嵌入式系統實驗, Fall 2016
Lab6 20161030

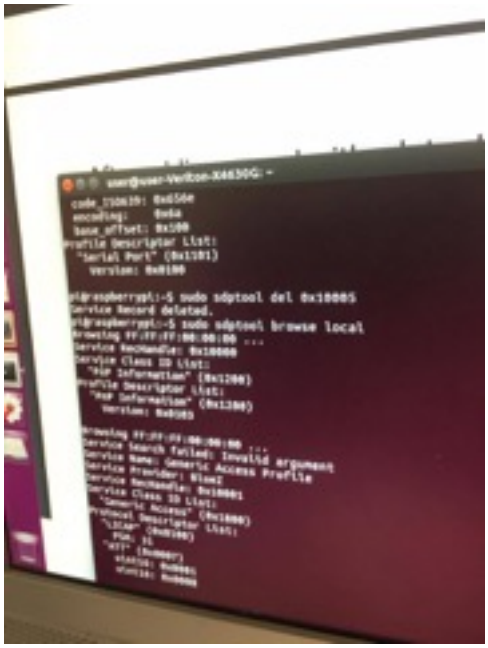
指導教授：王勝德 教授　　B03901023 電機三 許秉鈞

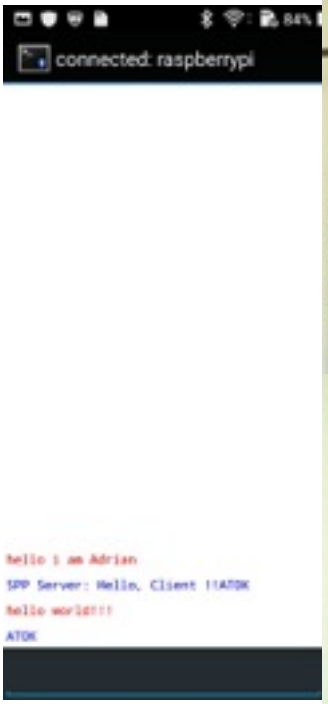
實驗目的

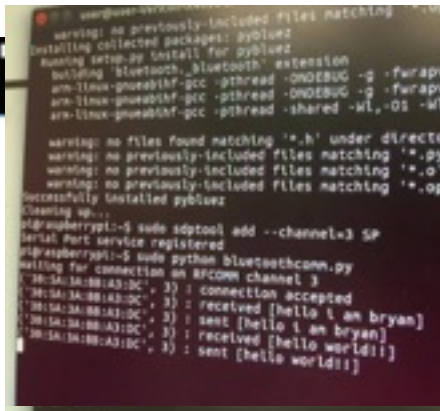
設定Rpi 3的環境，讓serial port成功運作。
成功把bluetooth packages安裝在raspbian OS上；用python, C做出bluetooth socket。

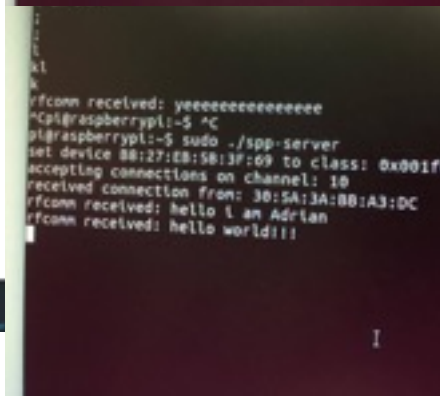
實驗紀錄

（見**lab6-result.txt** 或 下圖）









實驗流程

```
(target) sudo apt-get update
(target) sudo apt-get dost-upgrade
(target) sudo apt-get install bluez blueman libbluetooth-
dev
(target) sudo reboot
(target) sudo systemctl status bluetooth
(target) sudo vi /etc/systemd/system/dbus-
org.bluez.service
```

改這行：`ExecStart=/usr/lib/bluetooth/bluetoothd --compat`

```
(target) sudo systemctl daemon-reload
(target) sudo systemctl restart bluetooth
(target) sudo hciconfig hci0 up
PART 1
(target) sudo bluetoothctl
[Bluetooth]# agent on
[Bluetooth]# default-agent
scan on -> 找到後 -> trust -> pair
PART 2
(target) run python & c program, 結果見「實驗紀錄」
```

實驗問題

- **What do the terms profiles and protocols in Bluetooth mean?**
  - **Serial Port Profiles**

SPP emulates an RS-232 (序列資料通訊的介面標準)serial connection, thereby supporting raw binary communication between two Bluetooth devices.

  - **RFCOMM Protocols**

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. The protocol is based on the ETSI standard TS 07.10.

- **The example programs (Python or C) introduced here are for Bluetooth Classic. How to write Bluetooth Smart (BLE) programs in C or Python?**

