Integración de Toolchain PTXdist sobre IDE gráfico basado en Eclipse

Objetivos

- Integrar un toolchain basado en PTXdist sobre un IDE gráfico basado en Eclipse.
- Creación y compilación de un pequeño proyecto.
- Debug de una aplicación sobre el módulo.

Requisitos

- Toolchain para cros-compilación.
- Entorno Eclipse.

Notas Adicionales

- En el guión se han utilizado como referencia el **MTX-GATEWAY** y el **MTX-GTW** de Matrix Electrónica.

Referencias

- http://matrixembebidos.wikispaces.com

Requisitos previos

El presente tutorial se ha llevado a cabo sobre un PC con Ubuntu 12.04 LTS.

Se ha utilizado un Toolchain de PTXdist. La instalación de este toolchain está descrita en la siguiente página: http://matrixembebidos.wikispaces.com/mtx-gateway-sw

Respecto al entorno de Eclipse, se ha utilizado la versión Indigo ya que incorpora el soporte para cross-compilación. Si se utiliza la opción sudo apt-get install Eclipse-platform sobre Ubuntu 12.04 se instalará una versión que no tiene este soporte. Por este motivo se recomienda descargar el paquete completo directamente del siguiente enlace:

http://www.Eclipse.org/cdt/downloads.php

Configuración del entorno

Para una correcta ejecución de eclipse con el toolchain de compilación se deben crear los siguientes ficheros de configuración:

- Archivo de configuración de entorno: **setup_mtx.sh**

```
PATH=$PATH:/opt/OSELAS.Toolchain-2013.12.2/arm-v5te-linux-gnueabi/gcc-4.8.2-glibc-2.18-binutils-2.24-kernel-3.12-sanitized/bin

export CROSS_COMPILE=arm-v5te-linux-gnueabi-
```

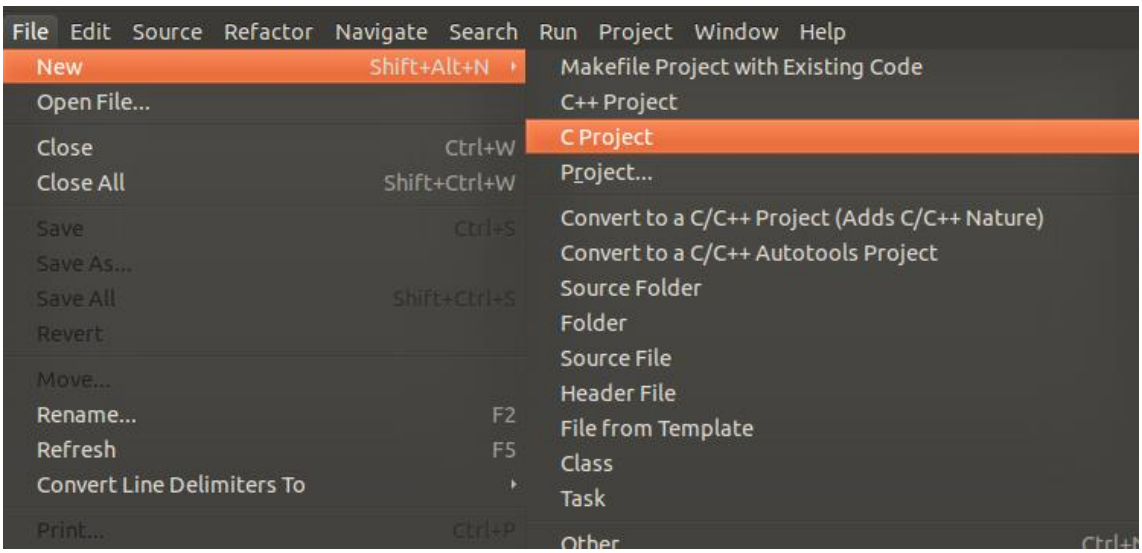
- Archivo de ejecución de eclipse: **eclipse_mtx.sh**

```
#!/bin/bash
source ~/setup_mtx.sh
cd ~/eclipse
./eclipse&
```

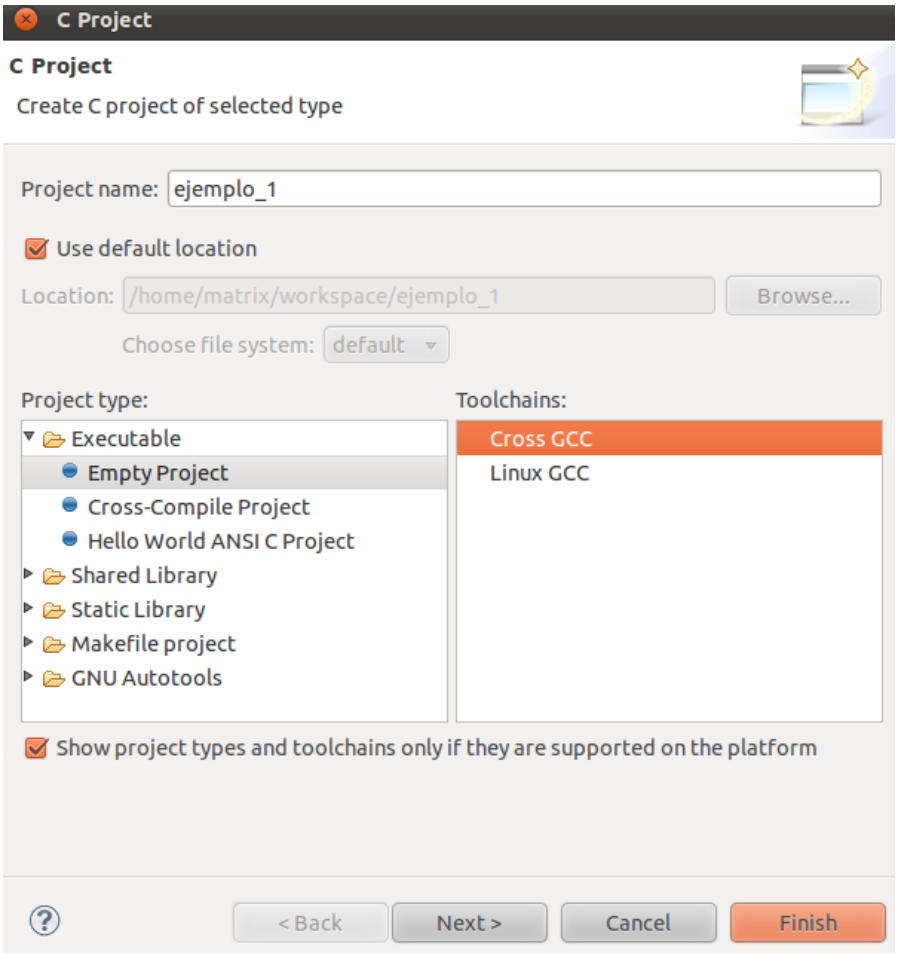
En este ejemplo se han creado ambos ficheros sobre la carpeta de usuario. Así mismo también se ha instalado eclipse directamente sobre la carpeta de usuario. Si se instala sobre otro directorio se deberán ajustar los paths correspondientes.

