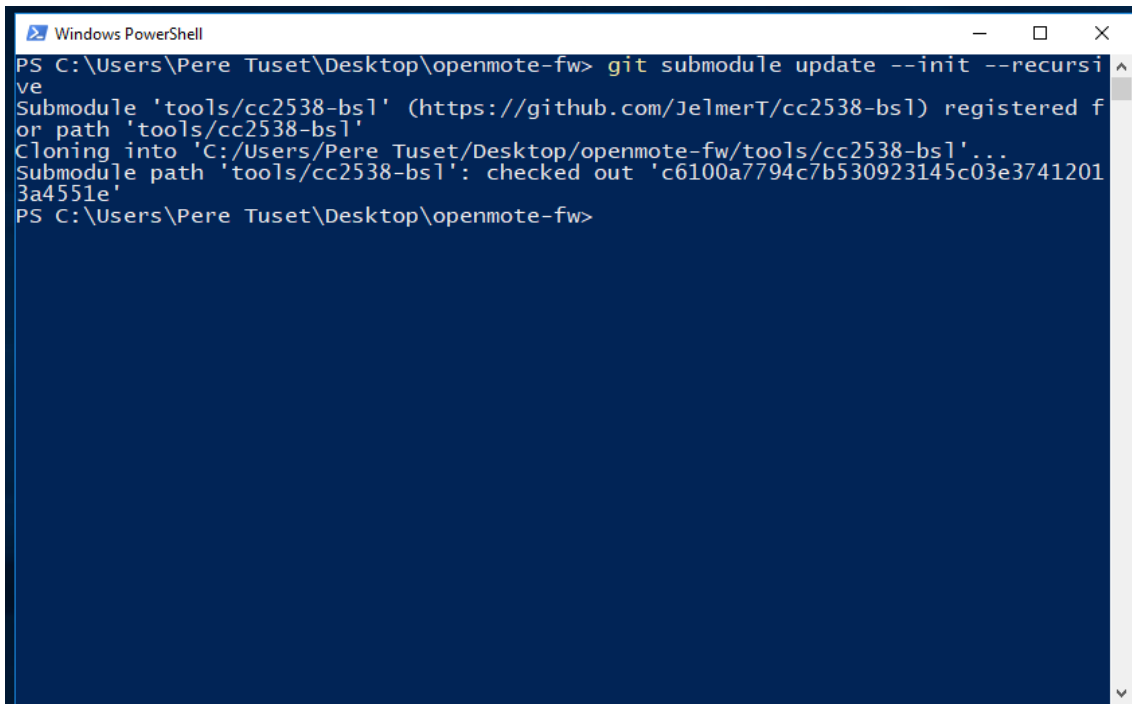
A screenshot of a Windows PowerShell terminal window. The title bar reads "Windows PowerShell". The command prompt shows the user at the C:\Users\Pere Tuset\Desktop directory. The command entered is "git clone https://peretuset@bitbucket.org/openmote/openmote-fw.git". The output shows the cloning process: "Cloning into 'openmote-fw'...", "remote: Counting objects: 7291, done.", "remote: Compressing objects: 100% (1241/1241), done.", "remote: Total 7291 (delta 708), reused 0 (delta 0)", "Receiving objects: 100% (7291/7291), 3.40 MiB | 3.02 MiB/s, done.", "Resolving deltas: 100% (4573/4573), done.", and the prompt returns to "PS C:\Users\Pere Tuset\Desktop>".

```
PS C:\Users\Pere Tuset\Desktop> git clone https://peretuset@bitbucket.org/openmote/openmote-fw.git
Cloning into 'openmote-fw'...
remote: Counting objects: 7291, done.
remote: Compressing objects: 100% (1241/1241), done.
remote: Total 7291 (delta 708), reused 0 (delta 0)
Receiving objects: 100% (7291/7291), 3.40 MiB | 3.02 MiB/s, done.
Resolving deltas: 100% (4573/4573), done.
PS C:\Users\Pere Tuset\Desktop>
```

After the OpenMote repository has been cloned, you need to initialize the repository submodules by executing the following command:

Git submodule update --init --recursive

If the command is successful you should see a screen similar to the next one.



