



Developer Study Guide:

Deploying BlueZ v5.50 on Raspberry Pi Board *Part 1 - Deployment*

BlueZ is the official Linux *Bluetooth*[®] protocol stack. From the release notes of BlueZ [v5.47](#), “this release comes with initial support for it in the form of a new *meshctl* tool. Using this tool, it’s possible to provision mesh devices through the GATT Provisioning Bearer (PB-GATT), as well as communicate with them (e.g. configure them) using the GATT Proxy protocol.” This Developer Study Guide, explains how to install the latest release, [BlueZ v5.50](#) on Raspberry Pi.

Author: Kai Ren

Version: 1.3

Revision Date: 26 July 2019

Revision History

Version	Date	Author	Changes
1.0	31 May 2018	Kai Ren	Initial Version
1.1	29 August 2018	Kai Ren	Upgrade BlueZ installation to v5.50
1.2	19 March 2019	Kai Ren	Updated the name to Developer Study Guide. Use latest Raspberry Pi release instead of master tree.
1.3	26 July 2019	Kai Ren	Add the support for Raspberry Pi 4 and update the kernel to raspberrypi-kernel_1.20190709-1 .

table of contents

1.0 Prerequisite	4
2.0 Install BlueZ v5.50	4
2.1 Remote Access R Pi3 Through SSH	4
2.2 Install Dependencies for BlueZ	5
2.3 Install json-c for BlueZ v5.50	5
2.4 Install e// for BlueZ v5.50	5
2.5 Get BlueZ v5.50 source code	5
2.6 Compile and Install BlueZ	6
3.0 Rebuild the Kernel for BlueZ v5.50	8
3.1 Install Kernel Building Dependencies	8
3.2 Checking Out Building Tool and R Pi3 Source Code	8
3.3 Configuring the Kernel	8
3.4 Build and Install the Kernel, Modules, and Device Tree blobs	11
3.5 Verifying Kernel Installation	11
4.0 Summary	13

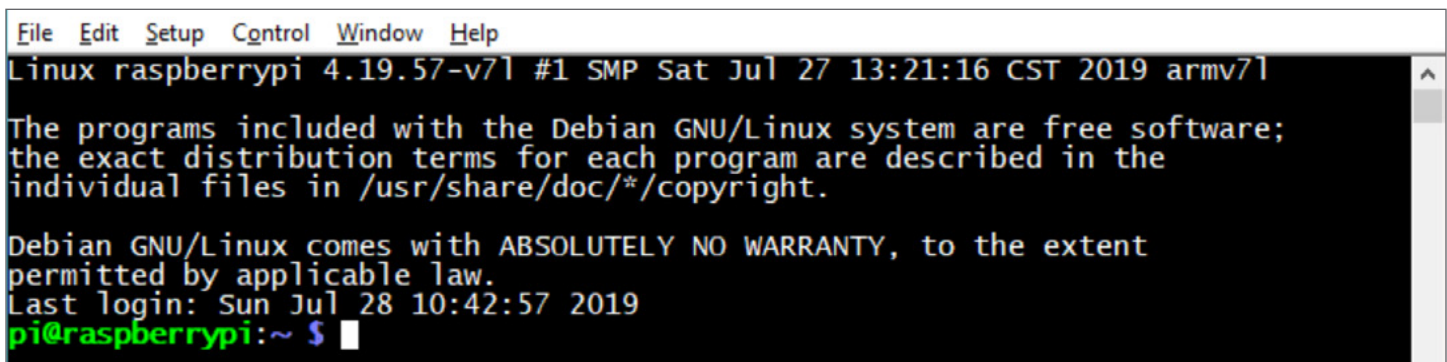
1.0 Prerequisite

This study guide has been tested on the following boards, calling them verified boards in this document:

- Raspberry Pi 2B
- Raspberry Pi 3B
- Raspberry Pi 3B+
- Raspberry Pi 4B

If you have one of above verified boards, please make sure that you:

- Follow this [guide](#) to setup your Raspberry Pi.
- Check if the operating system on your verified board is ready, and, if not, follow this [guide](#) to set up the software on your Raspberry Pi.
- Follow this [guide](#) to enable SSH to access the board remotely. The picture below shows the use of [Tera Term](#) on a Windows10 laptop through SSH to access the board remotely.



```
File Edit Setup Control Window Help
Linux raspberrypi 4.19.57-v7l #1 SMP Sat Jul 27 13:21:16 CST 2019 armv7l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 28 10:42:57 2019
pi@raspberrypi:~ $
```

- The board has been issued “apt-get update” and “apt-get upgrade” successfully, these two commands will ensure your board has the latest updates.

2.0 Install BlueZ v5.50

Once the board is setup correctly, you can start to install BlueZ v5.50.

2.1 Remote Access Board Through SSH

As mentioned in the [Prerequisite](#), you should remote login into the board through SSH. You need to make sure that your Windows computer is in the same LAN with the board and you know the IP address of the board.

2.2 Install Dependencies for BlueZ

```
sudo apt-get install -y git bc libusb-dev libdbus-1-dev libglib2.0-dev libudev-dev libical-dev libreadline-dev autoconf
```

2.3 Install json-c

```
cd ~  
wget https://s3.amazonaws.com/json-c_releases/releases/json-c-0.13.tar.gz  
tar -xvf json-c-0.13.tar.gz  
cd json-c-0.13/  
./configure --prefix=/usr --disable-static && make  
sudo make install
```

2.4 Install e// for BlueZ v5.50

```
cd ~  
wget https://mirrors.edge.kernel.org/pub/linux/libs/ell/ell-0.6.tar.xz  
tar -xvf ell-0.6.tar.xz  
cd ell-0.6/  
sudo ./configure --prefix=/usr  
sudo make  
sudo make install
```

2.5 Get BlueZ v5.50 Source Code

```
cd ~  
wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.50.tar.xz  
tar -xvf bluez-5.50.tar.xz  
cd bluez-5.50/
```

2.6 Compile and Install BlueZ

```
./configure --enable-mesh --prefix=/usr --mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var  
  
make  
  
sudo make install
```

To make sure the upgrade we you want to install is BlueZ to v5.50, tell systemd to use the new bluetooth daemon:

```
sudo vi /lib/systemd/system/bluetooth.service
```

After opening this file, `bluetooth.service`, make sure the `ExecStart` line points to your new daemon in `/usr/libexec/bluetooth/bluetoothd`, as shown in the screenshot below.



```
pi@raspberrypi: ~/bluez-5.50/mesh  
[Unit]  
Description=Bluetooth service  
Documentation=man:bluetoothd(8)  
ConditionPathIsDirectory=/sys/class/bluetooth  
  
[Service]  
Type=dbus  
BusName=org.bluez  
ExecStart=/usr/libexec/bluetooth/bluetoothd  
NotifyAccess=main  
#WatchdogSec=10  
#Restart=on-failure  
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE  
LimitNPROC=1  
ProtectHome=true  
ProtectSystem=full  
  
[Install]  
WantedBy=bluetooth.target  
Alias=dbus-org.bluez.service  
~
```

Up till now, that wasn't enough. You still need to create a symlink from the old `bluetoothd` to the new one. First, rename the old file for backup, type below command and you will find the backup file as below screenshot shown.

```
sudo cp /usr/lib/bluetooth/bluetoothd /usr/lib/bluetooth/bluetoothd-543.orig
```

```
pi@raspberrypi: /usr/lib/bluetooth
pi@raspberrypi:~/bluez-5.50/mesh $ cd ~
pi@raspberrypi:~ $ sudo cp /usr/lib/bluetooth/bluetoothd /usr/lib/bluetooth/bluetoothd-543.orig
pi@raspberrypi:~ $ cd /usr/lib/bluetooth/
pi@raspberrypi:~/usr/lib/bluetooth $ ls -l
total 1600
-rwxr-xr-x 1 root root 816996 Oct 20  2017 bluetoothd
-rwxr-xr-x 1 root root 816996 Aug 11 03:39 bluetoothd-543.orig
pi@raspberrypi:~/usr/lib/bluetooth $
```

