Intalar las herramientas de desarrollo Qt en el RPI
El primer paso es instalar las herramientas de desarrollo Qt en el RPI. El ultimo comando del codigo siguiente instala una suite completa de herramientas que necesita entre 60 y 200 MB:

```
apt-cache search qt5
sudo apt install gt5-default
```

Podemos comprobar la version de la instalacion usando lo siguiente:

```
qmake -version
```

A continuacion haremos una prueba que mostrara por pantalla una venta con el texto “Hola mundo!”.
El codigo es el siguiente:

```
#include <Qapplication>
#include <Qlabel>

int main(int argc, char *argv[]){
      Qaplication app(argc, argv);
      QLabel label(“Hola mundo!”);
      label.resize(200, 100);
      label.show();
      return app.exec();
}
```

Lo guardamos como simpleQt.cpp, este archivo sera el unico obligatorio en el directorioantes de que los pasos siguientes tengan lugar. El generador makefile multiplataforma qmake se puede usar entonces para crear un proyecto predeterminado.

```
~/exploringrpi/chap14/simpleQt $ ls
simpleQt.cpp
~/exploringrpi/chap14/simpleQt $ qmake -project
~/exploringrpi/chap14/simpleQt $ ls
simpleQt.cpp simpleQt.pro
~/exploringrpi/chap14/simpleQt $ more simpleQt.pro

// copiar aquí el .pro
```

Este archivo de proyecto describe los ajustes de proyecto y, si es necesario, se puede editar para añadir dependencias adicionales. En este caso, la linea:

```
QT += widgets
```

se debe añadir al archivo `.pro` (por ejemplo, entre las lineas `TEMPLATE` y `TARGET`), ya que, de otro modo, las librerias necesarias para mostrar los widgets de la interfaz grafica de usuario no se enlazarian correctamente. El generador qmake `Makefile` se puede ejecutar de nuevo, solo que esta vez sin el argumento `-project` :

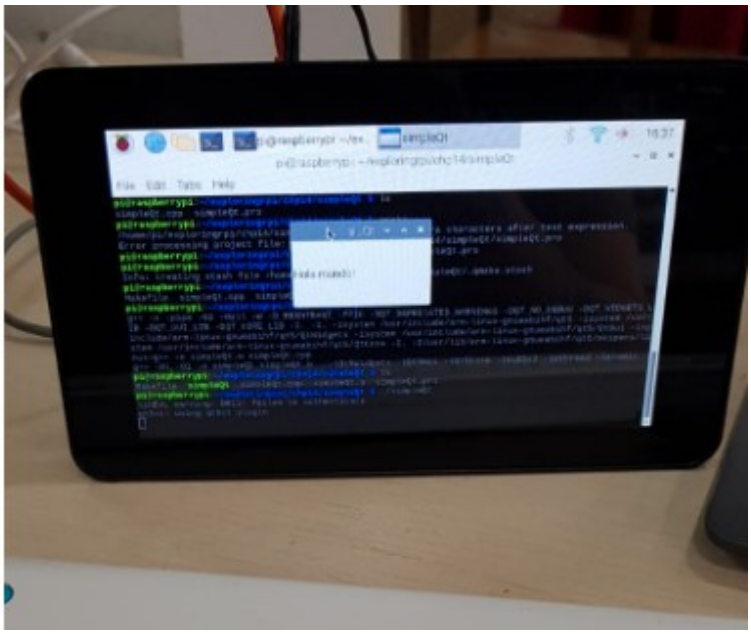
```
~/exploringrpi/chap14/simpleQt $ qmake
~/exploringrpi/chap14/simpleQt $ ls
Makefile simpleQt.cpp simpleQt.pro
```

Este paso tiene como resultado la creación de un archivo Makefile en el directorio actual, que permie al ejecutable ser compilado usando una llamada al programa make, que, a su vez, usa g++ para compilar la aplicación final:

```
~/exploringrpi/chap14/simpleQt $ make
g++ -c -pipe -O2 -Wall -W -D_REENTRANT -fPIE -DQT_NO_DEBUG …
```

El ejecutable esta ahora presente en el directorio y se puede ejecutar del siguiente modo:

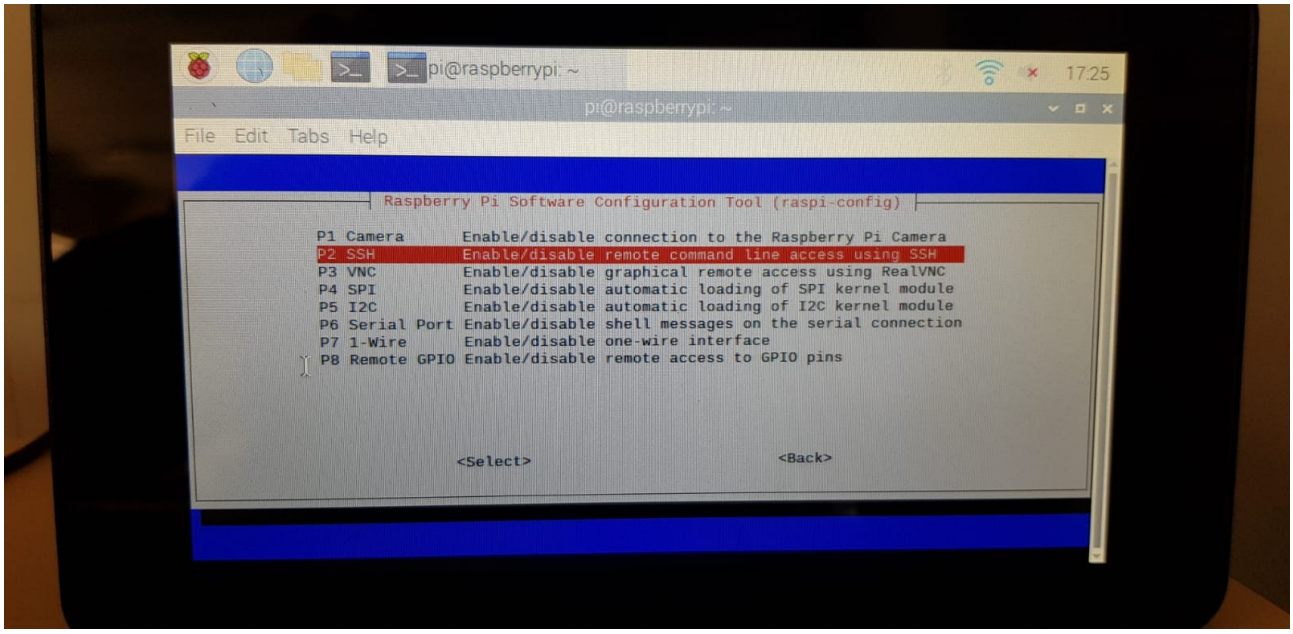
```
~/exploringrpi/chap14/simpleQt $ ls
Makefile simpleQt simpleQt.cpp simpleQt.pro
~/exploringrpi/chap14/simpleQt $ ./simpleQt
```



2- Aunque quizá, con lo hecho hasta aquí ya tendríamos para desarrollar las interfaces y programario para el tratado representado de la inflacionario recogida por los sensores, realmente seria mucho mas practico y eficiente utilizar la compilación cruzada.

Se llama compilación cruzada, cuando la compilación de la aplicación se compila en un sistema de características distintas al sistema en que dicha aplicación correrá, en nuestro caso, que utilizaremos como base de todo el sistema una Raspberry Pi 3 b+, que aunque potente, es un sistema limitado, seria mucho mas practico, rápido y deseable, hacer dicha compilación en nuestro ordenador habitual utilizado para programar ya que es mucho mas potente y capaz que la Raspberry.

1- Lo primero que haremos sera habilitar las conexiones ssh con la Raspberry, para esto, podemos abrir un terminal en la raspberry y escribir la orden `sudo raspi-config`, y habilitar el protocolo ssh . Intecface options > P2 SSH.



Instalación las librerías de desarrollo

Necesitamos instalar algunas librerias, asi que lo primero que debemos hacer es permitir al sistema instalar paquetes de la fuente, para esto solo debemos descomentar la linea `deb-src` en el fichero `/etc/apt/sources.list`, *el cual configura los repositorios.*

```
Sudo nano /etc/apt/sources.list
```

El siguiente paso sera actualizar y descargar los paquetes de desarrollo requeridos.

```
Sudo apt-get update
sudo apt-get build-dep qt4-x11
sudo apt-get build-dep libqt5gui5
sudo apt-get install libudev-dev libinput-dev libts-dev libxcb-xinerama0-dev
libxcb-xinerama0
```

Preparar el directorio objetivo

Creamos un directorio en nuestra Raspberry para el usuario pi. Este directorio (`/usr/local/qt5pi`) sera utilizado para desplegar Qt desde nuestro ordenador hacia la Raspberry Pi.

```
Sudo mkdir /user/local/qt5pi
Sudo chown pi:pi /usr/local/qt5pi
```

Crear un directorio de trabajo y configurar una cadena de herramientas

Creamos un directorio de trabajo en nuestro ordenador y descargamos la cadena de herramientas.

```
Mkdir ~/raspi
cd /raspi
git clone https://github.com/raspberrypi/tools
```

Crear y configurar un sysroot

Un sysroot es una estructura de directorios que incluye todos los directorios necesarios para correr un sistema en particular. Crearemos un sysroot para la compilación cruzada para Rasberry Pi en nuestro ordenador de trabajo.

```
Mkdir sysroot sysroot/usr sysroot/opt
```

Podemos utilisar rsync para sincronizar nuestro sysroot del ordenador y la Raspberry Pi. Asi, si hacemos cambios en nuestro ordenador, sera facil transferirlos a nuestro Raspberri Pi; raspberrypi_ip es el nombre de interfaz de red o la direccion ip de la Raspberry Pi.

```
rsync -avz pi@raspberrypi_ip:/lib sysroot
rsync -avz pi@raspberrypi_ip:/usr/include sysroot/usr
rsync -avz pi@raspberrypi_ip:/usr/lib sysroot/usr
rsync -avz pi@raspberrypi_ip:/opt/vc sysroot/opt
```

Ahora, necesitamos ajustar nuestros enlaces simbólicos en el sysroot para que sean relativos, ya que esta estructura de directorios esta tanto en nuestro ordenador como en el Raspberry Pi.

```
wget https://raw.githubusercontent.com/riscv/riscv-poky/master/scripts/sysroot-relativelinks.py
chmod +x sysroot-relativelinks.py
./sysroot-relativelinks.py sysroot
```

Descargar las Qt

```
wget http://download.qt.io/official_releases/qt/5.12/5.12.5/single/ qt-everywhere-src-5.12.5.tar.xz
tar xvf qt-everywhere-src-5.12.5.tar.xz
cd qt-everywhere-src-5.12.5
```

Configuracion de Qt para la compilacion cruzada

En la version de Raspbian, el sistema operativo que utiliza la Raspberry Pi, las librerias EGL tienen nombres diferentes a los que se asumen en los archivos de configuracion de Qt, asi que tendremos que editar el fichero ./qtbase/mkspecs/devices/linux-raspi-pi-g++/qmake.conf y sustituir todas las referencias a -LEGL y LGLESv2 por -lbrcmEGL y -lbrcmGLESv2, respectivamente.