Creación de un proyecto

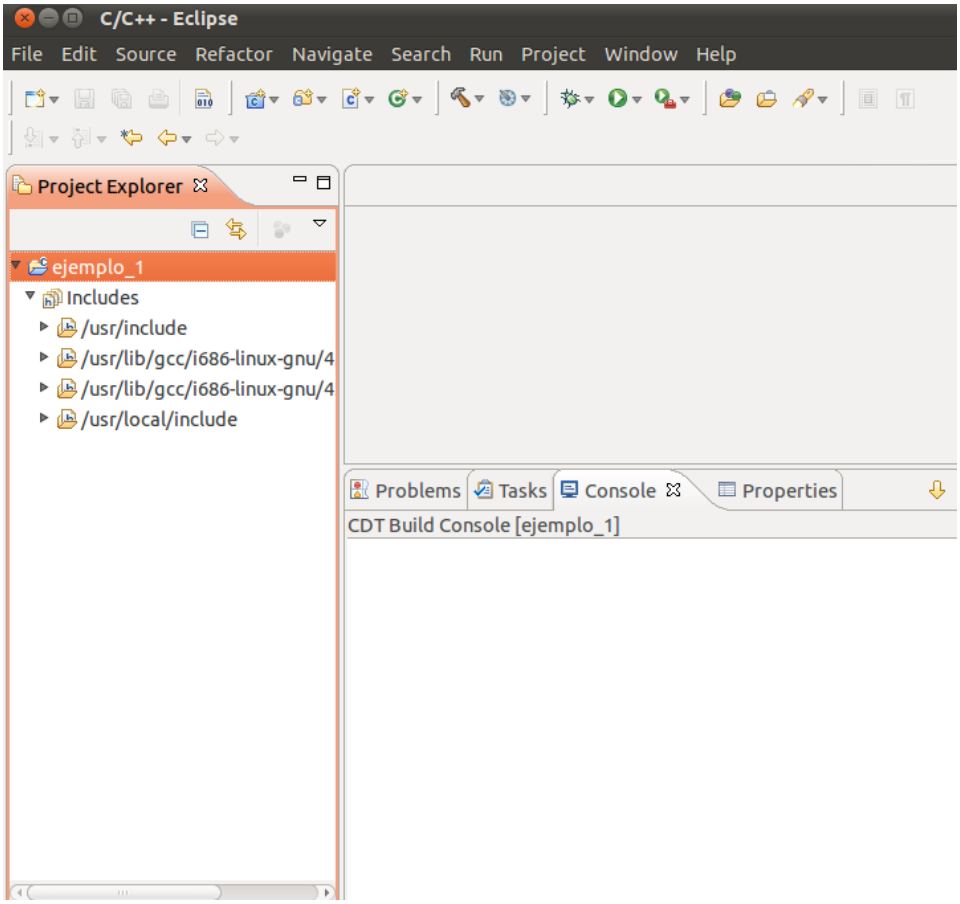
Para crear un nuevo proyecto se debe seleccionar la opción **File->New->C Project**:



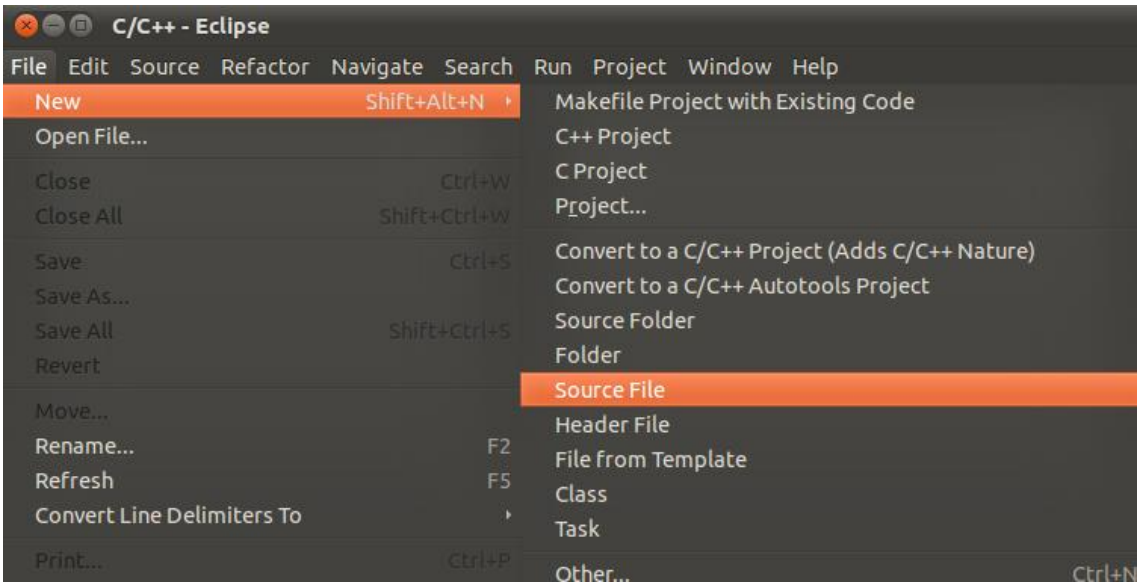
En la nueva ventana asignaremos un nombre al nuevo proyecto (en este ejemplo: ejemplo_1) y seleccionaremos **Empty Project** y **Cross GCC** dentro de las secciones **Project Type** y **Toolchains** respectivamente:



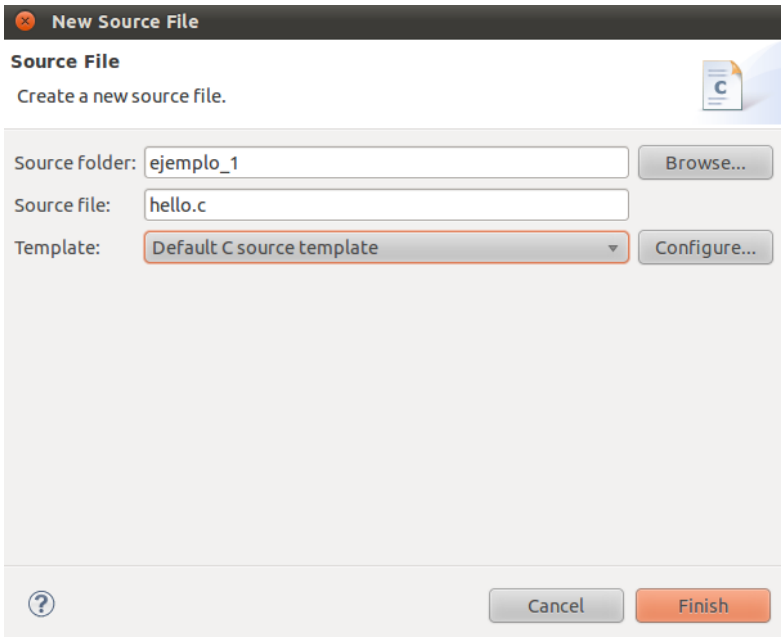
Por último seleccionaremos **Finish**, con lo que se generará el entorno necesario para nuestro nuevo proyecto:



