Bluetooth overview

Stable: 11.02.2019 - 13:20 / Revision: 29.01.2019 - 12:35

Summary
This article explains how a Bluetooth framework is composed, how to configure it, and how to use it.

Contents

1 Framework purpose 1
2 System overview 2
2.1 Component descriptions 2
2.2 APIs description 3
3 Configuration 3
3.1 Kernel configuration 3
3.2 Device tree 4
4 How to use Bluetooth 4
4.1 How to use the Bluetooth user space interface 4
5 How to trace and debug the framework 4
5.1 How to verify than Bluetooth driver is correctly probed 4
6 References 5

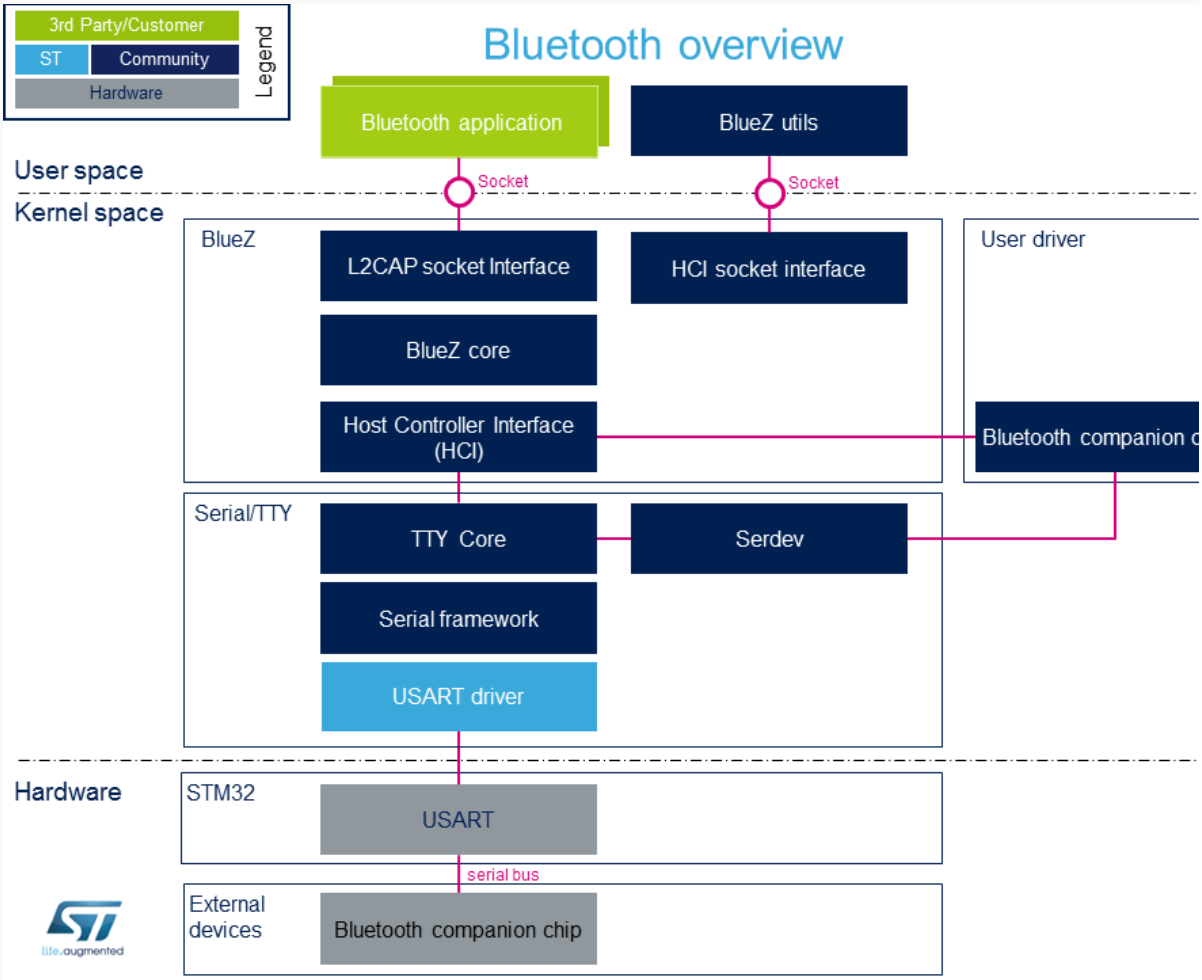
1 Framework purpose

Bluetooth is a protocol for wireless communication over short distances. It was developed in the 1990s to reduce the need for cable interconnects. Devices such as mobile phones, laptops, PCs, printers, digital cameras and video game consoles can connect to each other and exchange information using radio waves. This can be done securely. Bluetooth is only used for relatively short distances, typically of a few meters.
The Linux kernel has a popular Bluetooth stack: BlueZ. This stack is included in most Linux kernels, and runs in both the user space and kernel space of the Bluetooth protocol.
Bluetooth Low Energy is completely supported at the kernel level in Linux.

Bluetooth can be used in many different use cases, as mentioned in the How to use Bluetooth section:

- how to set up a Bluetooth connection Setup Bluetooth
- how to scan Bluetooth devices Scan Bluetooth devices
- how to scan BLE devices Scan BLE devices

2 System overview



Bluetooth Overview

2.1 Component descriptions

From User space to hardware

- **Bluetooth Applications** (User space)

Lots of applications use bluetooth:
bluetoothd [1]: bluetoothd daemon, which manages all the Bluetooth devices
...

- **BlueZ Utils** (User space)

Development and debugging utilities for the bluetooth protocol stack
There is a set of utilities to manage Bluetooth devices:
bluetoothctl [2]: Pairing a device from the shell is one of the simplest and most reliable options
To see all other utilities: https://www.archlinux.org/packages/extra/x86_64/bluez-utils/

- **BlueZ** (Kernel space)

BlueZ [3] is the official Linux Bluetooth stack. It provides, in a modular way, support for the core Bluetooth layers and protocols.
Currently BlueZ consists of many separate modules:
- bluetooth kernel subsystem core
- a "controller stack" containing the timing critical radio interface like HCI [4]
- a "host stack" dealing with high level data like L2CAP [5]

- **Bluetooth companion driver** (Kernel space)

Bluetooth companion driver registers and controls the Bluetooth device

- **Serial/TTY** (Kernel space)

See Serial TTY overview

- **SoC: USART** (Hardware)

See Serial TTY overview

2.2 APIs description

The BlueZ API [6] is documented in the Linux Kernel:
BlueZ exposes a socket API that is similar to network socket programming; the is socket created, used to communicate, PF_BLUETOOTH protocol family [7]

3 Configuration

3.1 Kernel configuration

Bluetooth must be enabled in the kernel configuration, as shown below. On top of this, the user has to activate STM32 support and STM32 USART support. The user can use the Linux Menuconfig tool: Menuconfig or how to configure kernel and select:

```
[*] Networking support  --->
    [*]   Bluetooth subsystem support
        [*]   Bluetooth Classic (BR/EDR) features
        [*]   Bluetooth High Speed (HS) features
        [*]   Bluetooth Low Energy (LE) features
        [*]   Bluetooth device drivers   --->
                [*]   HCI UART driver
```

```
[*] Device Drivers  --->
    [*]   Character devices  --->
        [*]   Serial device bus
[*] Security options  --->
    [*]   Enable access key retention support
```

For example if the companion chip is the Murata product 1DX[8]

```
[*] Networking support  --->
    [*]   Bluetooth subsystem support  --->
        [*]   Bluetooth device drivers   --->
                [*]   Broadcom protocol support
```

3.2 Device tree

DT bindings documentation deals with all of the required and optional device tree properties.

Detailed DT configuration for STM32 internal peripherals: Bluetooth device tree configuration.

4 How to use Bluetooth

4.1 How to use the Bluetooth user space interface

Please see the examples based on the following use cases:

- how to set up a Bluetooth connection Setup Bluetooth
- how to scan Bluetooth devices Scan Bluetooth devices
- how to scan BLE devices Scan BLE devices

5 How to trace and debug the framework

This part is an example based on the Murata companion chip

5.1 How to verify than Bluetooth driver is correctly probed

- In dmesg log, check "usart" logs :

```
[    0.485894] STM32 USART driver initialized
[    0.487163] 4000e000.serial: ttySTM1 at MMIO 0x4000e000 (irq = 21, base_baud = 4000000)
[    0.487514] stm32-usart 4000e000.serial: interrupt mode used for rx (no dma)
[    0.487531] stm32-usart 4000e000.serial: interrupt mode used for tx (no dma)
```

And, if the companion chip is the Murata 1DX :

```
[   13.755069] Bluetooth: HCI device and connection manager initialized
[   13.800349] Bluetooth: HCI socket layer initialized
[   13.837861] Bluetooth: L2CAP socket layer initialized
[   13.843218] Bluetooth: SCO socket layer initialized
[   14.279668] Bluetooth: HCI UART driver ver 2.3
[   14.282780] Bluetooth: HCI UART protocol H4 registered
[   14.288198] Bluetooth: HCI UART protocol Broadcom registered
```

```
[   14.289402] hci_uart_bcm serial0-0: No reset resource, using default baud rate
[   14.465008] Bluetooth: hci0: BCM: chip id 94
[   14.469843] Bluetooth: hci0: BCM: features 0x2e
[   14.497113] Bluetooth: hci0: BCM43430A1
[   14.499593] Bluetooth: hci0: BCM43430A1 (001.002.009) build 0000
```

6 References

1. ↑ [1], bluetoothd
2. ↑ [2], bluetoothctl
3. ↑ [3], BlueZ
4. ↑ [4], HCI
5. ↑ [5], L2CAP
6. ↑ [6], BlueZ API
7. ↑ [7], Socket
8. ↑ [8], 1DX

Bluetooth Low Energy .

TeleTYpewriter

Universal Synchronous/Asynchronous Receiver/Transmitter

Application programming interface

High Speed (MIPI® Alliance DSI standard)

Universal Asynchronous Receiver/Transmitter

Device Tree