

# Developer Study Guide: Deploying BlueZ v5.50 on Raspberry Pi3

## Part 1 - Deployment

- [Revision: 1.2](#)
- [Revision Date: Dec 19<sup>th</sup>, 2018](#)
- [Revision History](#)

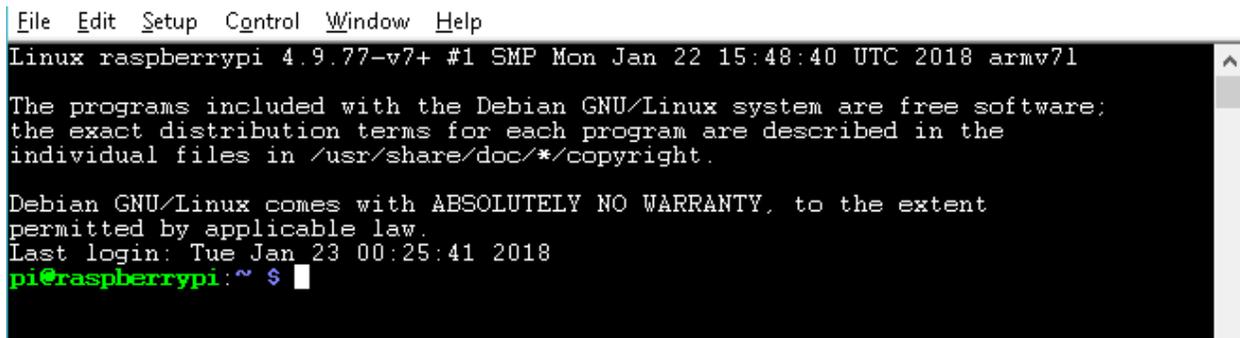
Revision Number	Date	Comments
1.0	2018-06-18	Initial Draft
1.1	2018-08-05	Upgrade BlueZ installation to v5.50.
1.2	2019-03-19	Updated the name to Developer Study Guide. Use latest Raspberry Pi <a href="#">release</a> instead of master tree.

BlueZ is 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.” In this Developer Study Guide, I will guide you how to install latest release, [BlueZ v5.50](#) on Raspberry Pi3 (in short as R Pi3).

## 0. Prerequisite

Before deploying BlueZ, you should have a setup R Pi3 board, setup means:

- The R Pi3 is powered by a USB Micro power supply which can supply at least 2A at 5V
- The TF card should be > 16GB, at least class 4
- The [Raspbian](#) is used in this guide:
  - RASPBIAN STRETCH WITH DESKTOP Image with desktop based on Debian Stretch
  - Version: March 2018
  - Release date:2018-03-13
  - Kernel version:4.9
- Flash the correct Raspbian, this [link](#) will show you how
- Change the login username and password if you want to, or remember the default one:  
Username: **pi**  
Password: **raspberrypi**  
It’s very important because we will use them to login R Pi3 remotely through SSH.
- Please follow this [guide](#) to enable SSH. After that, R Pi3 can be remote accessed through SSH. The image below shows that I use [Tera Term](#) on my Windows10 laptop to access R Pi3 remotely



```
File Edit Setup Control Window Help
Linux raspberrypi 4.9.77-v7+ #1 SMP Mon Jan 22 15:48:40 UTC 2018 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: Tue Jan 23 00:25:41 2018
pi@raspberrypi:~ $
```

- It’s already being “*apt-get update*” and “*apt-get upgrade*”.

## 1. Install BlueZ v5.50

Once R Pi3 is setup correctly, we can start to install BlueZ v5.50.

## Remote access R Pi3 through SSH

Like I mentioned in step 0, you should remote login into R Pi3 through SSH. You need to make sure that your Windows computer is in the same LAN with R Pi3 and you know the IP address of R Pi3.

## 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
```

## 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
```

## Install *ell* 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
```

## 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/
```

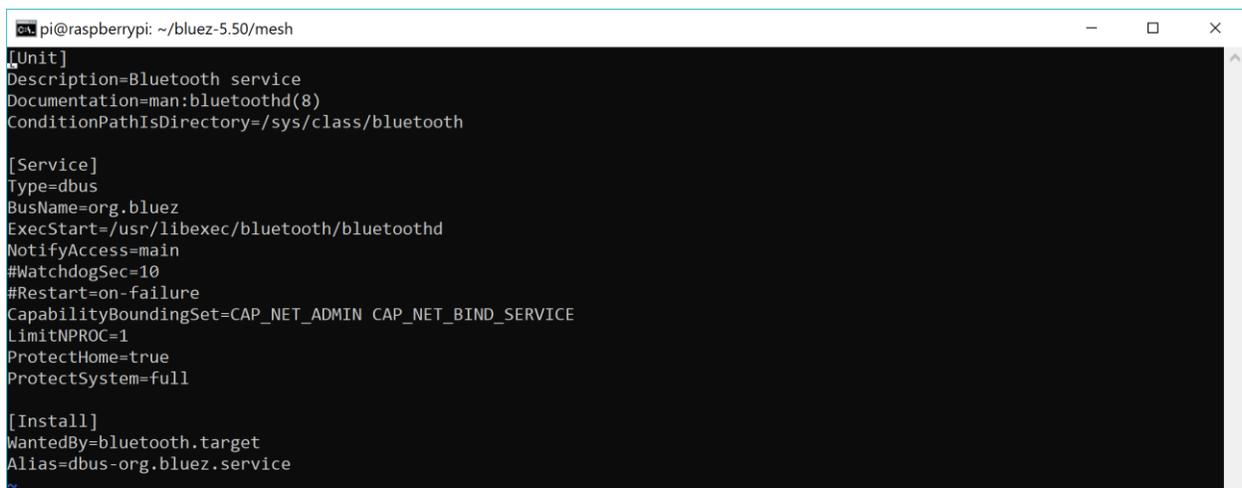
## Compile & Install BlueZ

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

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

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

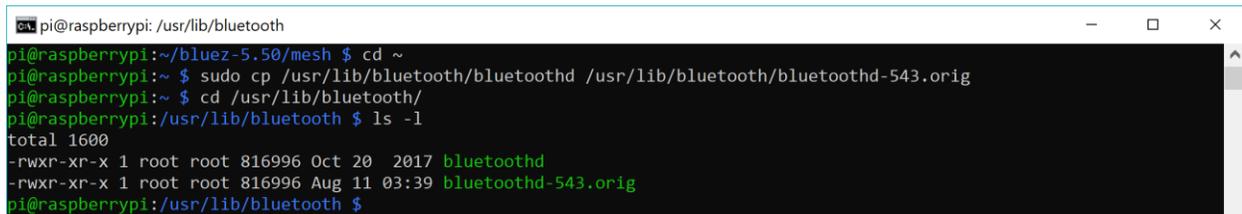
After opening this file, *bluetooth.service*, make sure the *ExecStart* line points to your new daemon in */usr/libexec/bluetooth/bluetoothd*, as shown in the screenshot below.



```
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. <sup>1</sup>



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

## 2. 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. For this article, I selected the local building method.

### Install kernel building dependencies

```
sudo apt-get install -y git bc libncurses5-dev
```

### Checking out building tool and R Pi3 source code

```
cd ~
wget https://github.com/raspberrypi/linux/archive/raspberrypi-kernel\_1.20190215-1.tar.gz
tar -xvf raspberrypi-kernel_1.20190215-1.tar.gz
```

### Configuring the kernel

```
cd ~
cd ./linux-raspberrypi-kernel_1.20190215-1
KERNEL=kernel7
```

---

<sup>1</sup> About upgrading *bluetoothd*, reference this article, <https://raspberrypi.stackexchange.com/questions/66540/installing-bluez-5-44-onto-raspbian>

```
make bcm2709_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'll 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.

