17-10-2019

Ayala Barbosa Jose Antonio

IIMAS PCIC UNAM

Tutorial

Puesta a punto Jetson TX2 y adición de GPUart al workspace

Tabla de contenido

[Puesta a punto de Jetson TX2 y Host Ubuntu 2](#_Toc22679665)

[Instalación de java en Host y Target 2](#_Toc22679666)

[Instalación de cuda en Host y Target: 2](#_Toc22679667)

[Instalación del CUDA Cross-Platform Environment 4](#_Toc22679668)

[Instalación del lado Host 5](#_Toc22679669)

[Banderas para compilación de NVCC 5](#_Toc22679670)

[Herramientas 6](#_Toc22679671)

[Referencias 6](#_Toc22679672)

[Importación de proyecto GPUart 7](#_Toc22679673)

[Referencias 18](#_Toc22679674)

# Puesta a punto de Jetson TX2 y Host Ubuntu

## Instalación de java en Host y Target

Es necesario instalar la versión 8 de java jdk y jre para el correcto funcionamiento de los componentes CUDA.

|  |
| --- |
| sudo apt-get update |
| sudo apt-get install openjdk-8-jdk |
| sudo apt-get install openjdk-8-jre-headless |

## Instalación de cuda en Host y Target:

Crear una cuenta de desarrollador en NVIDIA.

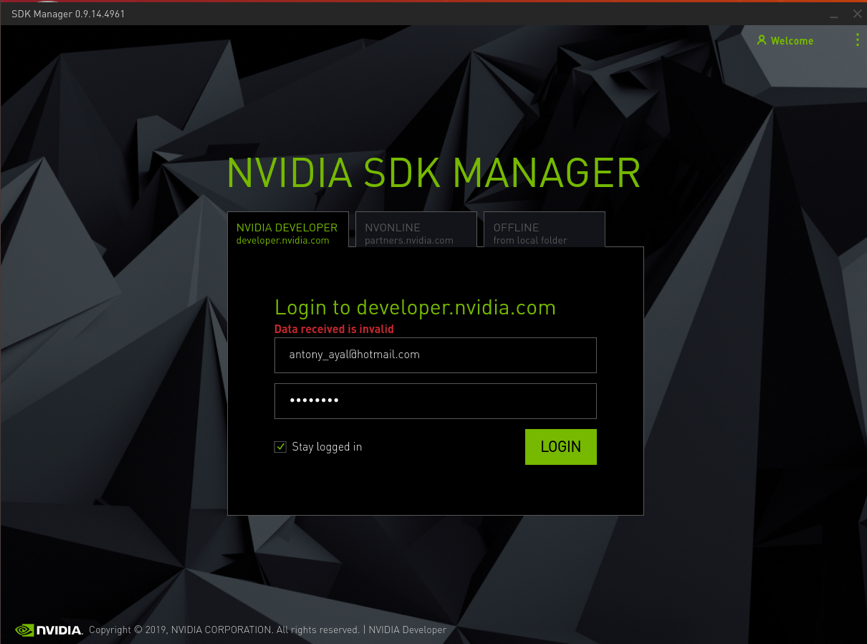
Instalción de SDK

Descargar NVIDIA JetPack SDK

|  |
| --- |
| <https://developer.nvidia.com/embedded/jetpack> |

Abrir el ejecutable descargado.

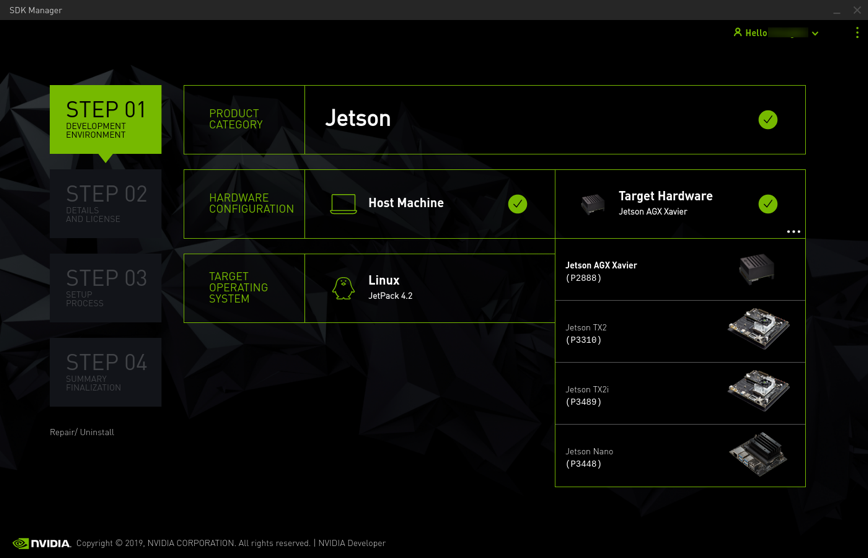
Loguearse y seguir instrucciones para instalar.



\* Es importante instalar la misma versión tanto en el Host (Ubuntu) y el target (TX2).

Paso 1:

* Seleccionar en Product Category: **Jetson.**
* Seleccionar Host Machine para instalar CUDA en el Host y seleccionar el sistema operativo.
* Seleccionar en Target Hardware Jetson TX2.

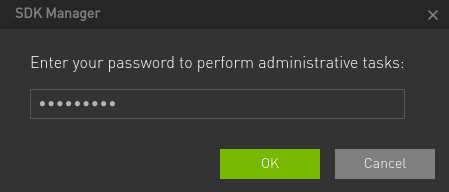


Paso 2:

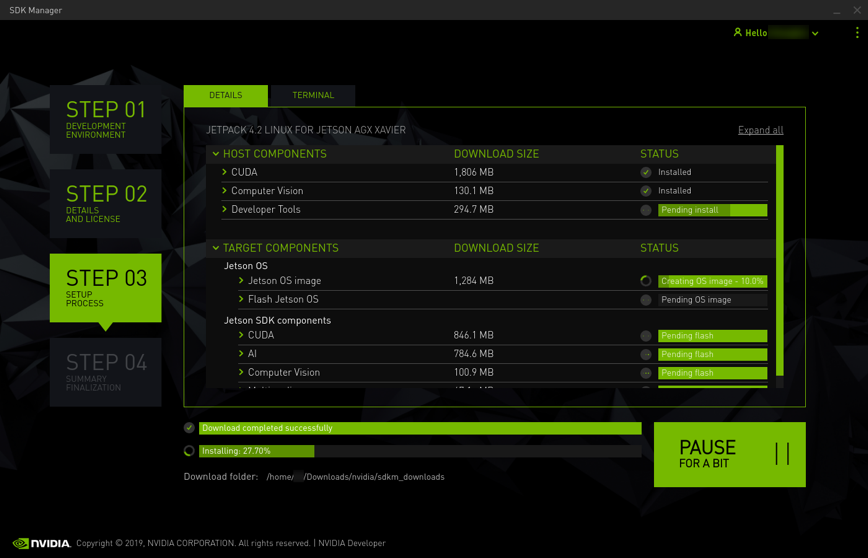
* Seleccionar los directorios para desargar e instalar.
* Aceptar los terminos y condiciones de la licencia.
* Click en continuar.

Paso 3:

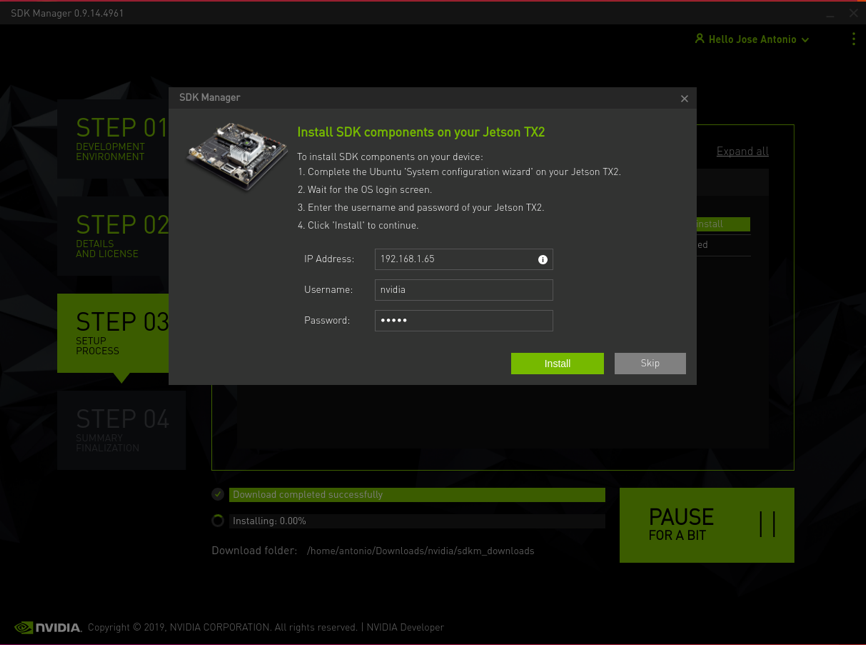
* Ingresar la contraseña sudo del host.



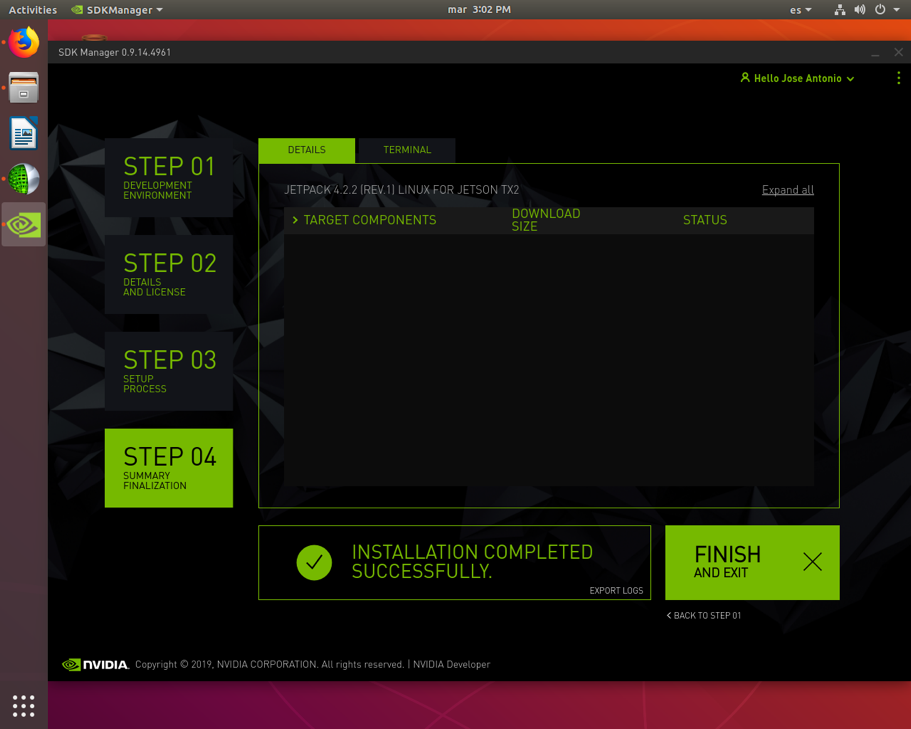
* Se mostrará el progreso de la instalación.