Create the symlink using the command below and double check the version of *bluetoothd* and *meshctl*.

```
sudo ln -sf /usr/libexec/bluetooth/bluetoothd /usr/lib/bluetooth/bluetoothd
sudo systemctl daemon-reload
bluetoothd -v
meshctl -v
```

As shown in the screenshot below, *bluetoothd* and *meshctl* are all v5.50. This means that BlueZ v5.50 installation is successful.¹

```
pi@raspberrypi: ~
pi@raspberrypi:~ $ cd ~
pi@raspberrypi:~ $ bluetoothd -v
5.50
pi@raspberrypi:~ $ meshctl -v
meshctl: 5.50
pi@raspberrypi:~ $
```

But if you type *meshctl* and click the *Enter* key to start the service, an error message, “Failed to parse provisioning database file prov_db.json”, will pop-up as below:

```
pi@raspberrypi:~ $ cd bluez-5.50/mesh
pi@raspberrypi:~/bluez-5.50/mesh $ meshctl
Failed to parse provisioning database file prov_db.json
pi@raspberrypi:~/bluez-5.50/mesh $
```

[The next section](#) will tell you how to solve this problem in order to initiate *meshctl* service.



¹ About upgrading *bluetoothd*, reference this article

<https://raspberrypi.stackexchange.com/questions/66540/installing-bluez-5-44-onto-raspbian>

3.0 Rebuilding the Kernel for BlueZ v5.50

There are two main methods for building the kernel. You can build locally on a Raspberry Pi, which will take a long time, or you can cross compile, which is much quicker but requires more setup. This article outlines the local building method.

3.1 Install Kernel Building Dependencies

```
sudo apt-get install -y git bc bison flex libssl-dev
```

3.2 Check Out Building Tool and Source Code

```
cd ~  
wget https://github.com/raspberrypi/linux/archive/raspberrypi-kernel_1.20190709-1.tar.gz  
tar -xvf raspberrypi-kernel_1.20190709-1.tar.gz
```

3.3 Configuring the Kernel

```
cd ~  
cd ./linux-raspberrypi-kernel_1.20190709-1/
```

Depending on your Raspberry Pi board version, run the following commands alternatively.

- Raspberry Pi 2, Pi 3, Pi 3+, and Compute Module 3² default build configuration

```
KERNEL=kernel7  
make bcm2709_defconfig  
make menuconfig
```

- Raspberry Pi 4

```
KERNEL=kernel7l  
make bcm2711_defconfig  
make menuconfig
```

After typing *menuconfig*, the kernel configuration menu will pop up. The *menuconfig* utility has simple keyboard navigation. After a brief compilation, you will be presented with a list of submenus containing all the options you can configure; there's a lot, so take your time to read through them and get acquainted.