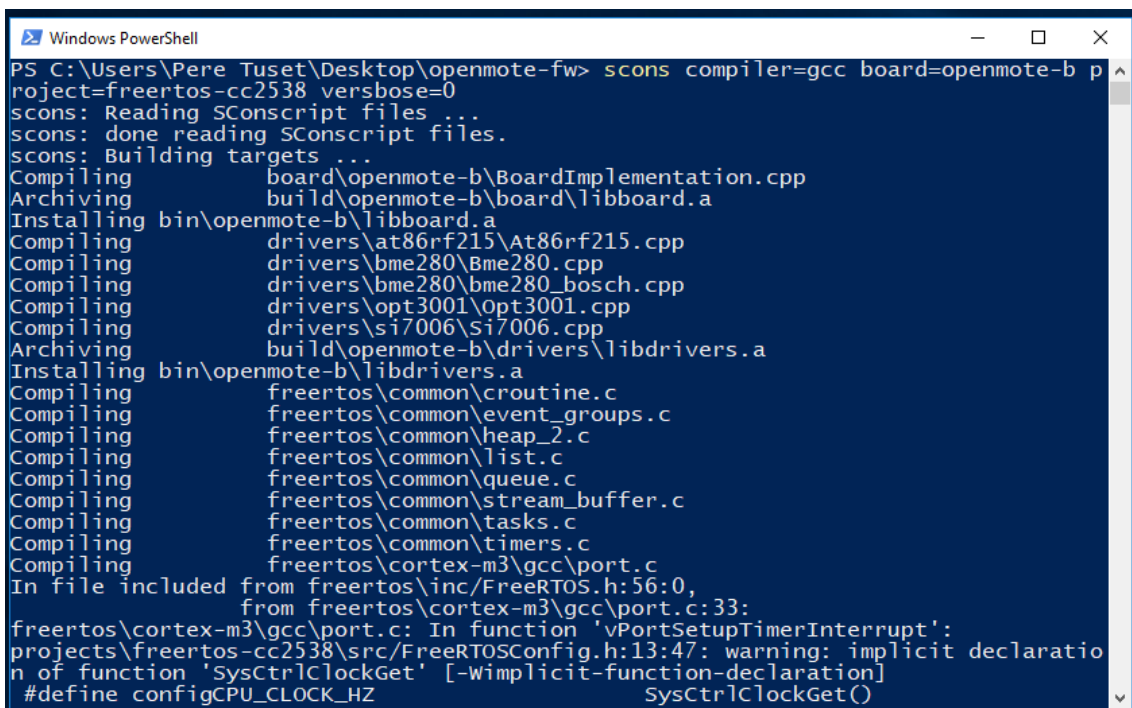
```

Windows PowerShell
PS C:\Users\Pere Tuset\Desktop\openmote-fw> git submodule update --init --recursive
Submodule 'tools/cc2538-bsl' (https://github.com/JelmerT/cc2538-bsl) registered for path 'tools/cc2538-bsl'
Cloning into 'C:/Users/Pere Tuset/Desktop/openmote-fw/tools/cc2538-bsl'...
Submodule path 'tools/cc2538-bsl': checked out 'c6100a7794c7b530923145c03e37412013a4551e'
PS C:\Users\Pere Tuset\Desktop\openmote-fw>

```

Once the initializations are made you can compile a project by the next command:

scons compiler=gcc board=openmote-b project=freertos-cc2538 verbose=0



```

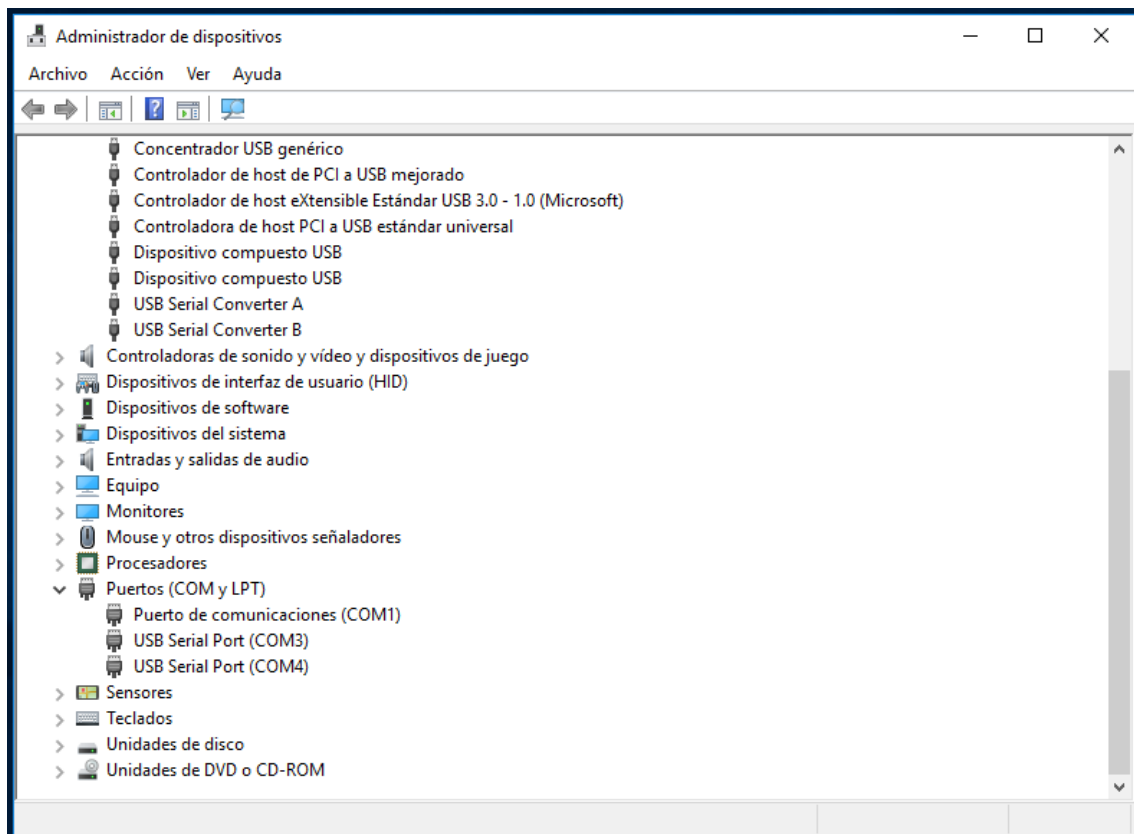
Windows PowerShell
PS C:\Users\Pere Tuset\Desktop\openmote-fw> scons compiler=gcc board=openmote-b project=freertos-cc2538 verbose=0
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
Compiling board\openmote-b\BoardImplementation.cpp
Archiving build\openmote-b\board\libboard.a
Installing bin\openmote-b\libboard.a
Compiling drivers\at86rf215\At86rf215.cpp
Compiling drivers\bme280\Bme280.cpp
Compiling drivers\bme280\bme280_bosch.cpp
Compiling drivers\opt3001\Opt3001.cpp
Compiling drivers\si7006\Si7006.cpp
Archiving build\openmote-b\drivers\libdrivers.a
Installing bin\openmote-b\libdrivers.a
Compiling freertos\common\croutine.c
Compiling freertos\common\event_groups.c
Compiling freertos\common\heap_2.c
Compiling freertos\common\list.c
Compiling freertos\common\queue.c
Compiling freertos\common\stream_buffer.c
Compiling freertos\common\tasks.c
Compiling freertos\common\timers.c
Compiling freertos\cortex-m3\gcc\port.c
In file included from freertos\inc\FreeRTOS.h:56:0,
                 from freertos\cortex-m3\gcc\port.c:33:
freertos\cortex-m3\gcc\port.c: In function 'vPortSetupTimerInterrupt':
projects\freertos-cc2538\src\FreeRTOSConfig.h:13:47: warning: implicit declaration of function 'SysCtrlClockGet' [-Wimplicit-function-declaration]
#define configCPU_CLOCK_HZ SysCtrlClockGet()

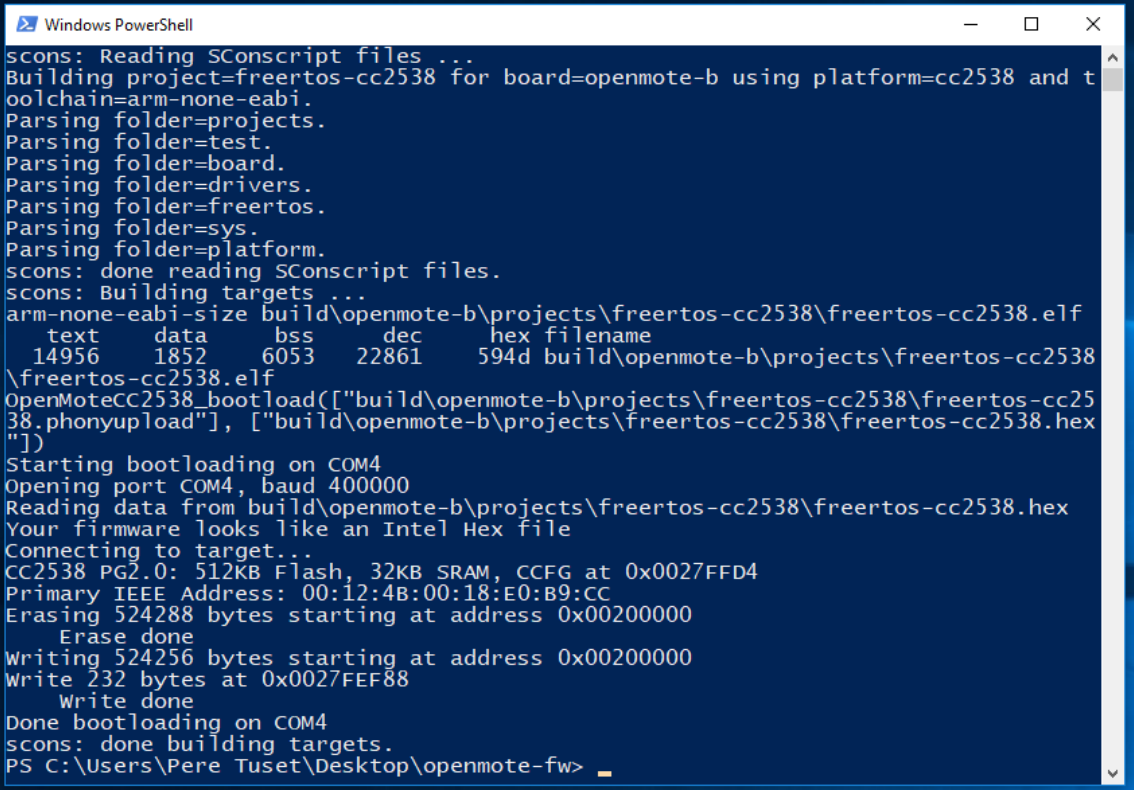
```

```

Windows PowerShell
Compiling platform\cc2538\cc2538_startup_gcc.c
Archiving build\openmote-b\platform\libplatform.a
Installing bin\openmote-b\libplatform.a
Compiling sys\src\Buffer.cpp
Compiling sys\src\CriticalSection.cpp
Compiling sys\src\Crc16.cpp
Compiling sys\src\Hdlc.cpp
Compiling sys\src\LedBlinker.cpp
Compiling sys\src\Mutex.cpp
Compiling sys\src\Rendezvous.cpp
Compiling sys\src\Scheduler.cpp
Compiling sys\src\Semaphore.cpp
Compiling sys\src\Serial.cpp
Compiling sys\src\Task.cpp
Archiving build\openmote-b\sys\libsys.a
Installing bin\openmote-b\libsys.a
Compiling projects\freertos-cc2538\src\main.cpp
Linking build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf
arm-none-eabi-objcopy --output-target=binary build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf build\openmote-b\projects\freertos-cc2538\freertos-cc2538.bin
arm-none-eabi-objcopy --output-target=ihex build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf build\openmote-b\projects\freertos-cc2538\freertos-cc2538.hex
arm-none-eabi-size build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf
   text    data     bss     dec     hex filename
  14956    1852     6053   22861   594d build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf
scons: done building targets.
PS C:\Users\Pere Tuset\Desktop\openmote-fw>

