

# ERIKA3 pre-built Virtual Machine

with support for Arduino and Jetson TX1/TX2

Version: 1.6

September 5, 2018



## About Evidence S.r.l.

Evidence is a company operating in the field of software for embedded real-time systems. It started in 2002 as a spin-off company of the Real-Time Systems (ReTiS) Lab of the Scuola Superiore Sant'Anna (Pisa, Italy). Today, Evidence is a dynamic company with collaborations in various fields including automotive, industrial, HVAC, and others.

People at Evidence are experts in the domain of embedded and real-time systems, with a deep knowledge on the design and specification flow of embedded software.

The main areas where Evidence is active are: Operating systems and firmware (both AUTOSAR and Linux based, Ubuntu Core, Yocto, Hypervisors); Model-based design (with experiences like E4Coder, Matlab/Simulink modeling and code generation, National Instruments LabView, and Eclipse/ECORE/XText/Acceleo technologies); Application development (C/C++/Qt/Linux/Windows).

For more information see: <http://www.evidence.eu.com>

## Contact Info

Evidence Srl,  
Via Carducci 56  
Località Ghezzano  
56010 S.Giuliano Terme  
PISA Italy

Tel: +39 050 99 11 224  
Fax: +39 050 99 10 812

For more information about Evidence products, please send an e-mail to the following address: [info@evidence.eu.com](mailto:info@evidence.eu.com). Other information about the Evidence product line can be found at the Evidence web site at: <http://www.evidence.eu.com>.

This document is Copyright 2017-2018 Evidence S.r.l.

Information and images contained within this document are copyright and the property of Evidence S.r.l. All trademarks are hereby acknowledged to be the properties of their respective owners. The information, text and graphics contained in this document are provided for information purposes only by Evidence S.r.l. Evidence S.r.l. does not warrant the accuracy, or completeness of the information, text, and other items contained in this document. Matlab, Simulink, Mathworks are registered trademarks of Matworks Inc. Microsoft, Windows are registered trademarks of Microsoft Inc. Java is a registered trademark of Sun Microsystems. in the USA. and other countries, and are used under license. All other trademarks used are properties of their respective owners. This document has been written using L<sup>A</sup>T<sub>E</sub>X and LyX.

# Contents

<b>1. Introduction</b>	<b>7</b>
1.1. Requirements . . . . .	7
1.2. Supported boards . . . . .	7
1.3. Licensing . . . . .	8
1.4. Feedback, bugs, and additional examples . . . . .	8
<b>2. Installing the Virtual Machine</b>	<b>9</b>
2.1. Installing VirtualBox . . . . .	9
2.2. Downloading the Virtual Machine . . . . .	9
2.3. VirtualBox settings . . . . .	10
2.4. Username and password of the Linux virtual machine . . . . .	10
<b>3. Compiling and running ERIKA3 On Jetson TX1/TX2</b>	<b>12</b>
3.1. Platform setup . . . . .	12
3.1.1. Serial interface . . . . .	12
3.1.2. Static IP address . . . . .	15
3.2. Building the Linux kernel and Jailhouse . . . . .	16
3.3. Installing the Linux kernel and Jailhouse . . . . .	17
3.4. Running Linux . . . . .	17
3.5. Compiling an ERIKA3 application . . . . .	18
3.6. Executing an ERIKA3 application . . . . .	18
3.7. Libc support . . . . .	23
3.8. Advanced configuration . . . . .	24
<b>4. Compiling and running ERIKA3 on Arduino UNO</b>	<b>26</b>
4.1. Platform setup . . . . .	26
4.1.1. Serial interface . . . . .	26
4.2. Running RT-Druid 3 . . . . .	29
4.3. Configuring RT-Druid 3 . . . . .	31
4.4. Compiling an ERIKA3 application . . . . .	35
4.5. Programming the Arduino UNO board . . . . .	38
<b>5. Acknowledgements</b>	<b>41</b>
<b>A. ERIKA3 version installed on the virtual machine</b>	<b>42</b>

# List of Figures

2.1.	Files obtained when unpacking the virtual machine. . . . .	9
2.2.	VirtualBox opened after clicking on the <code>vbox</code> file. . . . .	10
3.1.	Jetson serial interface. . . . .	12
3.2.	USB Type-A Connector. . . . .	13
3.3.	Assigning the serial interface to VirtualBox. . . . .	13
3.4.	How to run a terminal. . . . .	13
3.5.	<code>dmesg</code> command. . . . .	14
3.6.	<code>dmesg</code> command output. . . . .	14
3.7.	Putty and RT-Druid icons on Desktop. . . . .	15
3.8.	Putty configuration. . . . .	16
3.9.	RT-Druid Eclipse Workspace Selection. . . . .	18
3.10.	New RT-Druid project. . . . .	19
3.11.	Naming the RT-Druid project. . . . .	19
3.12.	Selecting the template for the RT-Druid project. . . . .	20
3.13.	New project created. . . . .	20
3.14.	Building the project. . . . .	21
3.15.	ERIKA3 application output on the serial interface. . . . .	23
3.16.	Eclipse project properties. . . . .	24
3.17.	RT-Druid generator properties. . . . .	25
4.1.	Arduino USB Type-B Connector. . . . .	26
4.2.	Arduino USB Type-A Connector. . . . .	27
4.3.	Arduino USB Setup. . . . .	27
4.4.	Terminal Icon. . . . .	27
4.5.	<code>dmesg</code> command. . . . .	28
4.6.	<code>dmesg</code> command output. . . . .	28
4.7.	PuTTY Configuration. . . . .	29
4.8.	RT-Druid Eclipse Workspace Selection. . . . .	30
4.9.	RT-Druid Eclipse Welcome. . . . .	30
4.10.	Preferences entry of the <b>Window</b> menu. . . . .	31
4.11.	Preferences Window. . . . .	31
4.12.	Arduino SDK Property selection. . . . .	32
4.13.	Arduino SDK Property setup. . . . .	32
4.14.	Arduino Serial Baudrate Property selection. . . . .	33
4.15.	Arduino Serial Baudrate Property setup. . . . .	33
4.16.	Arduino Serial Port Property selection. . . . .	34

4.17. Arduino Serial Port Property setup.	34
4.18. Arduino Properties confirmation.	35
4.19. Provide a name for the Eclipse project containing the demo.	35
4.20. How to select an Arduino UNO example.	36
4.21. The Arduino UNO demo after the creation and compilation of the Doxygen documentation.	37
4.22. Arduino UNO demo build successfull.	37
4.23. Arduino UNO build target creation.	38
4.24. Create Build Target dialog box.	38
4.25. Arduino UNO upload.	39
4.26. Arduino UNO serial output.	40

# About this document

This document describes the installation, first setup and first demo run of the ERIKA3 pre-built Virtual Machine for Nvidia Jetson TX1/TX2 and Arduino UNO.

## Function of the document

The function of this document is to provide a quick start guide for using the Virtual Machine with a demo example.

## Document history

Version	Date	Author	Company	Change Description
0.1	March 2018	Claudio Scordino	Evidence Srl	Initial version.
1.0	April 2018	Giuseppe Serano	Evidence Srl	Added Arduino UNO support.
1.1	April 2018	Paolo Gai	Evidence Srl	Document check.
1.2	April 2018	Claudio Scordino	Evidence Srl	Advanced Jailhouse configuration.
1.3	May 2018	Claudio Scordino	Evidence Srl	Commands changed for the new version of Jailhouse.
1.4	May 2018	Claudio Scordino	Evidence Srl	How to add libc support on TX1.
1.5	June 2018	Claudio Scordino	Evidence Srl	Support for TX2.
1.6	September 2018	Claudio Scordino	Evidence Srl	Information about the ERIKA version.

# 1. Introduction

Installing a complete development and debugging environment for an embedded board always involves a lot of work in installing compilers, debuggers, development environments, makefiles, and so on. This Virtual Machine aims at providing a quick solution for all these issues, providing a Linux platform with all software preinstalled and ready to work, allowing you to compile and flash OSEK/VDX applications on the specified target boards.

In particular, this Virtual Machine provides a complete virtual environment where you will be able to:

- edit the configuration of your OSEK/VDX application using the Eclipse-based RT-Druid environment;
- automatically generate the configuration files;
- build your application and link it to the ERIKA3 RTOS using a pre-installed open-source compiler.

Additionally, in case of the Nvidia Jetson development boards, the virtual machine also allows to configure and build the Linux kernel and the Jailhouse hypervisor for the board.

## 1.1. Requirements

The only requirement is the VirtualBox tool (<https://www.virtualbox.org/>) for running the Virtual machine.

## 1.2. Supported boards

The virtual machine has been tested on the following boards:

- Nvidia Jetson TX1;
- Nvidia Jetson TX2;
- Arduino UNO;

## 1.3. Licensing

The Virtual Machine described in this document includes various open-source software. The following items shortly describe the main licenses of the tools which have been integrated during this work:

- The Linux Distribution is a standard Ubuntu distribution. For more information about Ubuntu and the software licenses included in this Linux distribution please refer to the following website: <http://www.ubuntu.com/>.
- The ERIKA3 RTOS is distributed under the GPL2 license, whereas the RT-druid plugin is distributed under a proprietary license (see <http://www.erika-enterprise.com/index.php/erika3/licensing.html>); please contact Evidence for different licensing options.
- Eclipse, EMF, Acceleo and other Eclipse plugins are distributed under the EPL License ([http://en.wikipedia.org/wiki/Eclipse\\_Public\\_License](http://en.wikipedia.org/wiki/Eclipse_Public_License)).
- The additional compiler for AVR and Cortex-M pre-installed on this virtual machine are based on GCC, which is distributed under the GNU GPL License.
- The Linux kernel is distributed under the GPL License.
- The Jailhouse hypervisor is distributed under the GPL2 license; the Jailhouse inmate library is also available under the BSD license. For more information, please refer to the official project page: <https://github.com/siemens/jailhouse/> .

## 1.4. Feedback, bugs, and additional examples

We care about your feedback! Information, feedback, and new demos about ERIKA3 can be provided directly on the ERIKA3 website:

<http://www.erika-enterprise.com>

For commercial technical support, sales, pricing, order status, and general information and feedback, please contact Evidence Srl directly at the address and phone numbers available at the following web page:

<http://www.evidence.eu.com/en/contact-us.html>

## 2. Installing the Virtual Machine

### 2.1. Installing VirtualBox

VirtualBox can be freely downloaded and used also for commercial use from the following website:

<https://www.virtualbox.org/wiki/Downloads>

All you need to do is to download the VirtualBox installer, and install it on your PC. All the following screenshots will refer to the usage of VirtualBox on a Windows 7 Host machine.

### 2.2. Downloading the Virtual Machine

The ERIKA3 Virtual Machine can be downloaded from the following website:

<http://www.erika-enterprise.com>

The Virtual Machine is typically distributed as a compressed file. Please decompress it. You will find at least two files, as in Figure 2.1. Note that actual file names may vary. The file with the `vbox` extension is the file containing the settings of the virtual machine (describing the guest hardware, memory, disks, ...). The file with the `vdi` extension is the virtual hard disk used by the virtual machine.

On a typical VirtualBox setup, just double clicking on the file with `vbox` extension will open VirtualBox as in Figure 2.2. Just click on the Start button to boot the Virtual Machine.

