Learn Bluetooth Low Energy (BLE) on Raspberry Pi with Nuimo

By
Aravinth Panchadcharam
Maker | Embedded System Engineer
Senic GmbH

TechJAM Berlin 2017

Introduction

- Maker by Passion
- Masters in Electrical Engineering (Electronics, Robotics, Wireless Communication Technologies)
- 10 years of working experience with corporates, startups & research institutions
- Founding member of Sustainability Drinks Berlin
- Currently working as a lead embedded systems engineer at Senic GmbH (Nuimo)

www.aravinth.info
https://github.com/aravinthpanch

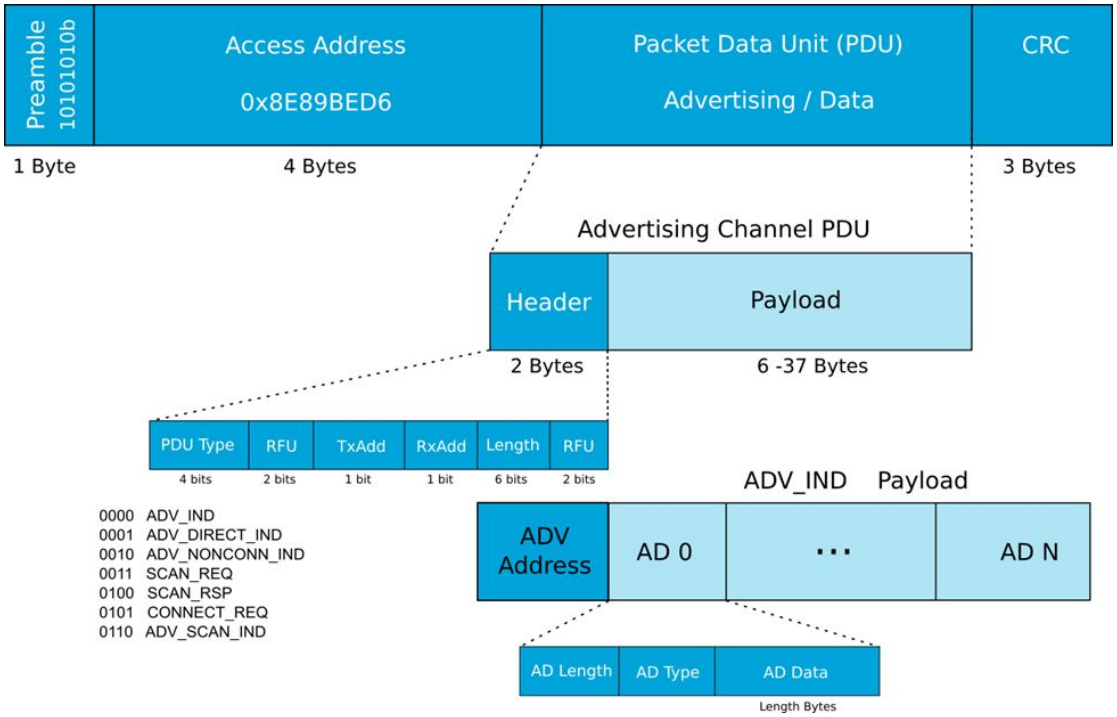
@AravinthPanch

Agenda

- Introduction to Bluetooth Low Energy (BLE)
- BLE on Raspberry Pi
- Senic Python SDKs for BLE
- Demo
- Bluetooth Cheatsheet

Introduction to Bluetooth Low Energy (BLE)

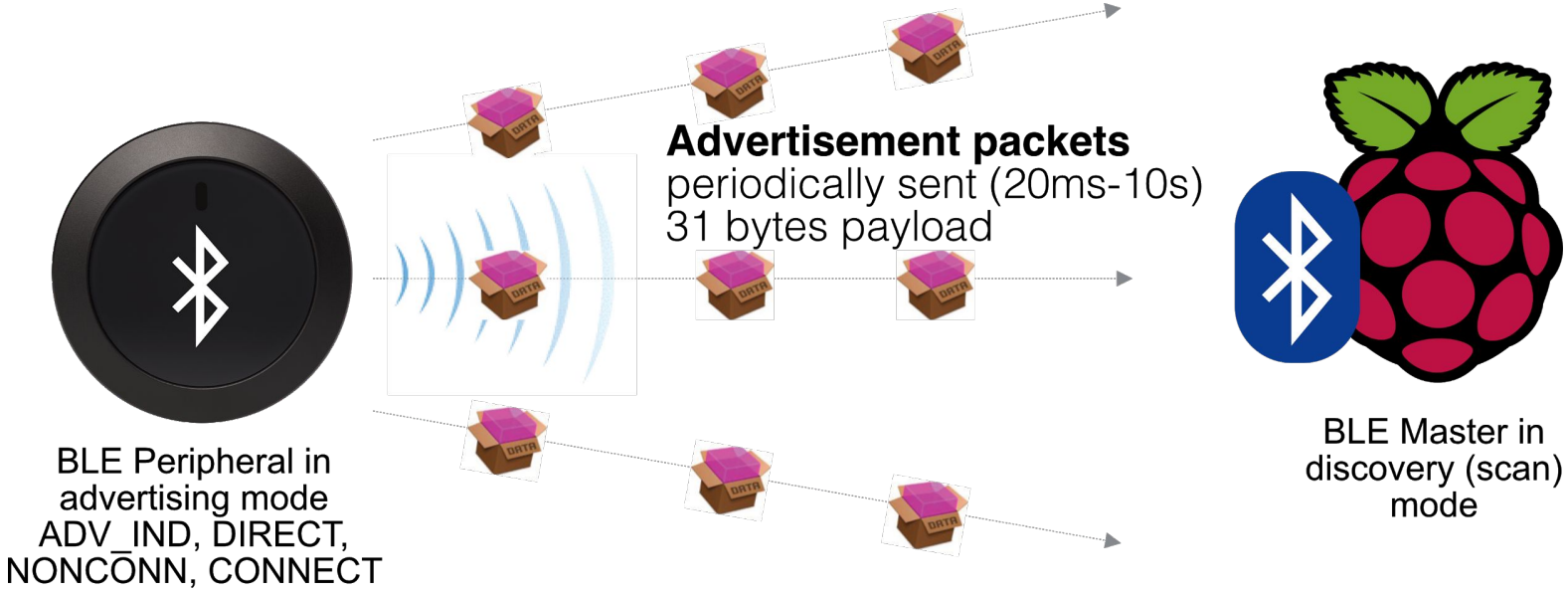
BLE Protocol



Source : argenox.com

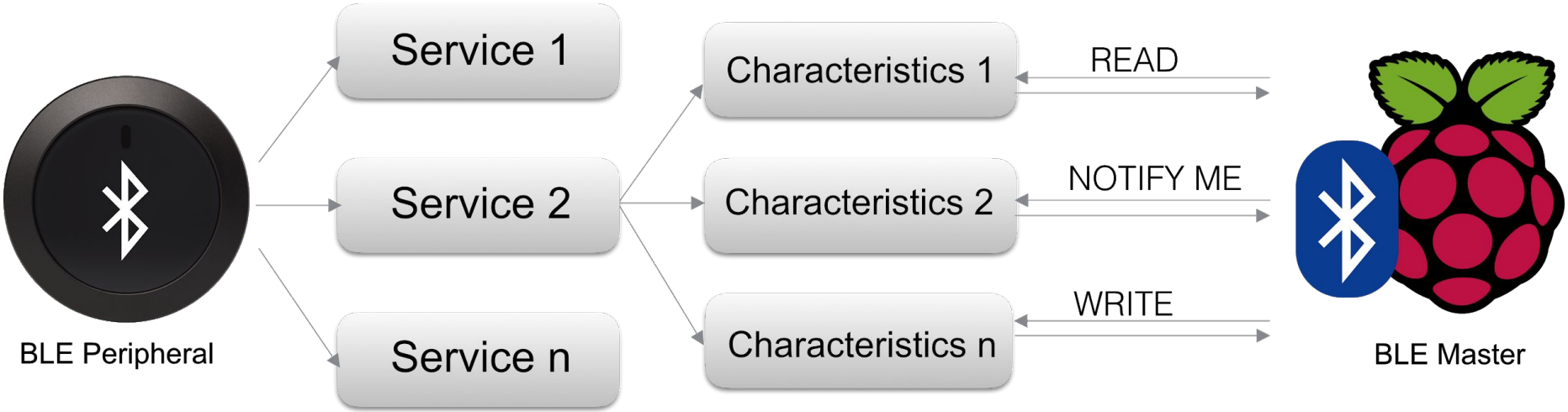
Introduction to Bluetooth Low Energy (BLE)