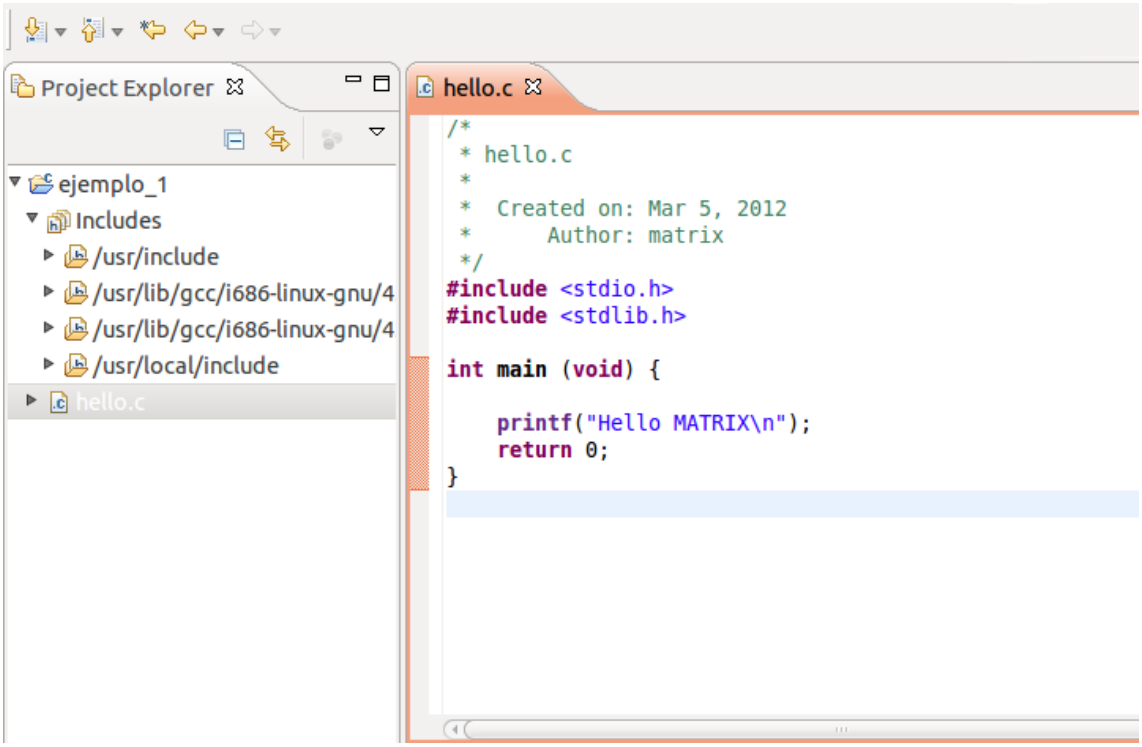
Para añadir el código fuente correspondiente a nuestra aplicación, seleccionaremos del menú: **File->New->Source File**



En la ventana emergente le indicaremos un nombre para el fichero (incluyendo la extensión) En este tutorial se ha utilizado **hello.c** como nombre para el fichero.

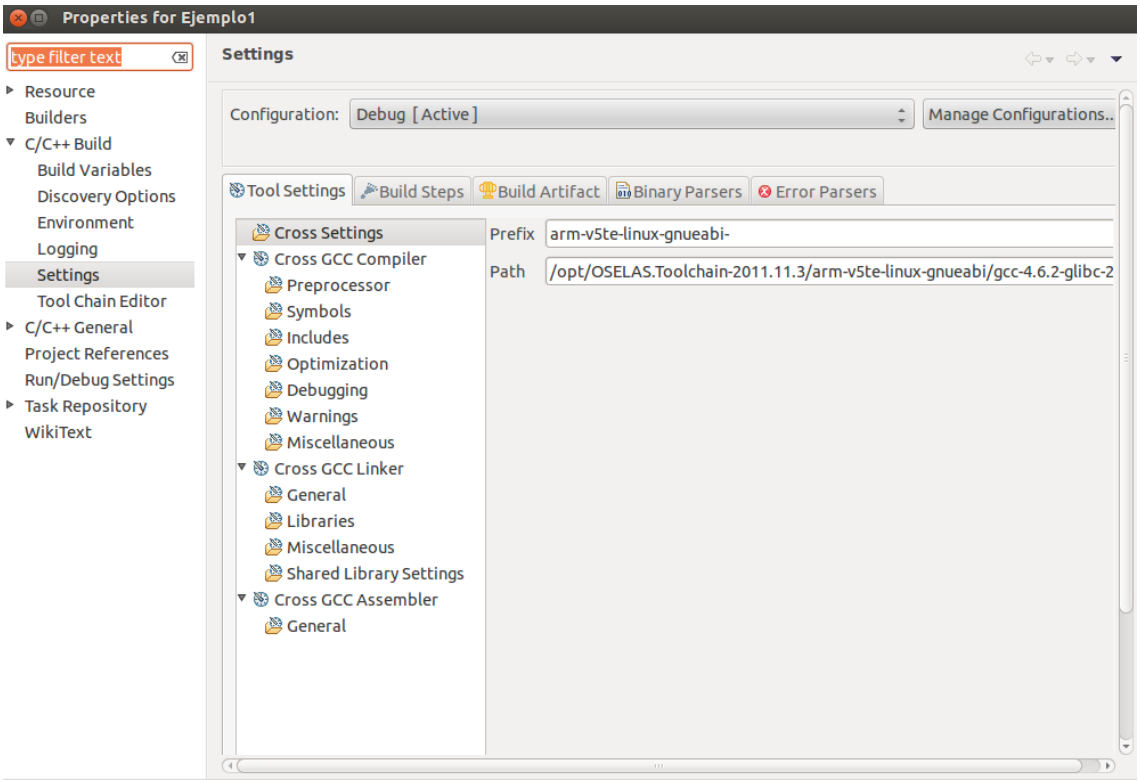


Una vez generado el fichero se debe añadir el código correspondiente a la aplicación. En el caso del presente tutorial se trata de un sencillo **Hello World**:



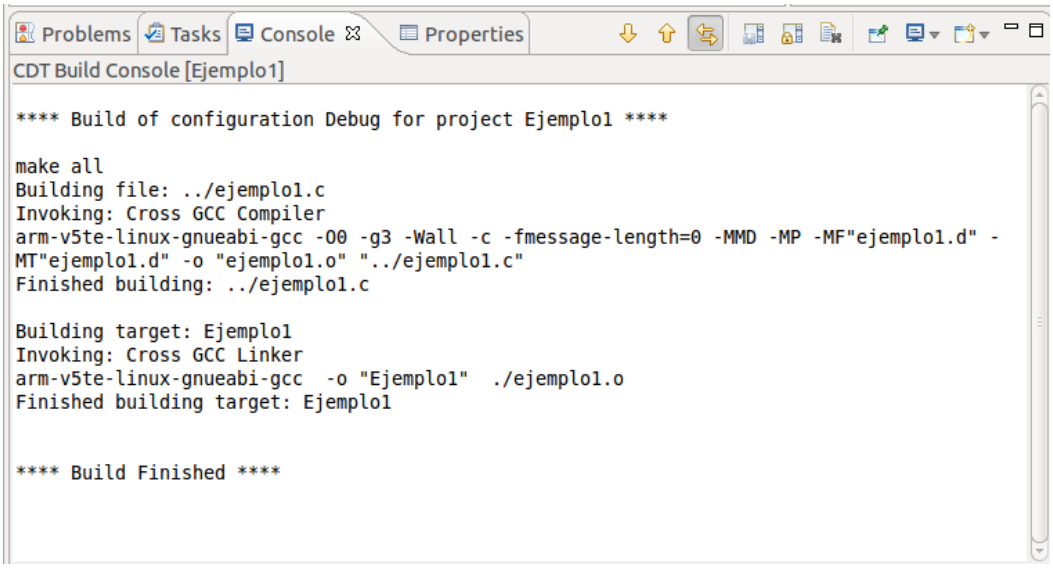
Como último paso antes de compilar debemos ajustar dos variables dentro del proyecto. Para realizar esta acción, seleccionaremos el proyecto pulsando con el botón derecho del ratón sobre **ejemplo_1** y seleccionaremos la última de las entradas **Properties**.

Dentro de la ventana emergente, expandiremos la opción **C/C++ Build** y seleccionaremos la sección **Settings**.



Para ajustar las variables tendremos que rellenar los campos **Prefix** y **Path** del apartado **Cross Settings**. Para el ejemplo de este tutorial seleccionaremos **arm-v5te-linux-gnueabi-** como Prefix y **/opt/OSELAS.Toolchain-2013.12.2/arm-v5te-linux-gnueabi/gcc-4.8.2--glibc-2.18-binutils-2.24-kernel-2.12-sanitized/bin** para la variable Path. Conviene destacar que debemos repetir esta operación dos veces: Una para la configuración **Debug** y otra para la configuración **Release**. Para cambiar entre configuraciones podemos utilizar el desplegable de esta misma ventana. Una vez realizados estos cambios pulsaremos sobre el botón **OK**.

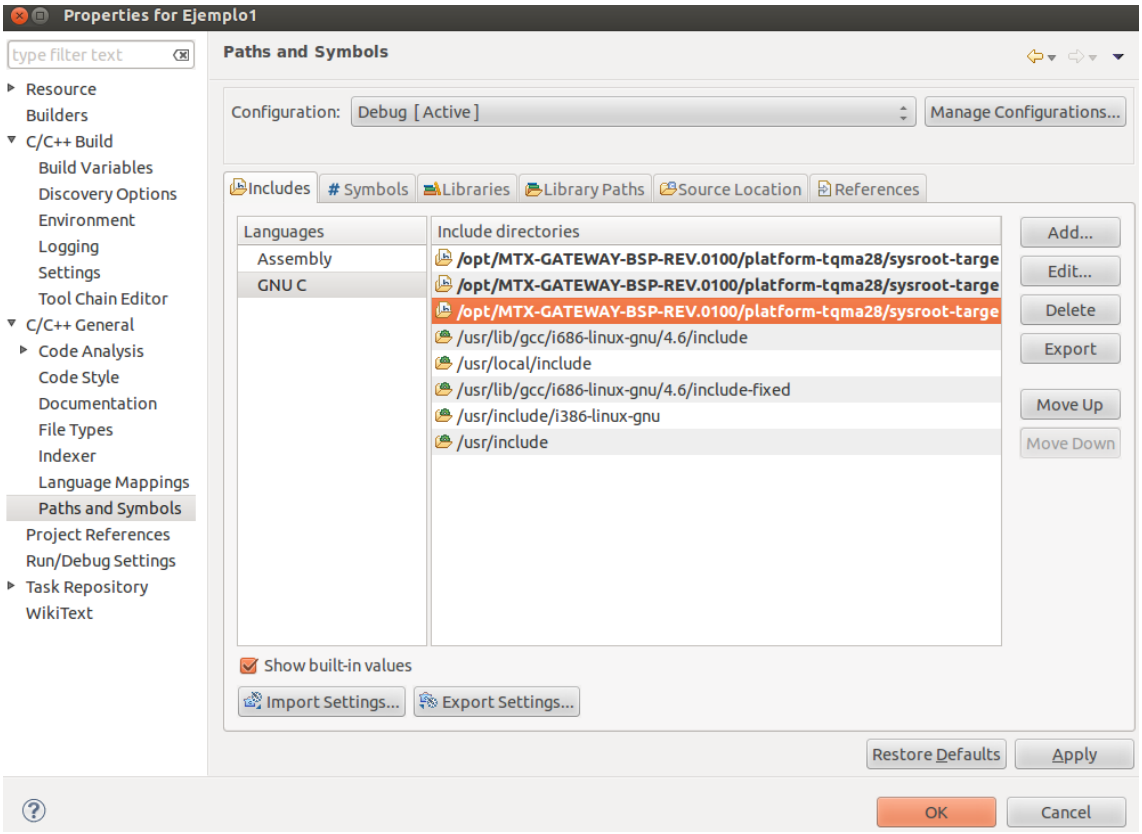
Para compilar el proyecto, lo seleccionaremos con el botón derecho y escogeremos **Build Project**. En la pestaña de **Console** situada en la parte inferior central deberíamos observar un log similar al siguiente:



Compilación incluyendo fuentes del kernel

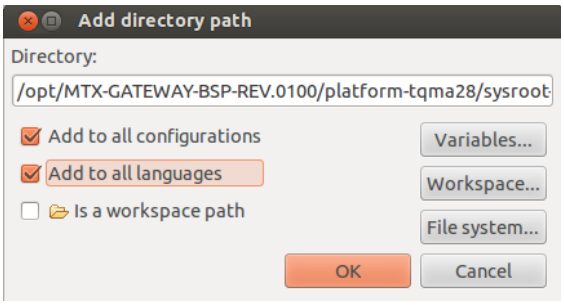
Para poder compilar un código fuente que haga referencia a los headers del Kernel, se deberá añadir una variable al proyecto para que el compilador sepa dónde tiene buscar dichos recursos.

