WioT - Postlab

Lab 2a: BLE Advertisements

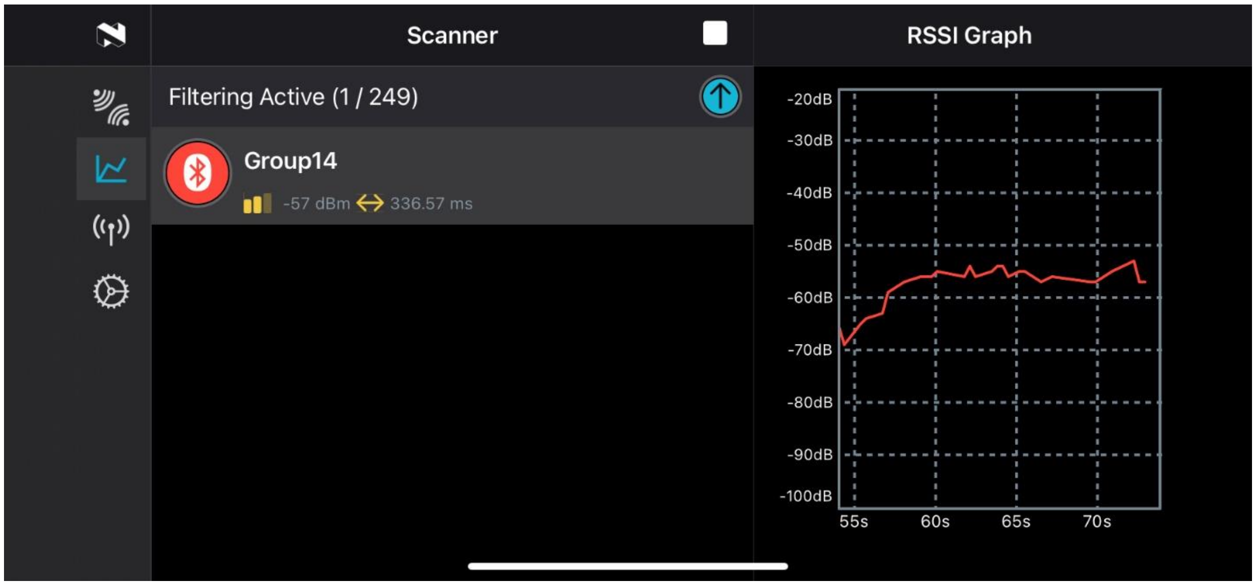
What to submit?

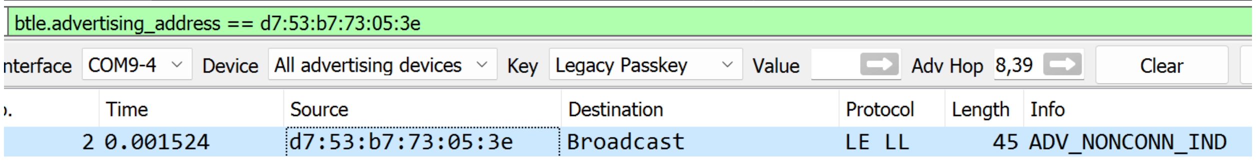
Please use this document as a template, add your responses directly, and export it as a PDF to Gradescope. Each group should submit one postlab.

