THE BEST OF THE BEST EMBEDDED SYSTEM DEVELOPERS
TOPGUN
10TH ANNIVERSARY
TESA RALLY 2016

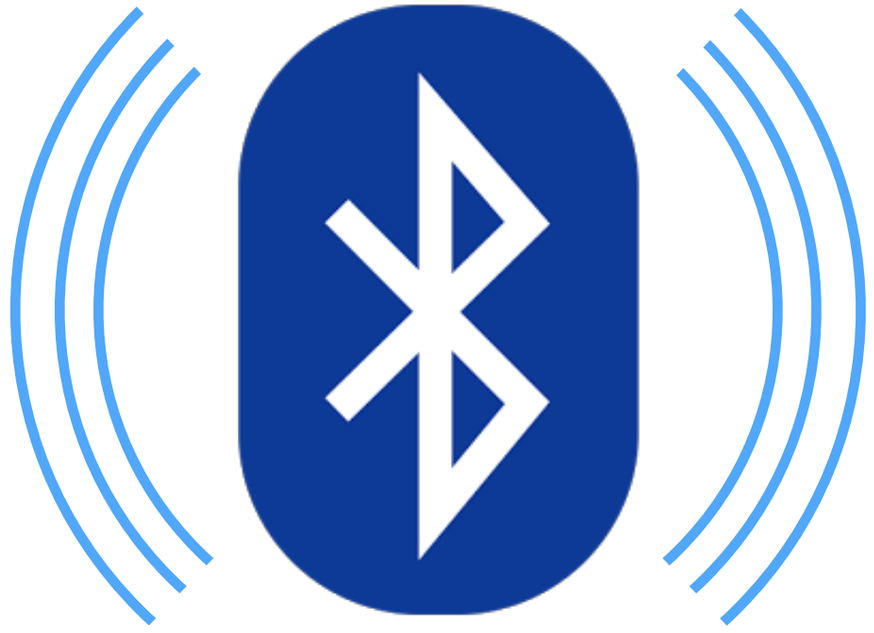
DEVELOPMENT TECHNIQUES FOR SMART CLASS ROOM

นายปริวัฒน์ เลื่อมสำราญ - คณะวิศวกรรมศาสตร์ ม.บูรพา

ภานุวัฒน์ พรหมศิริ, ปริวัฒน์ เลื่อมสำราญ, วสันต์ วยิะรันดร์,
กิตติศักดิ์ พรหมศรี, พีรเดช ลออธรรม - วิศวกร บริษัท เบสแล็บ จำกัด