```





```

Windows PowerShell
scons: Reading SConscript files ...
Building project=freertos-cc2538 for board=openmote-b using platform=cc2538 and toolchain=arm-none-eabi.
Parsing folder=projects.
Parsing folder=test.
Parsing folder=board.
Parsing folder=drivers.
Parsing folder=freertos.
Parsing folder=sys.
Parsing folder=platform.
scons: done reading SConscript files.
scons: Building targets ...
arm-none-eabi-size build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf
   text    data     bss       dec       hex filename
  14956    1852     6053    22861    594d build\openmote-b\projects\freertos-cc2538\freertos-cc2538.elf
OpenMoteCC2538_bootload(["build\openmote-b\projects\freertos-cc2538\freertos-cc2538.phonyupload"], ["build\openmote-b\projects\freertos-cc2538\freertos-cc2538.hex"])
Starting bootloading on COM4
Opening port COM4, baud 400000
Reading data from build\openmote-b\projects\freertos-cc2538\freertos-cc2538.hex
Your firmware looks like an Intel Hex file
Connecting to target...
CC2538 PG2.0: 512KB Flash, 32KB SRAM, CCFG at 0x0027FFD4
Primary IEEE Address: 00:12:4B:00:18:E0:B9:CC
Erasing 524288 bytes starting at address 0x00200000
Erase done
Writing 524256 bytes starting at address 0x00200000
Write 232 bytes at 0x0027FEF88
Write done
Done bootloading on COM4
scons: done building targets.
PS C:\Users\Pere Tuset\Desktop\openmote-fw>