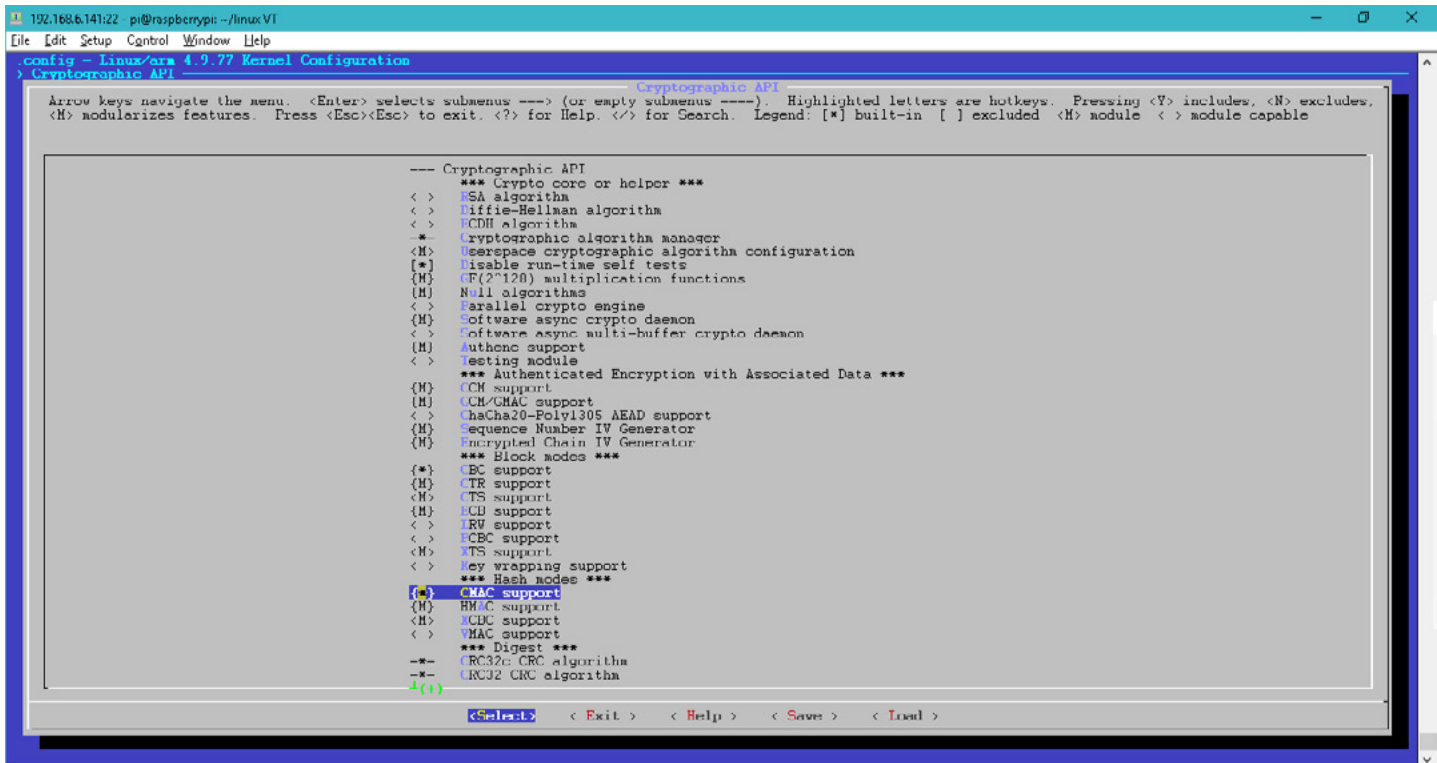
Use the arrow keys to navigate, the *Enter* key to enter a submenu (indicated by --->), *Escape* twice to go up a level or exit, and the space bar to cycle the state of an option. Some options have multiple choices, in which case they will appear as a submenu and the Enter key will select an option. You can press h on most entries to get help about that specific option or menu³.

Please include the three modules below:

Select *Cryptographic API* → *CMAC support*

Select *Cryptographic API* → *User-space interface for hash algorithms*

Select *Cryptographic API* → *User-space interface for symmetric key cipher algorithms*