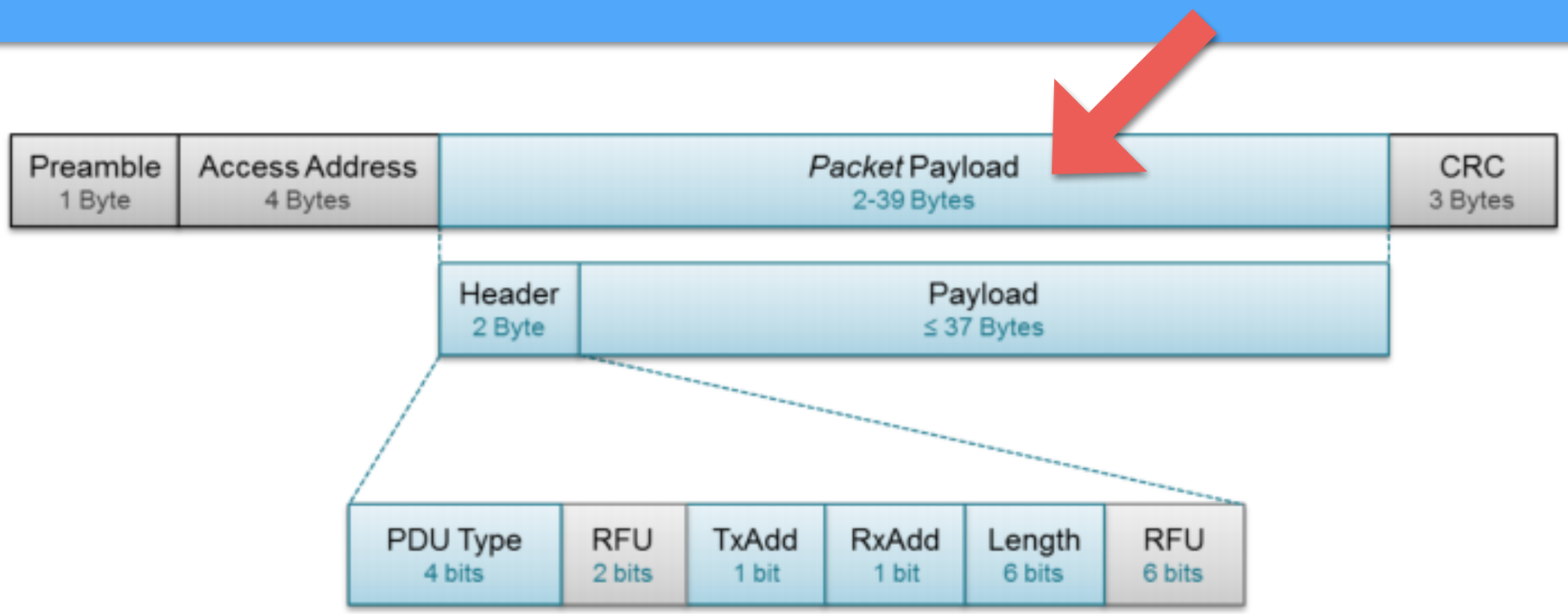
TOPICS

1. BLE Packet Data
   - Advertisement Types
   - Raw Data
2. Install Nodejs on Raspberry Pi
3. Scanning BLE using node.js
4. BLE Handling using node.js

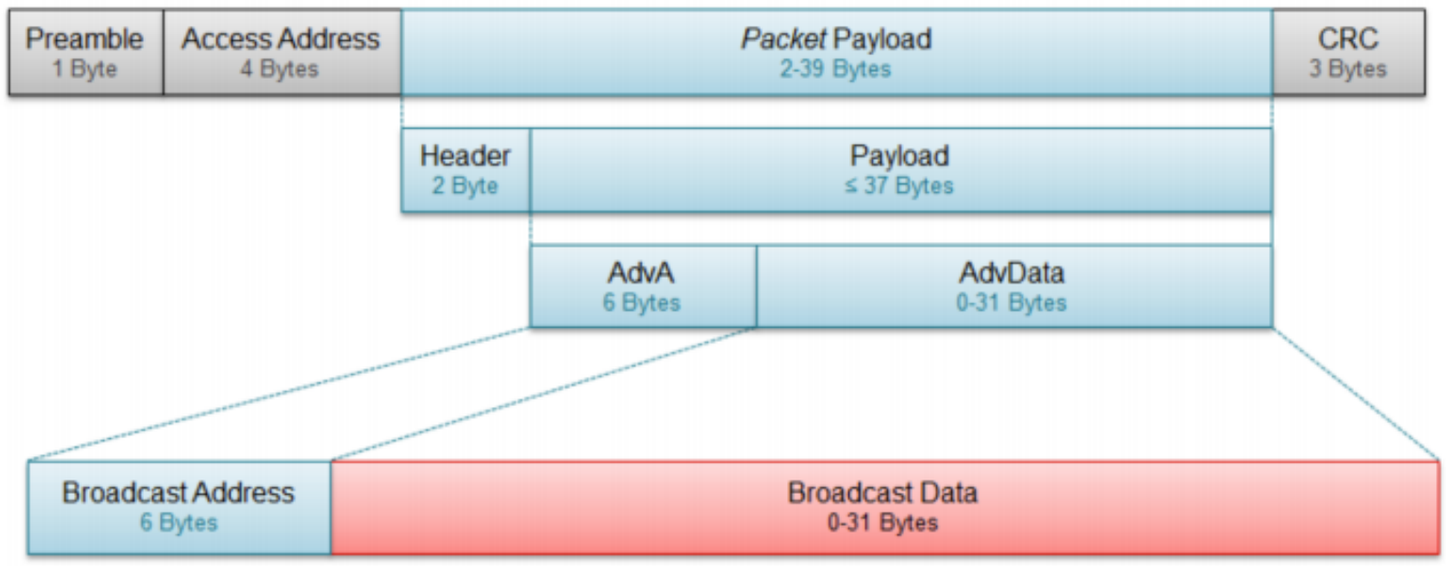


BLE Packet Data

Data Packet



Bluetooth Low Energy Data Packet



Bluetooth Low Energy Data Packet
(รูปภาพอ้างอิงจาก ww.ti.com)

