



# Instalación de Arch-Linux en BeagleBone Black

Ignacio Alvarez García – Octubre 2018

# Indice

Instalación de compiladores para ARM-V7-hf	.1
Instalación de Qt5 para ARM-V7-hf	.1
Instalación de depurador remoto	.2
Copia de archivos instalados en ARM	.2
Instalación de kit en Qt Creator	.3
Uso de kit en Qt Creator (programas modo consola)	.6
Uso de kit en Qt Creator (programas con widgets)	.7
Instalar aplicación para ejecución en el arranque	.8
Crear archivo con script de arranque	.8
Crear archivo de servicio	.8
Copiar archivos a BBB	.9
Instalar servicio en BBB	.9
Desinstalar servicio en BBB	.9





Es necesario instalar la compilación cruzada desde Qt para BBB. No se puede utilizar el compilador tal cual porque genera código para PC (Intel-x64), mientras que la BBB tiene un microprocesador ARM-V7-hf.

## Instalación de compiladores para ARM-V7-hf

• Entrar en /home/developer/BBB-Installs (o directorio similar que se haya creado):

```
$ cd /home/developer/BBB-Installs
```

Descargar y descomprimir compilador para ARM:

```
$ wget -c https://releases.linaro.org/components/toolchain/binaries/6.4-2018.05/ar
m-linux-gnueabihf/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
$ tar -xvf gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
$ export CC=`pwd`/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/arm-linu
x-gnueabihf-
```

## Instalación de Qt5 para ARM-V7-hf

 Crear directorio RootFS y descomprimir el root file system que ya se había instalado en la SD/EMMC anteriormente:

```
$ mkdir RootFS
```

- \$ sudo bsdtar -xpf ArchLinuxARM-am33x-latest.tar.gz -C RootFS
- Cambiar enlaces simbólicos en directorios RootFS:

```
$ wget https://raw.githubusercontent.com/riscv/riscv-poky/priv-1.10/scripts/sysroo
t-relativelinks.py
```

- \$ chmod +x sysroot-relativelinks.py
- \$ sudo ./sysroot-relativelinks.py RootFS
- Descargar última versión de Qt5 desde git:

```
$ mkdir -p git
$ cd git
$ git clone git://code.qt.io/qt/qt5.git
$ cd qt5
$ perl init-repository --module-subset=qtbase,qtserialport,qtimageformats
```

Configurar y compilar:

```
$ ./configure -v -qpa linuxfb -no-opengl -device linux-rasp-pi-g++ -device-option
CROSS_COMPILE=$CC -sysroot /home/developer/BBB-Installs/RootFS -opensource -confir
m-license -optimized-qmake -reduce-exports -release -make libs -nomake examples
-qt-zlib -qt-libpng -qt-libjpeg -prefix /opt/Qt/BBB
```

```
$ make -j4
$ sudo make install
```





## Instalación de depurador remoto

Descargar fuentes de gdb y compilar para ARM-V7-hf:

```
$ cd /home/developer/BBB-Installs
$ wget http://ftp.gnu.org/gnu/gdb/gdb-8.2.tar.gz
$ tar -xvf gdb-8.2.tar.gz
$ cd gdb-8.2/gdb/gdbserver
$ ./configure --host=arm-linux-gnueabihf --prefix=/home/developer/BBB-Installs/Roo
tFS/opt CC=${CC}gcc CXX=${CC}g++ CXXLD=${CC}g++
$ make -j4
$ sudo make install
```

Instalar gdb-multiarch:

\$ sudo apt-get install gdb-multiarch

### Copia de archivos instalados en ARM

Es necesario copiar los archivos en /opt, que sólo está permitido para el usuario root, así que lo primero es desbloquear el acceso ssh como root:

Entrar en BBB con ssh:

```
$ ssh alarm@192.168.100.23
$ su (clave root)
# nano /etc/ssh/sshd_config
Cambiar la línea:
```

#PermitRootLogin prohibit-password

Por:

PermitRootLogin yes

En la sección #Ciphers, añadir línea:

Ciphers +aes128-cbc

Guardar archivo.

Reiniciar servicio sshd y volver a usuario alarm:

```
# systemctl restart sshd
# exit
$
```

Abrir dos ventanas del gestor de archivos, una con la carpeta /home/developer/BBB-Installs/RootFS/opt y otra con la carpeta sftp://root@192.168.100.23/opt/ (clave root)





Copiar archivos de la 1ª carpeta a la 2ª.

Una vez finalizada la copia, cerrar ambas carpetas, y restaurar si se desea el PermitRootLogin al estado inicial (permitir login de root con ssh se considera un agujero de seguridad).