```

6.4.2 GNU/Linux

The first step is to clone the OpenMote firmware repository from BitBucket using the following command:

```
git clone https://peretuset@bitbucket.org/openmote/openmote-fw.git
```

```

openmote@openmote:~/Desktop$ git clone https://peretuset@bitbucket.org/openmote/openmote-fw.git
Cloning into 'openmote-fw'...
Password for 'https://peretuset@bitbucket.org':
remote: Counting objects: 6880, done.
remote: Compressing objects: 100% (847/847), done.
remote: Total 6880 (delta 454), reused 0 (delta 0)
Receiving objects: 100% (6880/6880), 2.91 MiB | 2.44 MiB/s, done.
Resolving deltas: 100% (4319/4319), done.

```

```
cd openmote-fw
```

```
git submodule update --init --recursive
```

```
openmote@openmote:~/Desktop/openmote-fw$ git submodule update --init --recursive
Submodule 'tools/cc2538-bsl' (https://github.com/JelmerT/cc2538-bsl) registered
for path 'tools/cc2538-bsl'
Cloning into '/home/openmote/Desktop/openmote-fw/tools/cc2538-bsl'...
Submodule path 'tools/cc2538-bsl': checked out 'c6100a7794c7b530923145c03e374120
13a4551e'
```

```
cd ti/cc2538/gcc
```

```
make
```

```
openmote@openmote:~/Desktop/openmote-fw/ti/cc2538/gcc$ make
Building 'libdriverlib' library...
Compiling ../../src/aes.c...
In file included from ../../src/aes.c:47:0:
../../src/aes.h:112:20: warning: implicit declaration of function 'asm' [-Wimplicit-function-declaration]
#define ASM_NOP      asm("NOP")
                      ^
../../src/aes.c:140:9: note: in expansion of macro 'ASM_NOP'
    ASM_NOP;
    ^~~~~~
Compiling ../../src/systick.c...
Compiling ../../src/pka.c...
Compiling ../../src/interrupt.c...
Compiling ../../src/adc.c...
Compiling ../../src/sys_ctrl.c...
Compiling ../../src/ioc.c...
Compiling ../../src/cpu.c...
Compiling ../../src/ccm.c...
```

After generating the driverlib for the Texas Instruments CC2538 micro-controller, you can return to the root directory using the 'cd -' command.