El siguiente comando configura la version de Qt open-source para la compilacion cruzada.

```
./configure -release -opengl es2 -device linux-rasp-pi-g++ -device-option
CROSS_COMPILE=~/raspi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/
bin/arm-linux-gnueabihf- -sysroot ~/raspi/sysroot -opensource -confirm-license -skip
qtwayland -skip qtlocation -skip qtscript -make libs -prefix /usr/local/qt5pi -
extprefix ~/raspi/qt5pi -hostprefix ~/raspi/qt5 -no-use-gold-linker -v -no-gbm
```

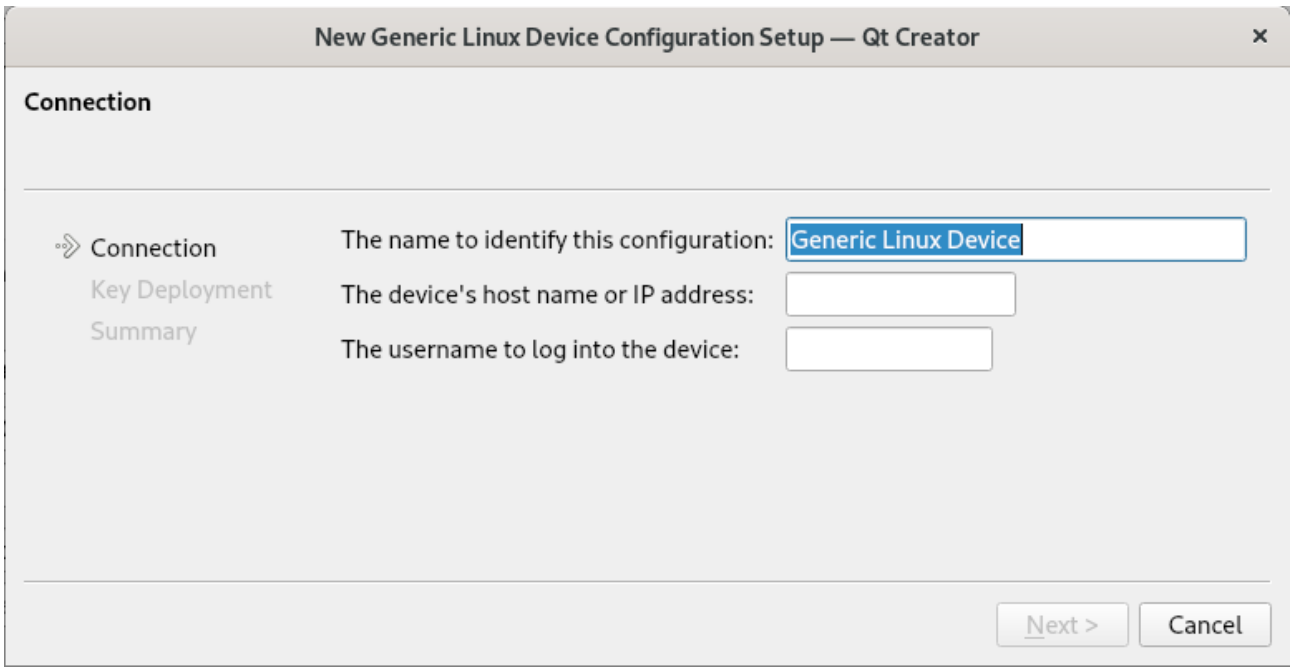
El tiempo exacto de compilacion dependera de nuestro ordenador, ahora haremos el make y el make install, esto a mi me llevo varios minutos, mas de media hora.

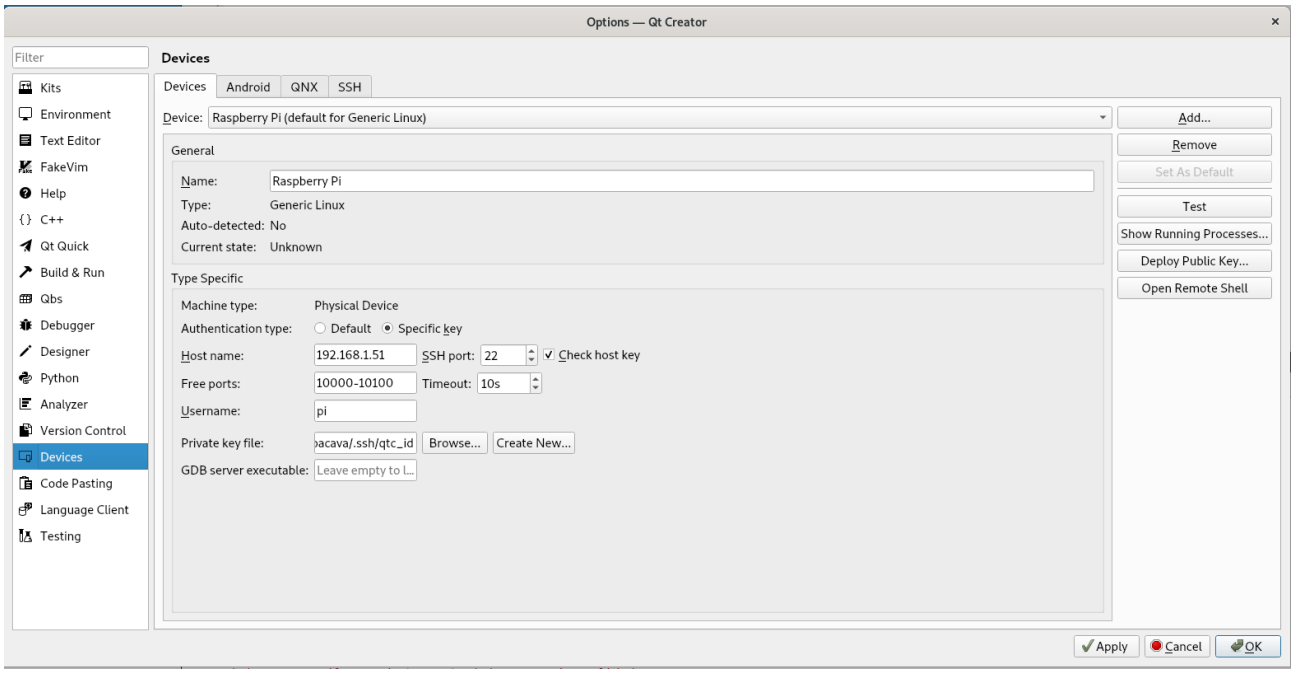
Una vez han sido compiladas las Qt, pueden ser desplegadas en la Raspberry Pi con el comando rsync.

```
rsync -avz qt5pi pi@raspberrypi_ip:/usr/local
```

