

POLITECNICO DI TORINO

SCUOLA DI INGEGNERIA
LAUREA MAGISTRALE IN CYBERSECURITY

Computer architectures and operating systems



Members of group 6:

Bertè Simona	s331981
Di Marco Davide	s332405
Oldani Edoardo	s332036
Polizzi Valeria	s330253
Scamardo Giovanni	s332372

ACADEMIC YEAR 2023 / 2024

Indice

1	Guide	1
1.1	Windows	1
1.1.1	Prerequisite	1
1.1.2	Download and install QEMU	2
1.1.3	Install an Eclipse	3
1.1.4	Install the GCC Toolchain	3
1.1.5	Install Make	4
1.1.6	Install FreeRTOS	5
1.1.7	Manage the PATH Environment variable	6
1.1.8	Import, Build and Execute a Demo Project in Eclipse	6
1.2	Ubuntu	8
1.2.1	Prerequisites	8
1.2.2	Download and install QEMU	8
1.2.3	Install Eclipse	8
1.2.4	Install GCC Toolchain	9
1.2.5	Install FreeRTOS	10
1.2.6	Manage the PATH Environment variable	10
1.2.7	Import, Build and Execute a Demo Project in Eclipse	12

Capitolo 1

Guide

QEMU stands for “Quick Emulator” and it is a popular open-source emulator that can be used for debugging and testing embedded software without the need for actual hardware. The purpose of this guide is to explain how to properly install and use it to run an Embedded OS in both Windows and Ubuntu. For this purpose, FreeRTOS will be used as our Embedded Operating System. In particular, FreeRTOS ARM Cortex-M3 GCC port on the MPS2 QEMU Demo was adopted for the development of our project.

1.1 Windows

1.1.1 Prerequisite

If Windows is not the native operating system of your machine, you need to install the latest Windows update (Windows 10) in a Virtual Machine.

Below there is a description of the steps to be taken.

- Install a tool for virtualizing computing architecture. This guide uses VirtualBox, which is a tool for virtualizing x86 and AMD64/Intel64 computing architecture. To use it download the VirtualBox installer for your opera-

ting system in the official web site <https://www.virtualbox.org/wiki/Downloads>, run it and define the installation options.

- Download the Windows 10 ISO following this link <https://www.microsoft.com/en-us/software-download/windows10ISO>.
- Create a new virtual machine in Virtual Box pressing the **New** button. Set **Type** to "Microsoft Windows and **Version** to "Windows 10". Make sure you match the x64 version with a 64-bit VM, and the x86 version with a 32-bit VM.
- Allocate RAM. Make sure you stay in the green part, otherwise, if you allocate too much RAM, you'll end up with serious performance issues.
- Create a virtual drive. Microsoft says that 16GB is the minimum space needed for the 32-bit version, but 20GB is required for the 64-bit version.
- Go into the settings for this virtual machine and navigate to the **Storage** tab. Click the disc icon with a green plus next to **Controller: SATA**. Click **Choose disk** and then locate the Windows 10 ISO you downloaded earlier.
- Press the **Start** button in VirtualBox, and begin the Windows 10 installation process. Follow the instructions on the screen.
- Once you're at the Windows 10 desktop, you'll need to install all of the proper drivers for VirtualBox. In the VirtualBox UI, go to **Devices**, and then select **Insert Guest Additions CD image**. Navigate to that disc image in Windows Explorer, and run the installer. Once you've gone through the entire process, you'll need to reboot the VM.

1.1.2 Download and install QEMU

Download and install the pre-built QEMU related binaries for 64 bit versions of Microsoft Windows. To do that following this link <https://qemu.weilnetz.de/w64/> and download QEMU.



[qemu-w64-setup-20231214.exe](#)

2023-12-14 13:38 159M QEMU Installer for Windows (64 bit)

1.1.3 Install an Eclipse

The project works with the base Eclipse version with Eclipse Embedded CDT plug-ins. To import and build the demo project follow the description below of the steps to be taken.

- Eclipse is a Java-based application. Current Windows distributions do not include Java, therefore to run Eclipse you need to install the Java Runtime Environment (JRE) on your system. Downloads the **x64 MSI Installer** from the official Oracle OpenJDK page <https://www.oracle.com/it/java/technologies/downloads/>. Once the file is downloaded, in a cmd.exe terminal check the version with the command `java --version`.
- Download the Eclipse Embedded CDT (C/C++ Development Tools) for Windows x86_64 following this link <https://www.eclipse.org/downloads/packages/> and extract it.



1.1.4 Install the GCC Toolchain

Download and install GNU Arm Embedded Toolchain. To do that following this link <https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads>; downloads the toolchain for Windows host **arm-gnu-toolchain-13.2.rel1-mingw-w64-i686-arm-none-eabi.zip**, which is available for x86 host architecture only but also compatible with x86_64. Once the file is downloaded, extract it.

1.1.5 Install Make

Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files. Make gets its knowledge of how to build your program from a file called the makefile, which lists each of the non-source files and how to compute it from other files. When you write a program, you should write a makefile for it, so that it is possible to use Make to build and install the program.