Advertisement Types

Advertising PDU Types for Broadcasting Data

PDU Type	Packet Name	Description
0000	ADV_IND	Connectable undirected advertising event
0010	ADV_NONCONN_IND	Non-connectable undirected advertising event
0110	ADV_SCAN_IND	Scannable undirected advertising event

Advertisement Data Types

AD Data Type	Data Type Value	Description
Flags	0x01	Device discovery capabilities
Service UUID	0x02 - 0x07	Device GATT services
Local Name	0x08 - 0x09	Device name
TX Power Level	0x0A	Device output power
Manufacturer Specific Data	0xFF	User defined

Advertisement Types

Advertisement Data Type, Flags

Byte	Bit	Flag/Value	Description
0		0x02	Length of this data
1		0x01	GAP AD Type Flags
2	0	LE Limited Discoverable Mode	180 s advertising
	1	LE General Discoverable Mode	Indefinite advertising time
	2	BR/EDR Not Supported	
	3	Simultaneous LE and BR/EDR (Controller)	
	4	Simultaneous LE and BR/EDR (Host)	
	5-7		Reserved

Manufacturer Specific Data

Byte	Value	Description
0	0x03-0x1F	Length of this data
1	0xFF	Manufacturer-Specific Data Flag
2	0x0D	Company ID
3	0x00	(Example, 0x000D – Texas Instruments)
4 - 31	-	User Defined Data (optional)

ตัวอย่างข้อมูลการปล่อยสัญญาณBLE

Raw Data

d6 be 89 8e 40 24 05 a2 17 6e 3d 71 02 01 1a 1a ff 4c 00 02 15 e2 c5 6d b5 df
fb 48 d2 b0 60 d0 f5 a7 10 96 e0 00 00 00 00 c5 52 ab 8d 38 a5

---

d6 be 89 8e # Access address for advertising data (this is always the same fixed value)
40 # Advertising Channel PDU Header byte 0. Contains: (type = 0), (tx add = 1), (rx add = 0)
24 # Advertising Channel PDU Header byte 1. Contains: (length = total bytes of the advertising payload + 6 bytes for the BLE mac address.)
05 a2 17 6e 3d 71 # Bluetooth Mac address (note this is a spoofed address)
02 01 1a 1a ff 4c 00 02 15 e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 00 00 00 00 c5 # Bluetooth advertisement
52 ab 8d 38 a5 # checksum

---

02 # Number of bytes that follow in first AD structure
01 # Flags AD type
1A # Flags value 0x1A = 000011010
bit 0 (OFF) LE Limited Discoverable Mode
bit 1 (ON) LE General Discoverable Mode
bit 2 (OFF) BR/EDR Not Supported
bit 3 (ON) Simultaneous LE and BR/EDR to Same Device Capable (controller)
bit 4 (ON) Simultaneous LE and BR/EDR to Same Device Capable (Host)
1A # Number of bytes that follow in second (and last) AD structure
FF # Manufacturer specific data AD type
4C 00 # Company identifier code (0x004C == Apple)
02 # Byte 0 of iBeacon advertisement indicator
15 # Byte 1 of iBeacon advertisement indicator
e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 # iBeacon proximity uuid
00 00 # major
00 00 # minor
c5 # The 2's complement of the calibrated Tx Power


TM
JS

How to Install NodeJS

สามารถเข้าไปอ่านรายละเอียดการติดตั้งได้ที่

https://github.com/nodesource/distributions

Command:

```
# apt-get update
# curl -sL https://deb.nodesource.com/setup_5.x | bash -
# apt-get install -y nodejs
```

```
Get:72 http://us.archive.ubuntu.com wily-backports/restricted Translation-en [28 B]
Get:73 http://us.archive.ubuntu.com wily-backports/universe Translation-en [270 B]
Fetched 31.9 MB in 2min 34s (206 kB/s)
Reading package lists... Done

## Run `apt-get install nodejs` (as root) to install Node.js 5.x and npm
```

```
update-alternatives: using /usr/bin/rlwrap to provide /usr/bin/readline-editor (readline-editor) in auto mode
Setting up nodejs (5.4.1-1nodesource1~wily1) ...
```