* Ingresar la Dirección IP, el Usuario y la Contraseña del sistema Target.



* Esperar a que se termine la instalación.



* Finalmente reiniciar ambos sistemas.

## Instalación del CUDA Cross-Platform Environment

Para que el host pueda compilar el código fuente que se ejecutará en el target es necesaria la instalación de arquitecturas foraneas.

En el host realizar lo siguiete.

* x86\_64: 64-bit x86 CPU architecture;

|  |
| --- |
| wget -c "https://developer.nvidia.com/compute/cuda/8.0/Prod2/local\_installers/cuda-repo-ubuntu1604-8-0-local-ga2\_8.0.61-1\_amd64-deb" |
| sudo dpkg --install cuda-repo-ubuntu1604-8-0-local-ga2\_8.0.61-1\_amd64-deb |
| sudo apt-get update |
| sudo apt-get install cuda |

* aarch64: 64-bit ARM CPU architecture, found on Jetson TX1 and TX2 and certain Android systems.

|  |
| --- |
| sudo dpkg --add-architecture arm64 |
| sudo apt-get update |

Instalación de paquetes cross-platform.

|  |
| --- |
| sudo apt-get install cuda-cross-aarch64 |
| sudo apt-get install g++-4.8-aarch64-linux-gnu |
| sudo apt-get install gcc g++ git gcc-\*-aarch64-linux-gnu g++-\*-aarch64-linux-gnu libncurses5-dev |

Reiniciar el host y posteriormente actualizar el path:

|  |
| --- |
| export PATH=/usr/local/cuda-8.0/bin:$PATH |
| export LD\_LIBRARY\_PATH=/usr/local/cuda-8.0/lib64:$LD\_LIBRARY\_PATH |

Verificar que en folder:

|  |
| --- |
| /usr/local/cuda-8.0/targets/ |

Se encuentren las arquitecturas:

* aarch64-linux
* x86\_64\_linux

## Instalación del lado Host

Instalación de Cuda

Para Ubuntu 16 o posterior

## Banderas para compilación de NVCC

En la siguiente liga se pueden encontrar las diferentes banderas con las que se pueden activar las diferentes modalidades del compilador NVCC

|  |
| --- |
| <https://blog.csdn.net/panda1234lee/article/details/84564540> |

O activando la bandera:

|  |
| --- |
| nvcc --help |

## Referencias

<https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/scripts/ubuntu/install_cuda.sh>

<https://devblogs.nvidia.com/cuda-jetson-nvidia-nsight-eclipse-edition/>

<https://askubuntu.com/questions/671695/solving-the-package-dependency-list-automatically-for-building-ubuntu-touch>

<https://askubuntu.com/questions/671695/solving-the-package-dependency-list-automatically-for-building-ubuntu-touch>

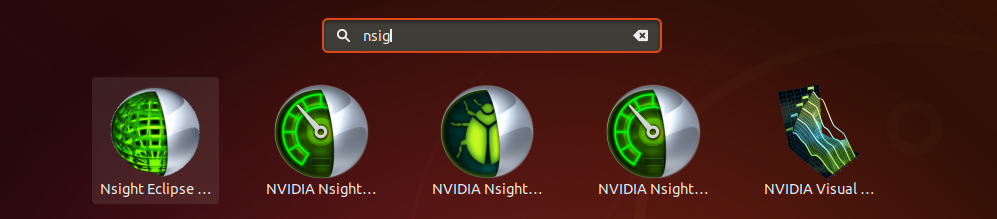
<https://stackoverflow.com/questions/4034392/makefile-error1>

# Importación de proyecto GPUart

En el Target crear una carpeta con el nombre del Proyecto.

|  |
| --- |
| /GPUart |

Abrir Nsight Eclipse, para ello buscarlo en la barra de actividades del escritorio



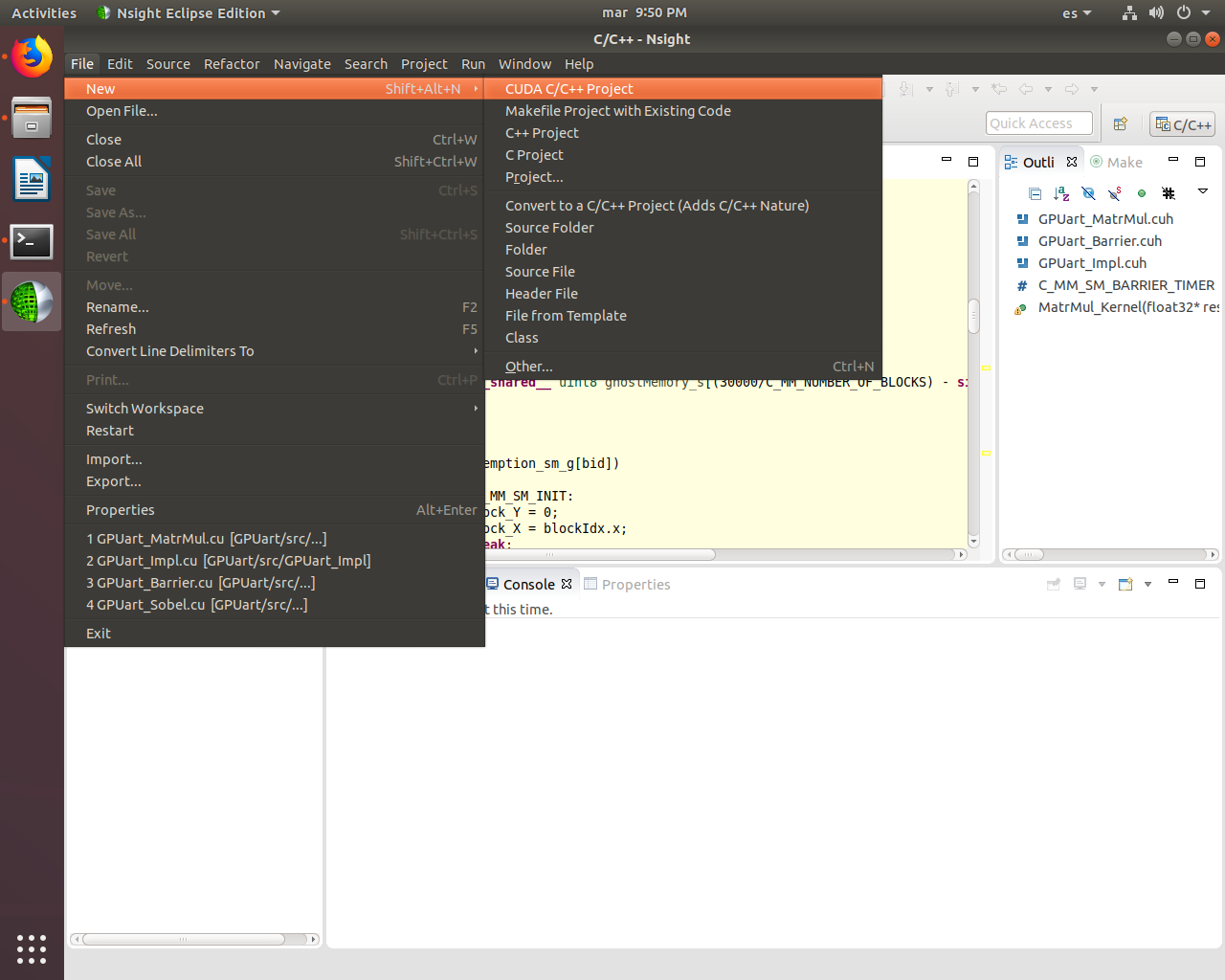


Seleccinar el workspace default.

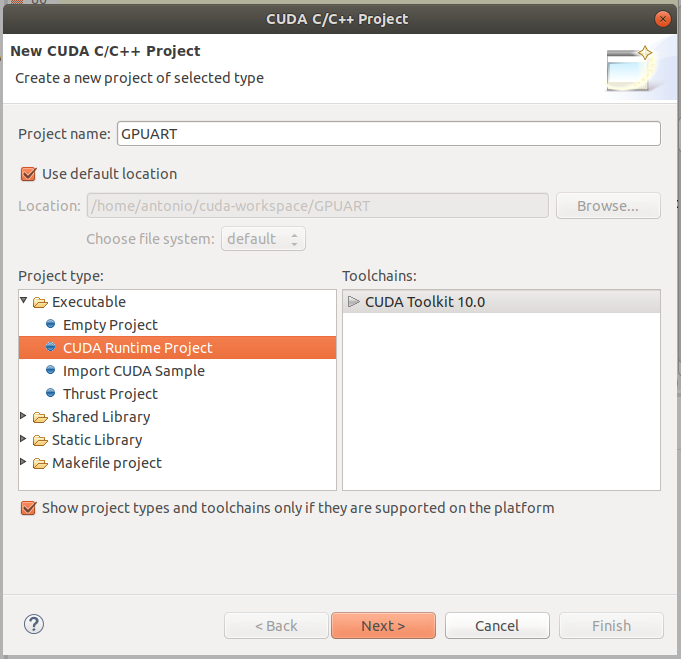


Crear un nuevo proyecto:

* File -> New -> CUDA C/C++ Project



* Escribir nombre del proyecto.
* Seleccionar el tipo CUDA Runtime Project.
* Dar click en siguiente.