Group name: Group 14
Team member names: Hudson Burke, Yann Donastien, Zach King

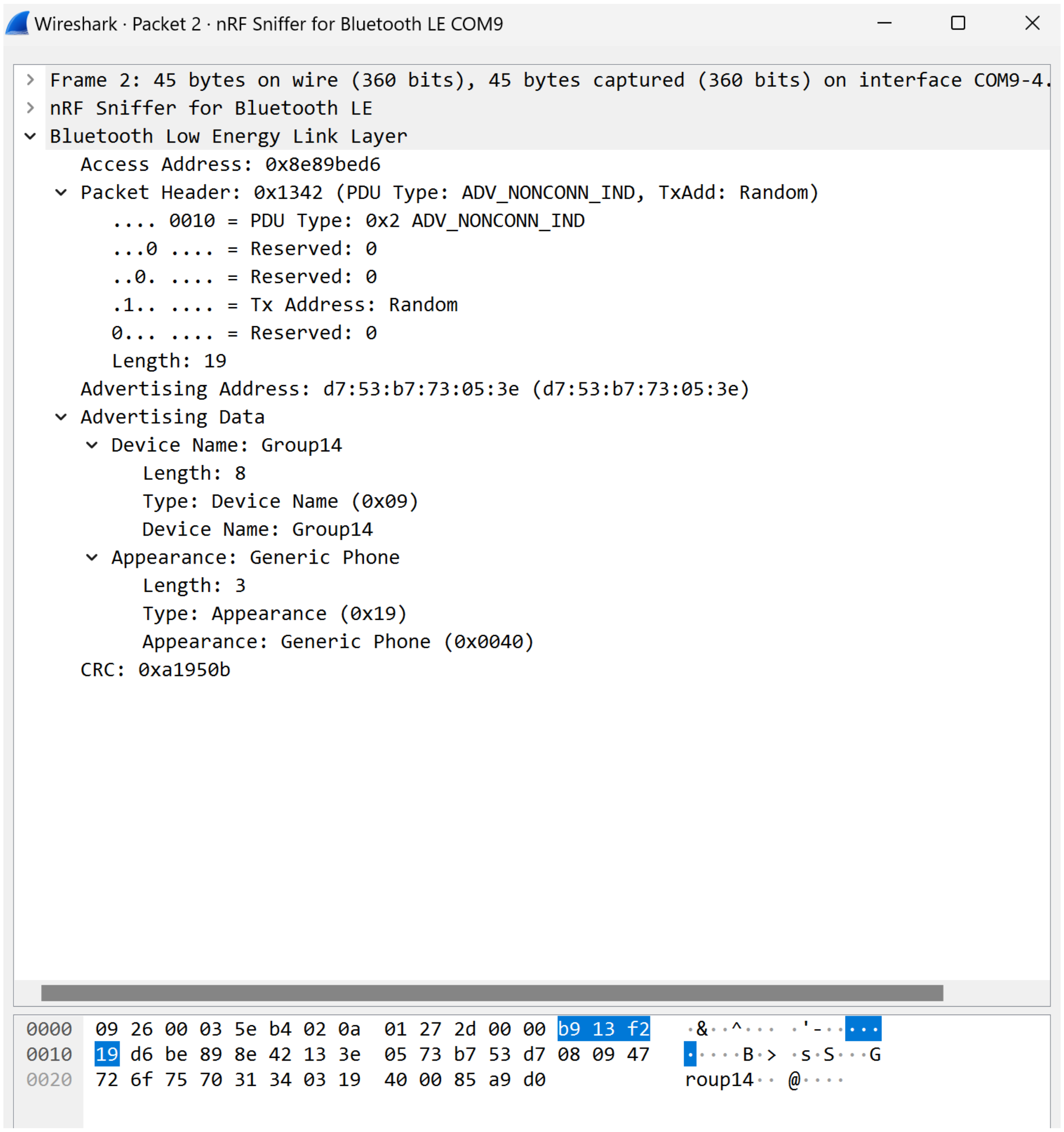
A: Programming a BLE Advertiser

[5pts] Show evidence you were able to create an advertiser with your custom name:





```
btle.advertising_address == d7:53:b7:73:05:3e
nterface COM9-4   Device All advertising devices   Key Legacy Passkey   Value   Adv Hop 8,39   Clear
).      Time        Source              Destination   Protocol  Length  Info
      2 0.001524    d7:53:b7:73:05:3e   Broadcast     LE LL         45  ADV_NONCONN_IND
```



```
Wireshark · Packet 2 · nRF Sniffer for Bluetooth LE COM9

> Frame 2: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface COM9-4.
> nRF Sniffer for Bluetooth LE
v Bluetooth Low Energy Link Layer
     Access Address: 0x8e89bed6
   v Packet Header: 0x1342 (PDU Type: ADV_NONCONN_IND, TxAdd: Random)
        .... 0010 = PDU Type: 0x2 ADV_NONCONN_IND
        ...0 .... = Reserved: 0