Scanning BLE using node.js

module : https://github.com/sandeepmistry/noble

Ubuntu/Debian/Raspbian

```
# sudo apt-get install bluetooth bluez libbluetooth-dev libudev-dev
# nam install noble
```

```
make: Leaving directory '/home/pariwatleamsumran/node_modules/bluetooth-hci-socket/build'
/home/pariwatleamsumran
└─┬ noble@1.3.0
  ├─┬ bluetooth-hci-socket@0.4.2
  │ └── nan@2.2.0
  ├── bplist-parser@0.0.6
  └─┬ debug@2.2.0
    └── ms@0.7.1
```

Actions Start scanning

```
noble.startScanning(); // any service UUID, no d

noble.startScanning([], true); // any service UU
```

Initial module

```
var noble = require('noble');
```

Actions Start scanning

```
noble.startScanning(); // any service UUID, no duplicates

noble.startScanning([], true); // any service UUID, allow duplicates

var serviceUUIDs = ["<service UUID 1>", ...]; // default: [] => all
var allowDuplicates = <false|true>; // default: false

noble.startScanning(serviceUUIDs, allowDuplicates[, callback(error)]); //
particular UUID's
```

Scanning BLE using node.js

Actions Stop

```
noble.stopScanning();
```

Event State Change

```
state = <"unknown" | "resetting" | "unsupported" | "unauthorized" | "poweredOff" |
“poweredOn">
```

```
noble.on('stateChange', callback(state));
```

Event Scan Start

```
noble.on('scanStart', callback);
```

Event Scan Stop

```
noble.on('scanStop', callback);
```

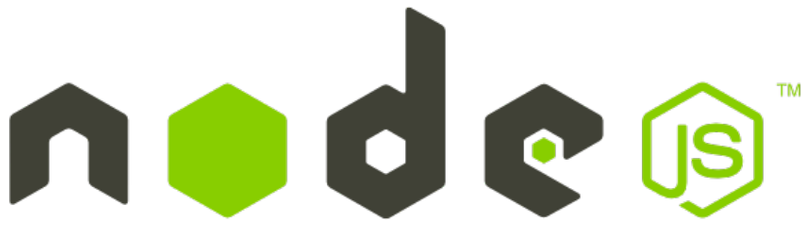
Scanning BLE using node.js

Peripheral discovered

```
peripheral = {
  id: "<id>",
  address: "<BT address">, // Bluetooth Address of device, or 'unknown' if not known
  addressType: "<BT address type>", // Bluetooth Address type (public, random)
  connectable: <connectable>, // true or false, or undefined if not known
  advertisement: {
    localName: "<name>",
    txPowerLevel: <int>,
    serviceUuids: ["<service UUID>", ...],
    manufacturerData: <Buffer>,
    serviceData: [
        {
            uuid: "<service UUID>"
            data: <Buffer>
        },
        ...
    ]
  },
  rssi: <rssi>
};

noble.on('discover', callback(peripheral));
```

ทดสอบเขียนโปรแกรมโดยใช้



Example

```
var noble = require('noble');
console.log('Program Start');

noble.on('stateChange', function(state) {
  console.log('on -> stateChange: ' + state);
  if (state === 'poweredOn') {
    noble.startScanning([],true);
  } else {
    noble.stopScanning();
  }
});

noble.on('scanStart', function() {
  console.log('on -> scanStart');
});
noble.on('scanStop', function() {
  console.log('on -> scanStop');
});

noble.on('discover', function(peripheral) {
     console.log('on -> discover: ' + peripheral);
});
```