Configuracion de Qt Creator para la compilacion cruzada para Raspberry Pi

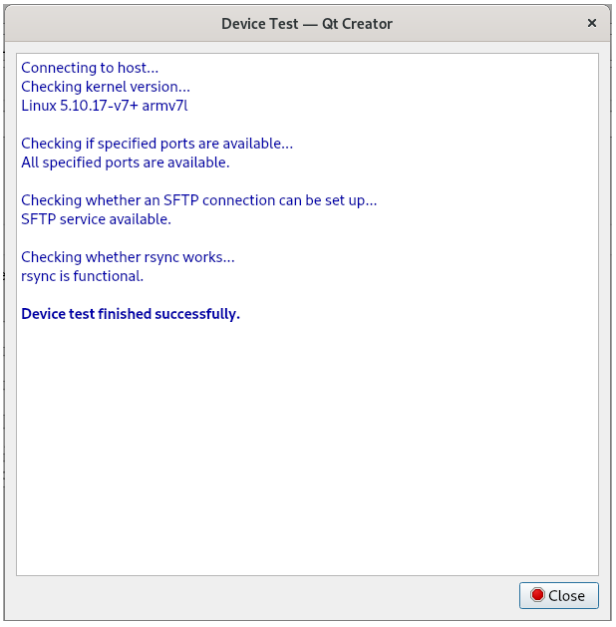
En primer lugar, vamos a Tools > Options, seleccionamos la seccion de Divices a la pestaña de Divices y añadimos (Add) un Generic Linux Device.





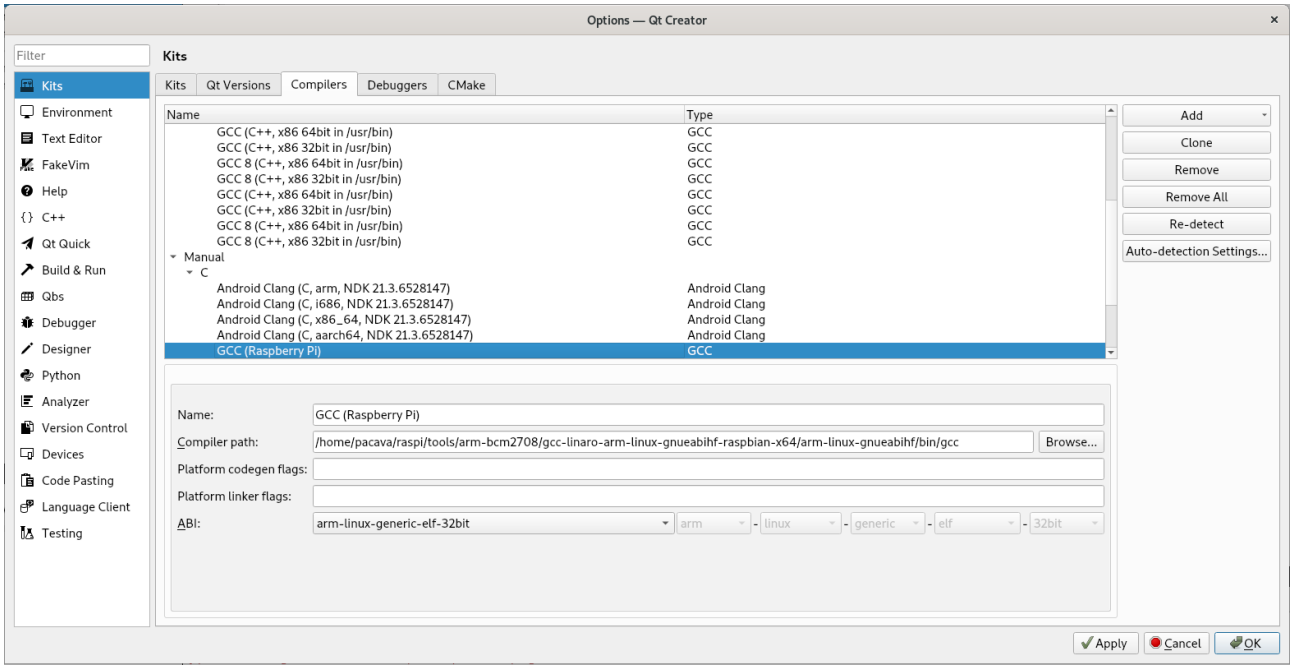
Luego seleccionamos un nombre, la direccion I.P. y el nombre de usuario del dispositivo, tambien una llave publica y privada para la conexión.