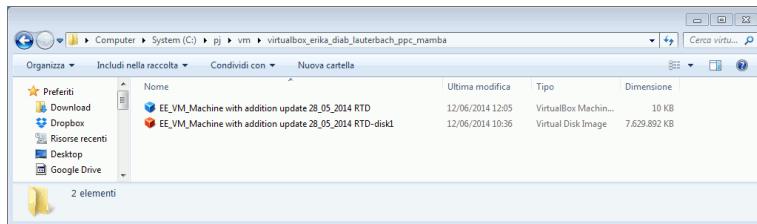


Figure 2.1.: Files obtained when unpacking the virtual machine.

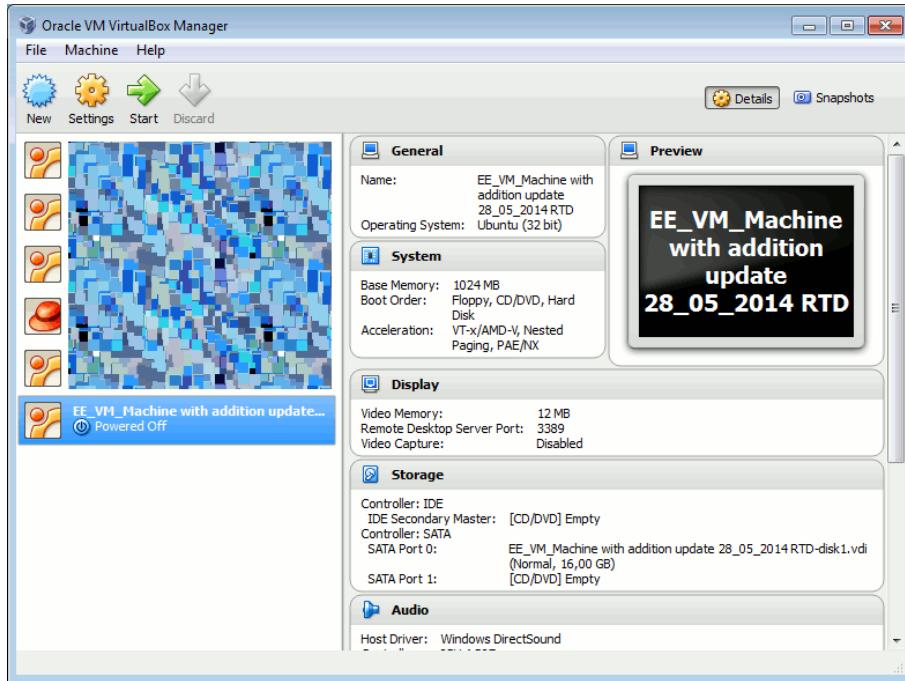


Figure 2.2.: VirtualBox opened after clicking on the `vbox` file.

## 2.3. VirtualBox settings

The following is a list of the main settings of the Virtual machine, useful if, for some reason, you need to recreate the `vbox` file from scratch. Those information must be set on a new virtual machine by clicking on the “Settings” button in Figure 2.2.

1. **General Tab, Basic subtab:** The Type of virtual machine must be Linux / Ubuntu 32 bit.
2. **System tab:** We suggest a 1Gb system memory, I/O APIC active, and as many processors as you have in your physical machine.

Please note that the virtual machine comes with the VirtualBox Guest Additions already installed. This turns out to be very convenient as the X Server will automatically recognize a resize of the VirtualBox window. In case you have a different version of the VirtualBox player you may need to re-install the Guest Additions (please, refer to the VirtualBox documentation for proper instructions).

Once powered up, Linux will boot, and the Ubuntu Desktop will appear.

## 2.4. Username and password of the Linux virtual machine

All the activities described in this document have been executed by the following user-name:

Username: **evidence**

Password: **evidence**

You will typically need to enter username and password when logging in, when using **sudo**, and everytime you are required to perform an action with administrator privileges.

# 3. Compiling and running ERIKA3 On Jetson TX1/TX2

## 3.1. Platform setup

### 3.1.1. Serial interface

A FTDI USB cable can be used to physically connect the platform's serial console to the host machine. This is particularly useful to get output messages from the ERIKA3 guest (as shown in the example below).

The Figure 3.1 shows how the UART pins must be connected on the Jetson platform. The USB Type-A connector side must be then connected to the PC running the Virtual Machine as shown in Figure 3.2

Once the physical connection has been established, proceed as follows:

1. On Virtual Box assign the peripheral to the virtual machine as shown in Figure 3.3 (note that the actual name of the USB-Serial converter may change on your machine).
2. Open a Terminal by clicking on the icon available on the Top-Bar of the virtual machine, shown in Figure 3.4 .
3. Type `dmesg` command and press return key as shown in Figure 3.5 .
4. Look at `dmesg` command output to identify the serial device identifier on which the Nvidia Jetson board is attached (usually `ttyUSB*`), as shown in Figure 3.6.

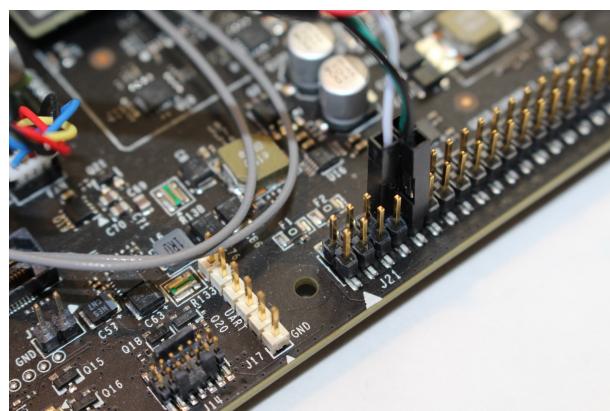


Figure 3.1.: Jetson serial interface.



Figure 3.2.: USB Type-A Connector.

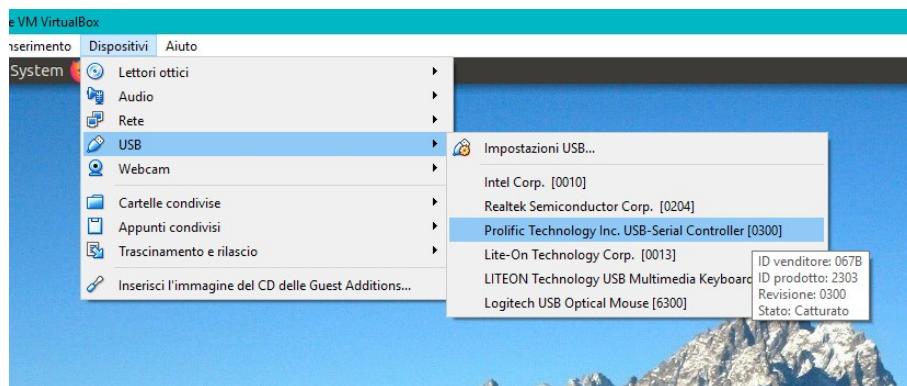


Figure 3.3.: Assigning the serial interface to VirtualBox.



Figure 3.4.: How to run a terminal.

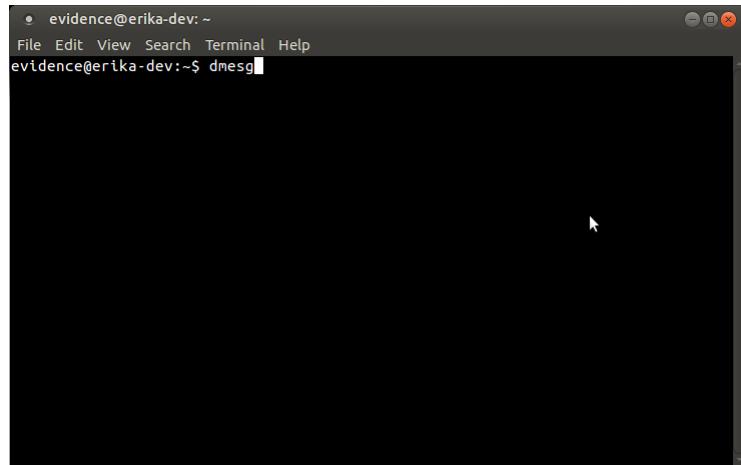
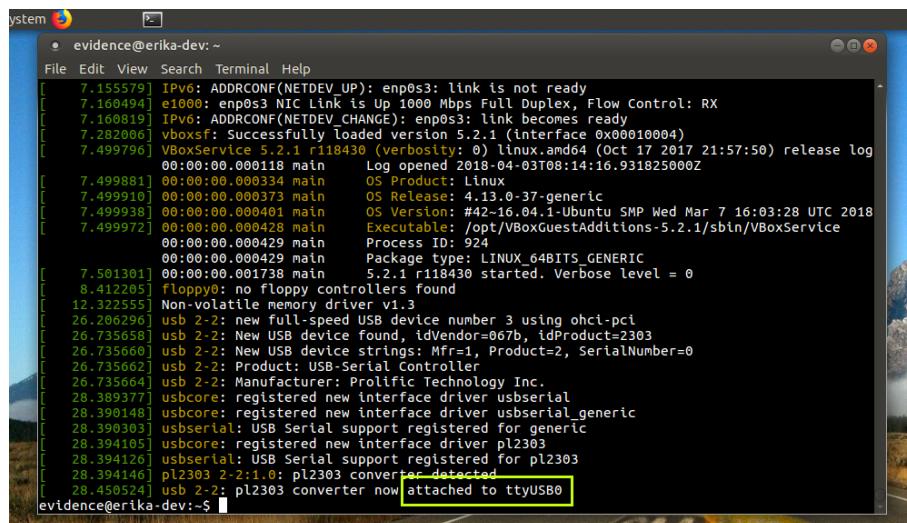


Figure 3.5.: `dmesg` command.



A screenshot of a terminal window titled 'evidence@erika-dev: ~'. The window displays the output of the 'dmesg' command. The output is a log of kernel messages, including network interface configuration, service startup, and USB device detection. A yellow box highlights the final message: 'evidence@erika-dev:~\$ [REDACTED] evidence@erika-dev:~\$ [REDACTED]'.

```

[ 7.155579] IPV6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 7.160494] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 7.166819] IPV6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[ 7.282006] vboxsf: Successfully loaded version 5.2.1 (interface 0x00010004)
[ 7.499796] VBoxService 5.2.1 r118430 (verbosity: 0) linux_and64 (Oct 17 2017 21:57:50) release log
  00:00:00.000118 main      Log opened 2018-04-03T08:14:16.931825000Z
[ 7.499881] 00:00:00.000334 main      OS Product: Linux
[ 7.499910] 00:00:00.000373 main      OS Release: 4.13.0-37-generic
[ 7.499938] 00:00:00.000401 main      OS Version: #42-16.04.1-Ubuntu SMP Wed Mar 7 16:03:28 UTC 2018
[ 7.499972] 00:00:00.000428 main      Executable: /opt/VBoxGuestAdditions-5.2.1/sbin/VBoxService
  00:00:00.000429 main      Process ID: 924
  00:00:00.000429 main      Package type: LINUX_64BITS_GENERIC
[ 7.501301] 00:00:00.001738 main      5.2.1 r118430 started. Verbose level = 0
[ 8.412205] floppyo: no floppy controllers found
[ 12.322555] Non-volatile memory driver v1.3
[ 26.206296] usb 2-2: new full-speed USB device number 3 using ohci-pci
[ 26.735658] usb 2-2: New USB device found, idVendor=067b, idProduct=2303
[ 26.735660] usb 2-2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 26.735662] usb 2-2: Product: USB-Serial Controller
[ 26.735664] usb 2-2: Manufacturer: Prolific Technology Inc.
[ 28.389377] usbcore: registered new interface driver usbserial
[ 28.390148] usbcore: registered new interface driver usbserial_generic
[ 28.390303] usbserial: USB Serial support registered for generic
[ 28.394105] usbcore: registered new interface driver pl2303
[ 28.394126] usbserial: USB Serial support registered for pl2303
[ 28.394146] pl2303 2-2:1.0: pl2303 converter detected
[ 28.450524] usb 2-2: pl2303 converter now attached to ttyUSB0
evidence@erika-dev:~$ [REDACTED]
evidence@erika-dev:~$ [REDACTED]
```

Figure 3.6.: `dmesg` command output.