* Device linker mode

|  |
| --- |
| Separate compilation. |

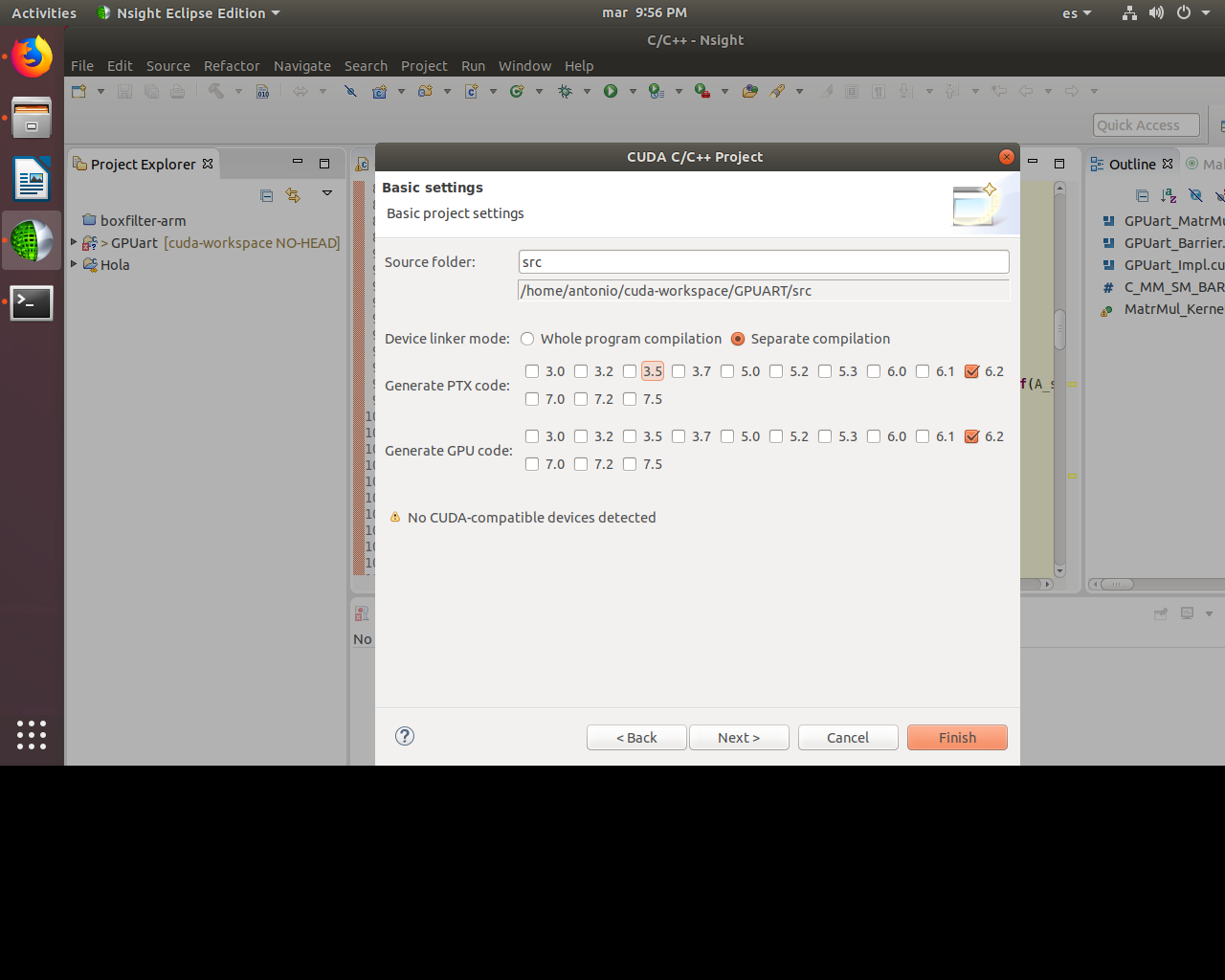
* Generate PTX code:

|  |
| --- |
| 6.2 |

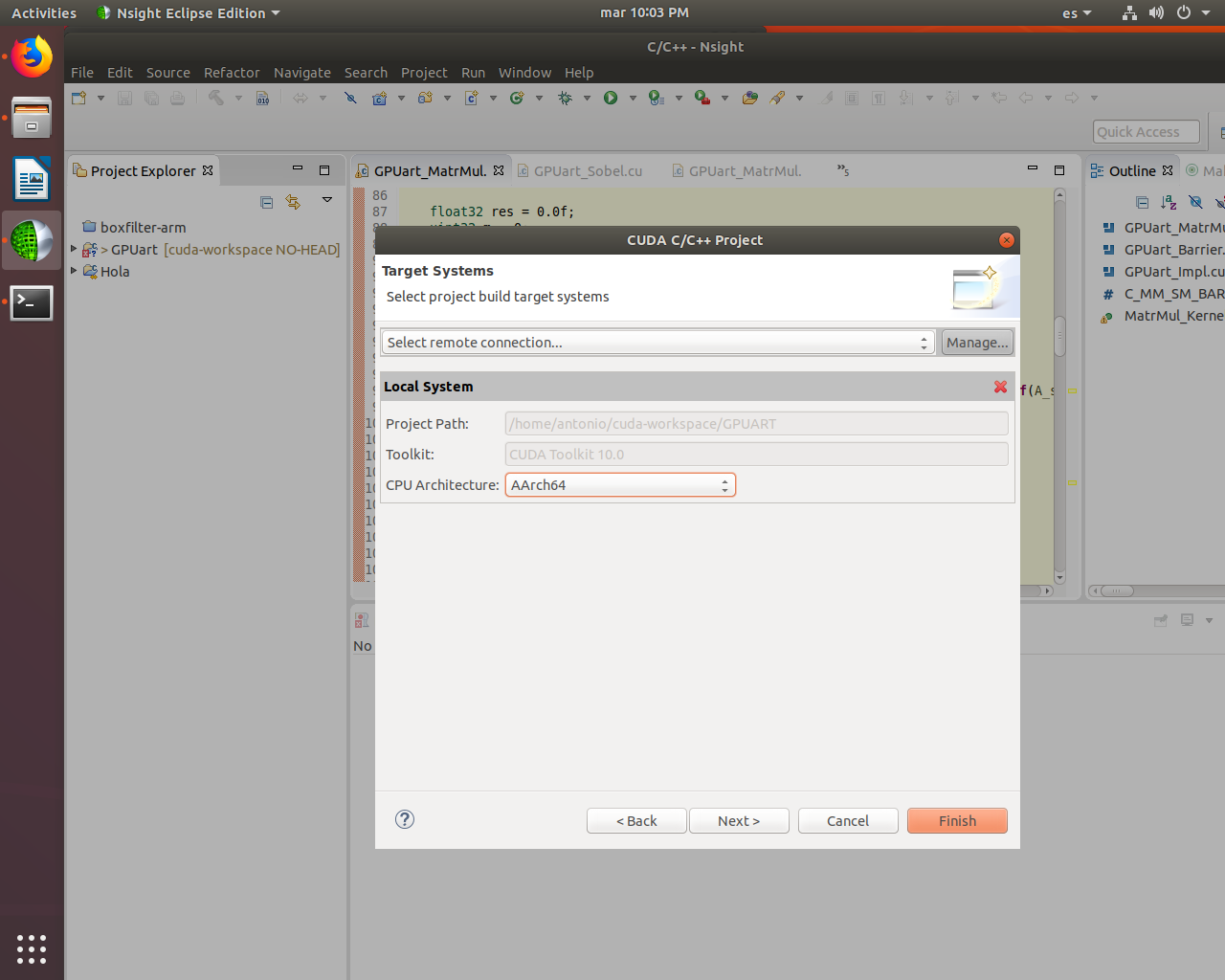
* Generate GPU code:

|  |
| --- |
| 6.2 |

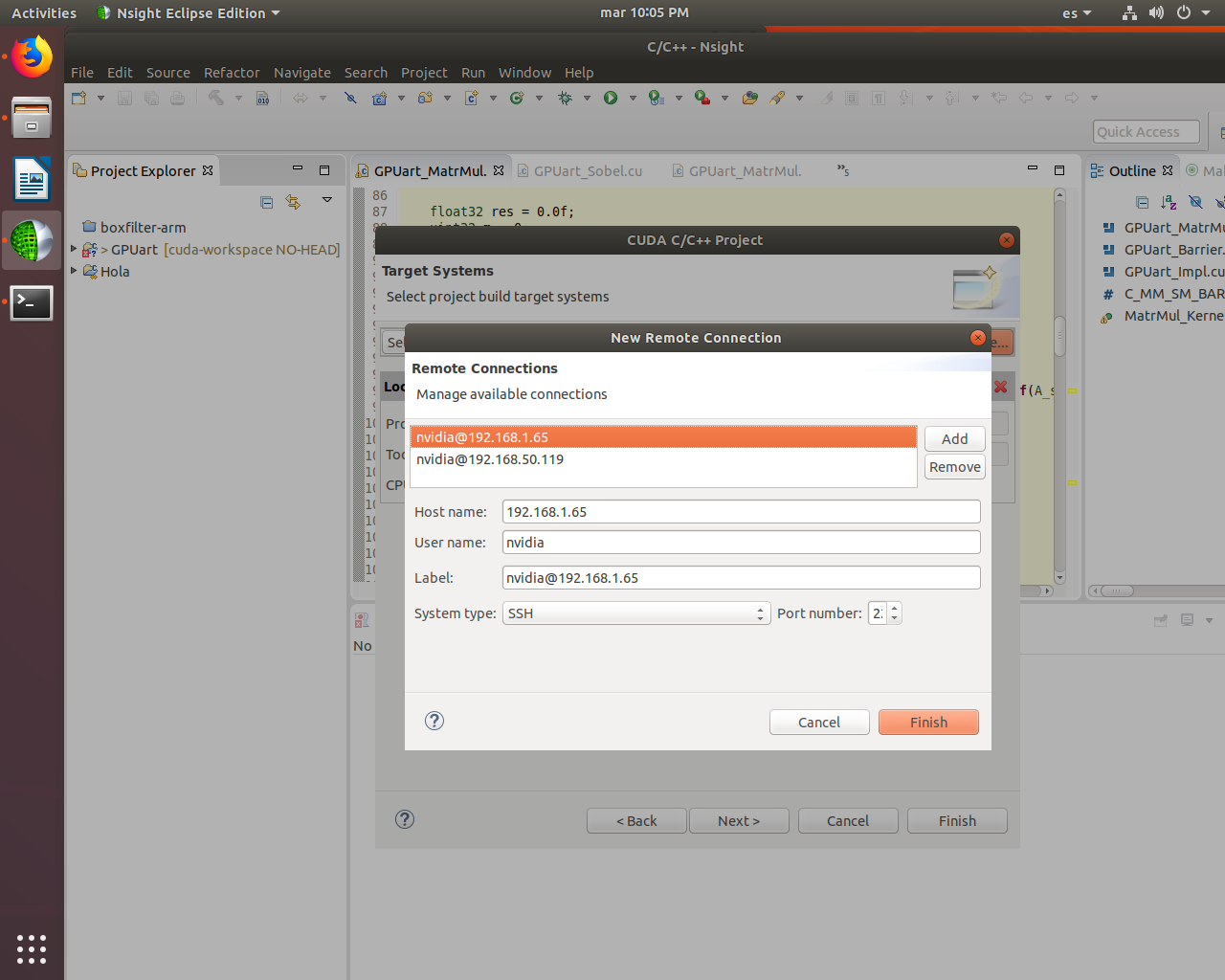
* Click en siguiente



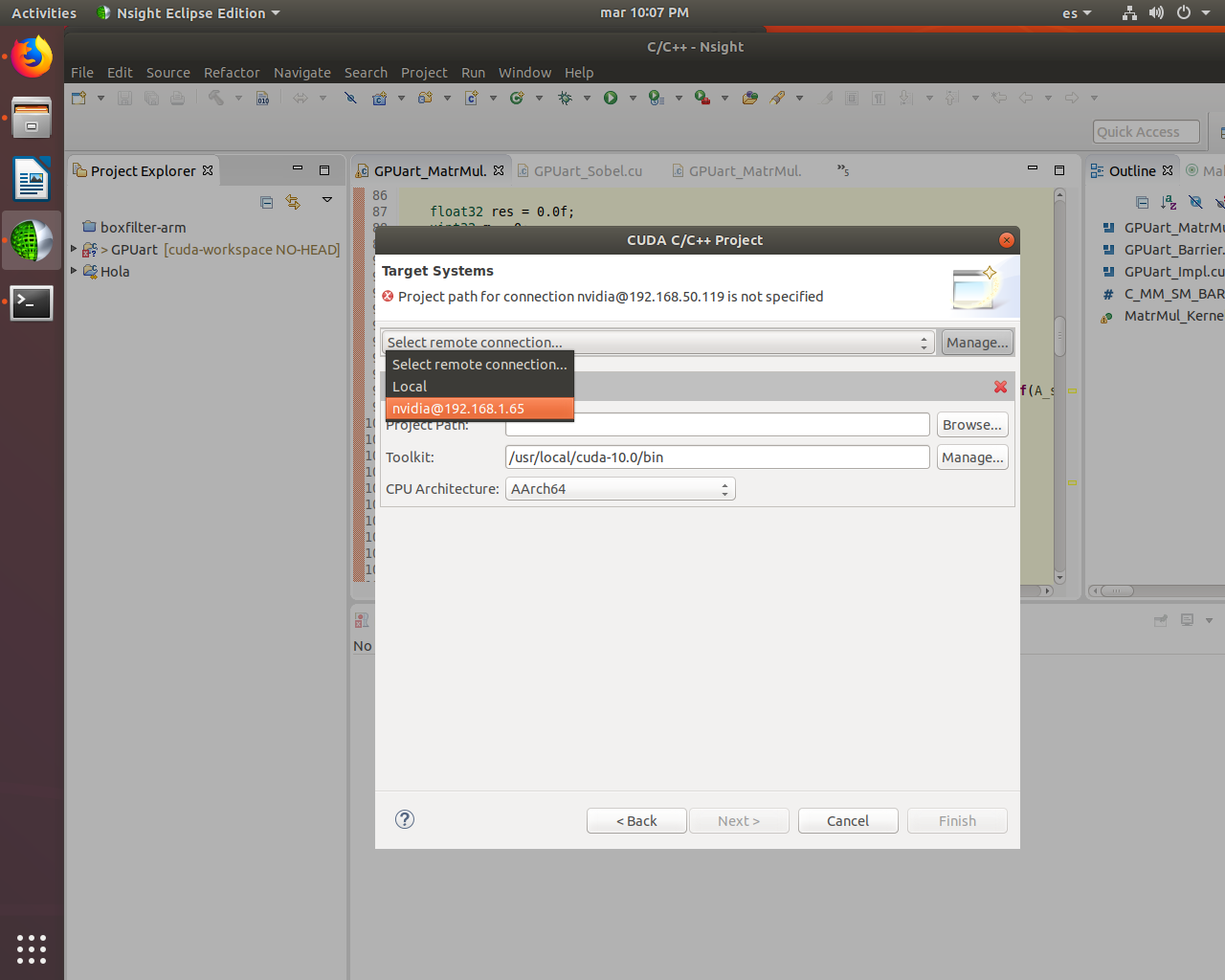
* En el botón Manage click.



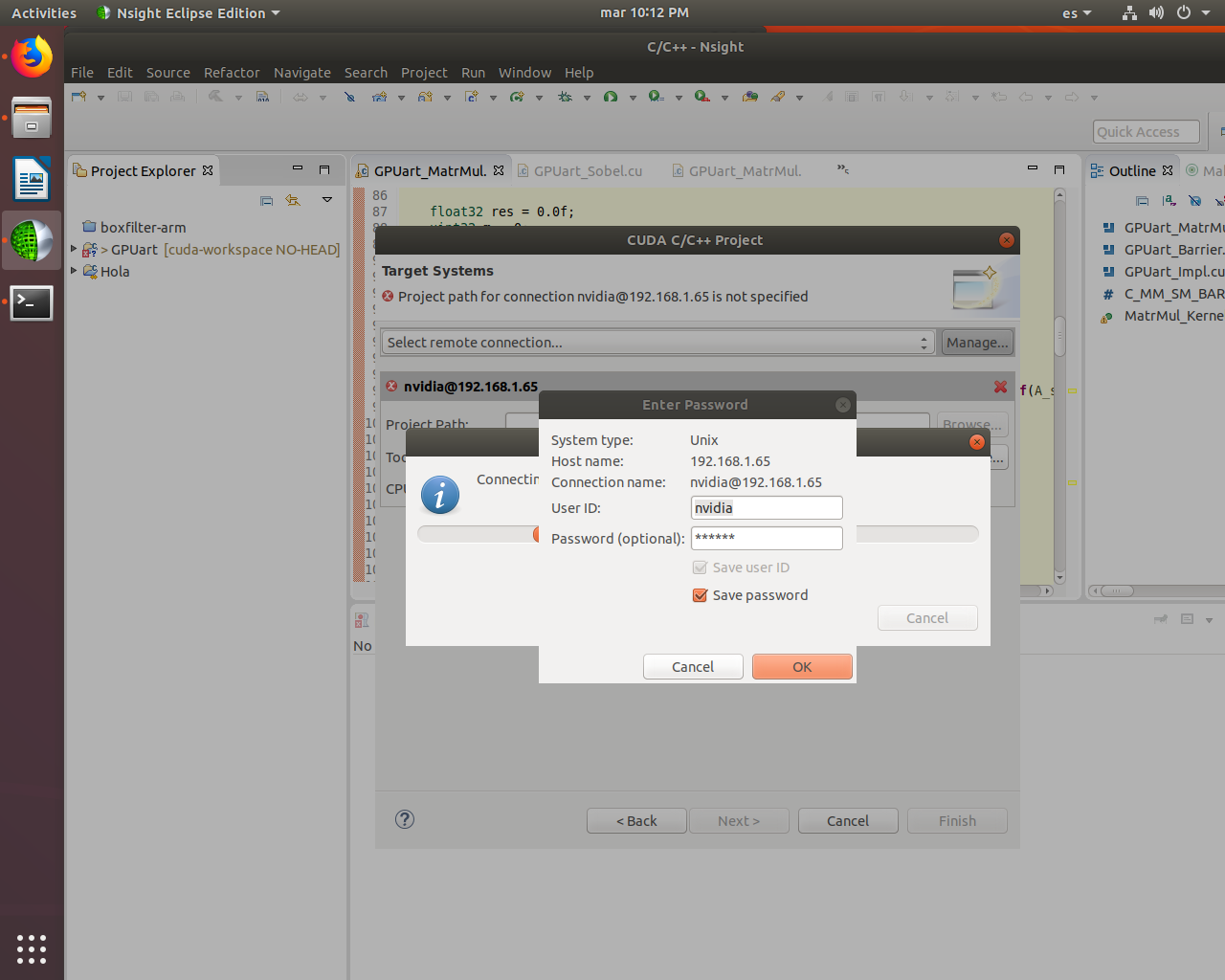
* Llenar los campos con los datos de la dirección y credenciales del Target.



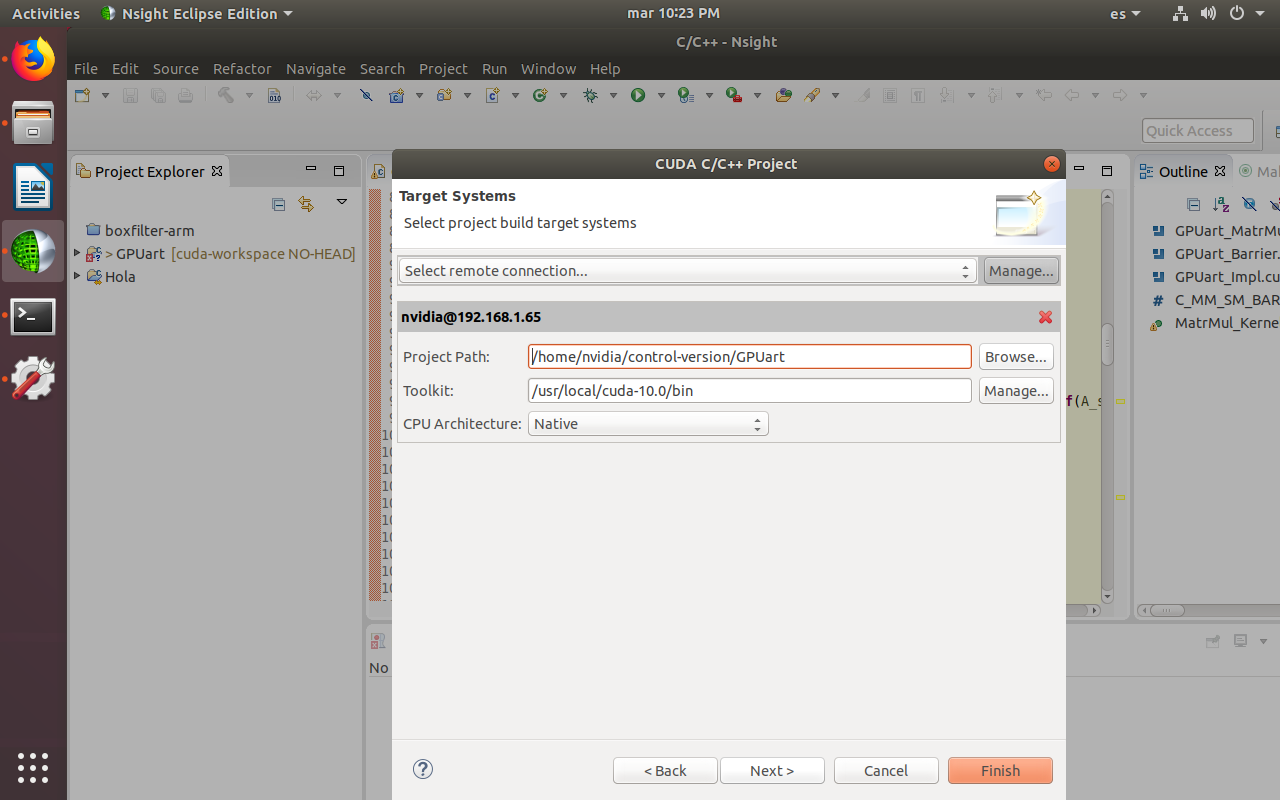
* Seleccionar la conexión remota del Target.