Podemos hacer una prueba de conexión con la Raspberry con el boton Test.

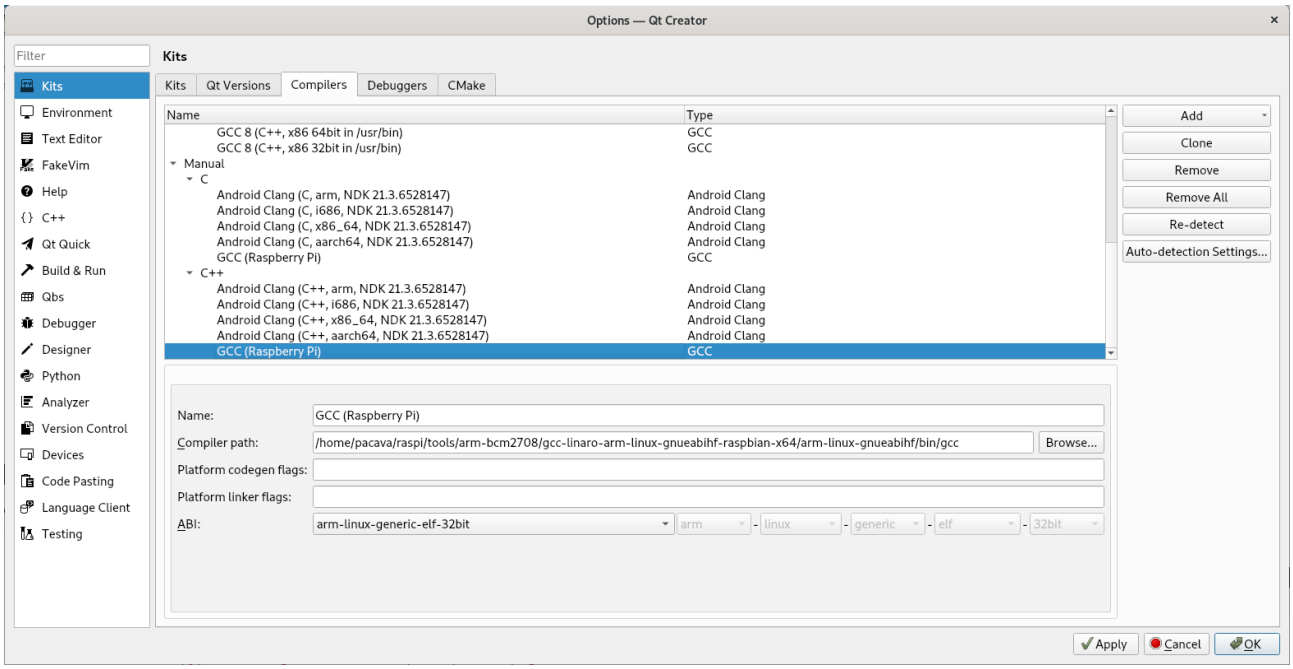


Luego, vamos a la seccion de Kits, a la pestaña Compilers y añadimos los compiladores GCC C y C++ para la compilacion cruzada para Raspberry Pi.

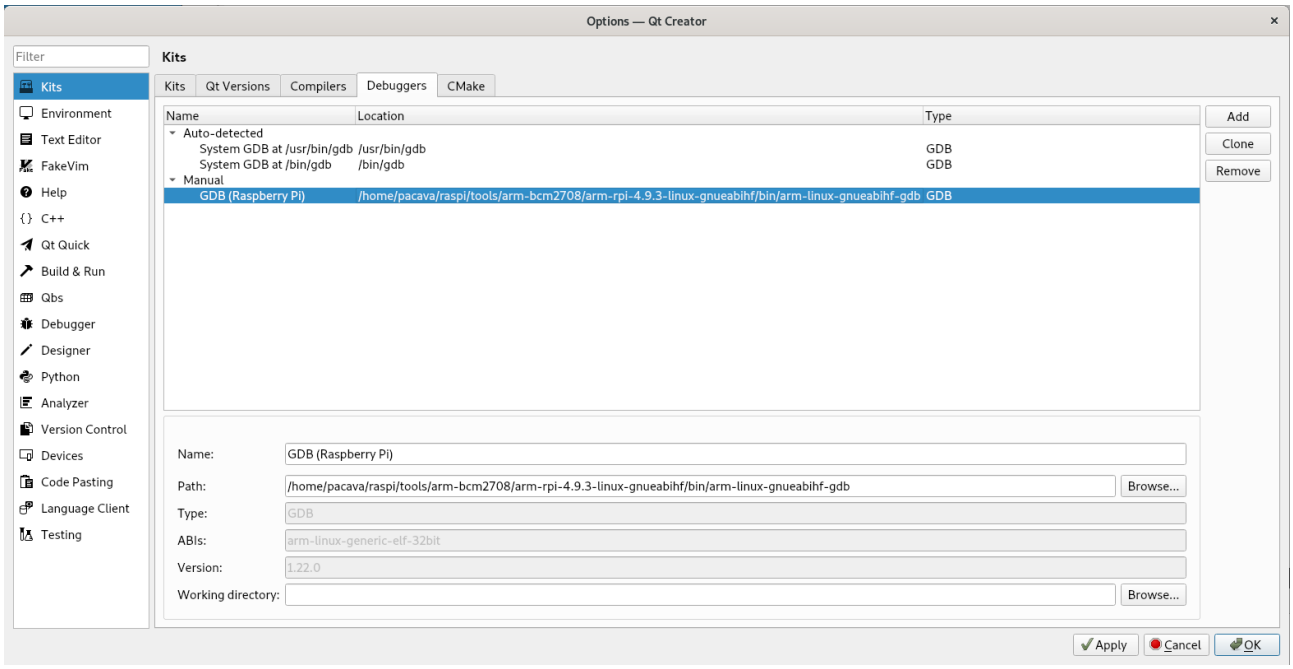
La ruta para el compilado gcc C para Raspberry Pi es **~/raspi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/arm-linux-gnueabihf/bin/gcc**.