Para ello abriremos la ventana de propiedades del proyecto (botón derecho sobre el proyecto y seleccionamos Properties), expandimos la sección **C/C++ General** y nos situamos sobre **Paths and Symbols**: (En la sección **Languages** seleccionaremos **GNU C**)



Para añadir la entrada correspondiente a la carpeta include del kernel sobre el que queremos hacer referencia pulsaremos sobre **Add…** y en la ventana emergente seleccionaremos **File System…** para añadir la ruta completa (De acuerdo con la instalación del BSP para el MTX-GATEWAY debemos utilizar el path hasta la carpeta **/opt/MTX-GATEWAY-BSP-REV.0101/platform-tqma28/sysroot-target/kernel-headers/include**)[1]. Antes de cerrar esta ventana emergente seleccionaremos la opción de añadir a todas las configuraciones (**Add to all configurations**) para no tener que repetir el proceso otra vez para este proyecto.

[1] Es necesario haber realizado previamente una compilación con ptxdist *(ptxdist go)* para que el directorio mencionado se encuentre disponible.



En la imagen también pueden apreciarse dos rutas más (marcadas en negrita) para añadir los headers asociados a paquetes y/o librerías añadidas a la imagen (como SDL, QT, readline…). Los paths adicionales son los siguientes:

/opt/MTX-GATEWAY-BSP-REV.0101/platform-tqma28/sysroot-target/usr/include

/opt/MTX-GATEWAY-BSP-REV.0101/platform-tqma28/sysroot-target/include

Debug de aplicaciones

Requisitos previos

Para el correcto desarrollo de esta sección es necesario que el MTX tenga una imagen con las siguientes utilidades:

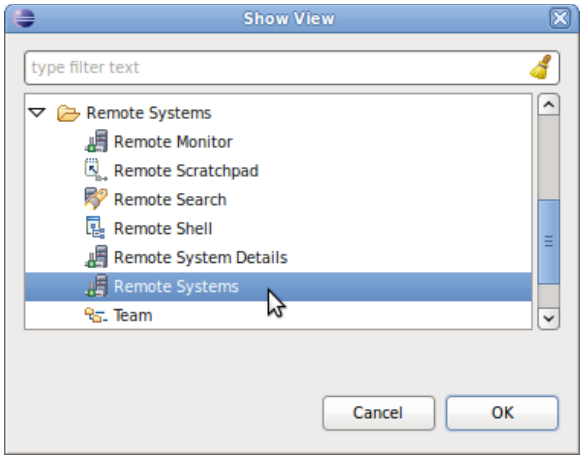
- gdbserver
- sftp-server

NOTA: La imagen de defecto de los MTX incluye por defecto ambas utilidades.

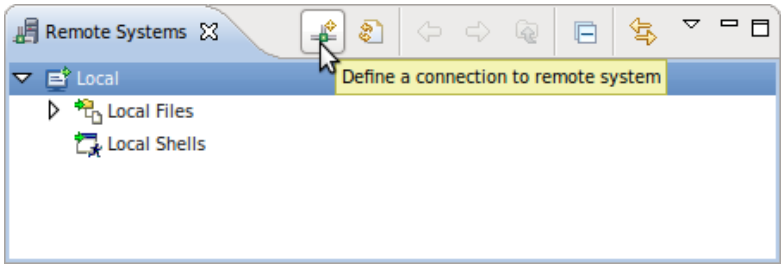
Configurar la conexión

El primer paso es establecer una conexión con el Target que en el caso de este tutorial es el MTX. Para realizar esta acción utilizaremos la herramienta RSE (Remote System Explorer)[2].

Para abrir la ventana correspondiente a esta herramienta seleccionaremos **Window > Show view > Other… >** Expandiremos **Remote Systems** y seleccionaremos **Remote Systems**.

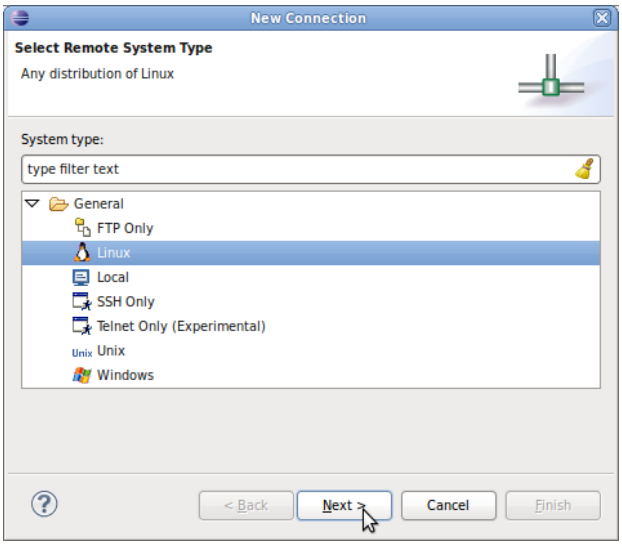


En segundo lugar añadiremos la configuración necesaria para poder conectarnos con el módulo mediante el icono de nueva conexión:

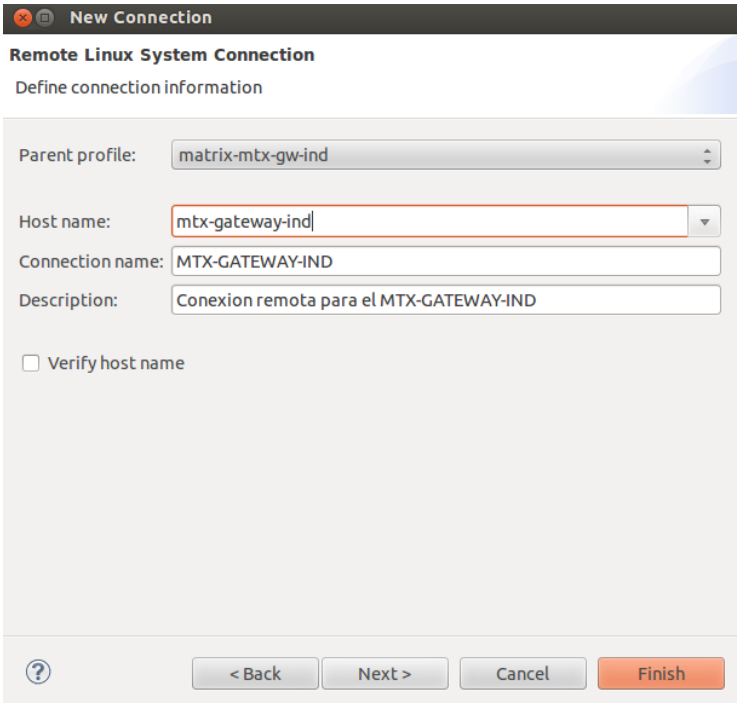


Aparecerá una nueva ventana emergente sobre la que seleccionaremos el tipo de sistema del Target, en nuestro caso Linux:

[2] Es necesario añadir los paquetes gdb-server y opnessh (incluyendo la opción sftp-server) a la imagen de defecto del módulo para poder realizar el debug remoto. Consultar http://matrixembebidos.wikispaces.com/mtx-gateway-sw para una descripción del proceso necesario para realizar esta tarea.