reference: https://developer.mbed.org/teams/Bluetooth-Low-Energy/code/BLE_GATT_Example/

```
#include "mbed.h"
#include "ble/BLE.h"

DigitalOut led(LED1, 1);
uint16_t customServiceUUID  = 0xA000;
uint16_t readCharUUID       = 0xA001;
uint16_t writeCharUUID      = 0xA002;

const static char     DEVICE_NAME[]        = "ChangeMe!!"; // change this
static const uint16_t uuid16_list[]        = {0xFFFF}; //Custom UUID, FFFF is reserved for
development

/* Set Up custom Characteristics */
static uint8_t readValue[10] = {0};
ReadOnlyArrayGattCharacteristic<uint8_t, sizeof(readValue)> readChar(readCharUUID, readValue);

static uint8_t writeValue[10] = {0};
WriteOnlyArrayGattCharacteristic<uint8_t, sizeof(writeValue)> writeChar(writeCharUUID, writeValue);

/* Set up custom service */
GattCharacteristic *characteristics[] = {&readChar, &writeChar};
GattService        customService(customServiceUUID, characteristics, sizeof(characteristics) /
sizeof(GattCharacteristic *));


/*
 *  Restart advertising when phone app disconnects
*/
void disconnectionCallback(const Gap::DisconnectionCallbackParams_t *)
{
    BLE::Instance(BLE::DEFAULT_INSTANCE).gap().startAdvertising();
}

/*
 *  Handle writes to writeCharacteristic
*/
void writeCharCallback(const GattWriteCallbackParams *params)
{
    /* Check to see what characteristic was written, by handle */
    if(params->handle == writeChar.getValueHandle()) {
        /* toggle LED if only 1 byte is written */
        if(params->len == 1) {
            led = params->data[0];
            (params->data[0] == 0x00) ? printf("led on\n\r") : printf("led off\n\r"); // print led
toggle
        }
        /* Print the data if more than 1 byte is written */
        else {
            printf("Data received: length = %d, data = 0x",params->len);
            for(int x=0; x < params->len; x++) {
                printf("%x", params->data[x]);
            }
            printf("\n\r");
        }
        /* Update the readChar with the value of writeChar */
        BLE::Instance(BLE::DEFAULT_INSTANCE).gattServer().write(readChar.getValueHandle(), params-
>data, params->len);
    }
}
/*
 * Initialization callback
 */
void bleInitComplete(BLE::InitializationCompleteCallbackContext *params)
{
    BLE &ble          = params->ble;
    ble_error_t error = params->error;

    if (error != BLE_ERROR_NONE) {
        return;
    }

    ble.gap().onDisconnection(disconnectionCallback);
    ble.gattServer().onDataWritten(writeCharCallback);

    /* Setup advertising */
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPORTED |
GapAdvertisingData::LE_GENERAL_DISCOVERABLE); // BLE only, no classic BT
```

```
    ble.gap().setAdvertisingType(GapAdvertisingParams::ADV_CONNECTABLE_UNDIRECTED); // advertising
type
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LOCAL_NAME, (uint8_t
*)DEVICE_NAME, sizeof(DEVICE_NAME)); // add name
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LIST_16BIT_SERVICE_IDS,
(uint8_t *)uuid16_list, sizeof(uuid16_list)); // UUID's broadcast in advertising packet
    ble.gap().setAdvertisingInterval(100); // 100ms.

    /* Add our custom service */
    ble.addService(customService);

    /* Start advertising */
    ble.gap().startAdvertising();
}

/*
 *  Main loop
*/
int main(void)
{
    /* initialize stuff */
    printf("\n\r********* Starting Main Loop *********\n\r");

    BLE& ble = BLE::Instance(BLE::DEFAULT_INSTANCE);
    ble.init(bleInitComplete);

    /* SpinWait for initialization to complete. This is necessary because the
     * BLE object is used in the main loop below. */
    while (ble.hasInitialized()  == false) { /* spin loop */ }

    /* Infinite loop waiting for BLE interrupt events */
    while (true) {
        ble.waitForEvent(); /* Save power */
    }
}
```

- **What is the difference about the pairing process for Bluetooth Classic and Bluetooth Smart (BLE)?**

the key difference is in Bluetooth 4.0's low power consumption. It's actually extremely positive when talking about M2M communication. With Bluetooth LE's power consumption, applications can run on a small battery for four to five years.

1. How do you know the device name corresponding to the just plug in usb devices?

ls -l /dev/disk/by-uuid/

2. How do you know the IP address of your RPi when it is connected to a LAN? (hint: use some network tools, like nmap, arp, ...)

1. To use nmap to scan the devices on your network, you need to know the subnet you are connected to. First find your own IP address, in other words the one of the computer you're using to find your Pi's IP address: (On Linux, type hostname -I into a terminal window)
2. you have the IP address of your computer, you will scan the whole subnet for other devices.
3. use the nmap command with the -sn flag (ping scan) on the whole subnet range. This may take a few seconds:

4. `nmap -sn 192.168.1.0/24`
5. Ping scan just pings all the IP addresses to see if they respond.
6. e.g.

```
Nmap scan report for ubuntu (192.168.1.5)
```

```
Host is up (0.0010s latency).
Nmap scan report for raspberrypi (192.168.1.8)
Host is up (0.0030s latency).
```

3. How do you copy files between a development Linux host and a RPi target?

[把remote pc 寫好的code,透過隨身碟丟到Rpi]

ref: http://www.otakuya.tw/1216/%E5%A6%82%E4%BD%95%E5%9C%A8linux%E8%B7%9Fraspberry-pi%E4%B8%8A%E6%8E%9B%E8%BC%89usb%E7%A1%AC%E7%A2%9F

1. 因為在Rpi的介面上不容易寫code,因此我們試著解決這問題
2. ls -l /dev/disk/by-uuid/
3. sudo mount /dev/sda1 /media/usb
4. 注意/media/usb裡面的檔案都是read only,要用chmod +x 改權限才丟到Rpi的家目錄。

4. How do you know the mounted partition in your current Linux machine?

`$ mount` is the most convenient method. For a complete and exact list of currently mounted filesystems, you should read the contents of /proc/mounts (e.g., with cat /proc/mounts).

5. How do you know the disk partition that appeared or detected by your Linux machine?

```
$ sudo fdisk -l
```

The -l options shows the partition tables for the specified devices and then exit. If no devices are given, those mentioned in /proc/partitions (if that exists) are used.