6.4.3 Blinking LED example

To test the build environment and the boards we will use the freertos-cc2538 project, which demonstrates using the FreeRTOS real-time operating system with the Texas Instruments CC2538 micro-controller on the OpenMote-B board. The code will be compiled with the gcc-arm-embedded toolchain. To compile the code issue the following command:

```
scons board=openmote-b project=freertos-cc2538 compiler=gcc verbose=0
```

The compile process for all the project modules will start, as displayed in the next figure.

```

openmote@openmote:~/Desktop/openmote-fw$ scon board=openmote-b project=freertos
-cc2538 compiler=gcc verbose=0
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
Compiling      board/openmote-b/BoardImplementation.cpp
Archiving      build/openmote-b/board/libboard.a
Indexing      build/openmote-b/board/libboard.a
Installing bin/openmote-b/libboard.a
Compiling      drivers/at86rf215/At86rf215.cpp
Compiling      drivers/bme280/Bme280.cpp
Compiling      drivers/bme280/bme280_bosch.cpp
Compiling      drivers/opt3001/Opt3001.cpp
Compiling      drivers/si7006/Si7006.cpp
Archiving      build/openmote-b/drivers/libdrivers.a
Indexing      build/openmote-b/drivers/libdrivers.a
Installing bin/openmote-b/libdrivers.a

```

If the process is correct the build system will display a successful message and will show the space occupied by the generated binary image, as depicted in the next figure.

```

Linking      build/openmote-b/projects/freertos-cc2538/freertos-cc2538
arm-none-eabi-objcopy --output-target=binary build/openmote-b/projects/freertos-
cc2538/freertos-cc2538 build/openmote-b/projects/freertos-cc2538/freertos-cc2538
.bin
arm-none-eabi-objcopy --output-target=ihex build/openmote-b/projects/freertos-cc
2538/freertos-cc2538 build/openmote-b/projects/freertos-cc2538/freertos-cc2538.h
ex
arm-none-eabi-size build/openmote-b/projects/freertos-cc2538/freertos-cc2538
  text    data    bss     dec     hex filename
 12664    2364    6129   21157   52a5 build/openmote-b/projects/freertos-cc253
8/freertos-cc2538
scons: done building targets.

```

Once the binary image has been generated, we can upload it to the OpenMote-B board using the bootloader and the cc2538-bsl Python script. To upload the firmware, issue the following command:

```
sudo scon board=openmote-b project=freertos-cc2538 compiler=gcc verbose=0 bootload=/dev/ttyUSB1
```

Notice that the command must be prepended with “sudo”, as accessing the serial port in Linux requires super-user permissions.

If the process is successful, the build system will display a success message, as depicted in the next image, and the OpenMote-B should start working: the green LED should blink briefly (100 ms) every second (1000 ms) and the user button should toggle the red LED.

```
openmote@openmote:~/Desktop/openmote-fw$ sudo scons board=openmote-b project=freertos-cc2538 compiler=gcc verbose=0 bootload=/dev/ttyUSB1
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
arm-none-eabi-size build/openmote-b/projects/freertos-cc2538/freertos-cc2538
      text    data     bss     dec     hex filename
    12664    2364     6129    21157    52a5 build/openmote-b/projects/freertos-cc2538/freertos-cc2538
OpenMoteCC2538_bootload(["build/openmote-b/projects/freertos-cc2538/freertos-cc2538.phonyupload"], ["build/openmote-b/projects/freertos-cc2538/freertos-cc2538.hex"])
Starting bootloading on /dev/ttyUSB1
Done bootloading on /dev/ttyUSB1
scons: done building targets.
```

7 Revision Table

Revision Number	Date	Changes
0	22/08/2019	First implementation

About Industrial Shields:

Address:

Fàbrica del Pont, 1-11
Sant Fruitós de Bages, 08272 (Barcelona)
Spain

Telephone:

(+34) 938 760 191 / (+34) 635 693 611

Mail:

info@industrialshields.com