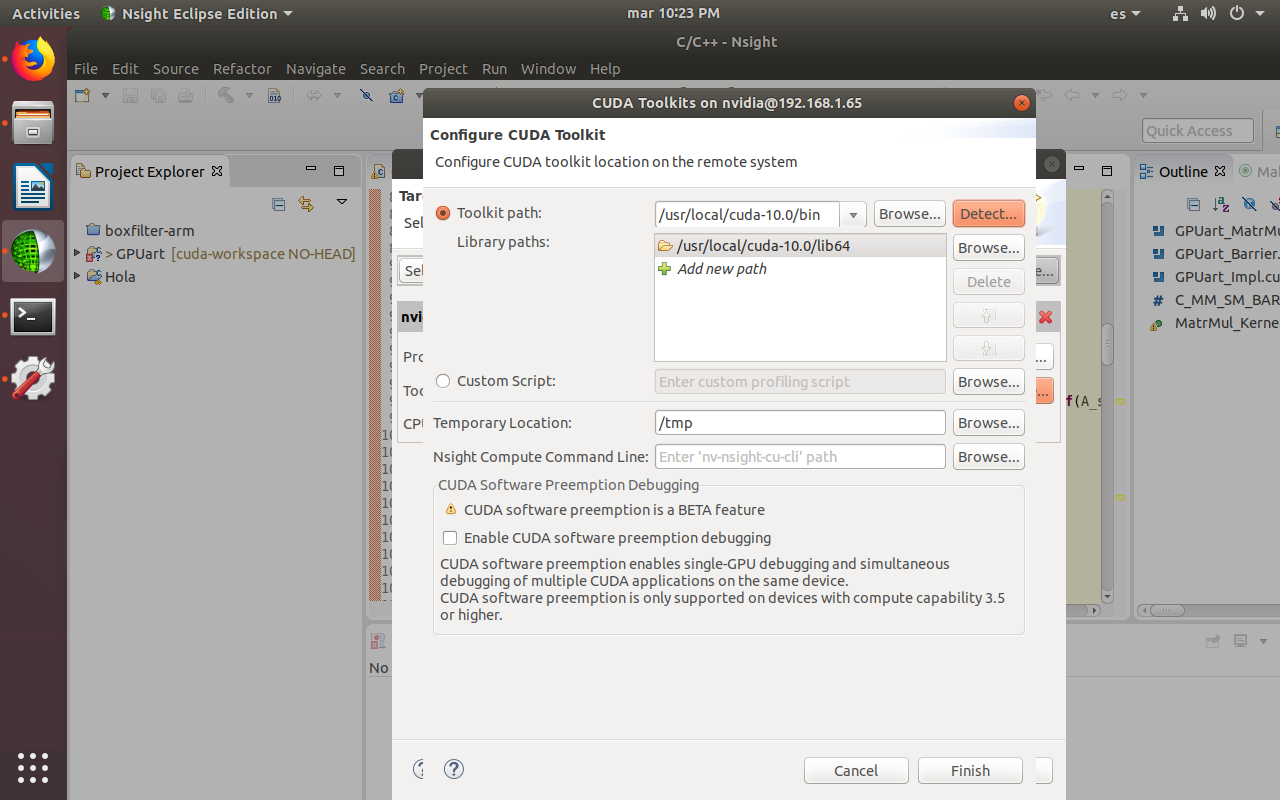
* Dar click en el botón de Browse en la caja de Project Pack.
* Confirmar contraseña del Target.



* Dar click en siguiente.
* En el botón browse del Project Path, seleccionar la dirección en que se creó la carpeta del proyecto en el Target.



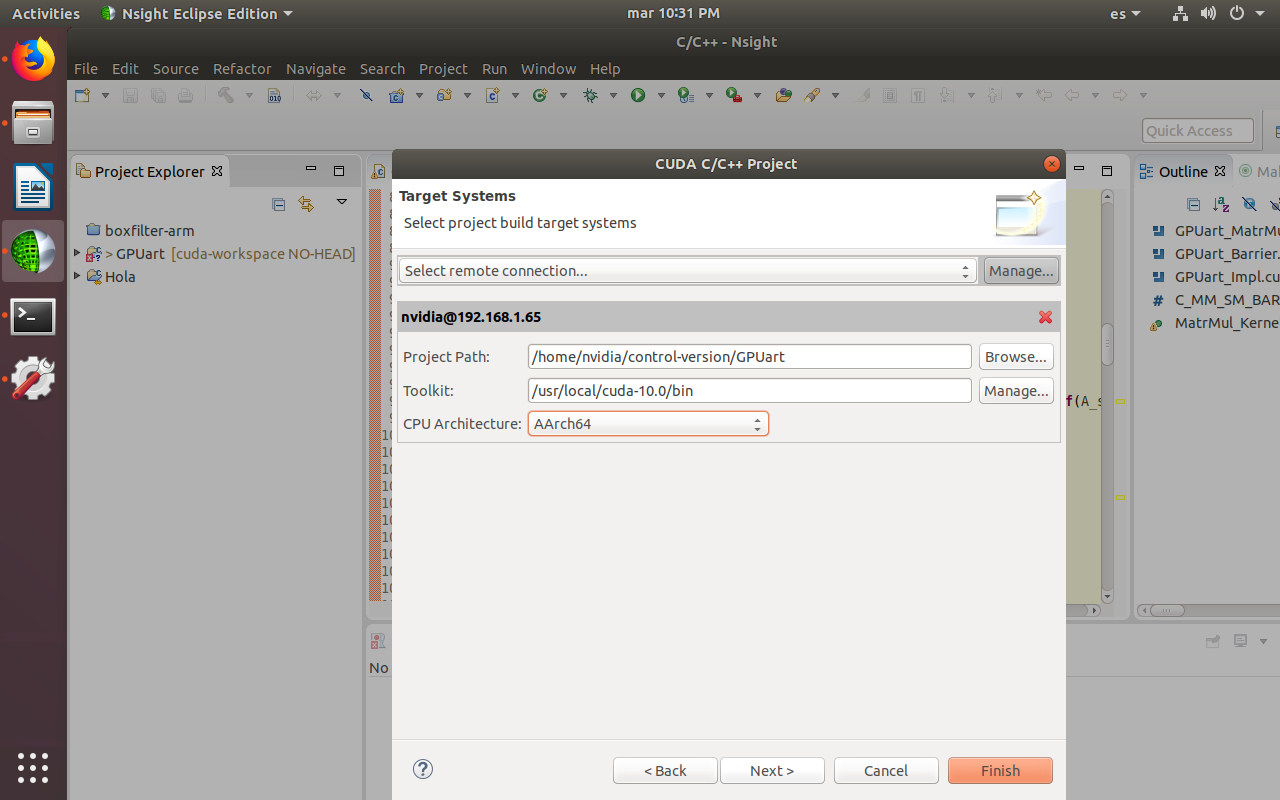
* Dar click en el Botón Manage de la opción Toolkit.
* En la nueva ventana presionar el botón Detect, lo cual llenará automáticamente las opciones.
* Dar click en Finish.



* Seleccionar para el target la arquitectura CPU:

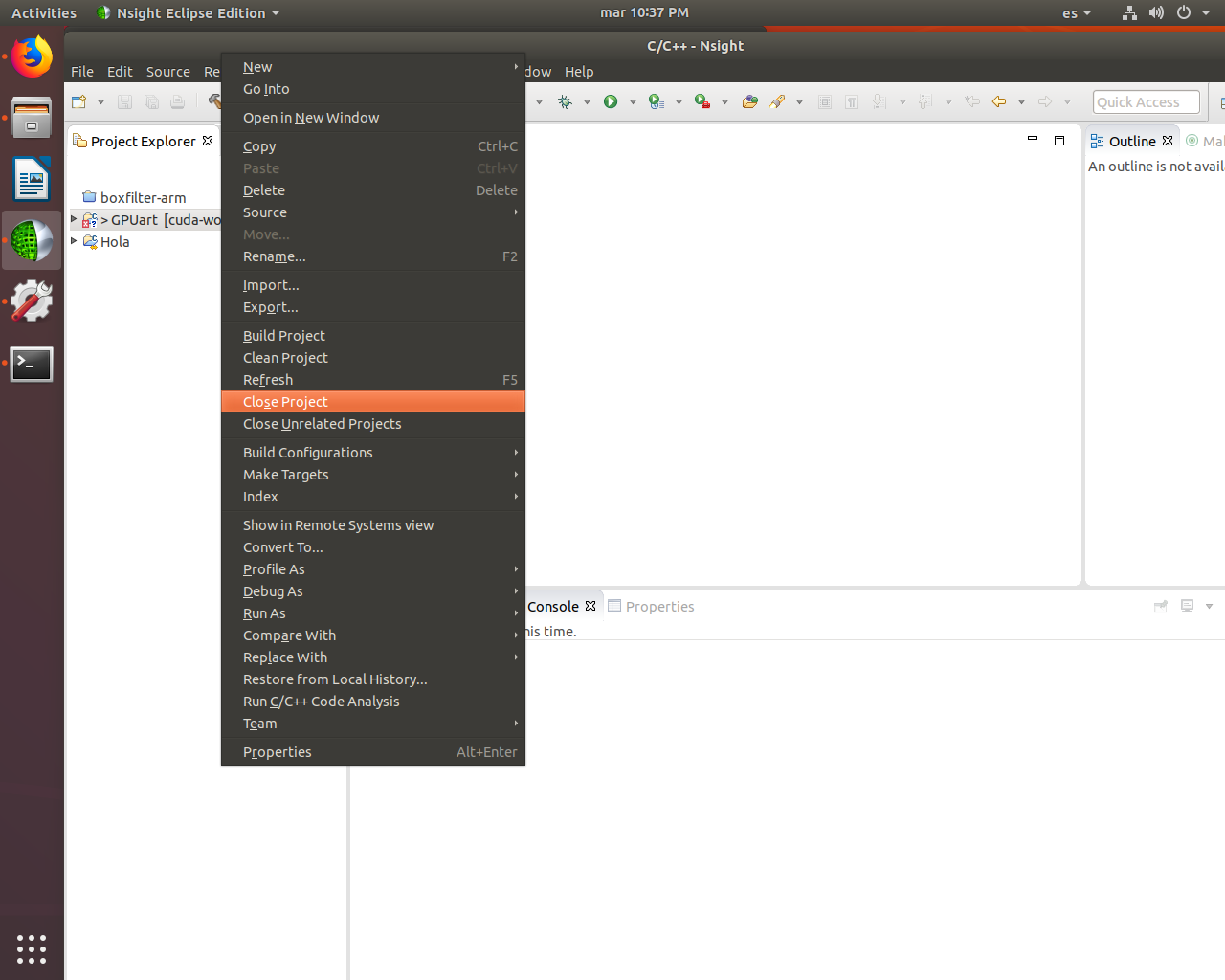
|  |
| --- |
| Aarch64 |

* Dar click en Finalizar.



Una vez creado el proyecto:

* Seleccionarlo en el explorador de proyectos.
* Botón derecho.
* Cerrar proyecto.