³ About this part, reference this article, <https://www.raspberrypi.org/documentation/linux/kernel/configuring.md>

```
192.168.6.141:22 pi@raspberrypi:~/linux/VI
File Edit Setup Control Window Help
config - Linux/arm 4.9.77 Kernel Configuration
) Cryptographic API
Cryptographic API
Arrow keys navigate the menu. <Enter> selects submenus --- (or empty submenu ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit. <?> for Help. </> for Search. Legend: [*] built-in [ ] excluded <M> module <> module capable

<-->
<> SHA3 digest algorithms
<M> Tiger digest algorithms
<M> Whirlpool digest algorithms
*** Ciphers ***
--*-- AES cipher algorithms
<> Anubis cipher algorithm
{M} ARC4 cipher algorithm
<> Blowfish cipher algorithm
<> Camellia cipher algorithm
<M> CAST5 (CAST-128) cipher algorithm
<> CAST6 (CAST-256) cipher algorithm
[*] DES and Triple DES EDE cipher algorithms
<> FCrypt cipher algorithm
<> Khazad cipher algorithm
<> Salsa20 stream cipher algorithm
<> ChaCha20 cipher algorithm
<> SEED cipher algorithm
<> Serpent cipher algorithm
<> TEA, XTEA and XETA cipher algorithms
<> Twofish cipher algorithm
*** Compression ***
{M} Deflate compression algorithm
{M} LZO compression algorithm
<> 842 compression algorithm
<M> LZ4 compression algorithm
<> LZ4HC compression algorithm
*** Random Number Generation ***
<> Pseudo Random Number Generation for Cryptographic modules
<M> NIST SP800-90A DRBG ----
<M> Jitterentropy Non-Deterministic Random Number Generator
[*] User-space interface for hash algorithms
<M> User-space interface for symmetric key cipher algorithms
<> User-space interface for random number generator algorithms
<> User-space interface for AEAD cipher algorithms
[ ] Hardware crypto devices ----
[ ] Asymmetric (public-key cryptographic) key type ----
[ ] Certificates for signature checking ----
[*] ARM Accelerated Cryptographic Algorithms ----

<Select> <Exit> <Help> <Save> <Load>
```

```
192.168.6.141:22 pi@raspberrypi:~/linux/VI
File Edit Setup Control Window Help
config - Linux/arm 4.9.77 Kernel Configuration
) Cryptographic API
Cryptographic API
Arrow keys navigate the menu. <Enter> selects submenus --- (or empty submenu ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit. <?> for Help. </> for Search. Legend: [*] built-in [ ] excluded <M> module <> module capable

<-->
<> SHA3 digest algorithms
<M> Tiger digest algorithms
<M> Whirlpool digest algorithms
*** Ciphers ***
--*-- AES cipher algorithms
<> Anubis cipher algorithm
{M} ARC4 cipher algorithm
<> Blowfish cipher algorithm
<> Camellia cipher algorithm
<M> CAST5 (CAST-128) cipher algorithm
<> CAST6 (CAST-256) cipher algorithm
[*] DES and Triple DES EDE cipher algorithms
<> FCrypt cipher algorithm
<> Khazad cipher algorithm
<> Salsa20 stream cipher algorithm
<> ChaCha20 cipher algorithm
<> SEED cipher algorithm
<> Serpent cipher algorithm
<> TEA, XTEA and XETA cipher algorithms
<> Twofish cipher algorithm
*** Compression ***
{M} Deflate compression algorithm
{M} LZO compression algorithm
<> 842 compression algorithm
<M> LZ4 compression algorithm
<> LZ4HC compression algorithm
*** Random Number Generation ***
<> Pseudo Random Number Generation for Cryptographic modules
<M> NIST SP800-90A DRBG ----
<M> Jitterentropy Non-Deterministic Random Number Generator
[*] User-space interface for hash algorithms
<M> User-space interface for symmetric key cipher algorithms
<> User-space interface for random number generator algorithms
<> User-space interface for AEAD cipher algorithms
[ ] Hardware crypto devices ----
[ ] Asymmetric (public-key cryptographic) key type ----
[ ] Certificates for signature checking ----
[*] ARM Accelerated Cryptographic Algorithms ----

<Select> <Exit> <Help> <Save> <Load>
```

Once you are done making the changes you want, press *Escape* until you're prompted to save your new configuration. By default, this will save to the `.config` file. You can save and load configurations by copying this file around.