To install Make for Windows use Chocolatey. Chocolatey is a software management solution that gives the freedom to create a simple software package and then deploy it anywhere there is Windows using any configuration or system management tools.

First you need to install Chocolatey package manager. To do that is necessary that you satisfy these requirements:

- Supported Windows client and server Operating Systems.
- PowerShell v2+.
- .NET Framework 4.8

First, ensure that you are using an **administrative shell**.

- Type the **PowerShell** in the search field.
- Right-click on the Windows PowerShell and then select the **run as administrator**. It will open a Windows PowerShell which will run as an administrator.

You can also run PowerShell as an Administrator from Command prompt

- Open the Command Prompt, and type the **Powershell** as a command, then press **Enter key**.
- Now, the command prompt will turn to Windows PowerShell.

- Type the command `start-process PowerShell -verb runas` and press **Enter** key. It will bring up an elevated Windows PowerShell as an administrator.

With PowerShell, you must ensure **Get-ExecutionPolicy** is not **Restricted**. We suggest using **Bypass** to bypass the policy to get things installed or **AllSigned** for quite a bit more security.

Run:

```
1 Get-ExecutionPolicy
```

If it returns **Restricted** then run

```
1 Set-ExecutionPolicy AllSigned
```

or

```
1 Set-ExecutionPolicy Bypass -Scope Process
```

Now run the following command into your shell:

```
1 Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.
  ServicePointManager]::SecurityProtocol = [System.Net.
  ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-
  Object System.Net.WebClient).DownloadString('https://community.
  chocolatey.org/install.ps1'))
```

Wait a few seconds for the command to complete and, if you don't see any errors, you are ready to use Chocolatey.

Once installed Chocolatey you simply need to install GNU make, to do that run the following command from the command line or from PowerShell (you may need to run it in an elevated/admin command prompt):

```
1 choco install make
```

1.1.6 Install FreeRTOS


Download the latest FreeRTOS packages: <https://www.freertos.org/index.html> and unzip it.

1.1.7 Manage the PATH Environment variable

We have a need to pass environment variable assignments containing commas to QEMU, GNU Arm Embedded Toolchain, Java, chocolately. The base Eclipse version relies on setting the environment variable PATH to reach the toolchain binaries. The PATH can be set from:

- the system global PATH setting or a per user PATH setting;
- the Eclipse workspace's common settings for all projects;
- the project's build configuration.

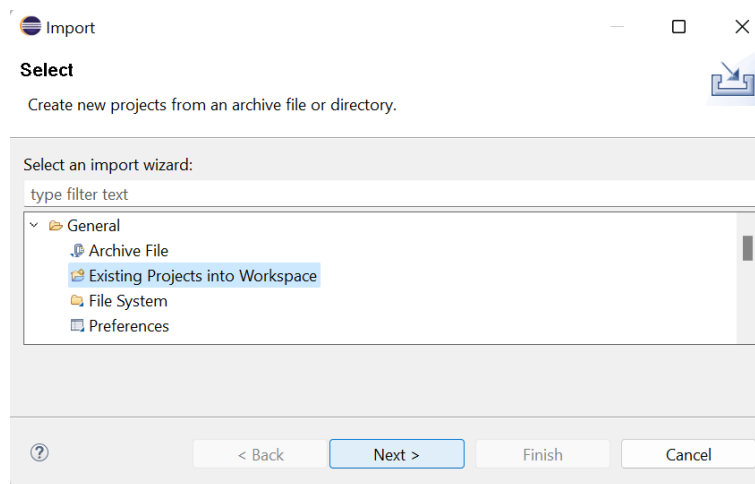
In this guide we set the PATH from the system global PATH setting. Make sure to have the global PATH as in the picture below.



```
C:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2021.10\bin;
C:\Program Files\qemu;
C:\Program Files\Java\jdk-21\bin;
C:\ProgramData\chocolatey\bin
```

1.1.8 Import, Build and Execute a Demo Project in Eclipse

- Start Eclipse, then select an existing, or create a new workspace when prompted.
- Select **Import** from the Eclipse **File** menu. The **Import** dialog box will open.
- In the Import dialog box, select **General** and then **Existing Project into Workspace** before clicking the **Next** button. The Import Projects dialog box will open.



- In the Import Projects dialog box, select the directory written below as the root directory and crucially ensure the **Copy projects into workspace** checkbox is unchecked, before clicking the **Finish** button to bring the project into Eclipse.

/FreeRTOS/Demo/CORTEX_MPS2_QEMU_IAR_GCC/build/gcc

- Open the *main.c* file, and set *mainCREATE_SIMPLE_BLINKY_DEMO_ONLY* to generate either the simply blinky demo, or the comprehensive test and demo application, as required.
- Select **Build All** from the Eclipse **Project** menu. A successful build creates the elf file.
- Open a command prompt then start QEMU with the following command line, replacing [path-to] with the correct path to the RTOSDemo.out file generated by the GCC build.

```
1 > qemu-system-arm -machine mps2-an385 -cpu cortex-m3 -kernel
    [path-to]/RTOSDemo.out -monitor none -nographic -serial
    stdio
```

1.2 Ubuntu

1.2.1 Prerequisites

The prerequisites to install **QEMU** on Ubuntu are:

- Having a running Ubuntu 23.10. This can be easily downloaded by clicking on the ‘**Start Download**’ button, making sure to select the 23.10 version, following this link: www.ubuntu-it.org/download.
- Sync all the installed packages with the latest available version, this decreases the chances of downloading insecure and incompatible software. This can be done by using the following command:

```
1 $ sudo apt update && sudo apt upgrade
```

After inserting that command on the bash, type **Y** or **Yes** when prompted for permission and let the system upgrade finish up.