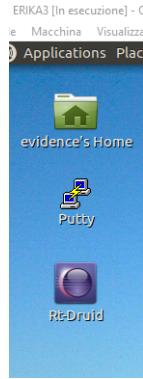


Figure 3.7.: Putty and RT-Druid icons on Desktop.

5. Run the Putty program by clicking on the icon available on the Desktop of the virtual machine, shown in Figure 3.7.
6. On Putty (see Figure 3.8 ), set:
  - Serial line: /dev/ttyUSB0
  - Baurdate: 115200
  - No flow control
7. Start the connection by clicking on Open.

### 3.1.2. Static IP address

There are mainly 3 ways for logging into the Jetson platform:

1. Through an SSH connection over the Ethernet cable.
2. Through the serial interface (see previous Section).
3. Through the graphical interface on HDMI .

If you want to assign a static IP address to the platform (useful for being able to log in into the machine even when the serial interface has been exclusively assigned to the ERIKA3 guest OS), open the `/etc/network/interfaces` file and append the following information:

```
auto eth0
iface eth0 inet static
address ...
netmask ...
gateway ...
```

To reach the external network, you may also need to explicitly add the DNS server by appending the following information to the `/etc/resolv.conf` file:

```
nameserver ...
```

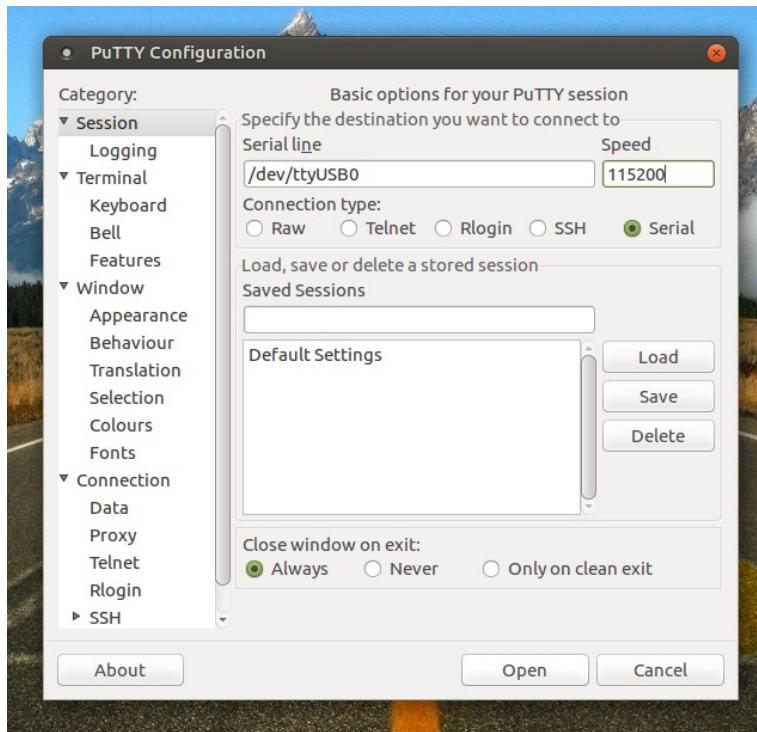


Figure 3.8.: Putty configuration.

## 3.2. Building the Linux kernel and Jailhouse

On the virtual machine, open a terminal as shown in Figure 3.4 and:

1. Enter the board-specific directory.

- For the tx1 board type:

```
cd /home/evidence/tx1
```

- For the tx2 board, instead, type:

```
cd /home/evidence/tx2
```

2. Build the Linux kernel and Jailhouse by typing:

```
./build.sh
```

Note that before running this command, it is possible to run `./configure.sh` for changing the Linux kernel configuration.

3. Transfer the file `target.tgz` to the Jetson platform. This can be done, for example, by using the `scp` command:

```
scp target.tgz nvidia@<Jetson IP address>:
```

### 3.3. Installing the Linux kernel and Jailhouse

This section explains how to install the Linux kernel and the Jailhouse hypervisor on the Jetson platform. It assumes that the `target.tgz` file (mentioned in the previous section) has been already transferred to the Jetson platform.

On the Linux console of the Jetson platform:

1. For installing Jailhouse, type:

```
sudo tar -xmf target.tgz -C /
cd /boot
sudo ./install.sh
```

2. Reboot the platform.

On the U-Boot console:

1. Stop the U-boot countdown by pressing a key.

2. On the U-Boot console type (only needed the first time):

```
load mmc 0:1 $scriptaddr /boot/linux-console-handler.scr
source $scriptaddr
```

3. Disable the Linux serial console by typing:

```
run linux_console_disable
```

4. If you want to disable the Linux serial console permanently<sup>1</sup> (i.e. also for the next times) then type:

```
saveenv
```

5. Continue booting Linux by typing:

```
boot
```

### 3.4. Running Linux

Once Linux has been configured as explained in the previous section, just select *jailhouse kernel* in the 2nd stage bootloader.

---

<sup>1</sup>Note that you will be always able to restore the Linux serial console by typing: `run linux_console_enable`.

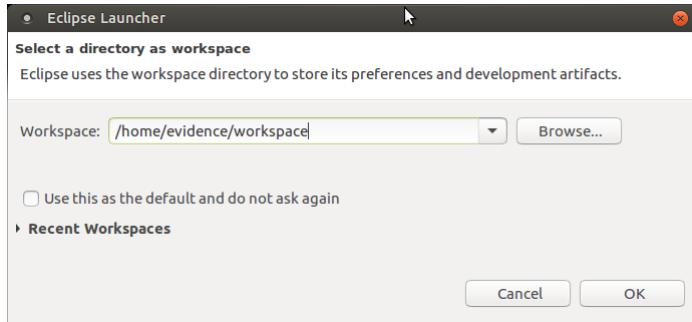


Figure 3.9.: RT-Druid Eclipse Workspace Selection.

### 3.5. Compiling an ERIKA3 application

On the host virtual machine:

1. Double click the RT-Druid Desktop icon for starting RT-Druid. Then confirm the default location `/home/evidence/workspace` as shown in Figure 3.9.
2. Create a new **RT-Druid v3 Oil and C/C++** project as shown in Figure 3.10.
3. Name the project (e.g., “mytest”) and select the Cross GCC toolchain as shown in Figure 3.14.
4. Check the box for using an existing template and select `aarch64→Jailhouse→Helloworld OSEK demo on Jailhouse` as shown in Figure 3.12.
5. Eclipse will then show the new project, and RT-Druid will generate the configuration files, as shown in Figure 3.13.
6. **Attention:** Ensure that the `SOC_DATA` variable is set to `NVIDIA_TEGRA_X1` for TX1 and to `NVIDIA_TEGRA_X2` for TX2 inside the `conf.oil` file.
7. Click with the right mouse key on the project and select **Build project** as shown in Figure 3.14. This will create the `erika_inmate.bin` file in the workspace.
8. Transfer the file `erika_inmate.bin` to the Jetson platform. This can be done, for example, by using the `scp` command:  
`scp erika_inmate.bin nvidia@<Jetson IP address>:`

### 3.6. Executing an ERIKA3 application

On the Jetson platform (Linux console):

1. Set the CPU to maximum performance by typing:

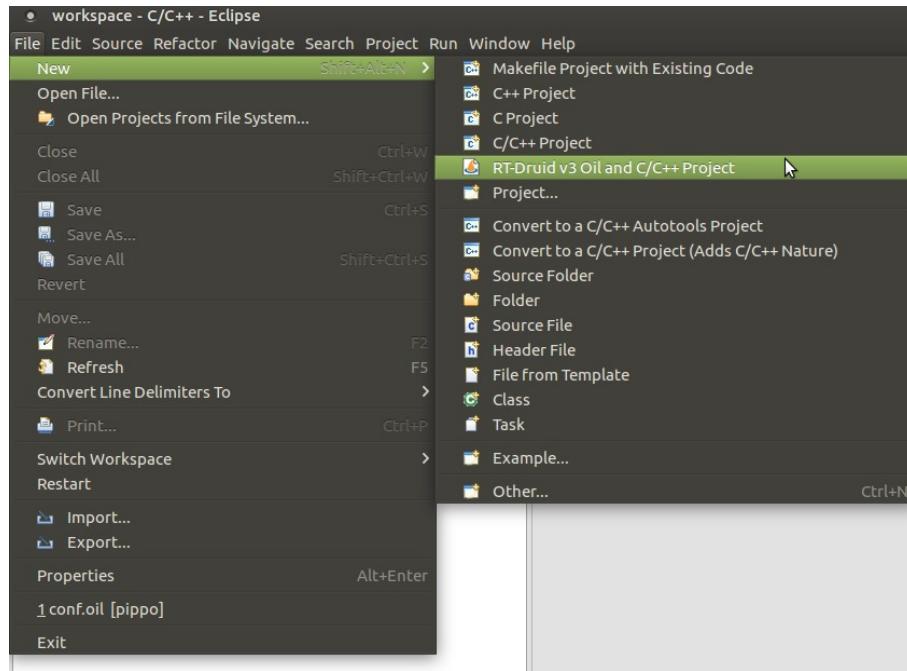


Figure 3.10.: New RT-Druid project.