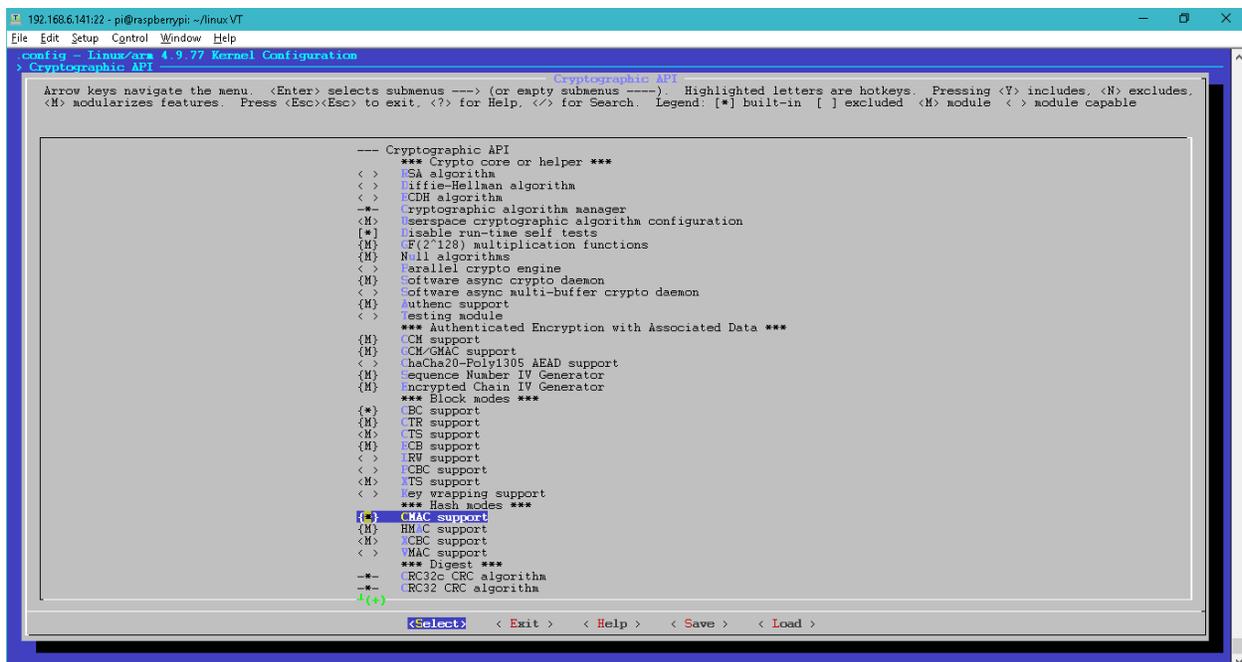
Use the arrow keys to navigate, the Enter key to enter a submenu (indicated by --->), Escape twice to go up a level or exit, and the space bar to cycle the state of an option. Some options have multiple choices, in which case they'll appear as a submenu and the Enter key will select an option. You can press h on most entries to get help about that specific option or menu. <sup>2</sup>

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*



```
192.168.6.141:22 - pi@raspberrypi: ~/linux/VT
File Edit Setup Control Window Help
config - Linux/ors 4.9.77 Kernel Configuration
> Cryptographic API
  Cryptographic API
  Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus --->). 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

  --- Cryptographic API
  *** Crypto core or helper ***
  < > ISA algorithm
  < > Diffie-Hellman algorithm
  < > ICDH algorithm
  -- Cryptographic algorithm manager
  <M> Userspace cryptographic algorithm configuration
  [*] Usable run-time self tests
  <M> F(2^128) multiplication functions
  <M> Null algorithms
  < > Parallel crypto engine
  <M> Software async crypto daemon
  < > Software async multi-buffer crypto daemon
  <M> Authenc support
  < > Testing module
  *** Authenticated Encryption with Associated Data ***
  <M> OCB support
  <M> OCB/CMAC support
  < > ChaCha20-Poly1305 AEAD support
  <M> Sequence Number IV Generator
  <M> Inencrypted Chain IV Generator
  *** Block modes ***
  [*] ECB support
  <M> CTR support
  <M> CTS support
  <M> ECB support
  < > FRV support
  < > CBC support
  <M> LTC support
  < > Key wrapping support
  *** Hash modes ***
  [*] CMAC support
  <M> HMAC support
  <M> CBC support
  < > MAC support
  *** Digest ***
  -- RC32c CRC algorithm
  -- RC32 CRC algorithm
  !(+)
```

<sup>2</sup> About this part, reference this article, <https://www.raspberrypi.org/documentation/linux/kernel/configuring.md>

```
192.168.6.141:22 - pi@raspberrypi: ~/linux/VT
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 algorithm
<M> Tiger digest algorithms
<M> Whirlpool digest algorithms
*** Ciphers ***
-- ES cipher algorithms
<> Anubis cipher algorithm
<M> ARC4 cipher algorithm
<> Blowfish cipher algorithm
<> Camellia cipher algorithms
<M> CAST5 (CAST-128) cipher algorithm
<> CAST6 (CAST-256) cipher algorithm
[*] DES and Triple DES EDE cipher algorithms
<> Crypt cipher algorithm
<> Hazad cipher algorithm
<> Salsa20 stream cipher algorithm
<> ChaCha20 cipher algorithm
<> EED cipher algorithm
<> Serpent cipher algorithm
<> TEA, XTEA and XETA cipher algorithms
<> Wolfish cipher algorithm
*** Compression ***
<M> Deflate compression algorithm
<M> LZ0 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> Litterentropy Non-Deterministic Random Number Generator
[*] User-space interface for hash algorithms
[*] 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 ----
[ ] Symmetric (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/VT
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 algorithm
<M> Tiger digest algorithms
<M> Whirlpool digest algorithms
*** Ciphers ***
-- ES cipher algorithms
<> Anubis cipher algorithm
<M> ARC4 cipher algorithm
<> Blowfish cipher algorithm
<> Camellia cipher algorithms
<M> CAST5 (CAST-128) cipher algorithm
<> CAST6 (CAST-256) cipher algorithm
[*] DES and Triple DES EDE cipher algorithms
<> Crypt cipher algorithm
<> Hazad cipher algorithm
<> Salsa20 stream cipher algorithm
<> ChaCha20 cipher algorithm
<> EED cipher algorithm
<> Serpent cipher algorithm
<> TEA, XTEA and XETA cipher algorithms
<> Wolfish cipher algorithm
*** Compression ***
<M> Deflate compression algorithm
<M> LZ0 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> Litterentropy Non-Deterministic Random Number Generator
[*] User-space interface for hash algorithms
[*] 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 ----
[ ] Symmetric (public-key cryptographic) key type ----
[ ] Certificates for signature checking ----
[*] ARM Accelerated Cryptographic Algorithms ----

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

Once you're 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.

## 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/

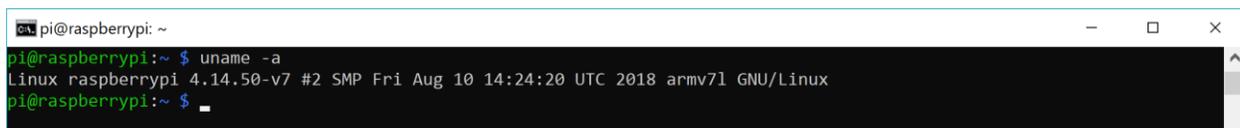
sudo cp arch/arm/boot/zImage /boot/$KERNEL.img

sudo reboot
```