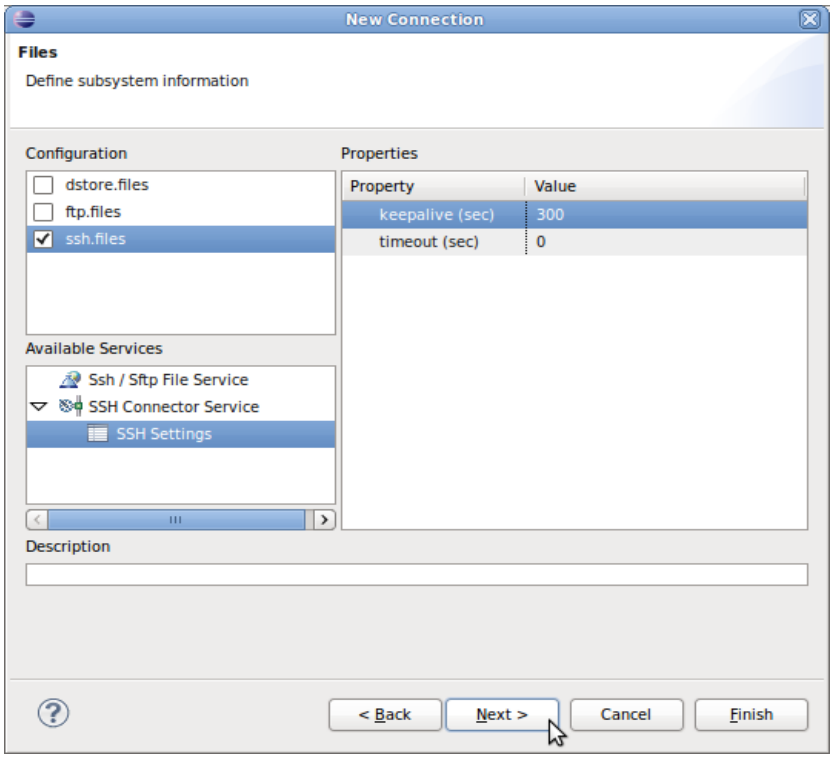


En la siguiente ventana introduciremos los datos correspondientes al Hostname[3] o IP del módulo y daremos un nombre a esta conexión:

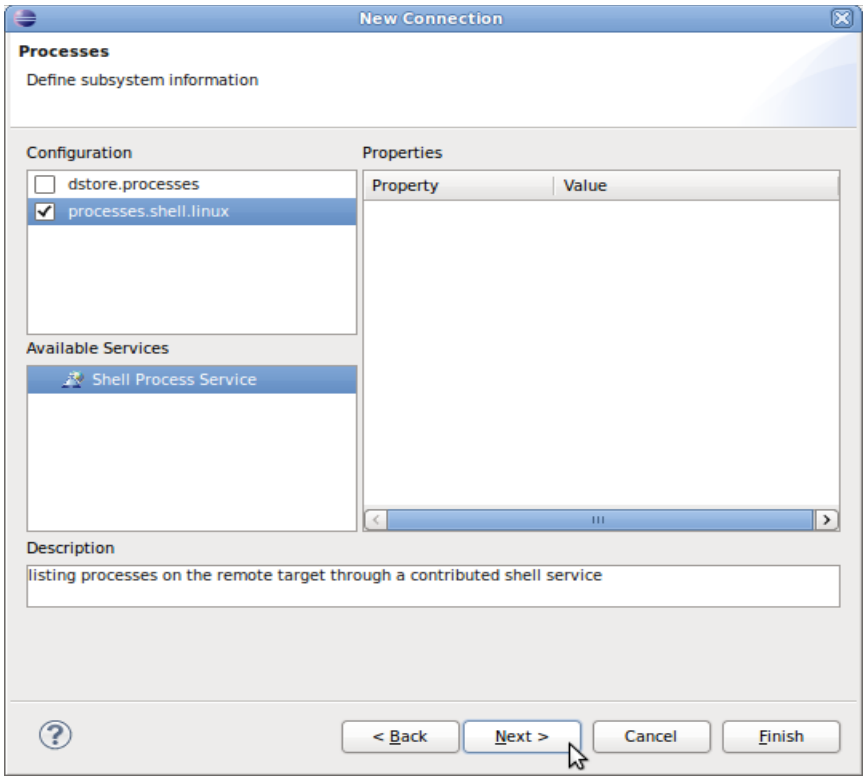


En la siguiente ventana seleccionaremos **ssh.files**

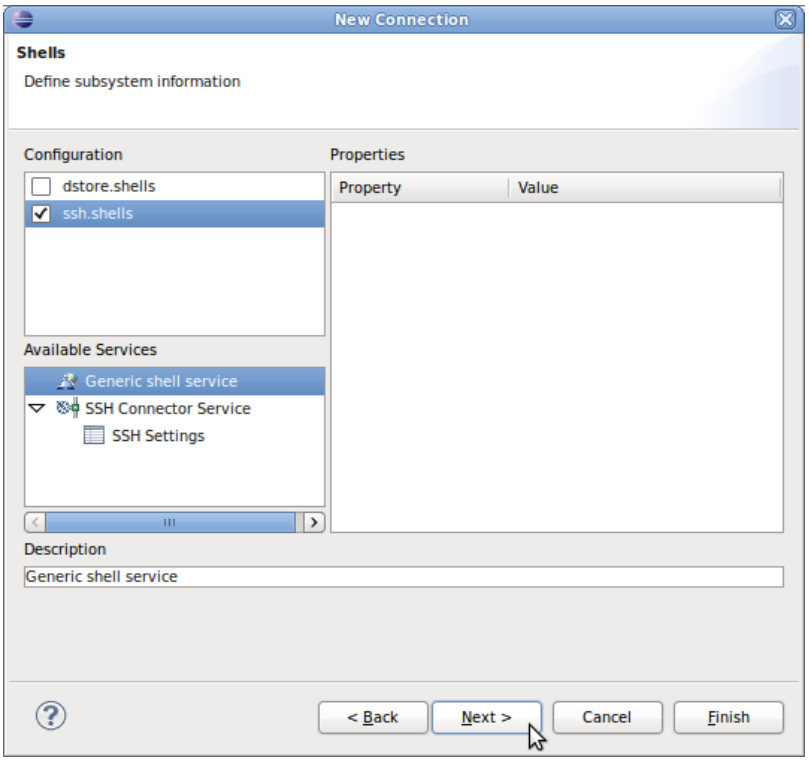
[3] Si el equipo no está conectado a la red mientras se realiza esta configuración, es necesario deshabilitar la opción **Verify host name**. Como alternativa se recomienda introducir la dirección IP del módulo.