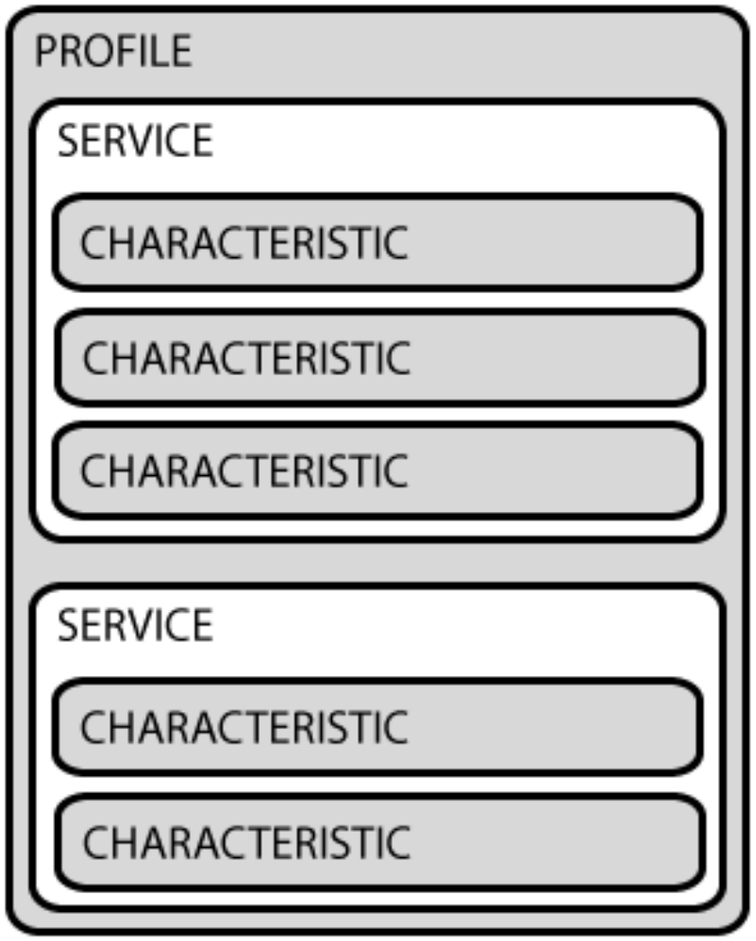
Example

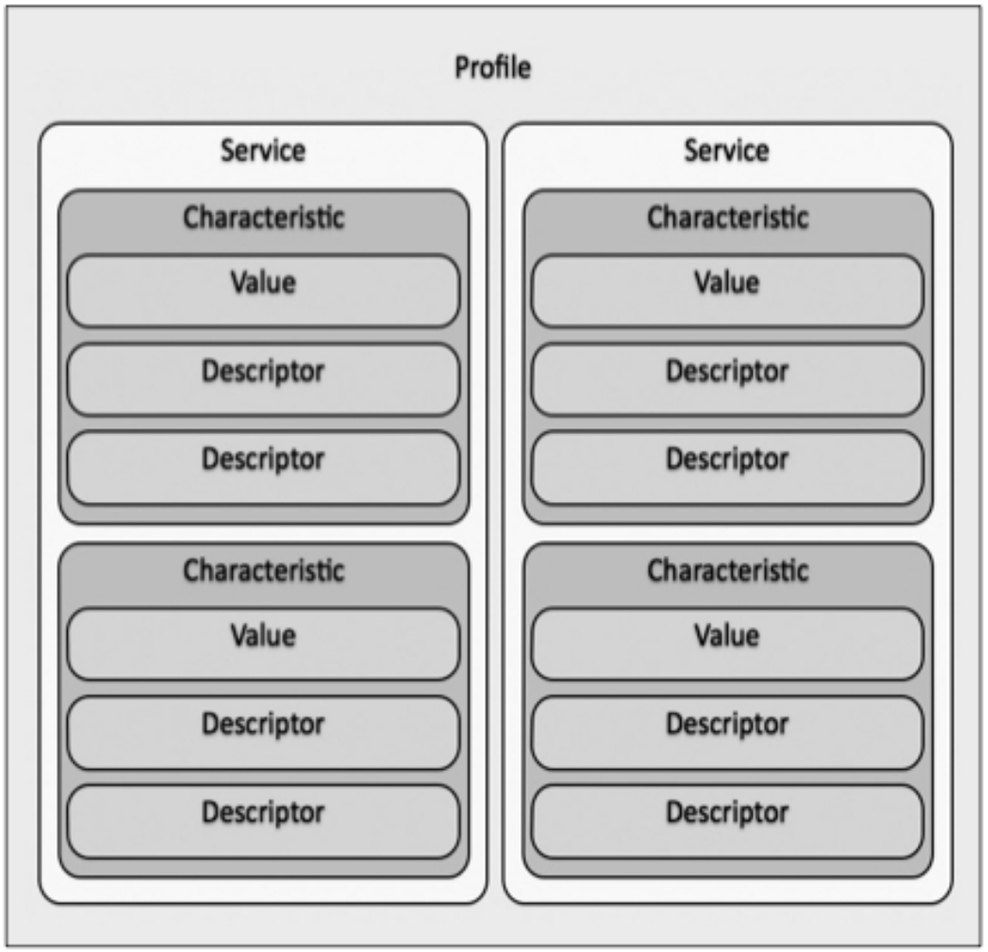
```
function hexstring(aa)
{
  var ret = "",
  i = 0,
  len = aa.length;
    while (i < len) {
      var h = aa[i].toString(16);
      if (h.length < 2)
        h = "0" + h;
      ret += h;
      i++;
    }
return ret;
}
```