BLE Advertising and discovery



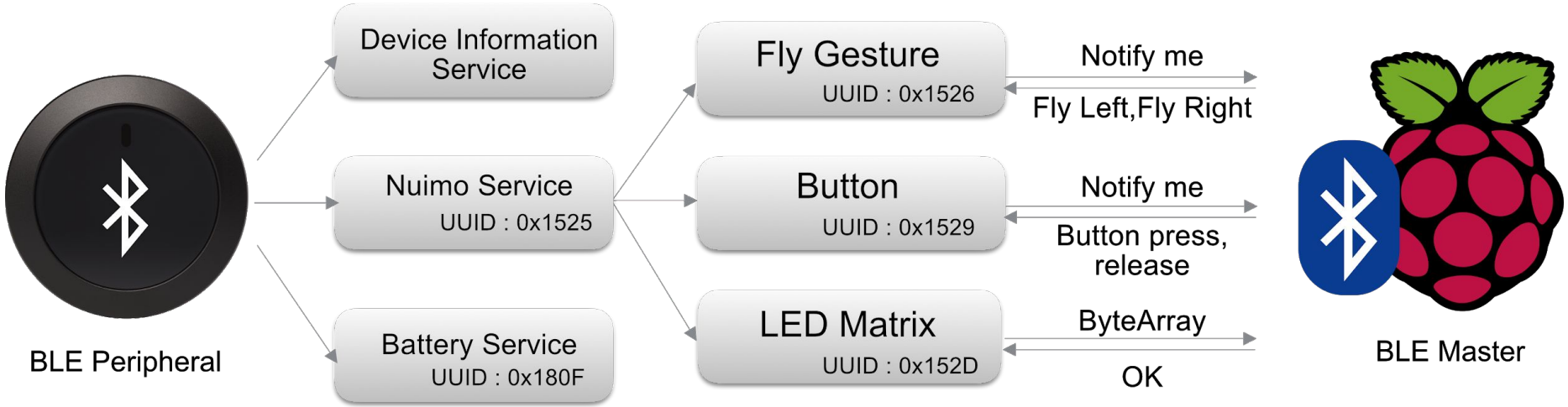
Introduction to Bluetooth Low Energy (BLE)

GATT => **G**eneric **ATT**ribute Profile => API of BLE Device



Introduction to Bluetooth Low Energy (BLE)

GATT Commands => Read, Write, Notify



BLE on Raspberry Pi

- BlueZ is the official Linux Bluetooth protocol stack
  - sudo apt-get install --no-install-recommends bluetooth
- hciconfig
  - Tool to configure Bluetooth Adapters
- bluetoothctl
  - Tool for discovery, connect,disconnect, scan, pair of Bluetooth devices
- hcitool
  - Tool to scan for Bluetooth devices
- gatttool
  - Tool to interact with GATT Server (BLE Peripheral)

BLE on Raspberry Pi - Discovering/Scanning

$ sudo hcitool lescan

LE Scan ...

6C:40:08:9F:31:35 ChromecastAudio3967

6C:40:08:9F:31:35 (unknown)

6C:40:08:9F:31:35 (unknown)

EA:69:48:31:64:61 Nuimo

MAC Address of the device

BLE on Raspberry Pi - Connecting, Reading, Writing

$ sudo gatttool -b EA:69:48:31:64:61 -t random -I

[EA:69:48:31:64:61][LE]> **connect**

Attempting to connect to EA:69:48:31:64:61

Connection successful

[EA:69:48:31:64:61][LE]> **characteristics**

handle: 0x0019, char properties: 0x0e, char value handle: 0x001a, uuid: f29b1526-cb19-40f3-be5c-7241ecb82fd1

Custom UUID of FLy Gesture of Nuimo Service

Control Nuimo with Senic Python SDKs

- Senic Nuimo Python SDK https://github.com/getsenic/nuimo-linux-python
  - sudo pip3 install nuimo
- Senic Gatt Python D-Bus https://github.com/getsenic/gatt-python

Dependencies

- BlueZ 5.44+
- D-Bus
  - A software bus that provides an inter-process communication (IPC) and remote procedure call (RPC) mechanism to allow communication between multiple processes.
- Python D-Bus
- Python 3.4 +

Source : WikiPedia

Control Nuimo with Senic Python SDKs - nuimoctl

$ sudo nuimoctl --connect EA:69:48:31:64:61

Nuimo controller EA:69:48:31:64:61 connecting...

Terminate with Ctrl+C

Nuimo controller EA:69:48:31:64:61 connected

Nuimo controller EA:69:48:31:64:61 did send gesture event Gesture.BUTTON_PRESS

Nuimo controller EA:69:48:31:64:61 did send gesture event Gesture.BUTTON_RELEASE

Nuimo controller EA:69:48:31:64:61 did send gesture event Gesture.SWIPE_LEFT

Nuimo controller EA:69:48:31:64:61 did send gesture event Gesture.SWIPE_RIGHT

Nuimo controller EA:69:48:31:64:61 did send gesture event Gesture.ROTATION,59

Control Nuimo with Senic Python SDKs - nuimo python package

```
import nuimo

manager = nuimo.ControllerManager(adapter_name='hci0')

controller = nuimo.Controller(mac_address='AA:BB:CC:DD:EE:FF', manager=manager)
controller.listener = nuimo.ControllerListener()
controller.connect()

manager.run()
```

Control any BLE device with Senic Python SDKs - gatt python package

```
import gatt

class AnyDeviceManager(gatt.DeviceManager):
    def device_discovered(self, device):
        print("Discovered [%s] %s" % (device.mac_address, device.alias()))

manager = AnyDeviceManager(adapter_name='hci0')
manager.start_discovery()
manager.run()
```

Demo

Checkout Bluetooth Cheatsheet at the end of the slides

Source : WikiMedia

aravinth@senic.com
www.aravinth.info

www.senic.com
Nuimo
github.com/getsenic

TechJAM Berlin
1st April 2017

TU Berlin

Bluetooth Cheatsheet

Install bluetooth kernel BlueZ library

sudo apt-get install --no-install-recommends bluetooth

#-> this installs bluez library (older version) and installs bluetoothctl and bluetoothd

bluetoothctl is useful to manage Bluetooth HCI or peripherals

bluetoothd is a daemon that should run from startup to use any other bluetooth tools

For c,c++ bluetooth development to install header files

sudo apt-get install libbluetooth-dev

Config files of BlueZ

/etc/bluetooth

Change bluetooth controller to allow only BLE mode

/etc/bluetooth/main.conf

ControllerMode = le

btmgmt # BlueZ command to directly manipulate bluetooth conf file

Bluetooth Cheatsheet

Display the available bluetooth device on the system

hciconfig dev

Enable Bluetooth

sudo hciconfig hci0 up

Disable Bluetooth

sudo hciconfig hci0 down

Scan for classic Bluetooth Devices

sudo hcitool scan

Scan for BLE Devices

sudo hcitool lescan

Reset bluetooth controller

hciconfig hci0 reset

Bluetooth Cheatsheet

Connect to the bluetooth in interactive mode

sudo gatttool -b FA:48:12:00:CA:AC -t random -I

to connect to the peripheral

connect

to list all the characteristics

characteristics

handle: 0x002d, char properties: 0x08, char value handle: 0x002e, uuid: f29b152c-cb19-40f3-be5c-7241ecb82fd2

char-write-req 0x002b 01

Write to a characteristics without interactive mode and listen to notifications

gatttool -b F9:8D:BB:F1:15:34 -t random --char-write-req -a 0x002c -n 01 --listen

To list the previously paired peripherals of the adapter

bluetoothctl devices

To find version of bluez

bluetoothd -v

Bluetooth Cheatsheet

Restart bluetooth service

service bluetooth restart

View information of BLE peripheral such as Nuimo

hcitool leinfo --random F9:8D:BB:F1:15:34

Create a connection to LE peripheral

hcitool lecc --random F9:8D:BB:F1:15:34

Read RSSI of BLE

btmon -w rssi.btsnoop # run this is background

hcitool lescan # run this for the need time of capturing packets and kill

btmon -r rssi.btsnoop > rssi.log # convert btsnoop to plain text

cat rssi.log | grep F9:8D:BB:F1:15:34 -A 10 # -A in grep shows 10 lines after the match

cat rssi.log | grep F9:8D:BB:F1:15:34 -A 10 | grep RSSI # shows all rssi of F9:8D:BB:F1:15:34 in the capture

Bluetooth Cheatsheet

BlueZ utils

is used to issue BlueCore commands to Cambridge Silicon Radio devices.

bccmd

is a Bluemoon configuration utility.

bluemoon

is the interactive Bluetooth control program.

bluetoothctl

is the Bluetooth daemon.

bluetoothd

provides access to the Bluetooth subsystem monitor infrastructure for reading HCI traces.

btmon

Bluetooth Cheatsheet

is used to set up, maintain, and inspect the CIP configuration of the Bluetooth subsystem in the Linux kernel.

ciptool

is used to attach a serial UART to the Bluetooth stack as HCI transport interface.

hciattach

is used to configure Bluetooth devices.

hciconfig

reads raw HCI data coming from and going to a Bluetooth device and prints to screen commands, events and data in a human-readable form.

hcidump

is used to configure Bluetooth connections and send some special command to Bluetooth devices.

http://manpages.ubuntu.com/manpages/zesty/man1/hcitool.1.html

hcitool

Bluetooth Cheatsheet

is used to convert a file needed by Broadcom devices to hcd (Broadcom bluetooth firmware) format.

hex2hcd

is used to send a L2CAP echo request to the Bluetooth MAC address given in dotted hex notation.

l2ping

is L2CAP testing program.

l2test

is used to test RFCOMM communications on the Bluetooth stack.

rctest

is used to set up, maintain, and inspect the RFCOMM configuration of the Bluetooth subsystem in the Linux kernel.

rfcomm

is used to perform SDP queries on Bluetooth devices.

sdptool