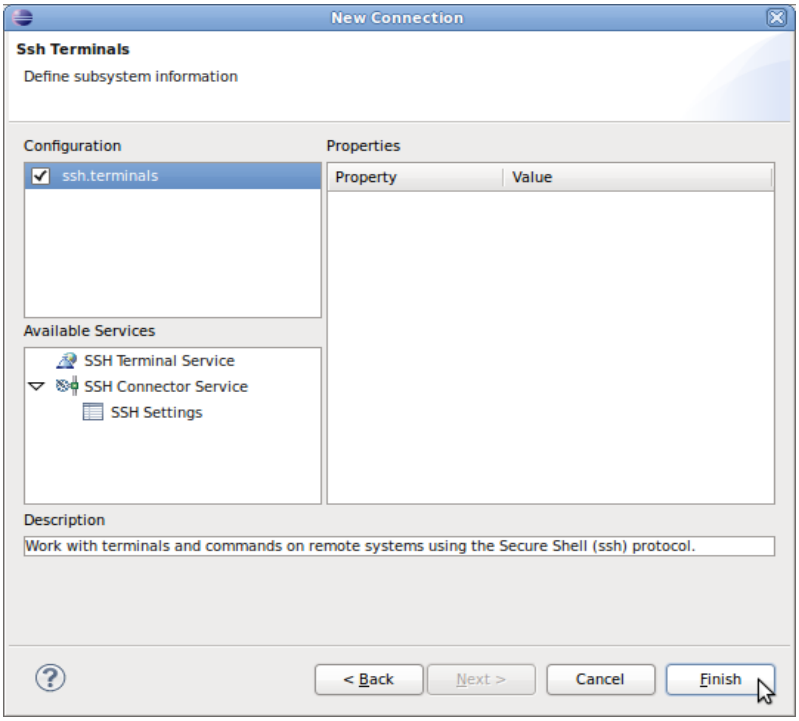
En la siguiente **processes.shell.linux**:



En la siguiente **ssh.shells**:



Por último seleccionaremos **ssh.terminals** y presionaremos **Finish**

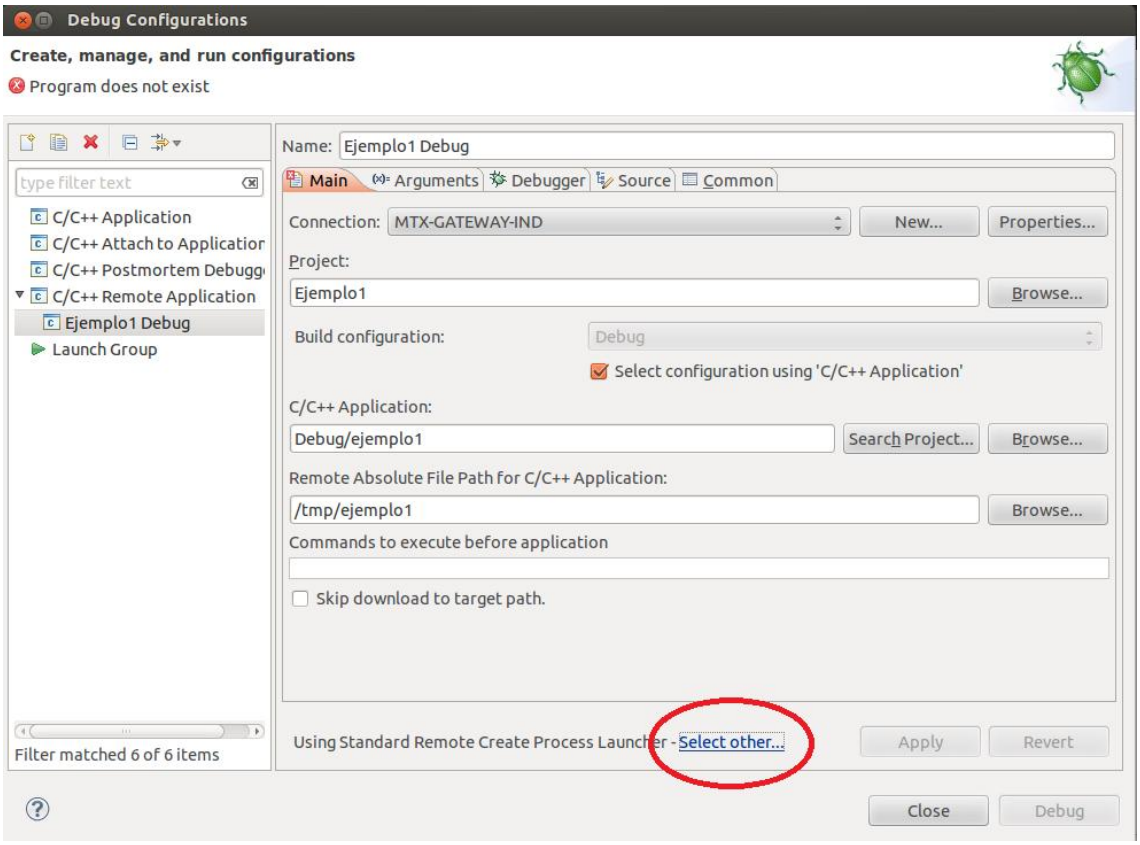


Una vez configurada la conexión podremos conectarnos con el target utilizando esta conexión. (El usuario y contraseña nos será requerido cuando inicialicemos la conexión)[4]

[4] Se debe asignar una contraseña al usuario root. Ejecutar sobre la consola del equipo el siguiente comando e introducir la contraseña deseada:
$: passwd

Inicial el Debug de una aplicación

Para abrir el debugger, seleccionaremos en el menú **Run > Debug Configurations…** (Es preferible realizar esta acción después de haber seleccionado el proyecto que queremos debugear para que muchos de los parámetros se completen de forma automática)



Crearemos una nueva **C/C++ Remote Application** pulsando con el botón derecho sobre esta entrada y seleccionando **New**