# Instalación de kit en Qt Creator

Crear nuevo dispositivo

Arrancar QtCreator, e ir a Tools -> Options \* Devices -> Add... Generic Linux Device Name : BBB IP : 192.168.100.23 User : alarm Pass: alarm [Next] \* New Generic Linux Device Configuration S

•	New Generic Linux Device Config	juration Setup	+	×
Connection				
Connection Summary	The name to identify this configuration: The device's host name or IP address: The username to log into the device: The authentication type: The user's password: The file containing the user's private key:	BBB 192.168.100.23 alarm Password Key Agent weloper/.ssh/id_rsa Browse		
		Next > Ca	ncel	

Una vez añadido, escribir el nombre del depurador remoto:

GDB server executable: /opt/bin/gdbserver





<b>-</b>	Options	+
Filter         Filter         Kits         Text Editor         FakeVim         Help         C++         Q t Quick         Build & Run         Debugger         Designer         E Analyzer         Westion Control	Options         Devices         Android QNX Devices         Device: BBB (default for Generic Linux)         General         Name: BBB         Type: Generic Linux         Auto-detected: No         Current state: Unknown         Type Specific         Machine type:       Physical Device         Authentication type:       Password Key Via ssh-agent         Host name:       192.168.100.23         ŞSH port:       22         Free ports:       10000-10100         Timeout:       10s	+ <u>A</u> dd <u>B</u> ernove <u>Set As Default</u> <u>Test</u> Show Running Processes <u>Deploy Public Key</u>
E       Analyzer         IVersion Control         IVersion Contro         IVersion Contro <td>Host name:       192.168.100.23       \$25H port:       \$2       \$       Check host key         Free ports:       10000-10100       Timeout:       10s       \$         Username:       alarm       *****       Show password         Private key file:       Browse       Create New         GDB server executable:       /opt/bin/gdbserve</td> <td></td>	Host name:       192.168.100.23       \$25H port:       \$2       \$       Check host key         Free ports:       10000-10100       Timeout:       10s       \$         Username:       alarm       *****       Show password         Private key file:       Browse       Create New         GDB server executable:       /opt/bin/gdbserve	
	√ Apply	× <u>C</u> ancel √ <u>O</u> K

Pulsar [Apply]

• Añadir depurador local:

Dentro de Options, seleccionar Kits -> Debugger y añadir nuevo depurador:

- Name: gdb-ARM
- Path: /usr/bin/gdb-multiarch

			options				
ilter	Kits						
TH Kits	Kits Qt Versions	Compilers	Debuggers	Qbs	CMake		
Environment	Name		Location		Туре		Add
Text Editor	<ul> <li>Auto-detected</li> <li>System GDB a</li> </ul>	t /usr/bin/adb	/usr/bin/adb		GDB		Clone
FakeVim	<ul> <li>Manual gdb-ARM</li> </ul>		/usr/bin/adb-r	nultiarch	GDB		Remove
Help			,, j				
(} C++							
Qt Quick							
> Build & Run							
🖡 Debugger							
Designer	Name:	gdb-ARM					
Analyzer	Path:	/usr/bin/gdb	-multiarch			Browse	
Version Control	Type:						
Devices	ABIs:	x86-linux-ger	neric-elf-64bit				
Code Pasting	Version:						
Testing	Working directory:					Browse	

Pulsar [Apply]

• Añadir compiladores:

Dentro de Options, seleccionar Kits -> Compiler y añadir nuevos compiladores GCC para C++:

- Name: gcc-ARM-C++





- Path: /home/developer/BBB-Installs/gcc-linaro-6.4.1-2018.05-x86\_64\_arm-linuxgnueabihf/bin/arm-linux-gnueabihf-gcc

Y para GCC C:

- Name: gcc-ARM-C
- Path: /home/developer/BBB-Installs/gcc-linaro-6.4.1-2018.05-x86\_64\_arm-linuxgnueabihf/bin/arm-linux-gnueabihf-gcc

<b>•</b>	Options + X			
Filter	Kits			
🖬 Kits	Kits Qt Versions Compilers Debuggers Qbs CMake			
Environment     Text Editor	Name         Type           GCC (C++, x86 64bit in /usr/bin)         GCC           GCC (C++, x86 64bit in /usr/bin)         GCC	Add -		
K FakeVim	GCC 7 (C++, X86 32bit in / Usr/bin) GCC GCC 7 (C++, X86 32bit in / usr/bin) GCC	Remove		
Help	GCC (C++, x86 64bit in /usr/bin) GCC GCC (C++, x86 24bit in /usr/bin) GCC GCC 7 (C++, x86 64bit in /usr/bin) GCC			
🗘 C++	GCC 7 (C++, x86 32bit in /usr/bin) GCC * Manual			
▶ Build & Run	GCC-ARM-C GCC ▼ C++ ▼			
Debugger				
✓ Designer	Name: GCC-ARM-C			
Analyzer	Compiler path: aro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gc			
Version Control	Platform codegen flags:			
Devices	Platform linker flags:			
Code Pasting	ABI: arm-linux v - linux v - generic v - elf v			
Testing				
	✓ Apply ¥ Cancel	<u>√о</u> к		