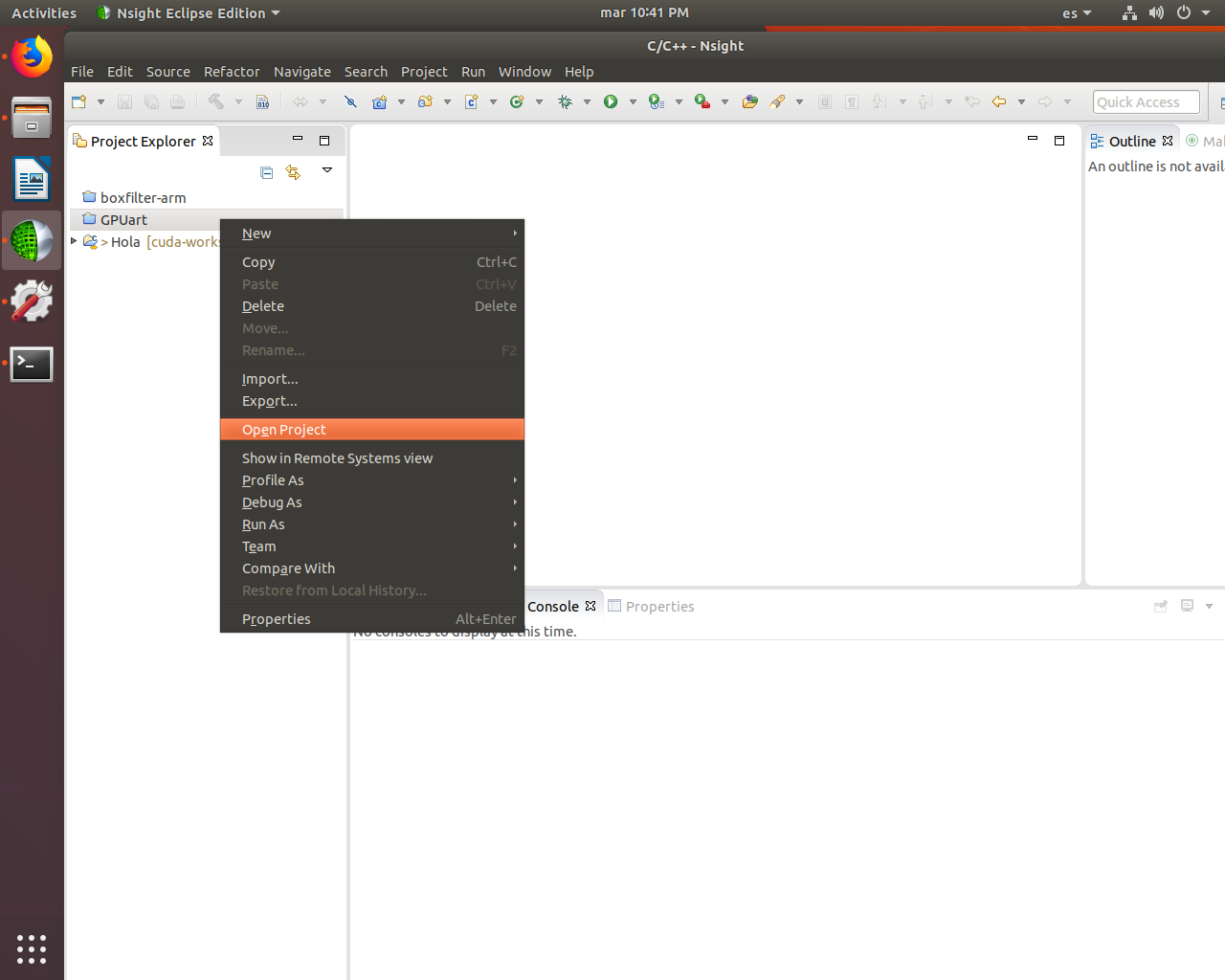


Abrir la terminal y descargar el proyecto del github de GPUart.

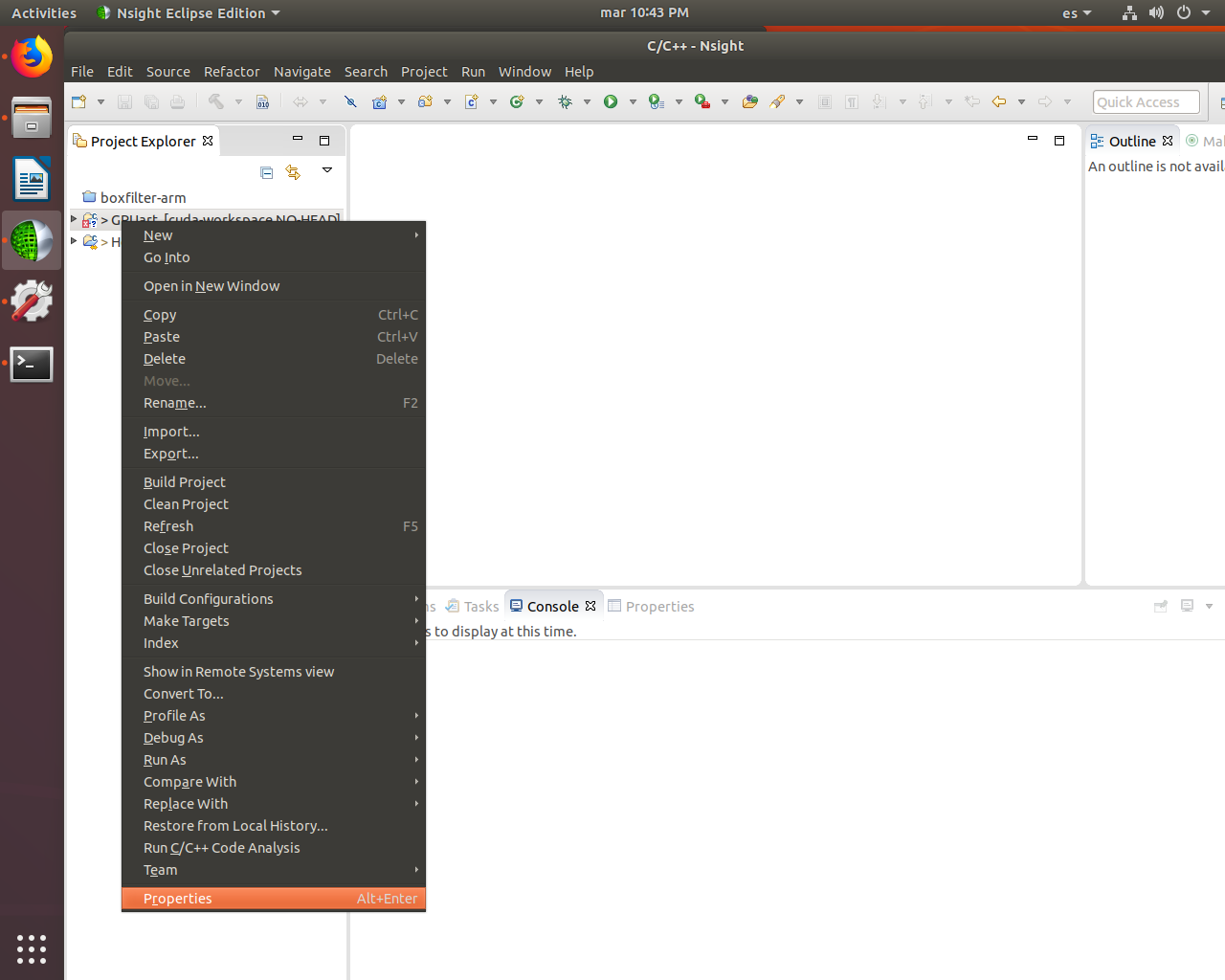
|  |
| --- |
| git clone https://github.com/gpuart/GPUart.git |

Sustituir el contenido de la carpeta del proyecto creado por Eclipse Nsight por el del proyecto descargado.

En eclipse volver a abrir el proyecto.



Click derecho al proyecto y seleccionar Properties.



En la opción Build -> Settings -> NVCC Compiler Sustituir la información:

* Command:

|  |
| --- |
| nvcc -lcudadevrt |

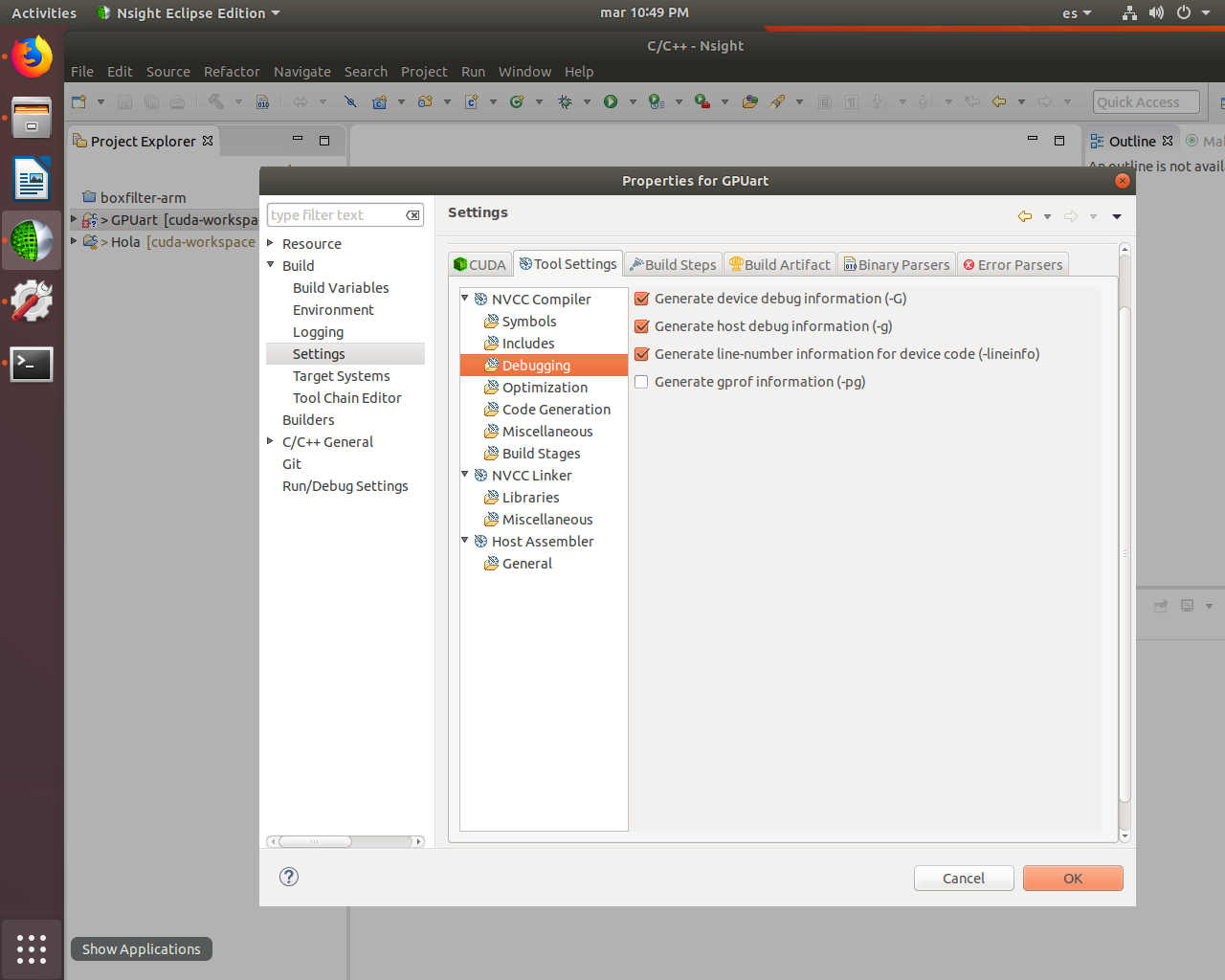
* All options

|  |
| --- |
| -G -g -lineinfo -O0 -m64 -ccbin aarch64-linux-gnu-g++ |

* Command line pattern:

|  |
| --- |
| ${COMMAND} ${FLAGS} -gencode arch=compute\_62,code=sm\_62 ${OUTPUT\_FLAG} ${OUTPUT\_PREFIX} ${OUTPUT} ${INPUTS} |

En la opción Build -> Settings -> NVCC Compiler -> Debugging Seleccionar las banderas:



En la opción Build -> Settings -> NVCC Linker Sustituir la información:

* Command:

|  |
| --- |
| nvcc |

* All options

|  |
| --- |
| --cudart static -Llinux/aarch64 -Xlinker --unresolved-symbols=ignore-in-shared-libs --relocatable-device-code=true -gencode arch=compute\_62,code=compute\_62 -gencode arch=compute\_62,code=sm\_62 -m64 -ccbin aarch64-linux-gnu-g++ |

* Command line pattern:

|  |
| --- |
| ${COMMAND} ${FLAGS} -gencode arch=compute\_62,code=sm\_62 ${OUTPUT\_FLAG} ${OUTPUT\_PREFIX} ${OUTPUT} ${INPUTS} -lcudadevrt |

En la opción Build -> Settings -> NVCC Linker -> Libraries -> Library search path (-L) Agregar:

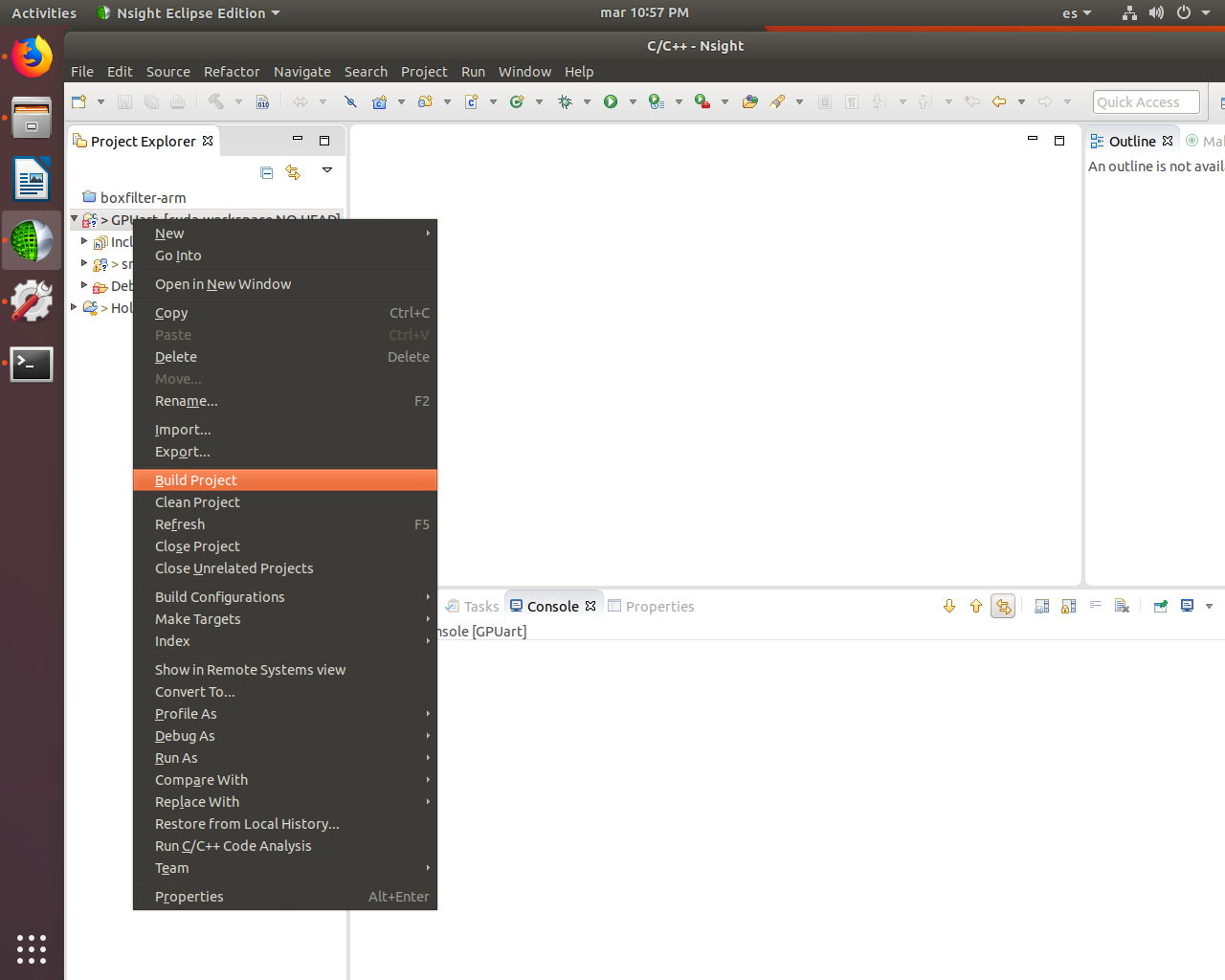
|  |
| --- |
| linux/aarch64 |

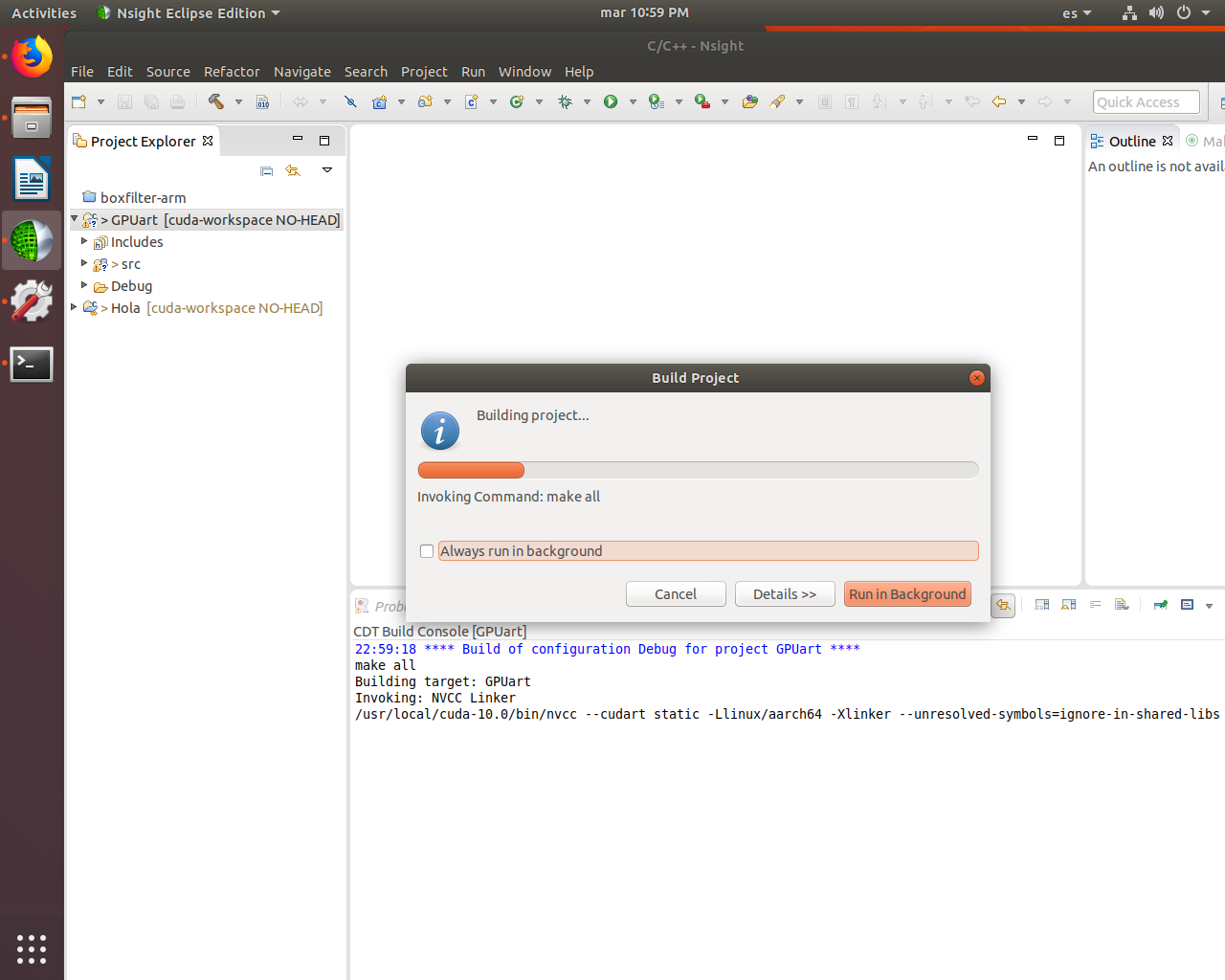
En la opción Build -> Settings -> NVCC Linker -> Miscellaneous -> Other options (-Xlinker) Agregar:

|  |
| --- |
| --unresolved-symbols=ignore-in-shared-libs |

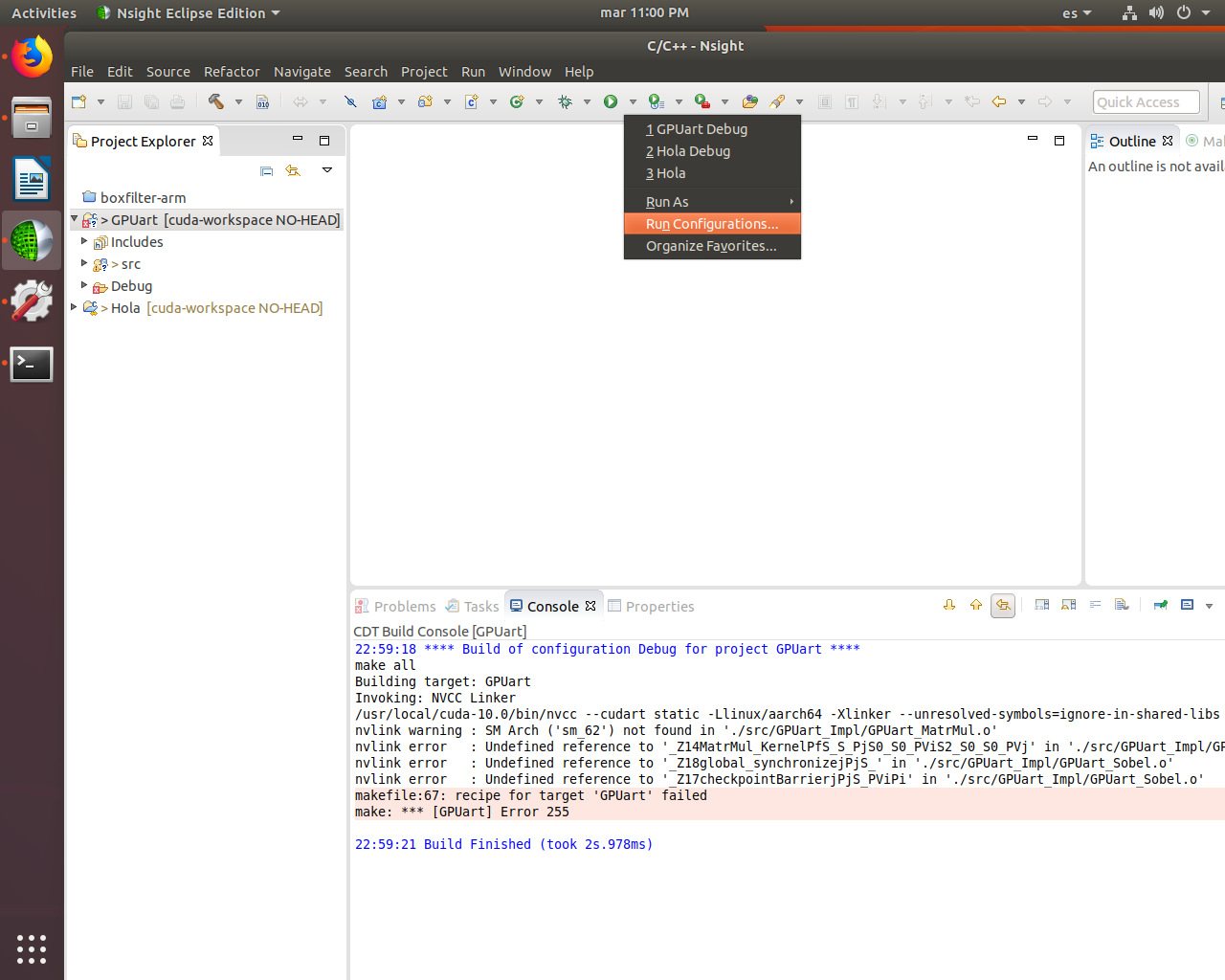
Dar click en OK.

Dar click derecho al Proyecto y seleccionar la opción Build Project.





En el boton de Run, seleccionar las opciones y Run Configurations.



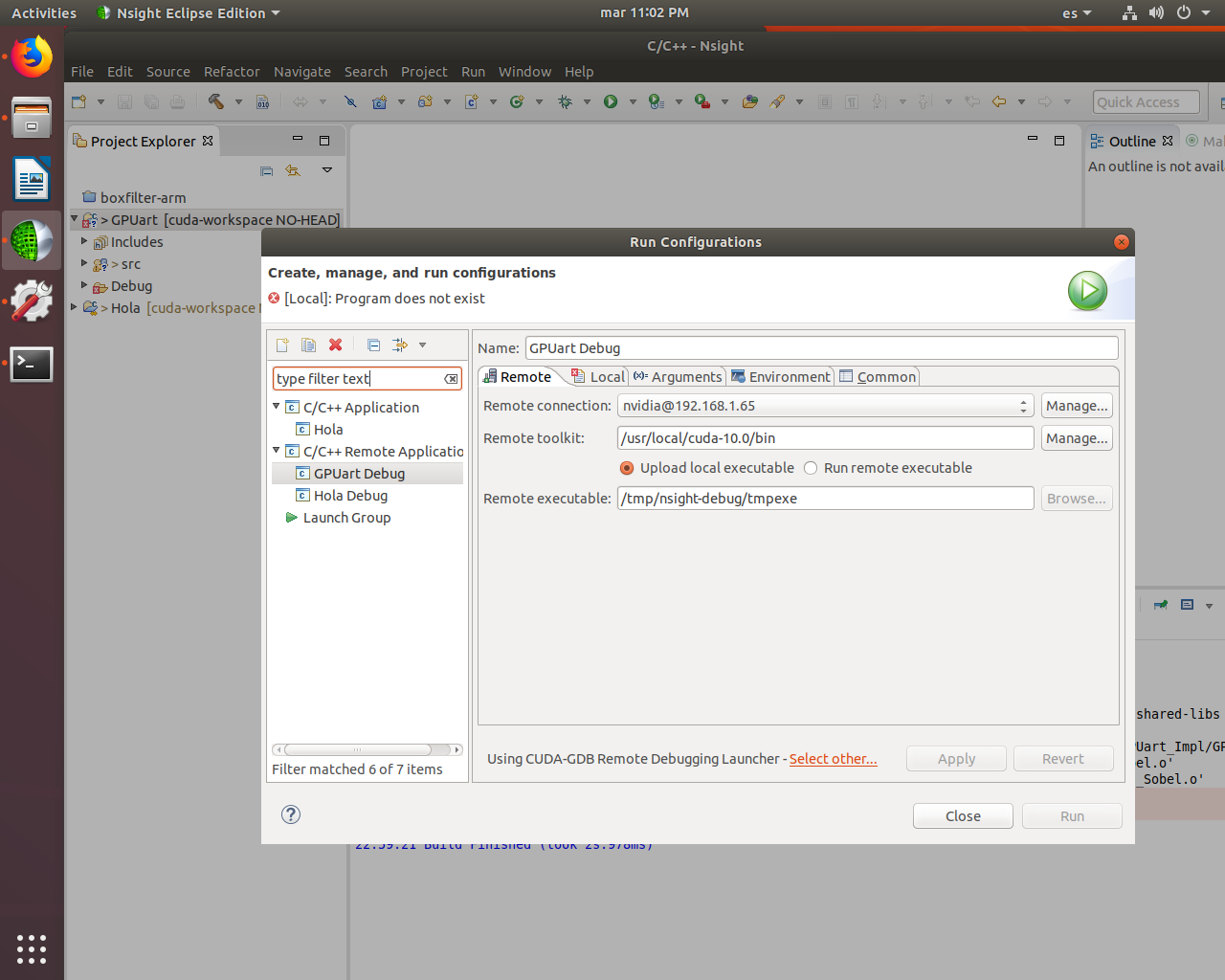
En la pestaña Remote

* Seleccionar la dirección del Target.
* Seleccionar el Toolkit por defecto.
* Seleccionar la opción:

|  |
| --- |
| Upload local executable |

* En Remote executable agregar la dirección:

|  |
| --- |
| /tmp/nsight-debug/tmpexe |



En la pestaña Local, modificar los siguientes parametros:

* C/C++ Application:

|  |
| --- |
| Debug/GPUart |

* Project:

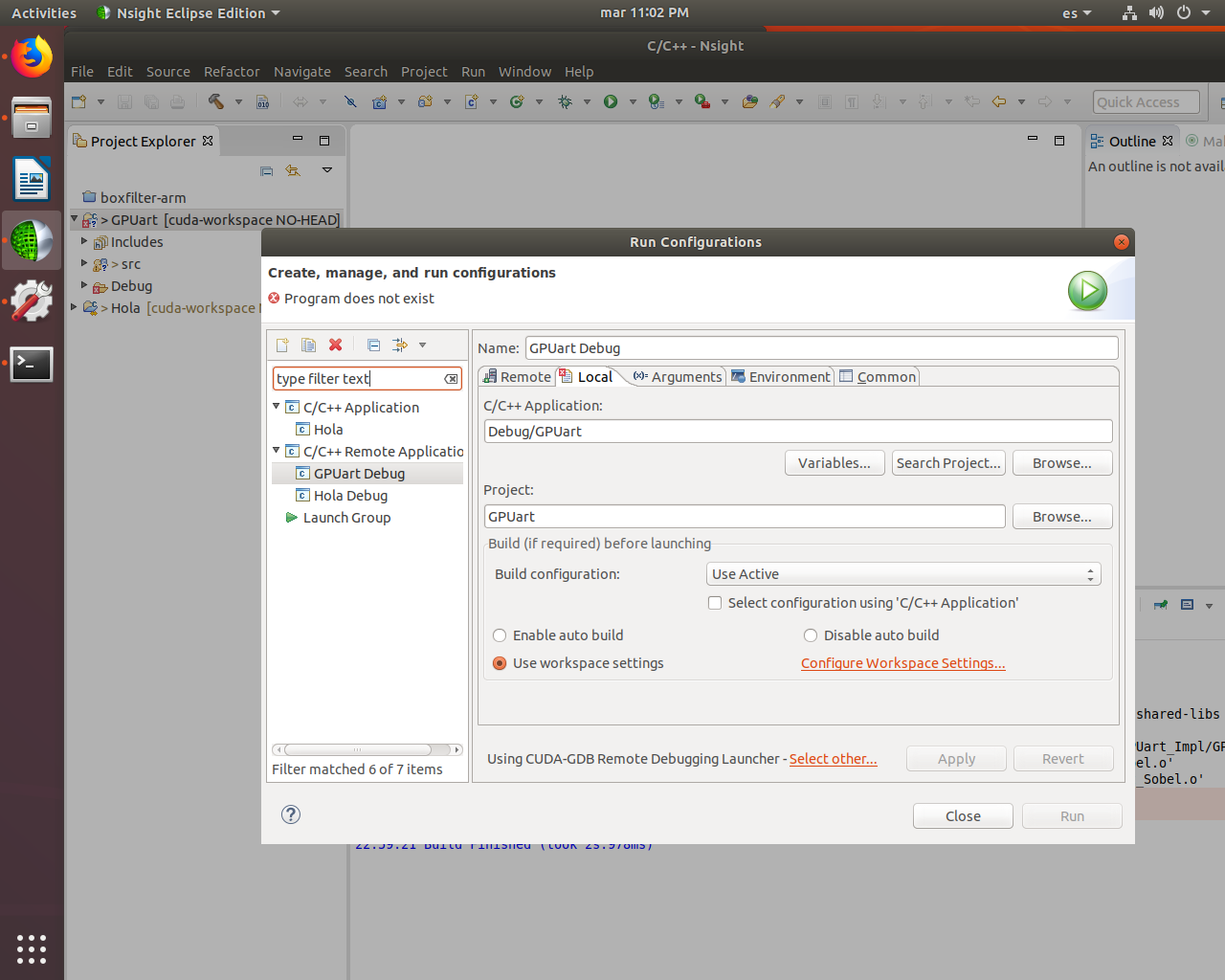
|  |
| --- |
| GPUart |

* Build Configuration:

|  |
| --- |
| Use Active |

* Seleccionar la opción

|  |
| --- |
| Use workspace settings |



Dar clicl en Apply.

Dar click en Run.

## Referencias

<https://devblogs.nvidia.com/cuda-jetson-nvidia-nsight-eclipse-edition/>

<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>

<https://docs.nvidia.com/nsight-compute/NsightComputeCli/index.html#quick-start>

<https://devtalk.nvidia.com/default/topic/1026470/using-nsight-running-cuda-8-0-0_simple-samples-cuda_print-show-error-compile-cross-tx2/>

<https://devtalk.nvidia.com/default/topic/1017827/jetson-tx2-cross-compile-error/>

<https://stackoverflow.com/questions/56767294/dynamic-parallelism-throwing-error-for-teslak80-gpu>

<https://stackoverflow.com/questions/38260577/generating-relocatable-device-code-using-nvidia-nsight>

<https://devtalk.nvidia.com/default/topic/847969/nsight-visual-studio-edition/how-to-specify-executable-for-graphics-debugging/>

<https://stackoverflow.com/questions/16945627/unknown-nvlink-error>

<https://devtalk.nvidia.com/default/topic/1010551/jetson-tx2/using-nsight-eclipse-edition-with-opencv-for-cross-compilation-/post/5175091/#5175091>

<https://devtalk.nvidia.com/default/topic/994137/nvcc-linker-error-while-compile-a-project-with-opencv/>

<https://devtalk.nvidia.com/default/topic/983098/jetson-tx1/opencv-3-1-with-usb-camera-support/>

Funciones virtuales

<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#virtual-functions>