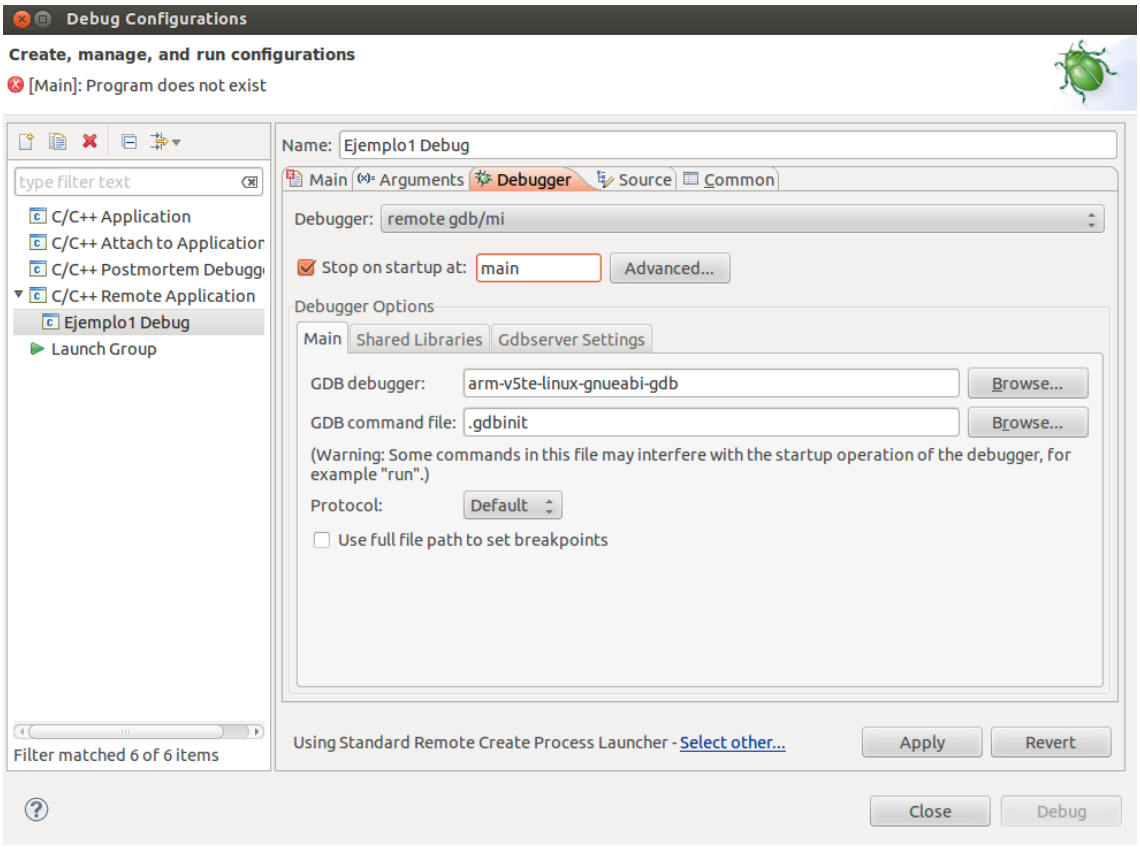
Cambiaremos el **Launcher** seleccionado **Select other…** (En la imagen aparece indicado dentro de una elipse roja). Activaremos la opción **Use configuration specific settings** y seleccionaremos **Remote Create Process Launcher**.

Aceptaremos y volveremos sobre la pantalla de Debug Configurations sobre la que deberemos especificar el directorio destino (apartado **Remote Absolute File Path for C/C++ Application**) donde nuestra aplicación será copiada y ejecutada (En el ejemplo se ha usado **/tmp/ejemplo1**. (Es recomendable utilizar el botón Browse para introducir un directorio existente)

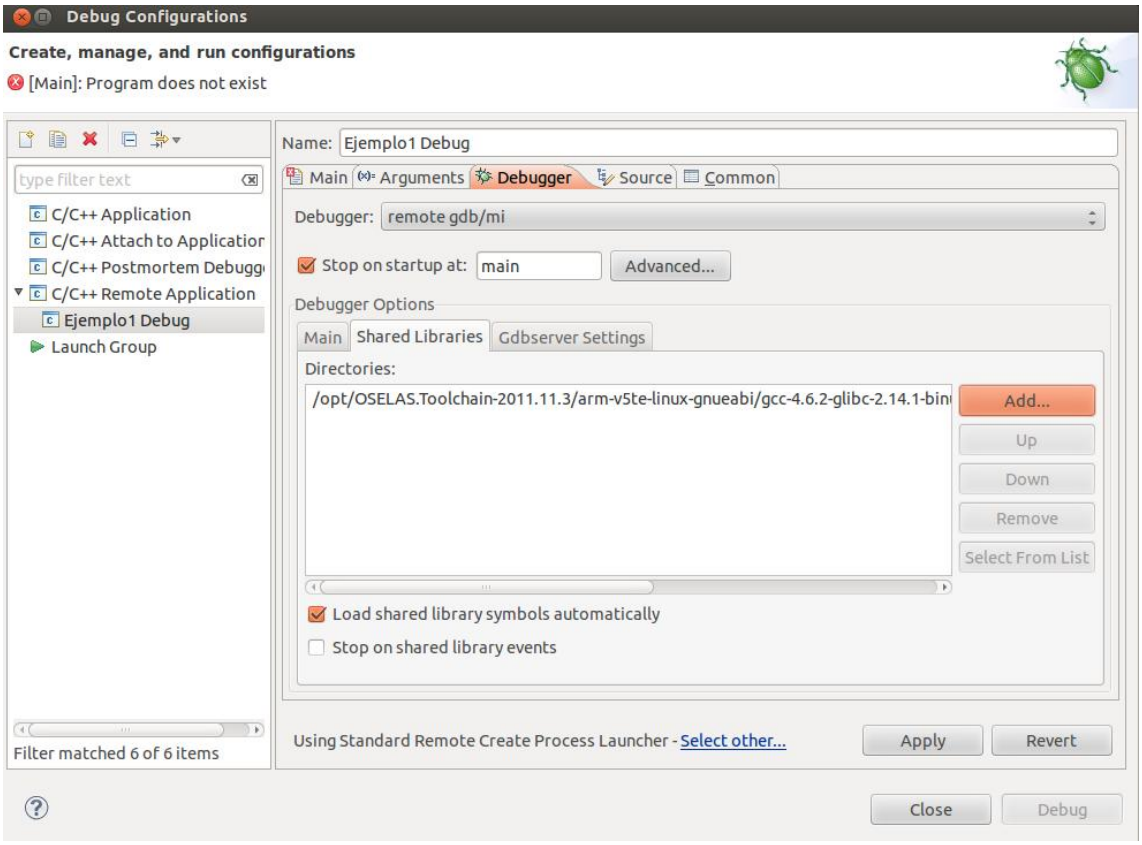
Sobre la pestaña **Arguments** pueden añadirse los argumentos requeridos por la aplicación si fuese necesario.

Dentro de la pestaña **Debugger** es necesario modificar el comando de GDB por **arm-v5te-linux-gnueabi-gdb**

NOTA: En algunos casos es necesario eliminar el contenido de la casilla "*GDB command file*".



Dentro de esta misma sección, sobre la pestaña **Shared Libraries** se debe añadir el path a las librerías de nuestro sistema: **/opt/OSELAS.Toolchain-2013.12.2/arm-v5te-linux-gnueabi/gcc-4.8.2-glibc-2.18-binutils-2.24-kernel-3.12-sanitized/sysroot-arm-v5te-linux-gnueabi/lib**



Por último pulsaremos sobre el botón **Apply** y a continuación sobre el botón **Debug**.

Si la aplicación requiere de consultar resultados sobre una **Shell Remota** (como es el caso de este tutorial debido al printf) se puede activar esta vista seleccionándola dentro de las opciones de **Display Selected Console**: