



# **Bluetooth for Linux Developers Study Guide**

## **Installation and Configuration**

Release : 1.0.1

Document Version: 1.0.0

Last updated : 16th November 2021

# Contents

|  |           |
|--|-----------|
| <b>1. REVISION HISTORY .....</b>                       | <b>3</b>  |
| <b>2. INTRODUCTION.....</b>                            | <b>4</b>  |
| <b>3. BLUEZ INSTALLATION.....</b>                      | <b>4</b>  |
| 3.1 Kernel Configuration .....                         | 4         |
| 3.2 Install BlueZ 5.58 or later and dependencies ..... | 5         |
| 3.3 Enable Bluetooth DBus Communication .....          | 6         |
| 3.4 Start the appropriate Bluetooth daemon .....       | 7         |
| 3.4.1 GAP/GATT support.....                            | 7         |
| 3.4.2 Bluetooth mesh support .....                     | 7         |
| 3.5 Debugging .....                                    | 7         |
| 3.5.1 -nd flag .....                                   | 7         |
| 3.5.2 btmon.....                                       | 9         |
| 3.5.3 btmgmt .....                                     | 10        |
| <b>4. PYTHON .....</b>                                 | <b>10</b> |

## 1. Revision History

| Version | Date               | Author                          | Changes  |
|---------|--------------------|---------------------------------|--|
| 1.0.0   | 16th November 2021 | Martin Woolley<br>Bluetooth SIG | <b>Release:</b><br>Initial release.<br><b>Document:</b><br>This document is new in this release. |

## 2. Introduction

Follow the instructions in this module to install and configure BlueZ for either GAP/GATT applications or for Bluetooth mesh. Bluetooth mesh uses cryptography functions such as AES-CMAC which must be supported by the Linux kernel.

A number of libraries must be installed and if your machine is to support Bluetooth mesh, the Linux Kernel must be compiled with the required cryptographic functions included.

Instructions for setting up Python3 for D-Bus development are also provided.

## 3. BlueZ Installation

### 3.1 Kernel Configuration

**Note:** If you are intending to develop Bluetooth mesh applications, the Linux kernel needs to be compiled with particular features enabled. A standard kernel as provided by a standard Raspbian image is suitable for using BlueZ with GAP/GATT applications. The steps in this section are therefore optional and only need to be carried out if developing for Bluetooth mesh.

1. Get your package lists up to date and then install dependencies.

```
sudo apt-get update

sudo apt-get install -y git bc libusb-dev libdbus-1-dev libglib2.0-dev libudev-dev libical-
dev libreadline-dev autoconf bison flex libssl-dev libncurses-dev glib2.0 libdbus-1-dev
```

2. Download the Linux kernel source and install its dependencies

```
cd ~

wget https://github.com/raspberrypi/linux/archive/raspberrypi-kernel_1.20210303-1.tar.gz

tar -xvf raspberrypi-kernel_1.20210303-1.tar.gz

cd ./linux-raspberrypi-kernel_1.20210303-1

sudo apt install bc bison flex libssl-dev make
```

3. Configure, build and the install the recompiled kernel

```
# If you're using a Pi Zero:
KERNEL=kernel
make bcmrpi_defconfig

# If you're using a Pi 4 or Pi 400:
KERNEL=kernel7l
make bcm2711_defconfig

make menuconfig

# In the menus that appear, select the following options

Cryptographic API-->
* CCM Support
* CMAC Support
* User-space interface for hash algorithms
```

```

* User-space interface for symmetric key cipher algorithms
* User-space interface for AEAD cipher algorithms

# Save to .config and exit menuconfig

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/

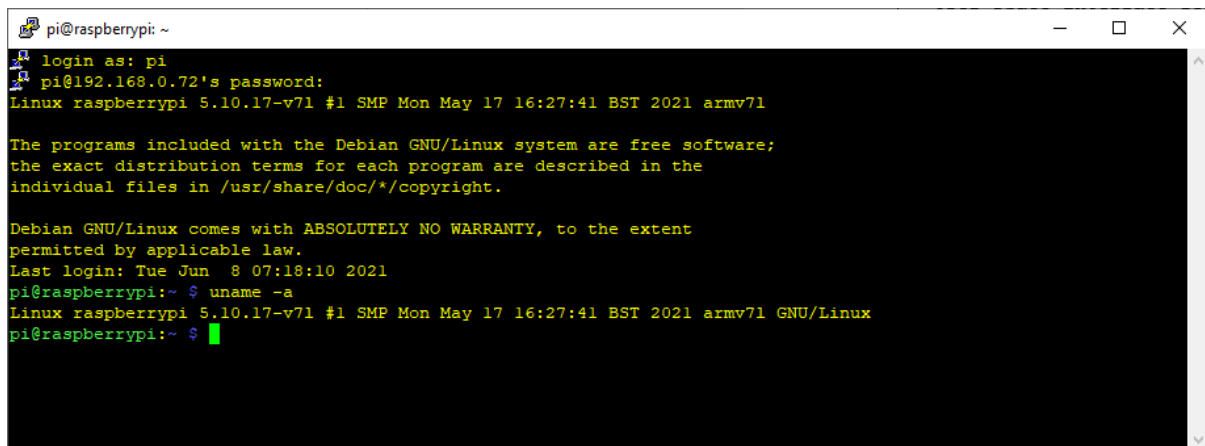
sudo cp arch/arm/boot/zImage /boot/$KERNEL.img

sudo reboot

```

#### 4. After rebooting, check the kernel version

```
uname -a
```



```

pi@raspberrypi: ~
login as: pi
pi@192.168.0.72's password:
Linux raspberrypi 5.10.17-v7l #1 SMP Mon May 17 16:27:41 BST 2021 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 Jun  8 07:18:10 2021
pi@raspberrypi:~$ uname -a
Linux raspberrypi 5.10.17-v7l #1 SMP Mon May 17 16:27:41 BST 2021 armv7l GNU/Linux
pi@raspberrypi:~$

```

### 3.2 Install BlueZ 5.58 or later and dependencies

Check the version of BlueZ running on your machine by launching the *bluetooth* tool and entering the *version* command.

```

pi@raspberrypi:~$ bluetoothctl
Agent registered
[bluetooth]# version
Version 5.50
[bluetooth]#

```

The version reported should be at least version 5.58. If an earlier version is reported, install dependencies and then download and build the required version of BlueZ. Note that this resource was created and tested using version 5.58. Later versions may be OK but this is not guaranteed.

Exit bluetoothctl by entering the *quit* command.

#### 1. Install libraries

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

## 2. Install JSON library

```
sudo apt install cmake

wget https://s3.amazonaws.com/json-c_releases/releases/json-c-0.15.tar.gz
tar xvf json-c-0.15.tar.gz

cd ~/json-c-0.15
mkdir build
cd build

cmake -DCMAKE_INSTALL_PREFIX=/usr -DCMAKE_BUILD_TYPE=Release -DBUILD_STATIC_LIBS=OFF ..
make

sudo make install
```

## 3. Install the Embedded Linux Library (ELL)

```
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
```

*You can probably ignore any warnings.*

## 4. Install BlueZ

```
cd ~

pi@raspberrypi:~ $ wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.58.tar.xz
URL transformed to HTTPS due to an HSTS policy
--2021-04-08 12:28:05-- https://www.kernel.org/pub/linux/bluetooth/bluez-5.58.tar.xz
Resolving www.kernel.org (www.kernel.org)... 136.144.49.103, 2604:1380:40b0:1a00::1
Connecting to www.kernel.org (www.kernel.org)|136.144.49.103|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://mirrors.edge.kernel.org/pub/linux/bluetooth/bluez-5.58.tar.xz [following]
--2021-04-08 12:28:06-- https://mirrors.edge.kernel.org/pub/linux/bluetooth/bluez-
5.58.tar.xz
Resolving mirrors.edge.kernel.org (mirrors.edge.kernel.org)... 147.75.101.1,
2604:1380:2001:3900::1
Connecting to mirrors.edge.kernel.org (mirrors.edge.kernel.org)|147.75.101.1|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 2060368 (2.0M) [application/x-xz]
Saving to: 'bluez-5.58.tar.xz'

bluez-5.58.tar.xz
100%[=====] 1.96M 6.12MB/s in 0.3s

2021-04-08 12:28:06 (6.12 MB/s) - 'bluez-5.58.tar.xz' saved [2060368/2060368]

tar -xvf bluez-5.58.tar.xz
cd bluez-5.58/
./configure --enable-mesh --enable-testing --enable-tools --prefix=/usr --
mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var
sudo make
sudo make install
```

### 3.3 Enable Bluetooth DBus Communication

The Bluetooth API uses the Linux DBus inter-process communication system. A default security policy restricts use of DBus for Bluetooth purposes to processes owned by users that are a member

of the *bluetooth* system group. You can see this in the `/etc/dbus-1/system.d/bluetooth.conf` file, here:

```
<!-- allow users of bluetooth group to communicate -->
<policy group="bluetooth">
  <allow send_destination="org.bluez"/>
</policy>
```

Edit `/etc/group` and add the *www-data* and *pi* users to the bluetooth group.

```
pi@raspberrypi:~ $ sudo vi /etc/group
pi@raspberrypi:~ $ cat /etc/group|grep bluet
bluetooth:x:112:www-data,pi
# you need to start a new bash shell for the change to be activated - do this by entering
# su - pi
# or exit this shell and start a new one

pi@raspberrypi:~ $ groups
pi adm dialout cdrom sudo audio www-data video plugdev games users input netdev bluetooth
gpio i2c spi
```

Now restart the DBus daemon:

```
pi@raspberrypi:~ $ sudo service dbus restart
```

## 3.4 Start the appropriate Bluetooth daemon

### 3.4.1 GAP/GATT support

For general GAP/GATT support, your device must run the *bluetooth* daemon. It can be started and stopped and its status checked using the *service* command.

```
sudo service bluetooth start
sudo service bluetooth status
sudo service bluetooth stop
```

The bluetooth-mesh daemon must not be running at the same time as the bluetooth daemon.

### 3.4.2 Bluetooth mesh support

For Bluetooth mesh support, your device must run the bluetooth-mesh daemon. It can be started and stopped and its status checked using the *service* command.

```
sudo service bluetooth-mesh start
sudo service bluetooth-mesh status
sudo service bluetooth-mesh stop
```

The bluetooth daemon must not be running at the same time as the bluetooth-mesh daemon.

## 3.5 Debugging

### 3.5.1 -nd flag

If you run into problems, it's possible to run the bluetooth and bluetooth-mesh daemons in debug mode (but not both at the same time). This is done by editing the associated Linux service file and adding the `-nd` flag to the end of the `ExecStart` line. Here's an example for the bluetooth daemon.

Note that the service file for the Bluetooth mesh daemon is in the same location and named `bluetooth-mesh.service`.

```
pi@raspberrypi:~/projects/ldsg/solutions/python/bluetooth $ sudo cat
/lib/systemd/system/bluetooth.service
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/libexec/bluetooth/bluetoothd -nd
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
```

After adding the `-nd` flag you must run:

```
sudo systemctl daemon-reload
```

Debug output will be written to `syslog` which can be monitored with `tail`:

```
tail -f /var/log/syslog
```

```
Dec 14 08:05:29 raspberrypi bluetoothd[820]: bluetoothd[820]: src/agent.c:agent_ref()
0x6dc0f8: ref=1
Dec 14 08:05:29 raspberrypi bluetoothd[820]: bluetoothd[820]: src/agent.c:register_agent()
agent :1.26
Dec 14 08:05:29 raspberrypi bluetoothd[820]: src/agent.c:agent_ref() 0x6dc0f8: ref=1
Dec 14 08:05:29 raspberrypi bluetoothd[820]: src/agent.c:register_agent() agent :1.26
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:start_discovery() sender :1.26
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:update_discovery_filter()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:start_discovery() sender :1.26
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:discovery_filter_to_mgmt_cp()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:trigger_start_discovery()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:cancel_passive_scanning()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:start_discovery_timeout()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:start_discovery_timeout() adapter->current_discovery_filter == 0
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:start_discovery_complete() status 0x00
Dec 14 08:05:34 raspberrypi bluetoothd[820]: bluetoothd[820]:
src/adapter.c:discovering_callback() hci0 type 7 discovering 1 method 0
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:update_discovery_filter()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:discovery_filter_to_mgmt_cp()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:trigger_start_discovery()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:cancel_passive_scanning()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:start_discovery_timeout()
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:start_discovery_timeout()
adapter->current_discovery_filter == 0
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:start_discovery_complete()
status 0x00
Dec 14 08:05:34 raspberrypi bluetoothd[820]: src/adapter.c:discovering_callback() hci0 type
7 discovering 1 method 0
```



### 3.5.2 btmon

You can also sometimes get clues using the *btmon* tool:

```
pi@raspberrypi:~ $ sudo btmon
Bluetooth monitor ver 5.58
= Note: Linux version 5.4.79-v7l+ (armv7l)
0.097785
= Note: Bluetooth subsystem version
2.22
0.097793
= New Index: DC:A6:32:0E:83:E2
(Primary,UART,hci0)
[hci0] 0.097796
= Open Index: DC:A6:32:0E:83:E2
[hci0] 0.097801
= Index Info: DC:A6:32:0E:83:E2 (Cypress Semiconductor)
[hci0] 0.097804
@ MGMT Open: bluetoothd (privileged) version 1.14
{0x0001} 0.097808
@ MGMT Command: Add Advertising (0x003e) plen 11
{0x0001} [hci0] 6.161926
    Instance: 1
    Flags: 0x00000000
    Duration: 0
    Timeout: 0
    Advertising data length: 0
    Scan response length: 0
< HCI Command: LE Set Advertising Data (0x08|0x0008) plen 32
#1 [hci0] 6.161979
    Length: 0
> HCI Event: Command Complete (0x0e) plen 4
#2 [hci0] 6.162365
    LE Set Advertising Data (0x08|0x0008) ncmd 1
    Status: Success (0x00)
< HCI Command: LE Set Scan Response Data (0x08|0x0009) plen 32
#3 [hci0] 6.162394
    Length: 0
> HCI Event: Command Complete (0x0e) plen 4
#4 [hci0] 6.162927
    LE Set Scan Response Data (0x08|0x0009) ncmd 1
    Status: Success (0x00)
< HCI Command: LE Set Random Address (0x08|0x0005) plen 6
#5 [hci0] 6.162953
    Address: 0C:2B:9E:27:57:51 (Non-Resolvable)
> HCI Event: Command Complete (0x0e) plen 4
#6 [hci0] 6.163259
    LE Set Random Address (0x08|0x0005) ncmd 1
    Status: Success (0x00)
< HCI Command: LE Set Advertising Parameters (0x08|0x0006) plen 15
#7 [hci0] 6.163287
    Min advertising interval: 62.500 msec (0x0064)
    Max advertising interval: 93.750 msec (0x0096)
    Type: Non connectable undirected - ADV_NONCONN_IND (0x03)
    Own address type: Random (0x01)
    Direct address type: Public (0x00)
    Direct address: 00:00:00:00:00:00 (OUI 00-00-00)
    Channel map: 37, 38, 39 (0x07)
    Filter policy: Allow Scan Request from Any, Allow Connect Request from Any (0x00)
> HCI Event: Command Complete (0x0e) plen 4
#8 [hci0] 6.163624
    LE Set Advertising Parameters (0x08|0x0006) ncmd 1
    Status: Invalid HCI Command Parameters (0x12)
@ MGMT Event: Command Status (0x0002) plen 3
{0x0001} [hci0] 6.163654
    Add Advertising (0x003e)
    Status: Invalid Parameters (0x0d)
= bluetoothd: src/advertising.c:add_client_complete() Failed to add advertisement: Invalid
Parameters (0x0d)
```

### 3.5.3 btmgmt

The btmgmt tool is another useful source of information:

```
pi@raspberrypi:~ $ sudo btmgmt
[mgmt]# info
Index list with 1 item
hci0: Primary controller
      addr DC:A6:32:0E:83:E2 version 9 manufacturer 305 class 0x000000
      supported settings: powered connectable fast-connectable discoverable bondable
link-security ssp br/edr le advertising secure-conn debug-keys privacy static-addr
      current settings: powered bondable ssp br/edr le secure-conn
      name raspberrypi
      short name
```

## 4. Python

All exercises and examples in this study guide use Python3. Ensure your machine has python3 installed like this:

```
pi@raspberrypi:~/projects/dbus/bluetooth_samples/c $ python3 -V
Python 3.7.3
```

If you don't already have python3 installed, install it as follows:

```
sudo apt update
sudo apt install python3
```

You will also need an *mainloop* implementation installed so that asynchronous messaging is supported in your applications. To that end, install a GLib library as follows:

```
pi@raspberrypi:~/projects/ldsg/solutions/python $ sudo apt-get install python3-gi
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  gir1.2-glib-2.0
The following NEW packages will be installed:
  gir1.2-glib-2.0 python3-gi
0 upgraded, 2 newly installed, 0 to remove and 228 not upgraded.
Need to get 0 B/304 kB of archives.
After this operation, 1,546 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Selecting previously unselected package gir1.2-glib-2.0:armhf.
(Reading database ... 96669 files and directories currently installed.)
Preparing to unpack .../gir1.2-glib-2.0_1.58.3-2_armhf.deb ...
Unpacking gir1.2-glib-2.0:armhf (1.58.3-2) ...
Selecting previously unselected package python3-gi.
Preparing to unpack .../python3-gi_3.30.4-1_armhf.deb ...
Unpacking python3-gi (3.30.4-1) ...
Setting up gir1.2-glib-2.0:armhf (1.58.3-2) ...
Setting up python3-gi (3.30.4-1) ...
```

Create a directory in which to create your own scripts. Copy into this folder the Python files packaged in the code/solutions directory of the study guide and whose names start with *bluetooth\_*. Your project directory should look like this:

```
pi@raspberrypi:~/projects/ldsg/solutions/python/bluetooth $ ls -l bluetooth_*
-rwxrwx--- 1 pi 2887 Nov  4 14:05 bluetooth_constants.py
-rwxrwx--- 1 pi  778 Nov  9 10:54 bluetooth_exceptions.py
-rwxrwx--- 1 pi 6930 Nov  4 14:26 bluetooth_gatt.py
```

```
-rwxrwx--- 1 pi 2072 Nov  4 10:32 bluetooth_utils.py
```