BLE Handling using node.js

Software model

Client, Server	ตัวเรียกแทนอุปกรณ์
Characteristic	ข้อมูลที่ใช้ติดต่อถึงกันระหว่าง Client กับ Server
Service	Collection ของข้อมูล โดยจะมีเก็บข้อมูลที่อยู่ในกลุ่มเดียวกัน
Descriptor	คำอธิบายเพิ่มเติมให้กับ Characteristic



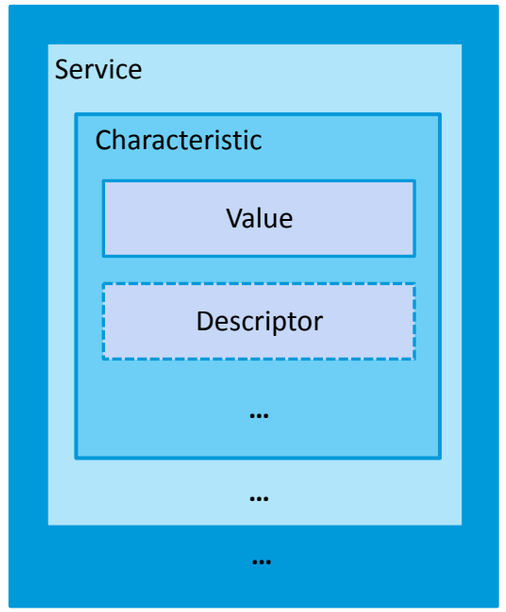


〈그림 1〉 BLE 장치에서 GATT의 역할

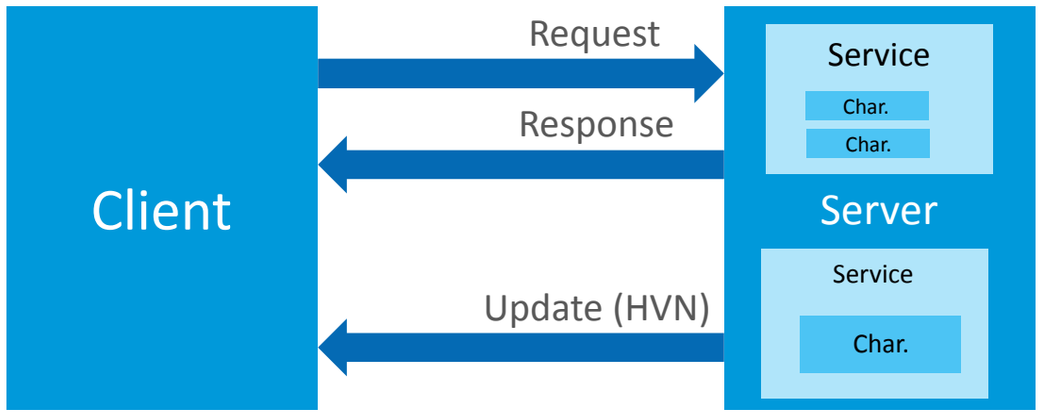
BLE Handling using node.js

GATT Overview

- **Identical Client Server Architecture as ATT**
- Data hierarchically categorized in Services
- Actual data values in Characteristics
- Defines Attribute Table layout and hierarchy
  - Service
  - Characteristic
  - Descriptor



ATT Table



BLE Handling using node.js

RSSI Update

```
this.updateRssi();

peripheral.on('rssiUpdate', callback(rssi));
```

Services discovered

```
this.discoverServices();

peripheral.on('servicesDiscover', callback(services));
```

Included services discovered

```
service.on('includedServicesDiscover', callback(includedServiceUuids));
```

Characteristics discovered

```
characteristic = {
  uuid: "<uuid>",
   // properties: 'broadcast', 'read', 'writeWithoutResponse', 'write', 'notify',
'indicate', 'authenticatedSignedWrites', 'extendedProperties'
  properties: [...]
};

service.on('characteristicsDiscover', callback(characteristics));
```

BLE Handling using node.js

Characteristic Data

```
characteristic.on('data', callback(data, isNotification));

characteristic.on('read', callback(data, isNotification)); // legacy
```

Characteristic Write

```
characteristic.on('write', withoutResponse, callback());
```

Characteristic Notify

```
characteristic.on('notify', callback(state));
```

ทดสอบเขียนโปรแกรมโดยใช้