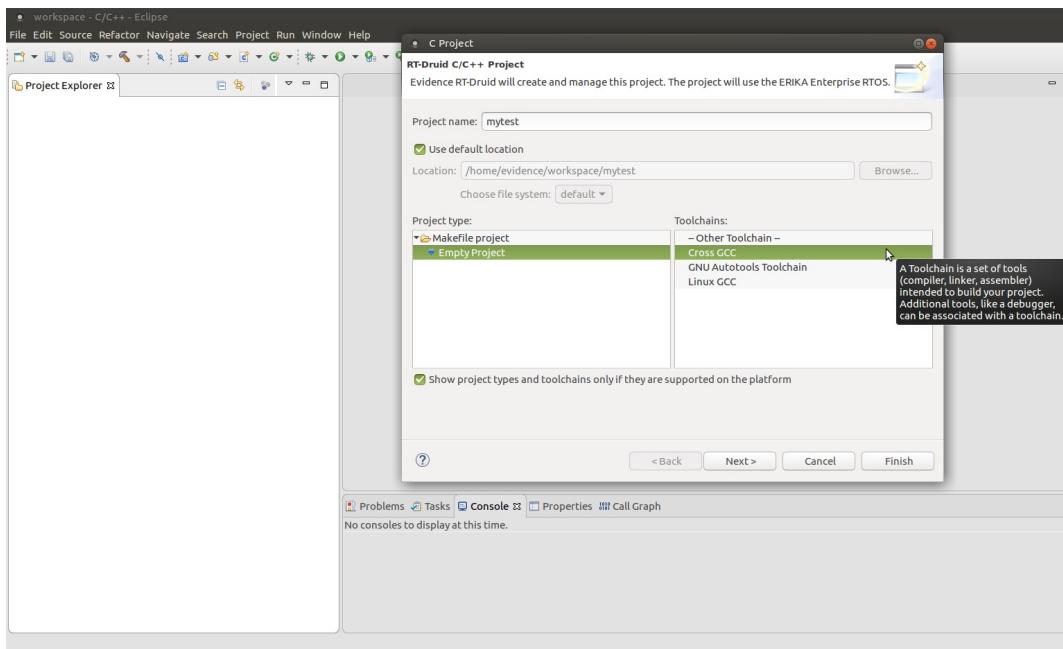


Figure 3.11.: Naming the RT-Druid project.

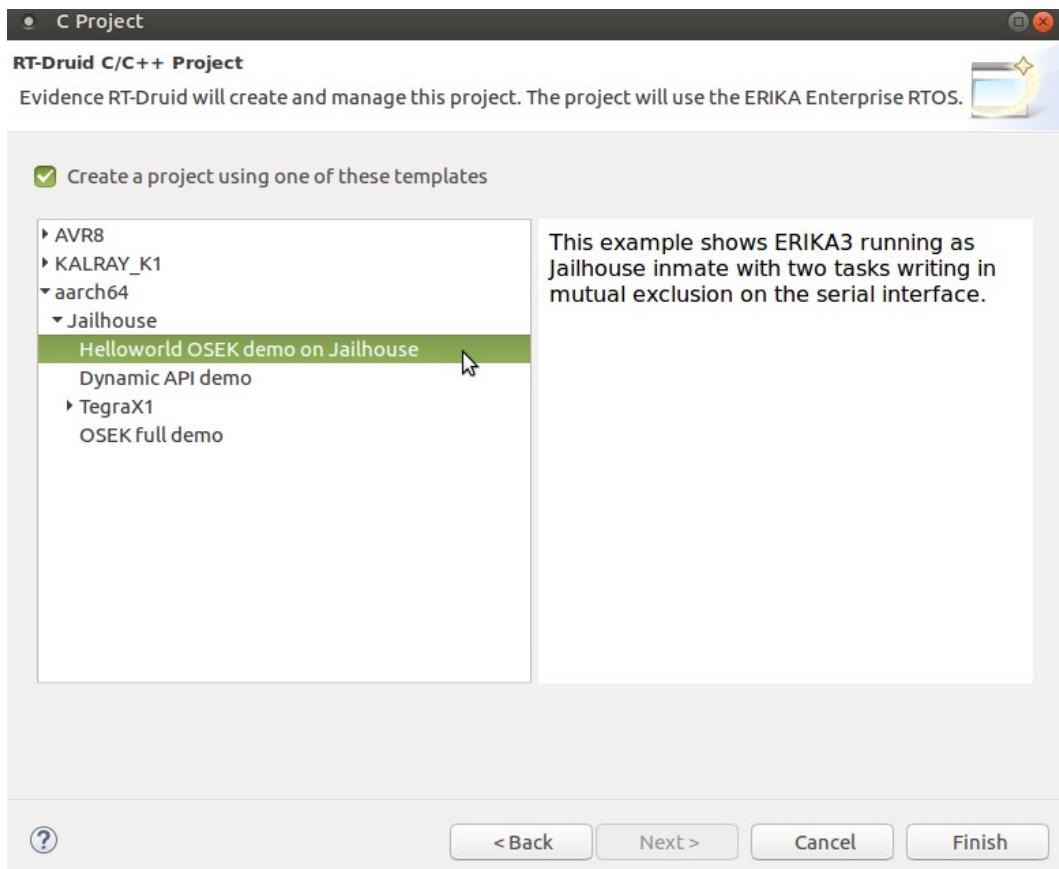


Figure 3.12.: Selecting the template for the RT-Druid project.

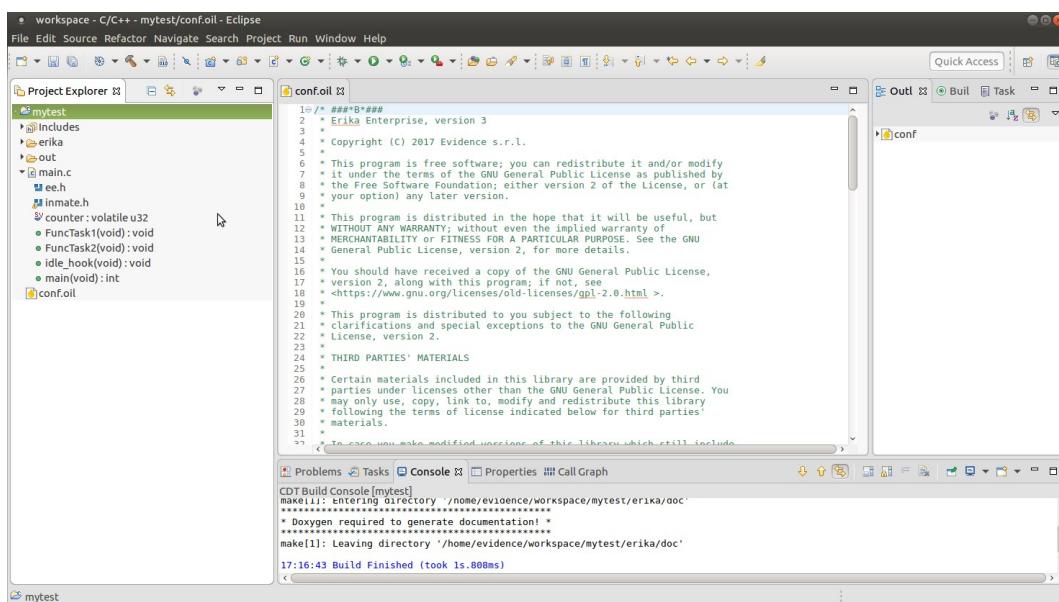


Figure 3.13.: New project created.

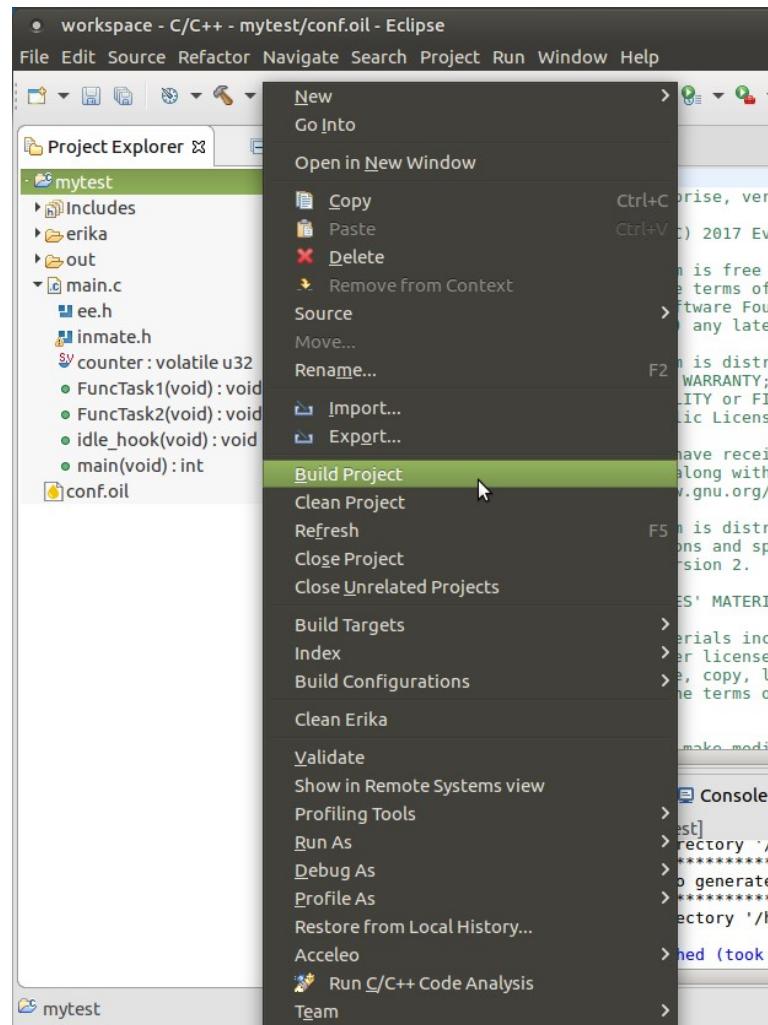


Figure 3.14.: Building the project.

```
sudo su

cd /sys/devices/system/cpu/cpufreq/policy0

echo performance > scaling_governor

exit
```

2. Insert the Jailhouse kernel module by typing:

```
sudo modprobe jailhouse
```

3. Enter the directory containing the Jailhouse cell configurations:

```
cd /jailhouse/configs/arm64/
```

4. Enable the Jailhouse hypervisor.

- For the TX1 board type:

```
sudo jailhouse enable jetson-tx1.cell
```

- For the TX2 board type:

```
sudo jailhouse enable jetson-tx2.cell
```

5. Create the Jailhouse cell for running the ERIKA3 application.

- For the TX1 board type:

```
sudo jailhouse cell create jetson-tx1-demo.cell
```

- For the TX2 board type:

```
sudo jailhouse cell create jetson-tx2-demo.cell
```

6. Load the ERIKA3 binary into the cell.

- For the TX1 board type:

```
sudo jailhouse cell load jetson-tx1-demo erika_inmate.bin
```

- For the TX2 board type:

```
sudo jailhouse cell load jetson-tx2-demo erika_inmate.bin
```

7. Start the ERIKA3 binary.

- For the TX1 board type:

```
sudo jailhouse cell start jetson-tx1-demo
```

```
Cell "jetson-tx1-demo" can be loaded
Started cell "jetson-tx1-demo"
Starting OS...
Starting communication over UART
Hello world from Task2! (0)
Hello world from Task1! (1)
Hello world from Task2! (2)
Hello world from Task1! (3)
Hello world from Task2! (4)
Hello world from Task1! (5)
Hello world from Task2! (6)
Hello world from Task1! (7)
Hello world from Task2! (8)
Hello world from Task1! (9)
Hello world from Task2! (10)
Hello world from Task1! (11)
Hello world from Task2! (12)
Hello world from Task1! (13)
```

Figure 3.15.: ERIKA3 application output on the serial interface.