        ..0. .... = Reserved: 0
        .1.. .... = Tx Address: Random
        0... .... = Reserved: 0
        Length: 19
     Advertising Address: d7:53:b7:73:05:3e (d7:53:b7:73:05:3e)
   v Advertising Data
      v Device Name: Group14
           Length: 8
           Type: Device Name (0x09)
           Device Name: Group14
      v Appearance: Generic Phone
           Length: 3
           Type: Appearance (0x19)
           Appearance: Generic Phone (0x0040)
     CRC: 0xa1950b

0000  09 26 00 03 5e b4 02 0a  01 27 2d 00 00 b9 13 f2   ·&··^···  ·'-·····
0010  19 d6 be 89 8e 42 13 3e  05 73 b7 53 d7 08 09 47   ·····B·>  ·s·S···G
0020  72 6f 75 70 31 34 03 19  40 00 85 a9 d0            roup14··  @····
```

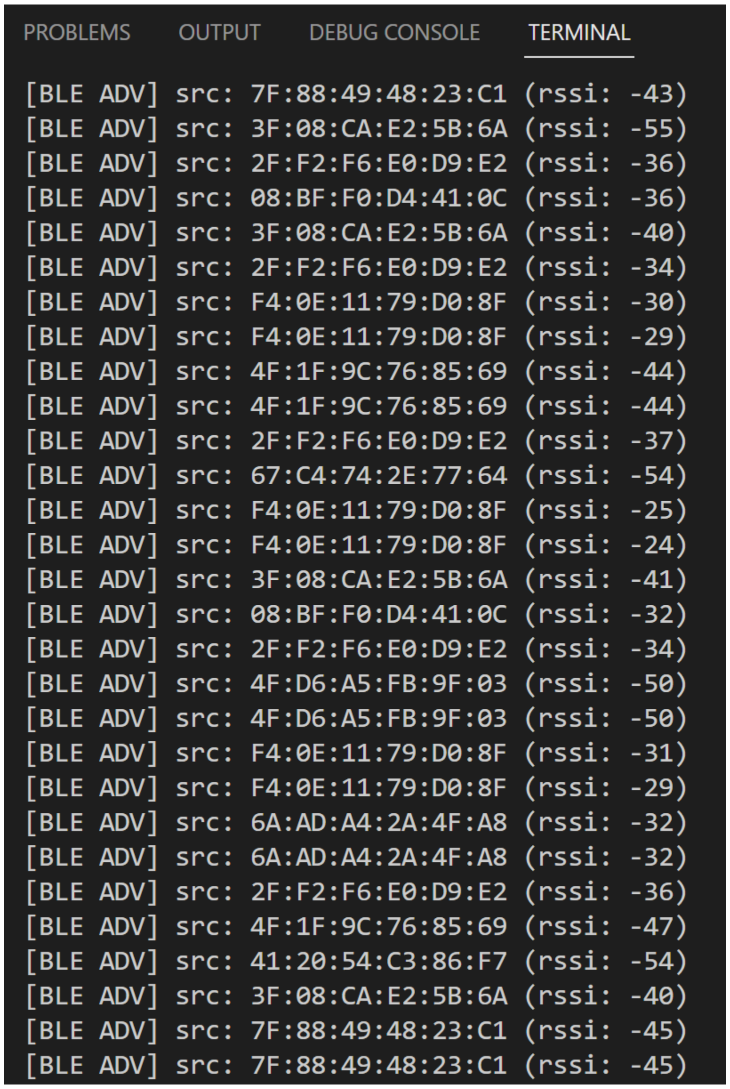
[5pts] Include your updated code:

```
/* main.c - Application main entry point */

/*
 * Copyright (c) 2015-2016 Intel Corporation
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr/types.h>
```

