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

Part 1 - Deployment

- **Revision:** 1.2
- **Revision Date:** Dec 19th, 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 release 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: ***raspberry***
  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

- 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/
```

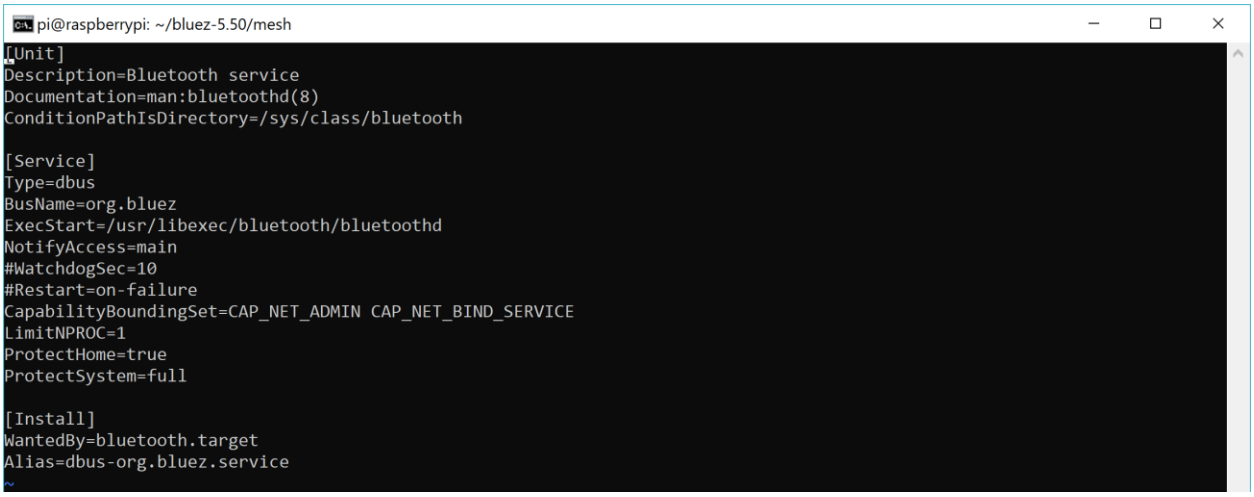
Compile & Install BlueZ

```
./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:

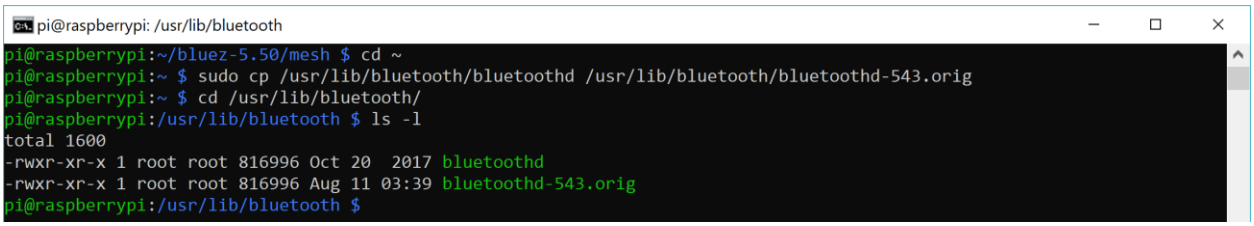
```
sudo vi /lib/systemd/system/bluetooth.service
```

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.



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
```



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]

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
```

[1] 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.

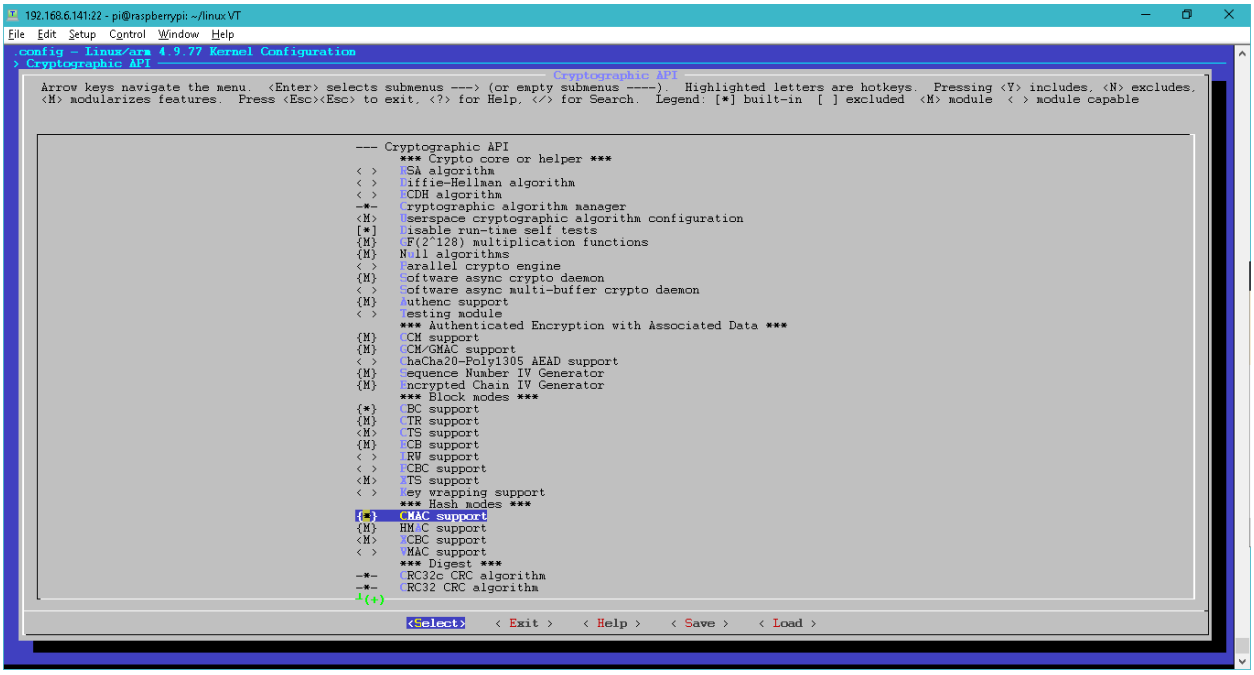
Use the arrow keys to navigate, the Enter key to enter a submenu (indicated by --->), Escape twice to go up a level or exit, and the space bar to cycle the state of an option. Some options have multiple choices, in which case they'll appear as a submenu and the Enter key will select an option. You can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

Please include the three modules below:

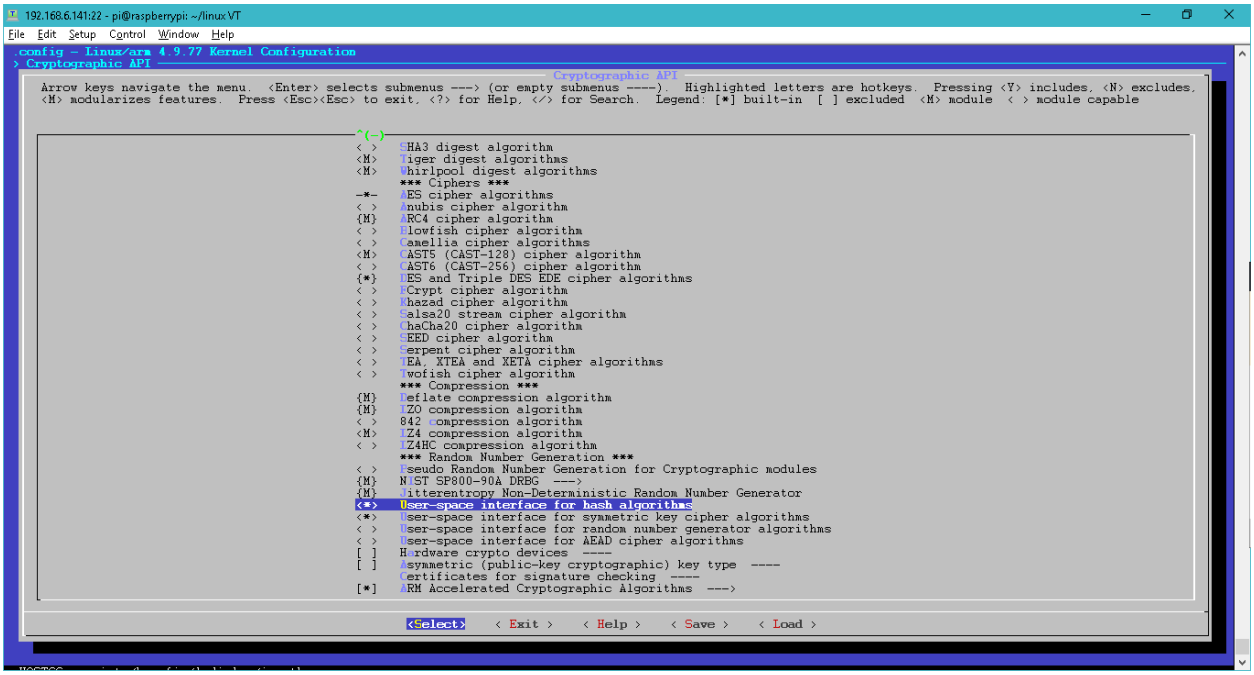
Select *Cryptographic API ---> CMAC support*

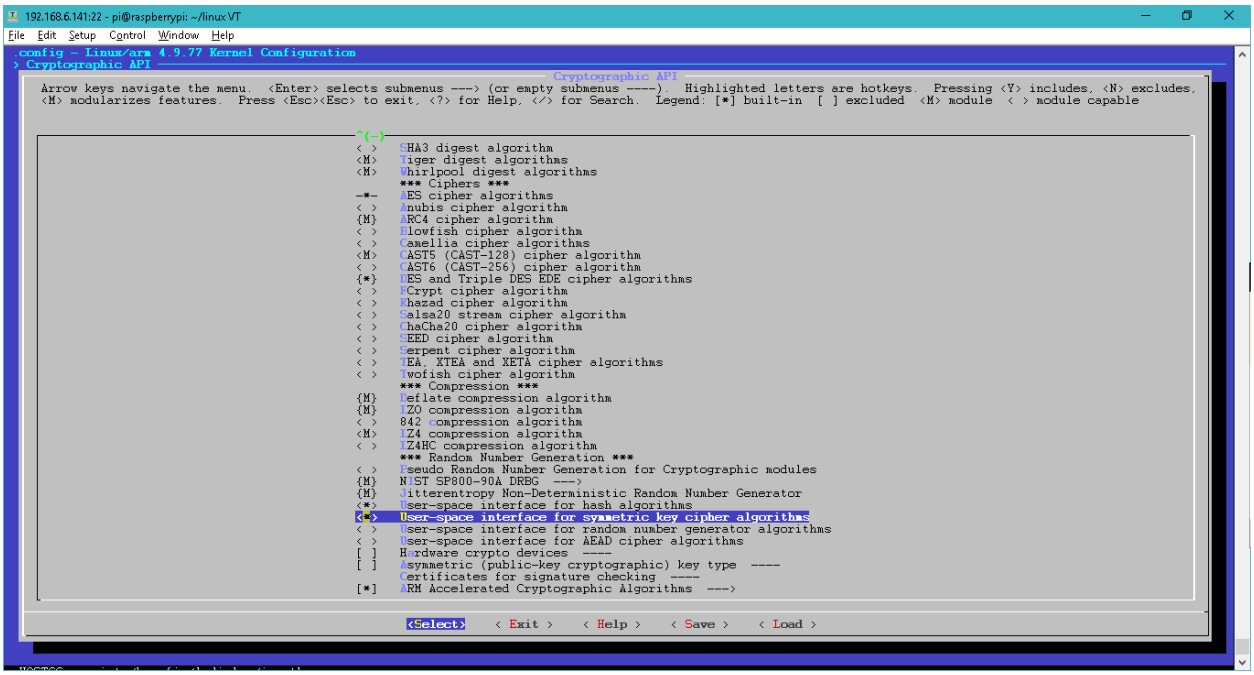
Select *Cryptographic API ---> User-space interface for hash algorithms*

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



[2] About this part, reference this article, https://www.raspberrypi.org/documentation/linux/kernel/configuring.md





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

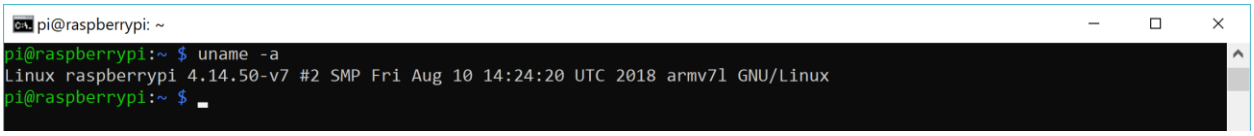
```
make -j4 zImage modules dtbs
```

```
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
```

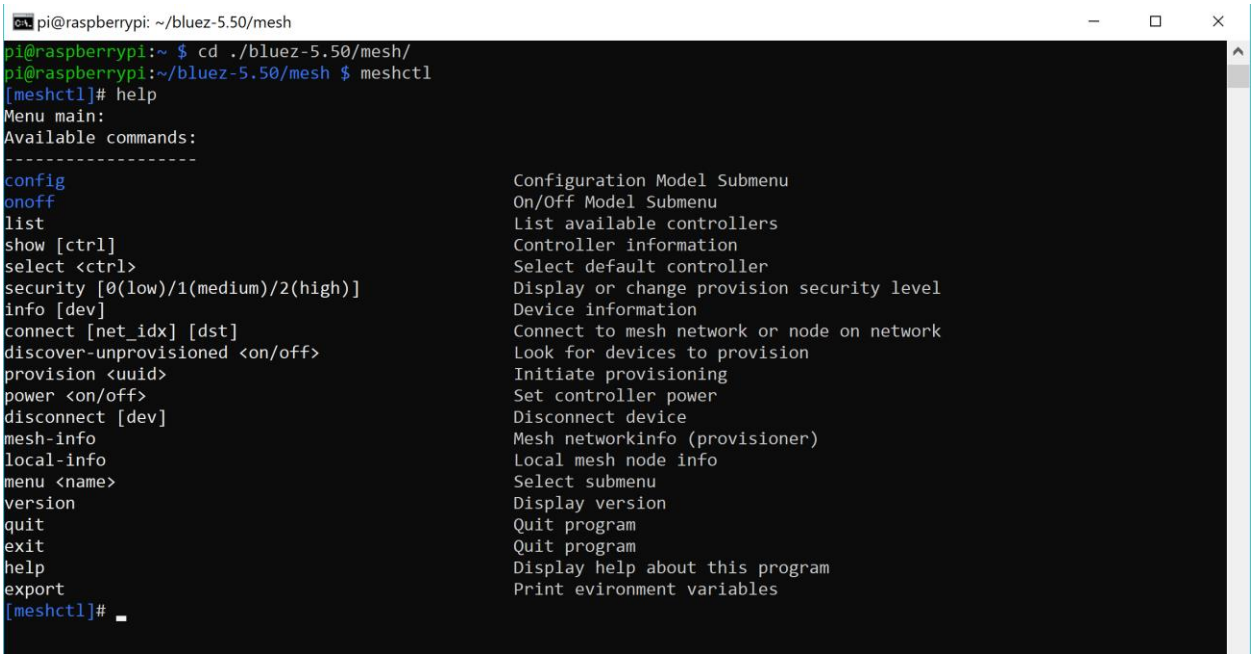
This process takes a long time, maybe 2 ~ 3 hours.

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.



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



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.