This process takes a long time, maybe 2 ~ 3 hours.

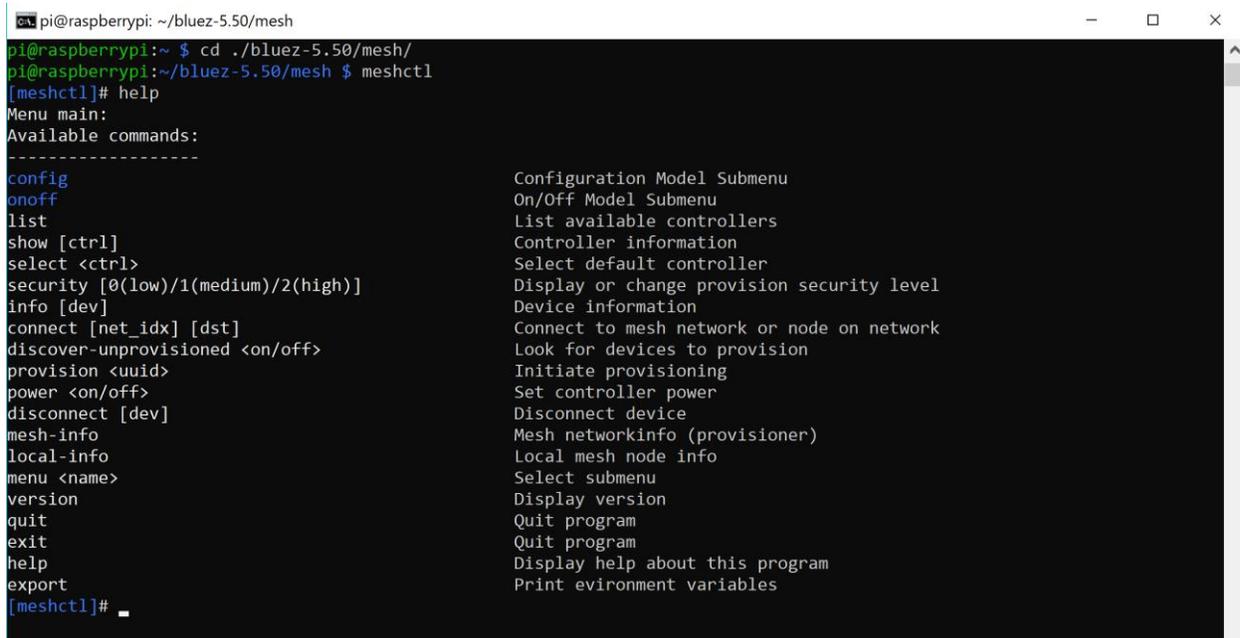
## Verifying kernel installation

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 Thu Mar 22 04:26:22 UTC 2018. That time and date was exactly when I built the kernel and it means the kernel building and installation was successful.



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ uname -a
Linux raspberrypi 4.14.50-v7 #2 SMP Fri Aug 10 14:24:20 UTC 2018 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]#
```

### 3. Summary

If you go through all the steps listed above, now, you've already have a Raspberry Pi3 board which can work as a provisioner to provision any dev kits/boards which support PB-GATT. In next guide, I will show you how to use *meshctl* to provision and configure a real Bluetooth mesh device.