- For the TX2 board type:

```
sudo jailhouse cell start jetson-tx2-demo
```

8. On the serial console, the ERIKA3 tasks will start printing the messages as shown in Figure 3.15.

9. Stop the ERIKA3 application.

- For the TX1 board type:

```
sudo jailhouse cell shutdown jetson-tx1-demo
```

- For the TX2 board type:

```
sudo jailhouse cell shutdown jetson-tx2-demo
```

## 3.7. Libc support

Most toolchains available in Ubuntu's repositories (e.g. `gcc-aarch64-linux-gnu`) are suitable for cross-compiling all the various components: the Linux kernel, Jailhouse's firmware, Jailhouse's kernel driver, the inmate library and the inmate containing ERIKA. However, they usually rely on libc libraries meant to be used on top of the Linux OS, and are not suitable for building bare-metal RTOSs like ERIKA. For this reason, the Virtual Machine provides a version of Jailhouse enhanced to use different toolchains for compiling the inmate library and the rest of the hypervisor.

The only action needed for adding libc support to an ERIKA application is to put the following statements inside the OIL file:

```
LIBS = "-lc";
LDFLAGS = "-L /home/evidence/aarch64-elf/aarch64-elf/libc/usr/lib";
```

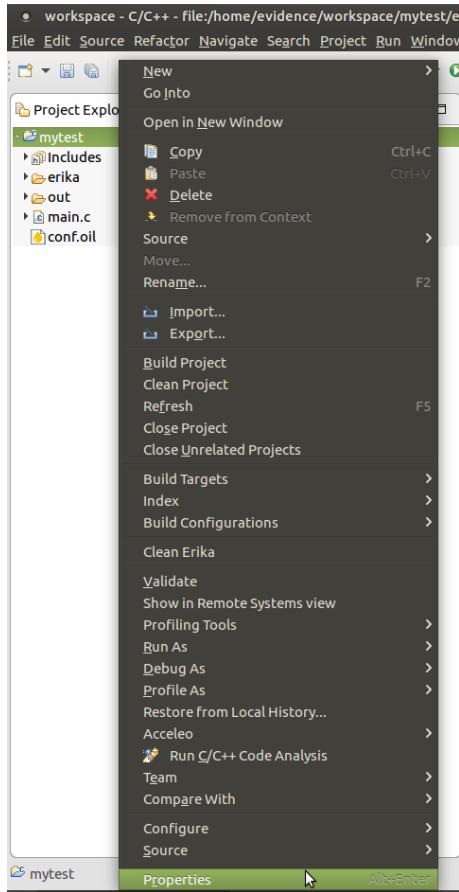


Figure 3.16.: Eclipse project properties.

Additionally, you should set proper alignment for accessing data structures by putting the following statements inside the OIL file:

```
CFLAGS = "-mstrict-align";
```

## 3.8. Advanced configuration

In case you want to build ERIKA Enterprise for a version of Jailhouse different than the one shipped within the Virtual Machine, press the right mouse button on the Eclipse project and select Properties as shown in Figure 3.16 .

Then, select Generator properties, enable project specific settings (as shown in Figure 3.17 ), and set the desired Jailhouse version and path (you can set the Compiler prefix equal to `aarch64-linux-gnu-`).

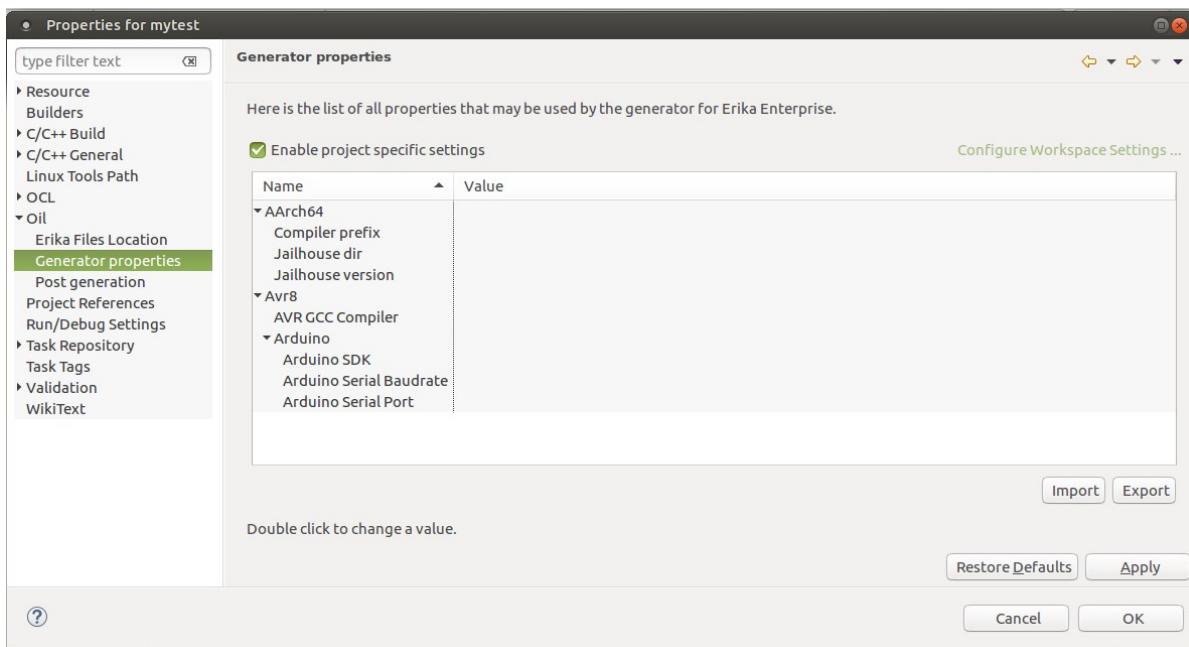


Figure 3.17.: RT-Druid generator properties.

## 4. Compiling and running ERIKA3 on Arduino UNO

### 4.1. Platform setup

#### 4.1.1. Serial interface

An USB (Type-A, Type-B) cable SHALL be used to physically connect the Arduino UNO to the host machine. This is particularly useful to program the board and get output messages from the ERIKA3 guest (as shown in the example below).

The Figure 4.1 shows how the USB Type-B connector must be connected on the Arduino UNO. The USB Type-A connector must be then connected to the PC running the Virtual Machine as shown in Figure 4.2 .

Once the physical connection has been established, proceed as follows:

1. On Virtual Box assign the peripheral to the Virtual Machine as shown in Figure 4.3. .
2. Open a Terminal by clicking on the icon available on the Top-Bar of the virtual machine, shown in Figure 4.4.
3. Type `dmesg` command and press return key as shown in Figure 4.5.
4. Look at `dmesg` command output to identify the serial device identifier on which Arduino UNO board is attached, as shown in Figure 4.6.



Figure 4.1.: Arduino USB Type-B Connector.



Figure 4.2.: Arduino USB Type-A Connector.

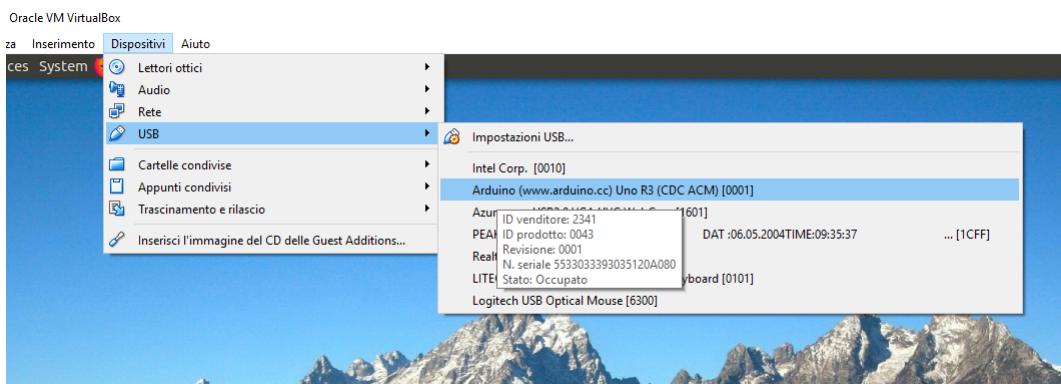


Figure 4.3.: Arduino USB Setup.



Figure 4.4.: Terminal Icon.

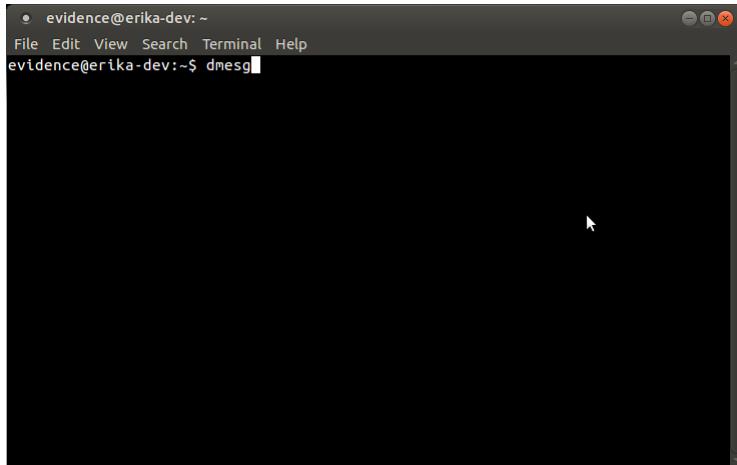


Figure 4.5.: `dmesg` command.

```
evidence@erika-dev: ~
File Edit View Search Terminal Help
[ 7.098201] VboxService 5.2.8 r121009 (verbosity: 0) linux.amd64 (Feb 26 2018 15:59:51) ^
ease log
          00:00:00.000062 main      Log opened 2018-03-29T08:29:20.774928000Z
[ 7.098231] 00:00:00.000117 main      OS Product: Linux
[ 7.098249] 00:00:00.000137 main      OS Release: 4.13.0-37-generic
[ 7.098266] 00:00:00.000155 main      OS Version: #42-16.04.1-Ubuntu SMP Wed Mar 7 16:03:
UTC 2018
[ 7.098288] 00:00:00.000171 main      Executable: /opt/VBoxGuestAdditions-5.2.8/sbin/VBox
vice
          00:00:00.000172 main      Process ID: 870
          00:00:00.000172 main      Package type: LINUX_64BITS_GENERIC
[ 7.099575] 00:00:00.001454 main      5.2.8 r121009 started. Verbose level = 0
[ 8.159830] floppy: no floppy controllers found
[ 8.159867] work still pending
[ 12.185242] Non-volatile memory driver v1.2
[ 1578.066571] usb 1-2: new full-speed USB device number 3 using ohci-pci
[ 1578.588241] usb 1-2: New USB device found, idVendor=2341, idProduct=0043
[ 1578.588244] usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=220
[ 1578.588246] usb 1-2: Manufacturer: Arduino (www.arduino.cc)
[ 1578.588247] usb 1-2: SerialNumber: f533033393035120A080
[ 1579.119485] cdc_acm 1-2:1.0: ttyACM0  USB ACM device
[ 1579.139349] usbcore: registered new interface driver cdc_acm
[ 1579.139351] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
evidence@erika-dev:~$
```

Figure 4.6.: `dmesg` command output.

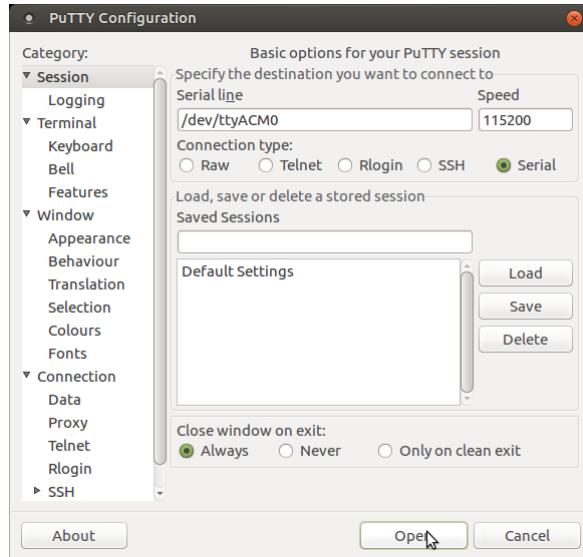


Figure 4.7.: PuTTY Configuration.

5. Run the Putty program by clicking on the icon available on the Desktop of the virtual machine, shown in Figure 3.7.
6. On Putty (see Figure 4.7), set:
  - Serial line: depending the serial device Arduino UNO board is attached (E.g. /dev/ttyACM0).
  - Baurdate: 115200.
  - No flow control.
7. Start the connection by clicking on Open.

## 4.2. Running RT-Druid 3

The following steps will guide you in the compilation of a simple ERIKA3 application for Arduino UNO:

1. To compile your first application with ERIKA3, you need to open the Eclipse IDE. There is an **Rt-Druid** link on the Desktop as shown in Figure 3.7.
2. Double click on it, and Eclipse will open requiring the workspace location. Please confirm the default location `/home/evidence/workspace` as in Figure 4.8. The Eclipse welcome screen will appear as in Figure 4.9. Click on the **Workbench** icon as highlighted in Figure 4.9, and the default Eclipse view will appear.

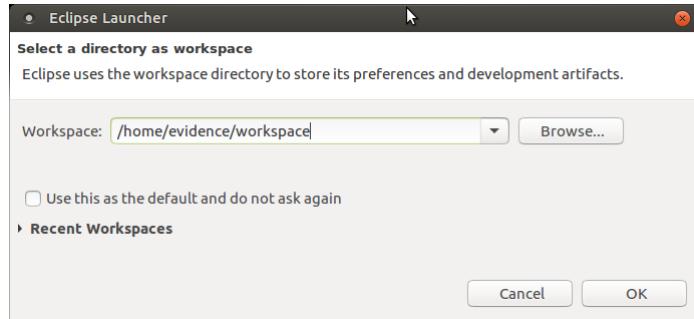


Figure 4.8.: RT-Druid Eclipse Workspace Selection.

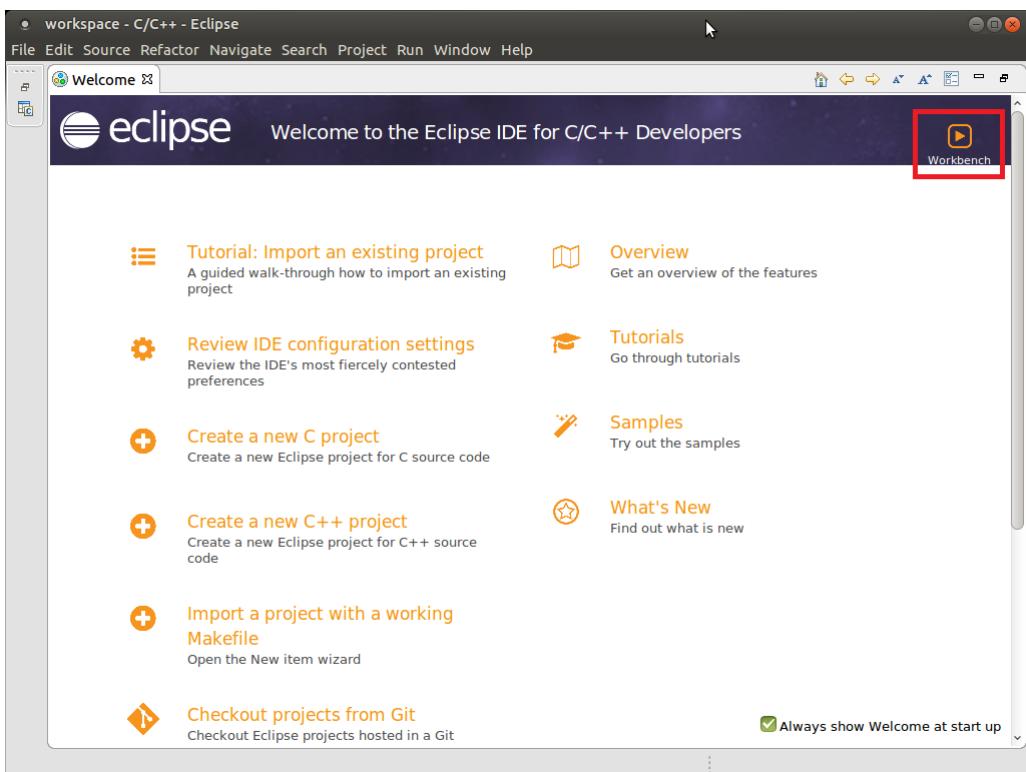


Figure 4.9.: RT-Druid Eclipse Welcome.

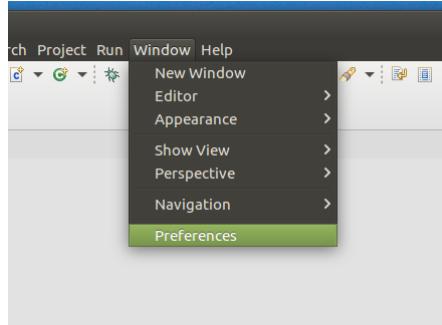


Figure 4.10.: Preferences entry of the Window menu.

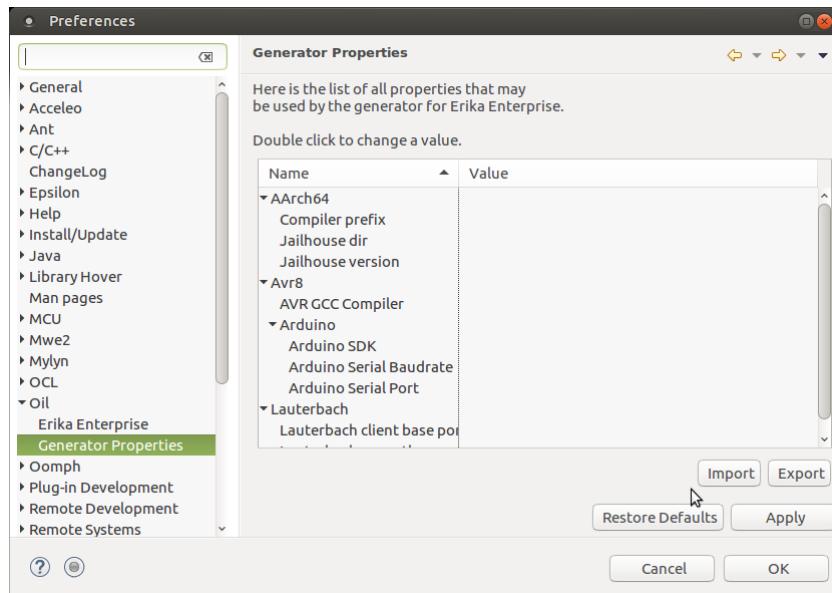


Figure 4.11.: Preferences Window.

### 4.3. Configuring RT-Druid 3

The following steps will guide you in the RT-Druid configuration for Arduino UNO board:

1. Click on the **Preferences** entry of the **Window** menu as shown in Figure 4.10.
2. The **Preferences** window will appear as shown in Figure 4.11. From the left panel expand **Oil** entry and select **Generator Properties** as shown in Figure 4.11.
3. In the center panel double-click on the **Arduino SDK Property** entry as shown in Figure 4.12.
4. The **Arduino SDK Property** setup window will appear: browse the file-system to select the correct path of Arduino distribution (E.g. /home/evidence/arduino-1.8.5) and click **OK** button as shown in Figure 4.13.

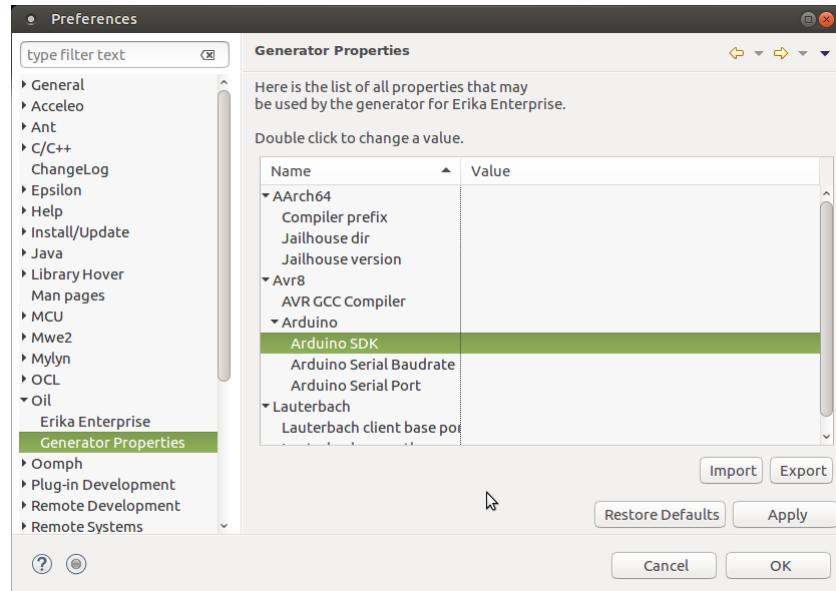


Figure 4.12.: Arduino SDK Property selection.

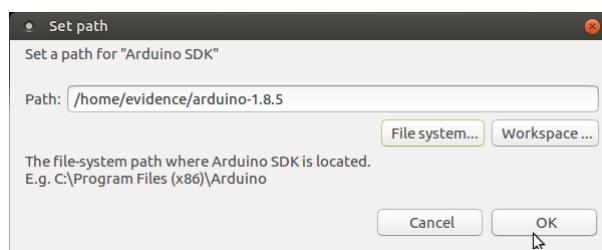


Figure 4.13.: Arduino SDK Property setup.

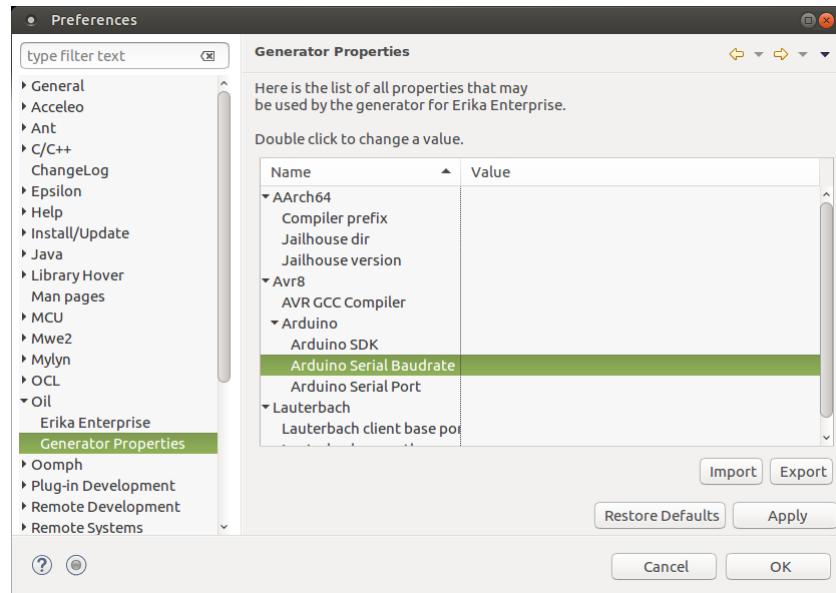


Figure 4.14.: **Arduino Serial Baudrate Property** selection.

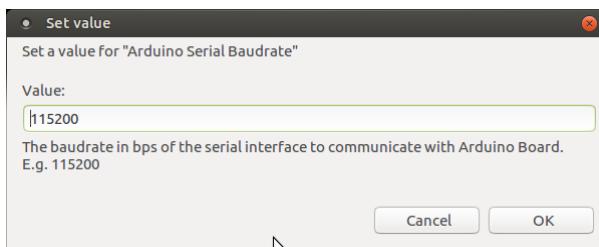


Figure 4.15.: **Arduino Serial Baudrate Property** setup.

5. Please note that there is no need to specify the **AVR GCC Compiler** property, as it is automatically inherited from the Arduino SDK location.
6. In the center panel of **Preferences** window double-click on the **Arduino Serial Baudrate Property** entry as shown in Figure 4.14.
7. The **Arduino Serial Baudrate Property** setup window will appear: type the value 115200 and click **OK** button as shown in Figure 4.15.
8. In the center panel of **Preferences** window double-click on the **Arduino Serial Port Property** entry as shown in Figure 4.16.
9. The **Arduino Serial Port Property** setup window will appear: type the USB Device identifier retrieved by `dmesg` command (E.g. `ttyACM0`) and click **OK** button as shown in Figure 4.17.
10. Finally click on **Apply** button and then click on **OK** button as shown in Figure 4.18.

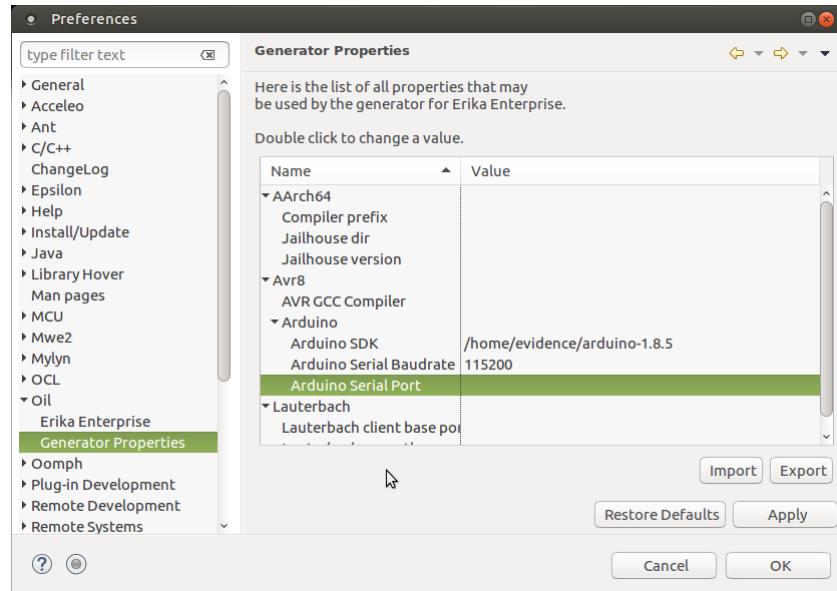


Figure 4.16.: **Arduino Serial Port Property** selection.

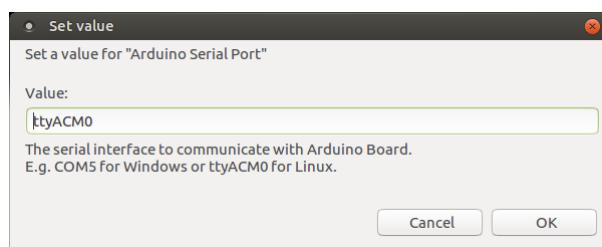


Figure 4.17.: **Arduino Serial Port Property** setup.

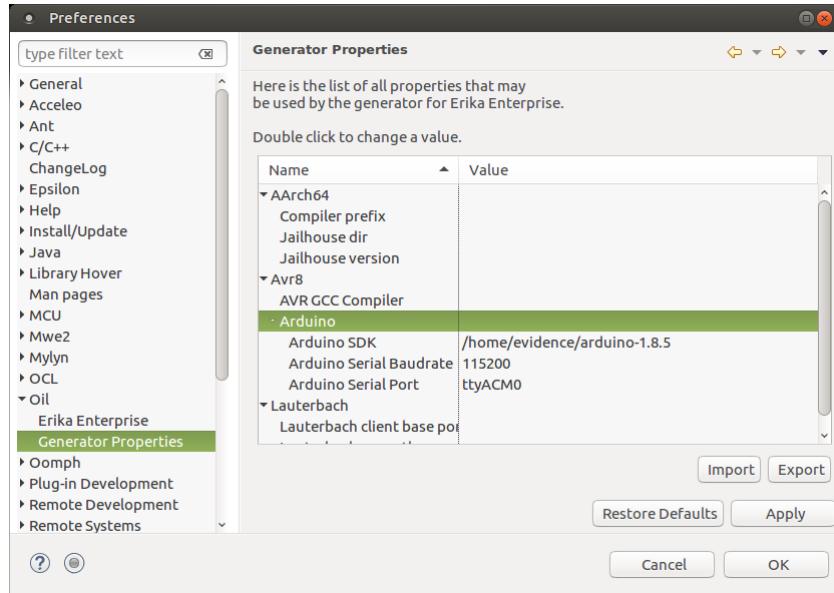


Figure 4.18.: Arduino Properties confirmation.

## 4.4. Compiling an ERIKA3 application

The following steps will guide you in the compilation of a simple ERIKA3 application for Arduino UNO:

1. Create a new **RT-Druid v3 Oil and C/C++** project as shown in Figure 3.10.
2. A Dialog Box will appear as in Figure 4.19. Provide a name for the project (E.g. ArduinoEE3), select **Cross GCC** and press **Next** button.

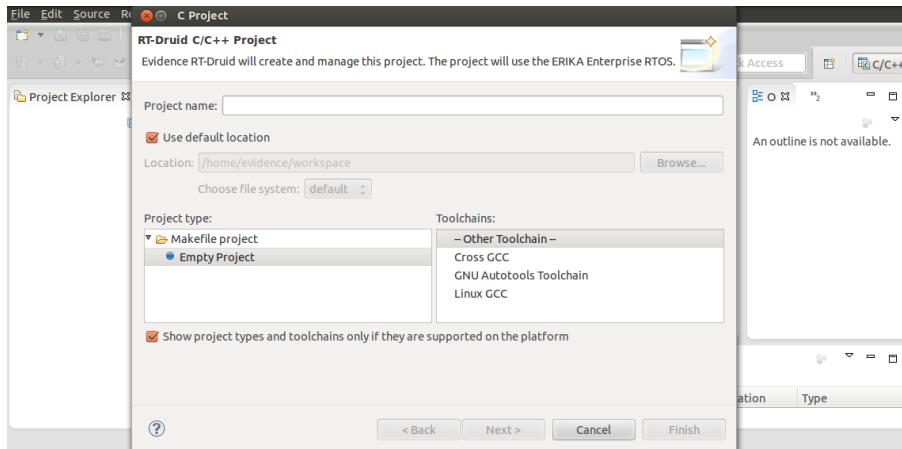


Figure 4.19.: Provide a name for the Eclipse project containing the demo.

3. Select the checkbox “Create a project using one of these templates”, and select the **Full Demo 2** for the Arduino UNO board (see Figure 4.20).

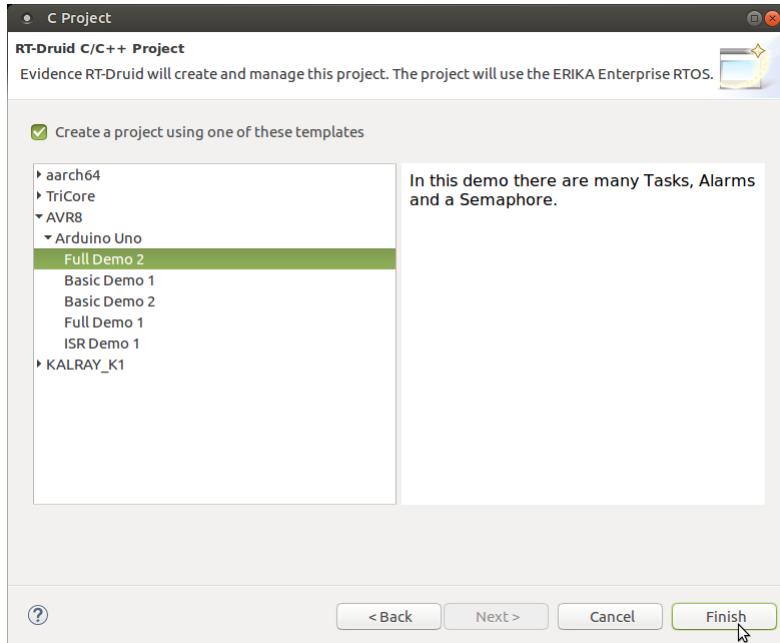


Figure 4.20.: How to select an Arduino UNO example.

4. Click on the Finish button to create the example: the RT-Druid generator will pull ERIKA files into your project and the html documentation will be generated as shown on Figure 4.21.
5. At this point, you can explore the demo examples:
  - They are typically composed by several files:
    - a `conf.oiL` file, containing the OIL description needed to statically configure the kernel and other files;
    - a `code.cpp`, containing the application code.
  - The target board selected for the specific demo is listed in the OIL file parameters.
  - A description of the typical layout of an EE3 workspace can be found at this link:  
[http://www.erika-enterprise.com/wiki/index.php?title=Quick\\_start\\_guide#Anatomy\\_of\\_an\\_ERIKA\\_v3\\_Project](http://www.erika-enterprise.com/wiki/index.php?title=Quick_start_guide#Anatomy_of_an_ERIKA_v3_Project)
6. To compile the project, just right click on the project name and select “Build Project” as shown in Figure 3.14. As a result, the project is compiled using the GCC cross-compiler for the specific board. The output is printed on the Console view as shown on Figure 4.22.

You are now ready to program the resulting files on the target board.

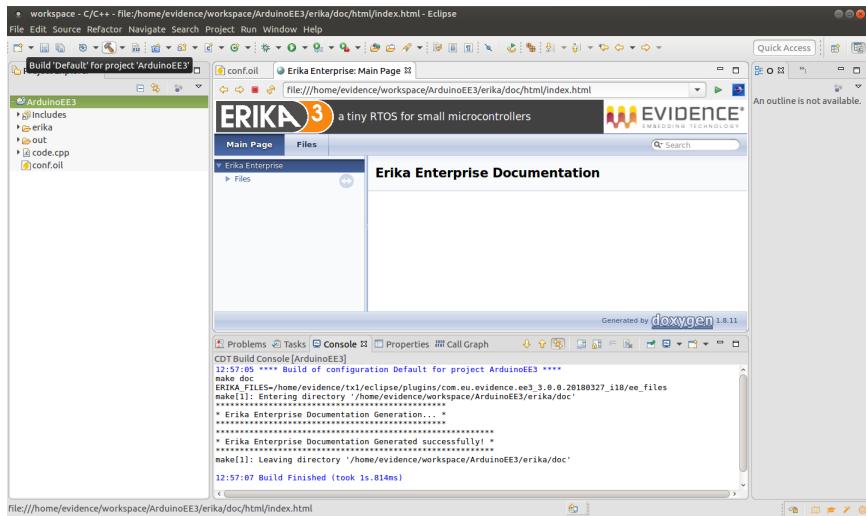


Figure 4.21.: The Arduino UNO demo after the creation and compilation of the Doxygen documentation.

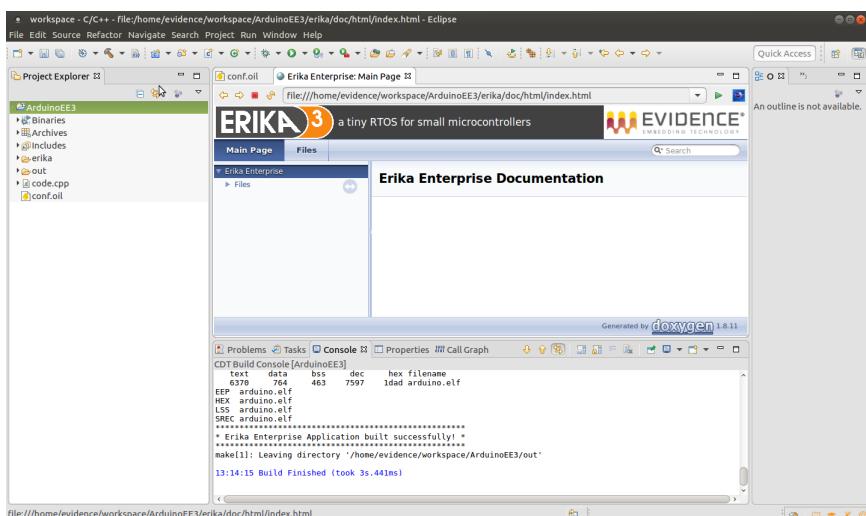


Figure 4.22.: Arduino UNO demo build successfull.

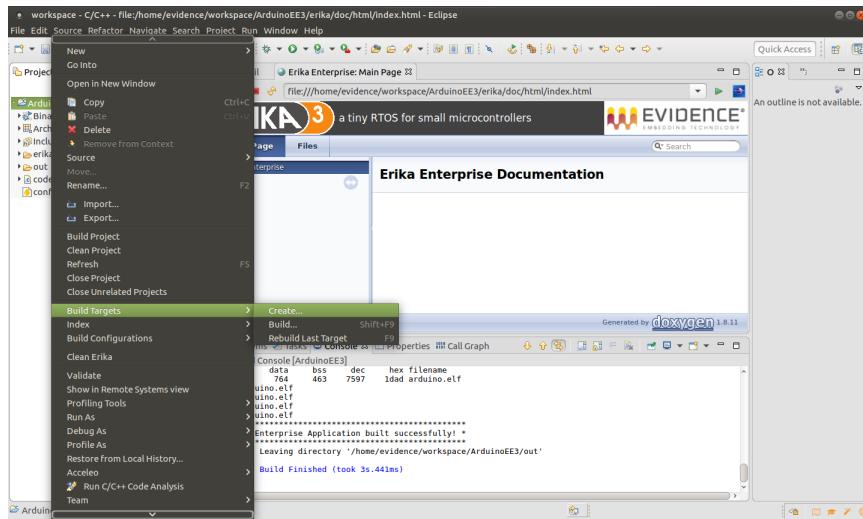


Figure 4.23.: Arduino UNO build target creation.

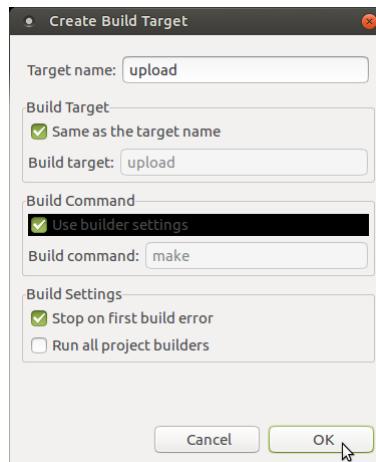


Figure 4.24.: Create Build Target dialog box.

## 4.5. Programming the Arduino UNO board

To program the Arduino UNO board from the virtual machine, you can do the following steps:

1. Right click on the project name and select “Create...” entry from “Build Targets” menu as shown in Figure 4.23.
2. The “Create Build Target” Dialog Box will appear as in Figure 4.24. Provide a name for the build target (E.g. upload), uncheck the **Run all project builders** setting and press **OK** button.
3. The “upload” build target will be created on your project. Double-click on it to program the Arduino UNO board: the output is printed on the Console view as

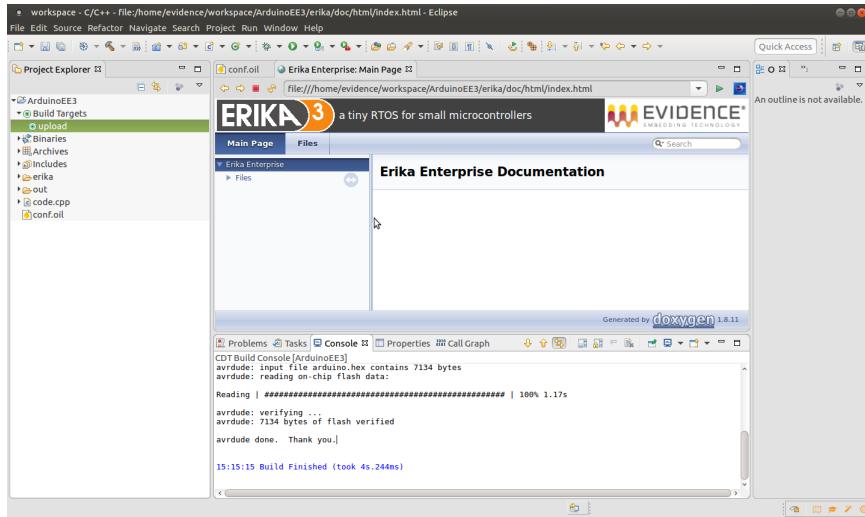


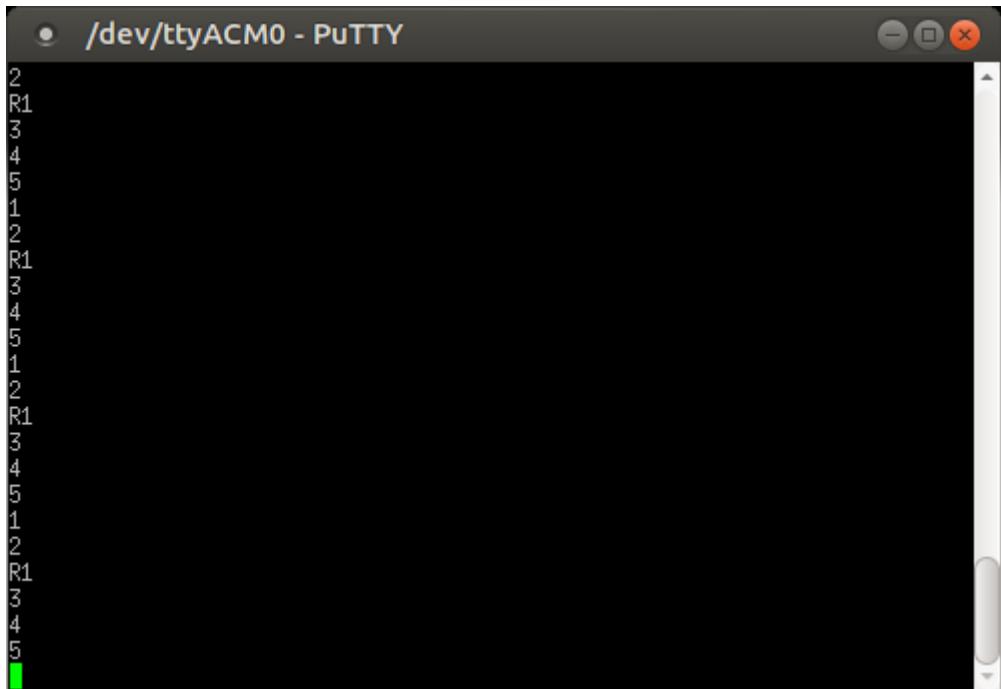
Figure 4.25.: Arduino UNO upload.

shown on Figure 4.25.

4. After that, the application will run on the Arduino UNO board and you can connect a Serial Terminal (e.g. PuTTY<sup>1</sup>) to the board to see the application output<sup>2</sup> as shown on Figure 4.26.

<sup>1</sup>Please select the same device you listed in Figure 4.17, as example /dev/ttyACM0. Also, please select 115200 for the Speed, and Serial for the Connection type.

<sup>2</sup>Please note that not all the Arduino demos use the serial output. For instance, the **Full Demo 2** demo suggested at the beginning of this chapter uses the serial output.



The screenshot shows a terminal window titled '/dev/ttyACM0 - PuTTY'. The window contains the following text output:

```
2
R1
3
4
5
1
2
R1
3
4
5
1
2
R1
3
4
5
1
2
R1
3
4
5
```

Figure 4.26.: Arduino UNO serial output.

## **5. Acknowledgements**

The activity of porting Jailhouse and the ERIKA3 RTOS on the TX1 and TX2 boards has been funded by the European Commission under the HERCULES H2020 project (<http://hercules2020.eu>).

## A. ERIKA3 version installed on the virtual machine

In order to understand the ERIKA3 version you are currently running on the virtual machine, you can do as follows:

1. Open Eclipse, and select Help → About Eclipse;
2. Click on the Evidence Logo. The list of Eclipse plugins provided by Evidence will appear.
3. In the version column, you will find a version code in the format:

`release_series.date_version`

As an example, the code:

`3.0.0.20180328_gh30`

refers to:

**3.0.0** ...which is ERIKA3;

**20180328** ...which is the build day, 28th March, 2018;

**GH30** ...which is the build number for the GitHub series, build number 30.

**ATTENTION:** the version of ERIKA contained in this Virtual Machine is an unofficial version (i.e. version “GH41”) that solves some issues with MMU and cache enabling on the Jetson platforms. These changes will be integrated into the next official release of ERIKA.

For more information about ERIKA3 version numbers, please refer to the following link:

[http://www.erika-enterprise.com/wiki/index.php?title=Release\\_schedule\\_and\\_build\\_numbers](http://www.erika-enterprise.com/wiki/index.php?title=Release_schedule_and_build_numbers)