1.2.2 Download and install QEMU

The easiest way to install QEMU is using the following command:

```
1 $ sudo apt install qemu-system
```

Note: this may not install the latest version of QEMU. To make sure to have the most recent version, download it from the official website by simply following the **Build Instructions** commands.

After the installation, it is possible to verify the correct installation of the qemu metapackage by inserting the command below:

```
1 $ dpkg --get-selections | grep qemu
```

1.2.3 Install Eclipse

As explained in the introduction, QEMU can be used for running an embedded operating system such as FreeRTOS. If it is wanted to simulate and test the

code in an emulated environment together with QEMU, in this manual, it is used Eclipse.

Eclipse is an integrated development environment (IDE) that can streamline the programming and debugging of FreeRTOS code.

Since Eclipse is a Java-based application and requires a Java Runtime Environment (JRE) to work. Install the default openjdk package with the following command:

```
1 $ sudo apt install default-jre
```

Eclipse can be downloaded from [here](#).

Once installed, extract the Eclipse package that it was downloaded from the Eclipse official website performing this command:

```
1 $ sudo tar xf eclipse-embeddedcpp-2023-12-R-linux-gtk-x86_64.tar.gz/  
   opt
```

To run Eclipse from any location in the system, without having to specify the full path executable, we can use the command below:

```
1 $ sudo ln -s /opt/eclipse/eclipse /usr/local/bin/
```

This command creates a symbolic link (which is a file type that serves as a pointer to another file or directory in the system) called “eclipse” in the directory “/usr/local/bin”, which points to the Eclipse executable located in “/opt/eclipse/eclipse”. Finally, Eclipse can be run by simply inserting “eclipse” command on the terminal as shown in the following picture:

```
1 $ eclipse
```

1.2.4 Install GCC Toolchain

Since we are using FreeRTOS ARM Cortex-M3 GCC, it is needed to install GNU ARM Embedded Toolchain from [armDeveloper website](#). Scroll down the page until you have found the following content:

x86_64 Linux hosted cross toolchains**AArch32 bare-metal target (arm-none-eabi)**

- [arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-eabi.tar.xz](#)
- [arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-eabi.tar.xz.asc](#)
- [arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-eabi.tar.xz.sha256asc](#)

Now, download only the first and extract:

```
1 $ tar -xf arm-gnu-toolchain-13-2.rel1-x86_64-arm-none-eabi.tar.xr
```

1.2.5 Install FreeRTOS

After extracting the file above, go to the FreeRTOS official website and download:

Download FreeRTOS

Download the latest FreeRTOS and Long Term Support (LTS) packages below.

The [FAQ](#) describes the difference between individual libraries and library packages, and provides [links to individual library repositories](#).

FreeRTOS 202212.01

Package containing the [FreeRTOS Kernel](#), [FreeRTOS-Plus libraries](#) and [AWS IoT libraries](#), along with example projects. Source code is also available on [GitHub](#). Separately, the latest FreeRTOS Kernel can also be downloaded from [here](#).

Download

To unzip the freeRTOS file use:

```
1 $ unzip FREERTOSv202212.01.zip
```

1.2.6 Manage the PATH Environment variable

In order to obtain the access rights, the command “chown” is used:

```
1 $ sudo chown ubuntu /opt
```

By using this command, the owner of the directory ‘/opt’ is changed to ‘ubuntu’. After executing this command, the user ‘ubuntu’ will become the owner of the directory ‘/opt’ and will have all the related access rights.

Then, move freeRTOS folder from the /Download folder to /Home folder:

```
1 $ export FREERTOS_PATH=~ / FREERTOSv202212.01 /
```

This command sets the environment variable **FREERTOS_PATH** in an environment of shell Linux.

To check the value of the environment variable:

```
1 $ echo $FREERTOS_PATH
```

Modify with nano command the file ‘.profile’ in the following way:

```
1 $ nano .profile
```

By inserting ‘export FREERTOS_PATH=~ / FreeRTOSv202212.01’ in the file .profile, every time it is needed to access a new shell environment it will set automatically the environment variable FREERTOS_PATH.

The command in the photo below will apply the configurations and the environment variables defined in the **.profile** file (1.2.6).

```
GNU nano 7.2
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

export FREERTOS_PATH=~/.FreeRTOSv202212.01
```

1.2.7 Import, Build and Execute a Demo Project in Eclipse

In order to import, build and execute a demo project in Eclipse see subsection 1.1.8.