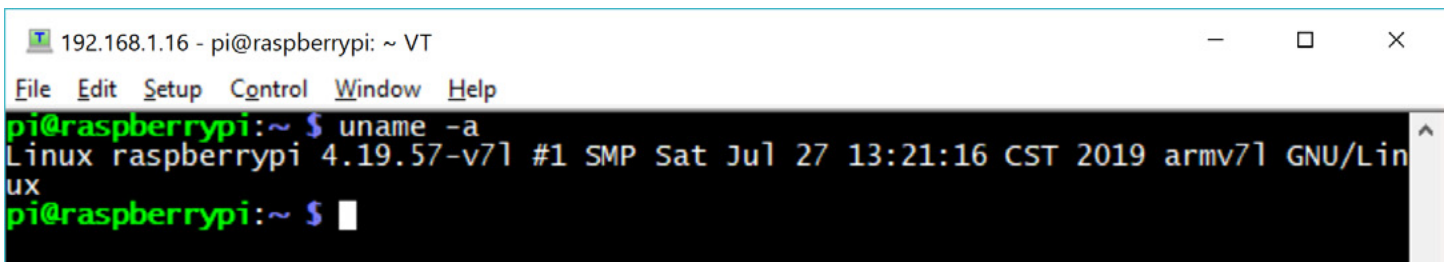
3.4 Build and Install the Kernel, Modules, and Device Tree blobs

```
make -j4 zImage modules dtbs
sudo make modules_install
sudo cp arch/arm/boot/dts/*.dtb /boot/
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
sudo reboot
```

This process takes a longtime, maybe 2 to 3 hours.

3.5 Verifying Kernel Installation

After the board restart, issue command `uname -a` and a new build time will be shown. In the image below, you can see the build time is **Sat Jul 27 13:21:16 CST 2019**. That time and date was exactly when the kernel was built and it means the kernel building and installation was successful.



```
192.168.1.16 - pi@raspberrypi: ~ VT
File Edit Setup Control Window Help
pi@raspberrypi:~ $ uname -a
Linux raspberrypi 4.19.57-v7l #1 SMP Sat Jul 27 13:21:16 CST 2019 armv7l GNU/Linux
pi@raspberrypi:~ $ █
```

Type `meshctl` in folder `~/bluez-5.50/mesh` to ensure it will work correctly, as shown in the image below.

```
pi@raspberrypi: ~/bluez-5.50/mesh
pi@raspberrypi:~ $ cd ./bluez-5.50/mesh/
pi@raspberrypi:~/bluez-5.50/mesh $ meshctl
[meshctl]# help
Menu main:
Available commands:
-----
config                Configuration Model Submenu
onoff                 On/Off Model Submenu
list                  List available controllers
show [ctrl]           Controller information
select <ctrl>         Select default controller
security [0(low)/1(medium)/2(high)] Display or change provision security level
info [dev]            Device information
connect [net_idx] [dst] Connect to mesh network or node on network
discover-unprovisioned <on/off> Look for devices to provision
provision <uuid>      Initiate provisioning
power <on/off>        Set controller power
disconnect [dev]      Disconnect device
mesh-info             Mesh networkinfo (provisioner)
local-info            Local mesh node info
menu <name>           Select submenu
version              Display version
quit                 Quit program
exit                 Quit program
help                 Display help about this program
export               Print environment variables
[meshctl]#
```

4.0 Summary

If you go through all the steps listed above, you will have a Raspberry Pi board that can work as a provisioner to provision any dev kits/boards that support PB-GATT. This guide, “[Deploying BlueZ v5.50 on Raspberry Pi 3 and Use It, Part 2 — Provisioning](#)”, shows you how to use *meshctl* to provision and configure a real Bluetooth mesh device.