Pulsar [Apply]

• Añadir versión Qt:

Dentro de Options, seleccionar Kits -> Qt Versions y añadir versión de Qt seleccionando el path para qmake:

/home/developer/BBB-Installs/RootFS/opt/Qt/BBB/bin/qmake

ter	Kits	
Kits	Kits Qt Versions Compilers Debuggers Qbs CMake	
D Environment	Name • qmake Location	Add
Text Editor	<ul> <li>Auto-detected Qt 5.11.2 GCC 64bit /opt/Qt/5.11.2/gcc_64/bin/qmake</li> </ul>	Remove
FakeVim	Manual     Qt 5.11.2 (BBB) //home/developer/BBB-Installs/RootF5/opt/Qt/BBB/bin/qmake	
Help		
C++		
Qt Quick		
Build & Run		
Debugger		
Designer		
Analyzer		
Version Control		
Devices	Version name: Qt %(Qt:Version) (BBB)	
Code Pasting	qmake location: /home/developer/BBB-Installs/RootFS/opt/Qt/BBB/bin/qmake Browse	
Testing	Qt version 5.11.2 for Embedded Linux Details *	
A lesting	Qt version 5.11.2 for Embedded Linux Details ~	

Pulsar [Apply]





• Finalmente, añadir kit BBB, utilizando los elementos anteriores:

 Dentro de Options, seleccionar Kits -> Kits y añadir nuevo kit: Name : BBB
 Device type : Generic Linux Device
 Device : BBB
 Sysroot: /home/developer/BBB-Installs/RootFS
 Compiler C: GCC-ARM-C
 Compiler C++: GCC-ARM-C++
 Debugger: gdb-ARM
 Qt versión: Qt-5.11.2 (BBB)

TT Kits	Kits Ot Versions Compilers Debuggers Obs CMake	
	Name	Add
Environment	Auto-detected Desktop 0t 5 11 2 GCC 64bit (default)	Clone
Text Editor	<ul> <li>Manual</li> </ul>	Remove
FakeVim		Make Default
Help		Make Default
() C++	Name: BBB	<b>.</b>
	File system name:	]
- QUQUICK	Device type:	
Build & Run	Device: BBB (default for Generic Linux)	Manage
🗰 Debugger	Compate (development DDD (development)	Desures
✓ Designer	Systool: Informe/developer/BBB-Installs/RootES	Browse
Analyzer	C: GCC-ARM-C *	Manage
Version Control	C++: GCC-ARM-C++	]
	Environment: No changes to apply.	Change
	Debugger: adb-ARM	Manage
Code Pasting	Othersion: Ot 5 11 2 (PPP)	Manago
Testing		ivianaye
	Qt mkspec:	
	Additional Qbs Profile Settings	Change
	CMake Tool:	Manage
	CMake generator: <none> - <none>, Platform: <none>, Toolset: <none></none></none></none></none>	Change
	CMake Configuration CMAKE CXX COMPILER:STRING=%{Compiler:Executable:Cxx}; CMAKE C COMPILER:STRING=	Change

Pulsar [Apply]

# Uso de kit en Qt Creator (programas modo consola)

Una vez instalado el nuevo kit, cuando se crea un nuevo proyecto solicitará los kits para los que debe estar disponible: seleccionar ambos (Desktop y BBB).

Modificar archivo .pro, cambiando el directorio para el deployment (target.path), que debe existir en la BBB (si no existe, crearlo desde BBB con \$ mkdir /home/alarm/Tests ):

```
# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /home/alarm/Tests/
!isEmpty(target.path): INSTALLS += target
```

Seleccionar el kit a aplicar:





-			main.cpp @ Qt-Test - Qt Creator
<u>Eile E</u> dit	Build Debug Analyze Tool	ls <u>W</u> indow <u>H</u> elp	
	Projects 🗢 🕈 🗸 👄	▶ 🗄 🖬 < 🔺 🗉	nain.cpp 🗢 🗢 🗘
Welcome	<ul> <li>Qt-Test</li> <li>Qt-Test.pro</li> <li>Cources</li> </ul>	1 2	<pre>#include <qcoreapplication> #include <qdebug></qdebug></qcoreapplication></pre>
Edit	💼 main.cpp	3 4 • 5	<pre>int main(int argc, char *argv[]) {</pre>
		6	QcoreApplication a(argc, argv);
Ŵ		8	<pre>qDebug() &lt;&lt; "HELLO WORLD";</pre>
Debug		10	<pre>return a.exec();</pre>
Juni Insta		11	}
Projects Projects			
	Project: <b>Qt-Test</b> Deploy: <b>Deploy to Remote Linux</b> Run: <b>Qt-Test</b>	x Host	
Qt-Test	Kit	Build	
Γ,	BBB	Debug	
Debug	Desktop Qt 5.11.2 GCC 64bit	Profile	
		Release	
Â			
>	Type to locate (Ctrl+K)	1 Issues	1 2 Search Results 3 Application Output 4 Compile Output 5 Debugger Console 8 Test R

Si se selecciona el kit Desktop, la aplicación se ejecutará en el equipo de desarrollo.

Si se selecciona el Kit BBB, la aplicación se descargará y ejecutará en la BBB.

### Uso de kit en Qt Creator (programas con widgets)

También se pueden ejecutar en la BBB programas tipo gráfico (con widgets). En este caso, es necesario:

 Copiar en la carpeta adecuada del disco de la BBB (micro-SD o EMMC) los archivos con fuentes (tipos de letra):

```
$ mkdir /home/developer/BBB-Installs/fonts
$ find /usr/share/fonts -name *.ttf -exec cp {} /home/developer/BBB-Installs/fonts/ \;
```

Mediante una ventana de navegador de archivos, entrar como root en sftp://192.168.100.23, y moverse a la carpeta /opt/Qt/BBB/lib. Crear en ella el directorio fonts y moverse a dicho directorio.

En otra ventana del navegador, abrir la carpeta /home/developer/BBB-Installs/fonts.

Copiar los contenidos de la 2ª carpeta a la 1ª.

• Autorizar al usuario 'alarm' a utilizar la pantalla de vídeo:

\$ ssh alarm@192.168.100.23	(clave alarm)
[alarm@alarm ~]\$ su	(clave root)
<pre>[root@alarm alarm]# usermod -a</pre>	-G video alarm
[root@alarm alarm]# reboot now	(provocará un reinicio de la BBB)

• Ya se puede conectar una pantalla al conector mini-HDMI disponible, y teclado/ratón a través del USB, y ejecutar aplicaciones con widgets:







## Instalar aplicación para ejecución en el arranque

Una vez se dispone de una aplicación operativa, se puede instalar como un servicio para que se ejecute en el arranque de la BBB.

El siguiente ejemplo supone que se desean ejecutar dos aplicaciones en el arranque:

/home/alarm/Tests/AplicacionAEjecutar1 param11 param12

/home/alarm/Tests/AplicacionAEjecutar2 param2

### Crear archivo con script de arranque

En el equipo host, con el editor de texto, crear archivo Shell con los ejecutables a arrancar (ej. initRover.sh):

#!/bin/bash
# ttyS1 = 115200 baud
eval stty -F /dev/ttyS1 115200

# Arrancar programas
/home/alarm/Tests/AplicacionAEjecutar1 param11 param12 &
/home/alarm/Tests/AplicacionAEjecutar2 param2

### Crear archivo de servicio

En el equipo host, con el editor de texto, crear archivo con el nombre del servicio (ej. initRover.service):

[Unit] Description=init script After=syslog.target network.target [Service] Type=simple ExecStart=/home/alarm/Tests/initRover.sh [Install] WantedBy=multi-user.target





#### Copiar archivos a BBB

Se puede hacer desde ventana de comandos en el equipo host:

```
$ cd /home/developer/.../directorio_de_archivos_sh_y_service
$ scp initRover.* alarm@192.168.100.23:/home/alarm/Tests/
```

#### Instalar servicio en BBB

```
$ ssh alarm@192.168.100.23 (clave alarm)
[alarm@alarm ~]$ su (clave root)
[root@alarm alarm]# cp /home/alarms/Tests/initRover.service /etc/systemd/system/
[root@alarm alarm]# systemctl start initRover.service
[root@alarm alarm]# systemctl enable initRover.service
```

En el siguiente arranque, la aplicación se ejecutará automáticamente.

Desinstalar servicio en BBB

\$ ssh alarm@192.168.100.23 (clave alarm)
[alarm@alarm ~]\$ su (clave root)
[root@alarm alarm]# systemctl stop initRover.service
[root@alarm alarm]# systemctl disable initRover.service

En el siguiente arranque, la aplicación no se ejecutará.