

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 qt5-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 :

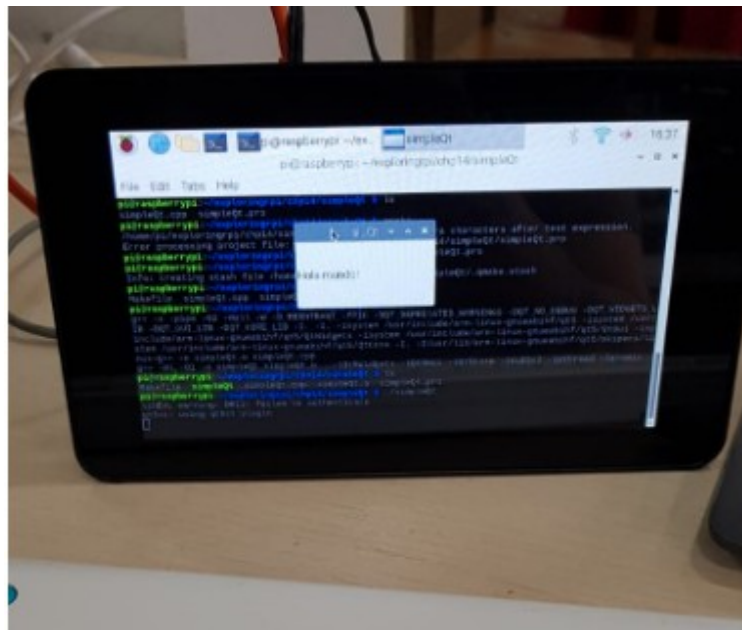
```
~/exploringrpi/chap14/simpleQt $ qmake
~/exploringrpi/chap14/simpleQt $ ls
Makefile simpleQt.cpp simpleQt.pro
```

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:

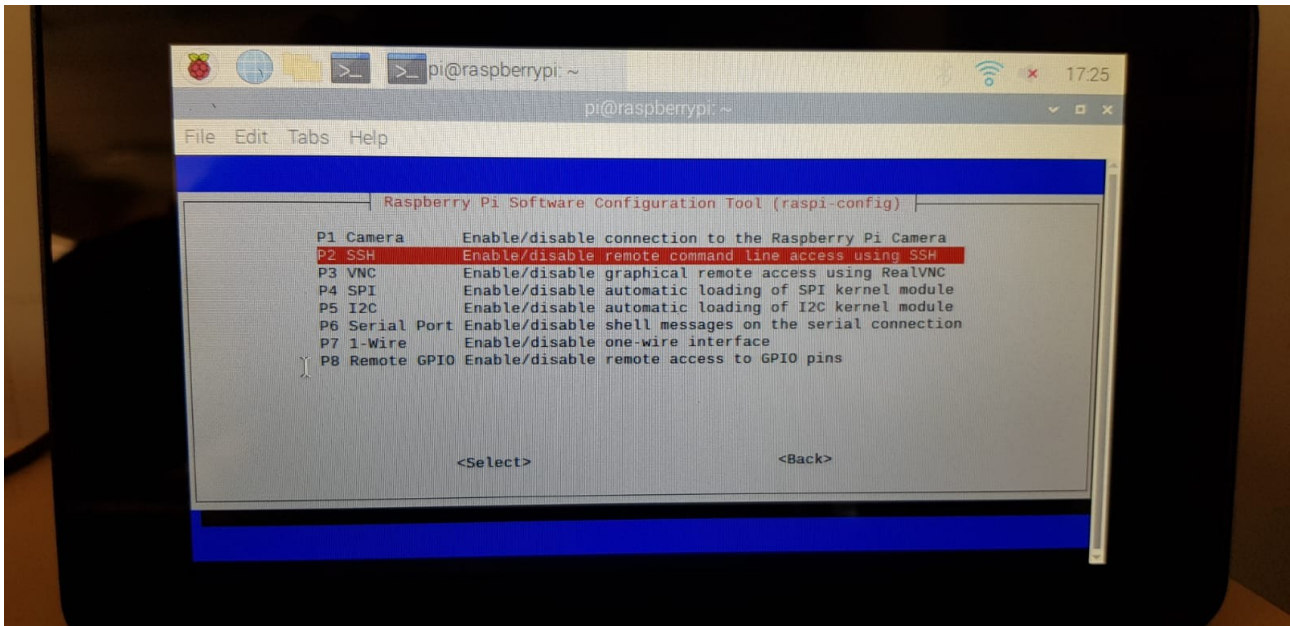
```
~/exploringrpi/chap14/simpleQt $ ls
Makefile simpleQt simpleQt.cpp simpleQt.pro
~/exploringrpi/chap14/simpleQt $ ./simpleQt
```



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 . Inteface options > P2 SSH.



Instalación las librerías de desarrollo

Necesitamos instalar algunas librerías, 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 Raspberry Pi en nuestro ordenador de trabajo.

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

Podemos utilizar rsync para sincronizar nuestro sysroot del ordenador y la Raspberry Pi. Así, si hacemos cambios en nuestro ordenador, será fácil transferirlos a nuestro Raspberry Pi; raspberrypi_ip es el nombre de interfaz de red o la dirección 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 está 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
```