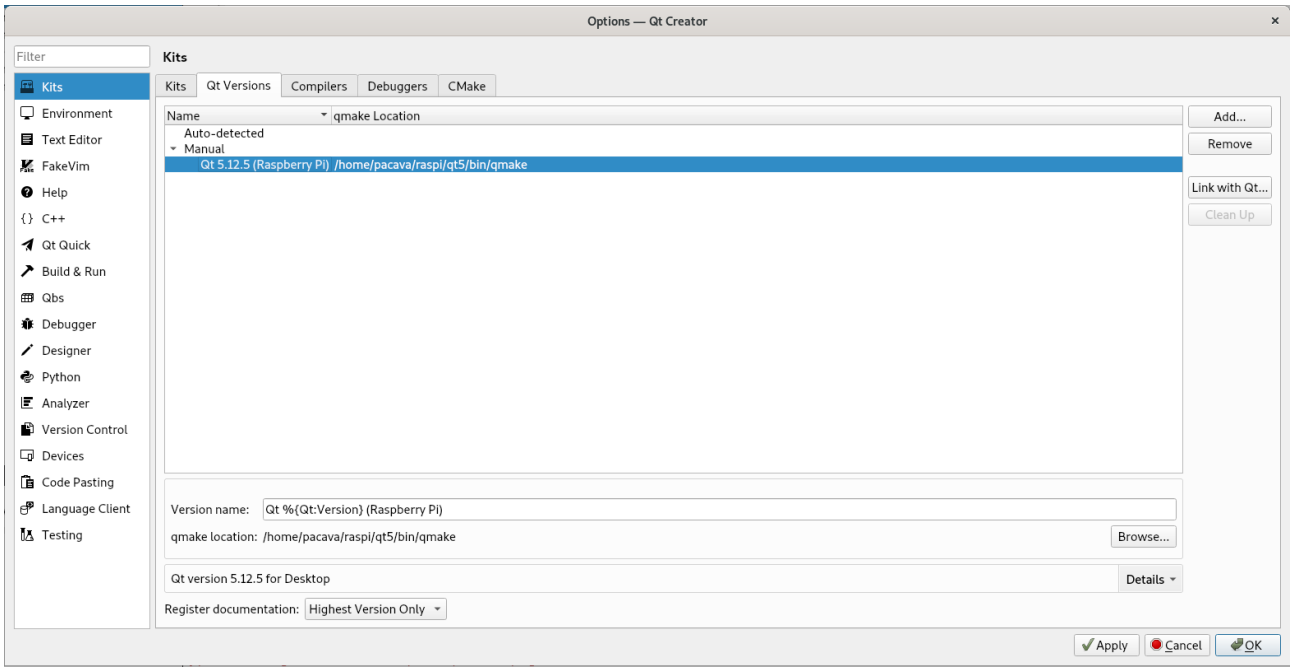
La ruta para el GCC C++ es la misma que para el compilador GCC C.



Ahora iremos a la pestaña Debuggers y añadiremos uno, colocaremos la ruta **~/raspbi/tools/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/bin/arm-linux-gnueabihf-gdb**.

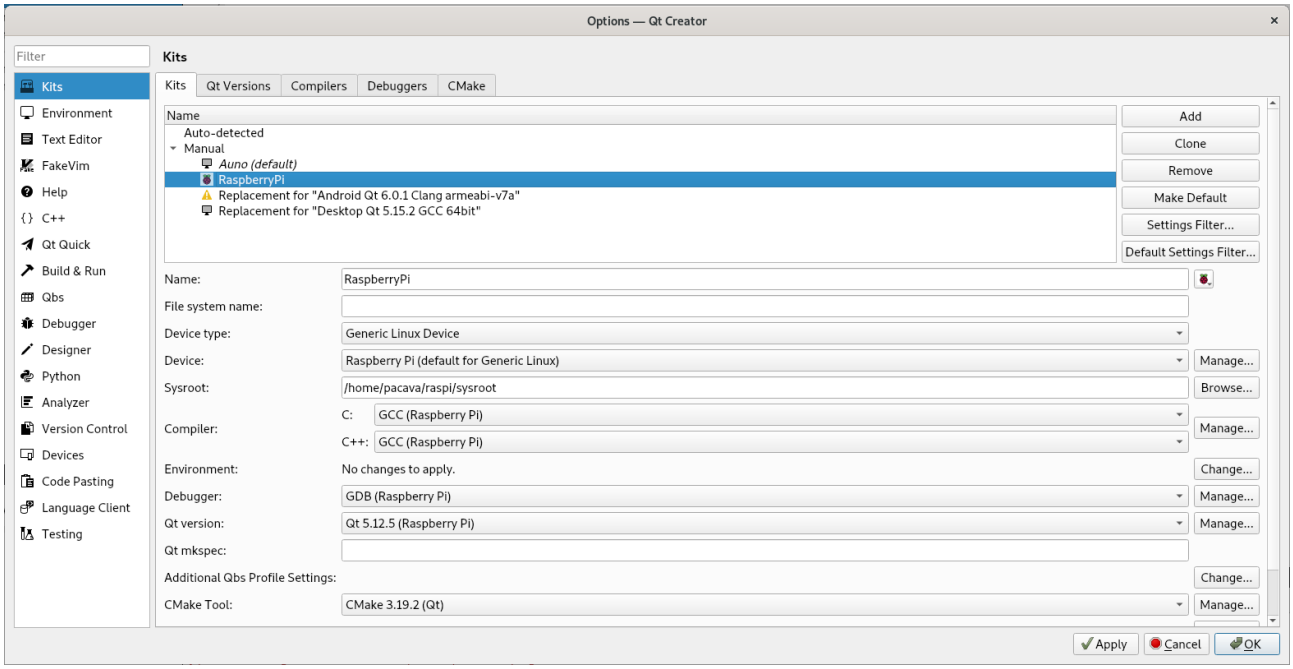


Luego, vamos a la pestaña de Qt Versions y añadimos una version: Qt 5.12 (raspberry Pi), y pondremos la ruta al qmake **~/raspi/qt5/bin/qmake**.

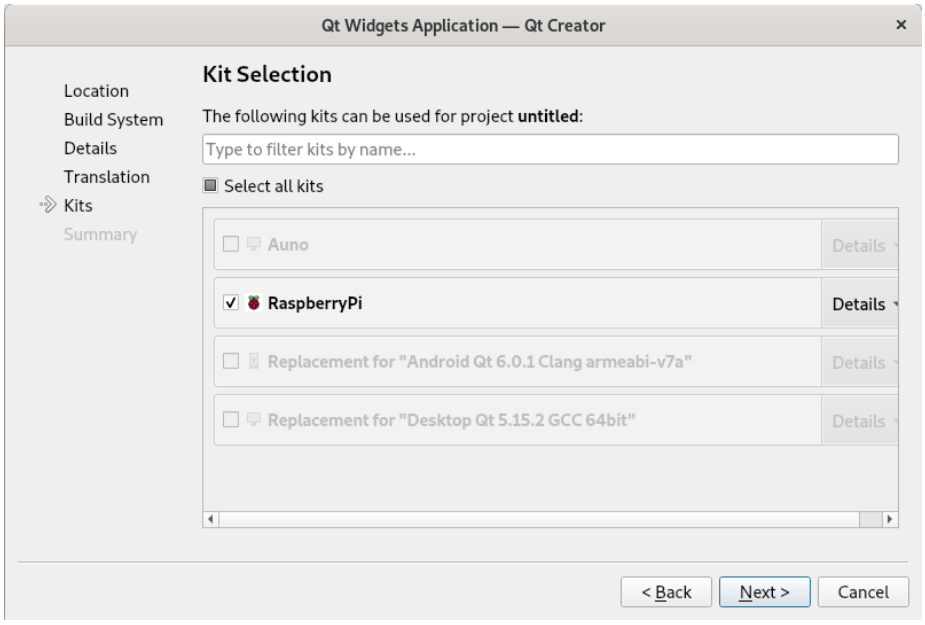


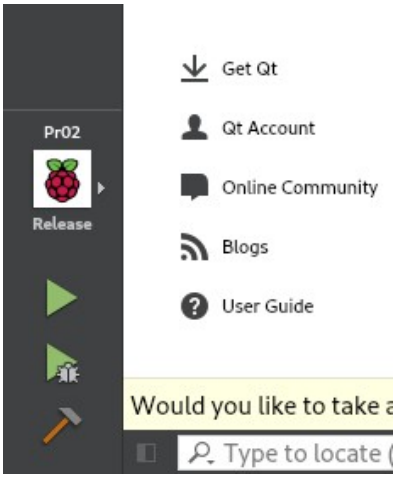
Y finalmente iremos a la pestaña de Kits, Add a new kit, colocamos el nombre y el icono de nuestra preferencia, y prestaremos especial atencion a los siguientes campos:

- Device type: Generic Linux Device
- Device: Raspberry Pi (default for Generic Linux)
- Sysroot: ~/raspi/sysroot
- Compiler C: GCC (Raspberry Pi)
- Compiler C++: GCC (Raspberry Pi)
- Debugger: GDB (Raspberry Pi)
-Qt Version: Qt 5.12 (Raspberry Pi)

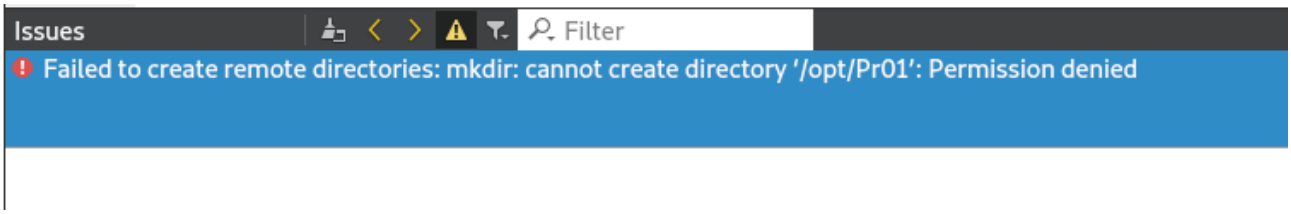


Ya todas las configuraciones estan listas. Podemos probar nuestra configuracion creando un nuevo proyecto y seleccionando el kit definido anteriormente: Raspberry Pi.





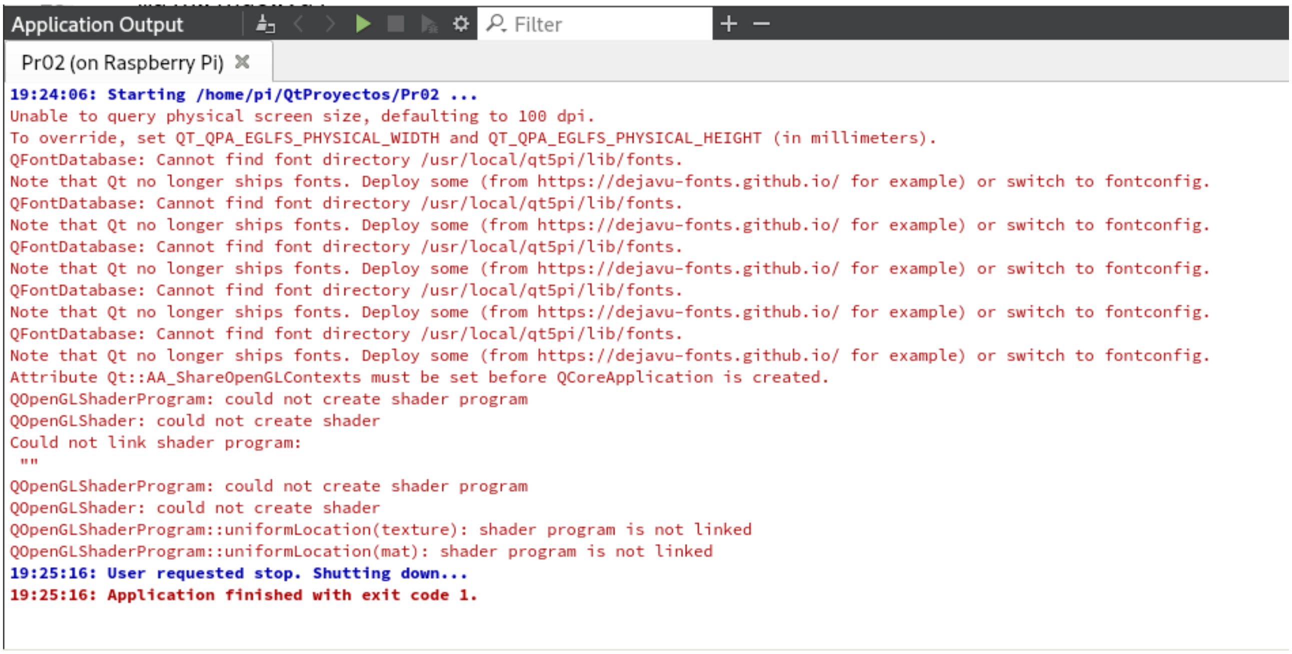
Sin embargo, luego de haber hecho todos estos pasos tuve hasta ahora dos problemas, el primero es que en el archivo de configuracion del proyecto, el archivo .pro, estaba seleccionado un directorio en la carpeta opt de la raspberry que necesita permisos de super usuario para ser escrito, asiq ue tuve que cambiar esta ruta por otra que permita al usuario pi escritura:




```
20
21 # Default rules for deployment.
22 qnx: target.path = /tmp/$${TARGET}/bin
23 else: unix:!android: target.path = /home/pi/QtProyectos
24 !isEmpty(target.path): INSTALLS += target
25
```

Una vez corregido esto, la aplicación si corre en la raspberry, pero tengo otro problema que hasta la fecha no he resuelto, la aplicación se ve en negro.

Y me indica los siguientes problemas:




```
19:24:06: Starting /home/pi/QtProyectos/Pr02 ...
Unable to query physical screen size, defaulting to 100 dpi.
To override, set QT_QPA_EGLFS_PHYSICAL_WIDTH and QT_QPA_EGLFS_PHYSICAL_HEIGHT (in millimeters).
QFontDatabase: Cannot find font directory /usr/local/qt5pi/lib/fonts.
Note that Qt no longer ships fonts. Deploy some (from https://dejavu-fonts.github.io/ for example) or switch to fontconfig.
QFontDatabase: Cannot find font directory /usr/local/qt5pi/lib/fonts.
Note that Qt no longer ships fonts. Deploy some (from https://dejavu-fonts.github.io/ for example) or switch to fontconfig.
QFontDatabase: Cannot find font directory /usr/local/qt5pi/lib/fonts.
Note that Qt no longer ships fonts. Deploy some (from https://dejavu-fonts.github.io/ for example) or switch to fontconfig.
QFontDatabase: Cannot find font directory /usr/local/qt5pi/lib/fonts.
Note that Qt no longer ships fonts. Deploy some (from https://dejavu-fonts.github.io/ for example) or switch to fontconfig.
QFontDatabase: Cannot find font directory /usr/local/qt5pi/lib/fonts.
Note that Qt no longer ships fonts. Deploy some (from https://dejavu-fonts.github.io/ for example) or switch to fontconfig.
Attribute Qt::AA_ShareOpenGLContexts must be set before QCoreApplication is created.
QOpenGLShaderProgram: could not create shader program
QOpenGLShader: could not create shader
Could not link shader program:
 ""
QOpenGLShaderProgram: could not create shader program
QOpenGLShader: could not create shader
QOpenGLShaderProgram::uniformLocation(texture): shader program is not linked
QOpenGLShaderProgram::uniformLocation(mat): shader program is not linked
19:25:16: User requested stop. Shutting down...
19:25:16: Application finished with exit code 1.
```