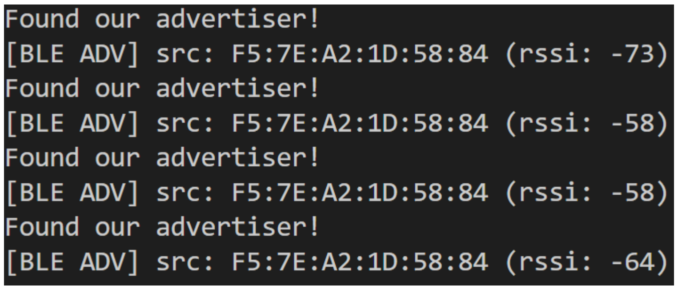
```c
#include <stddef.h>
#include <sys/printk.h>
#include <sys/util.h>

#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>

#define DEVICE_NAME "Group14"
#define DEVICE_NAME_LEN (sizeof(DEVICE_NAME) - 1)
#define PHONE_APPEARANCE 0x0040

#define CUSTOM_BT_LE_ADV_NCONN_IDENTITY
BT_LE_ADV_PARAM(BT_LE_ADV_OPT_USE_IDENTITY, \
                         0x0214, \
                         0x0215, \
                         NULL)


static const struct bt_data ad[] = {
    BT_DATA(BT_DATA_NAME_COMPLETE, DEVICE_NAME, DEVICE_NAME_LEN),
    BT_DATA_BYTES(BT_DATA_GAP_APPEARANCE, 0x40, 0x00)
};


static void bt_ready(int err)
{
    // Note: printk() works the same as printf(), just designed to be used
    // within the "k"ernel.

    if (err) {
        printk("Bluetooth init failed (err %d)\n", err);
        return;
    }

    printk("Bluetooth initialized\n");

    // Start advertising
    err = bt_le_adv_start(CUSTOM_BT_LE_ADV_NCONN_IDENTITY, ad, ARRAY_SIZE(ad),
NULL, 0);
    if (err) {
        printk("Advertising failed to start (err %d)\n", err);
        return;
    }

    // Print the device address.
    char addr_s[BT_ADDR_LE_STR_LEN];
    bt_addr_le_t addr = {0};
    size_t count = 1;

    bt_id_get(&addr, &count);
    bt_addr_le_to_str(&addr, addr_s, sizeof(addr_s));

    printk("Beacon started, advertising as %s\n", addr_s);
}

void main(void)
{
    int err;

    printk("Starting Beacon Demo\n");

    // Initialize the Bluetooth Subsystem. This will call `bt_ready()` when
    // bluetooth is ready.
    err = bt_enable(bt_ready);
    if (err) {
        printk("Bluetooth init failed (err %d)\n", err);
    }
}
```

B: Programming a BLE Scanner

[5pts] Show evidence you were able to implement a scanner with the nRF52840DK:



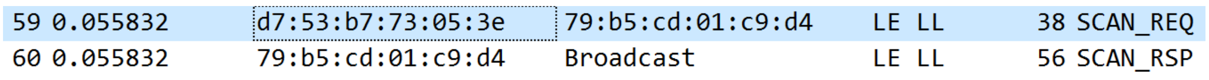
```
[BLE ADV] src: 7F:88:49:48:23:C1 (rssi: -43)
[BLE ADV] src: 3F:08:CA:E2:5B:6A (rssi: -55)
[BLE ADV] src: 2F:F2:F6:E0:D9:E2 (rssi: -36)
[BLE ADV] src: 08:BF:F0:D4:41:0C (rssi: -36)
[BLE ADV] src: 3F:08:CA:E2:5B:6A (rssi: -40)
[BLE ADV] src: 2F:F2:F6:E0:D9:E2 (rssi: -34)
[BLE ADV] src: F4:0E:11:79:D0:8F (rssi: -30)
[BLE ADV] src: F4:0E:11:79:D0:8F (rssi: -29)
[BLE ADV] src: 4F:1F:9C:76:85:69 (rssi: -44)
[BLE ADV] src: 4F:1F:9C:76:85:69 (rssi: -44)
[BLE ADV] src: 2F:F2:F6:E0:D9:E2 (rssi: -37)
[BLE ADV] src: 67:C4:74:2E:77:64 (rssi: -54)
[BLE ADV] src: F4:0E:11:79:D0:8F (rssi: -25)
[BLE ADV] src: F4:0E:11:79:D0:8F (rssi: -24)
[BLE ADV] src: 3F:08:CA:E2:5B:6A (rssi: -41)
[BLE ADV] src: 08:BF:F0:D4:41:0C (rssi: -32)
[BLE ADV] src: 2F:F2:F6:E0:D9:E2 (rssi: -34)
[BLE ADV] src: 4F:D6:A5:FB:9F:03 (rssi: -50)
[BLE ADV] src: 4F:D6:A5:FB:9F:03 (rssi: -50)
[BLE ADV] src: F4:0E:11:79:D0:8F (rssi: -31)
[BLE ADV] src: F4:0E:11:79:D0:8F (rssi: -29)
[BLE ADV] src: 6A:AD:A4:2A:4F:A8 (rssi: -32)
[BLE ADV] src: 6A:AD:A4:2A:4F:A8 (rssi: -32)
[BLE ADV] src: 2F:F2:F6:E0:D9:E2 (rssi: -36)
[BLE ADV] src: 4F:1F:9C:76:85:69 (rssi: -47)
[BLE ADV] src: 41:20:54:C3:86:F7 (rssi: -54)
[BLE ADV] src: 3F:08:CA:E2:5B:6A (rssi: -40)
[BLE ADV] src: 7F:88:49:48:23:C1 (rssi: -45)
[BLE ADV] src: 7F:88:49:48:23:C1 (rssi: -45)
```



```
Found our advertiser!
[BLE ADV] src: F5:7E:A2:1D:58:84 (rssi: -73)
Found our advertiser!
[BLE ADV] src: F5:7E:A2:1D:58:84 (rssi: -58)
Found our advertiser!
[BLE ADV] src: F5:7E:A2:1D:58:84 (rssi: -58)
Found our advertiser!
[BLE ADV] src: F5:7E:A2:1D:58:84 (rssi: -64)
```

[5pts] Show evidence <u>your device</u> asked for a scan response and another device sent a scan response:

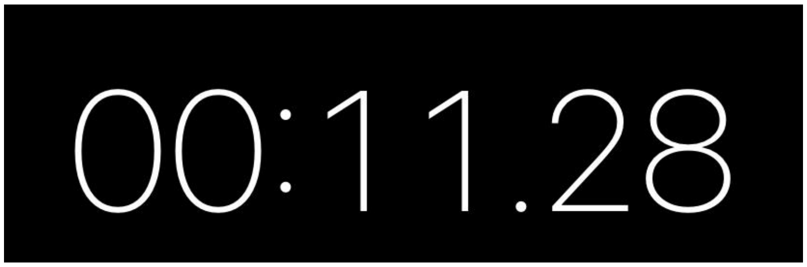
Our scanner MAC address = D7:53:B7:73:05:3E as shown in Part A



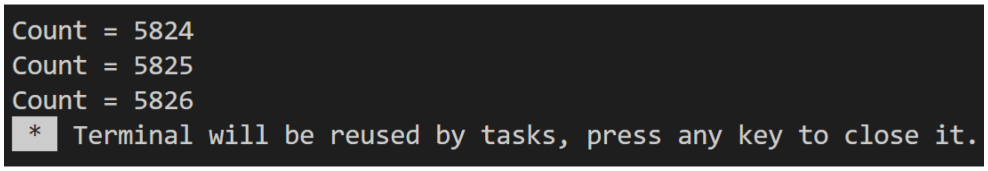
```
59 0.055832      d7:53:b7:73:05:3e   79:b5:cd:01:c9:d4   LE LL     38 SCAN_REQ
60 0.055832      79:b5:cd:01:c9:d4   Broadcast           LE LL     56 SCAN_RSP
```

[5pts] What is the overall average advertising interval (in ms)?

Timed on stopwatch from reset to Ctrl+C:



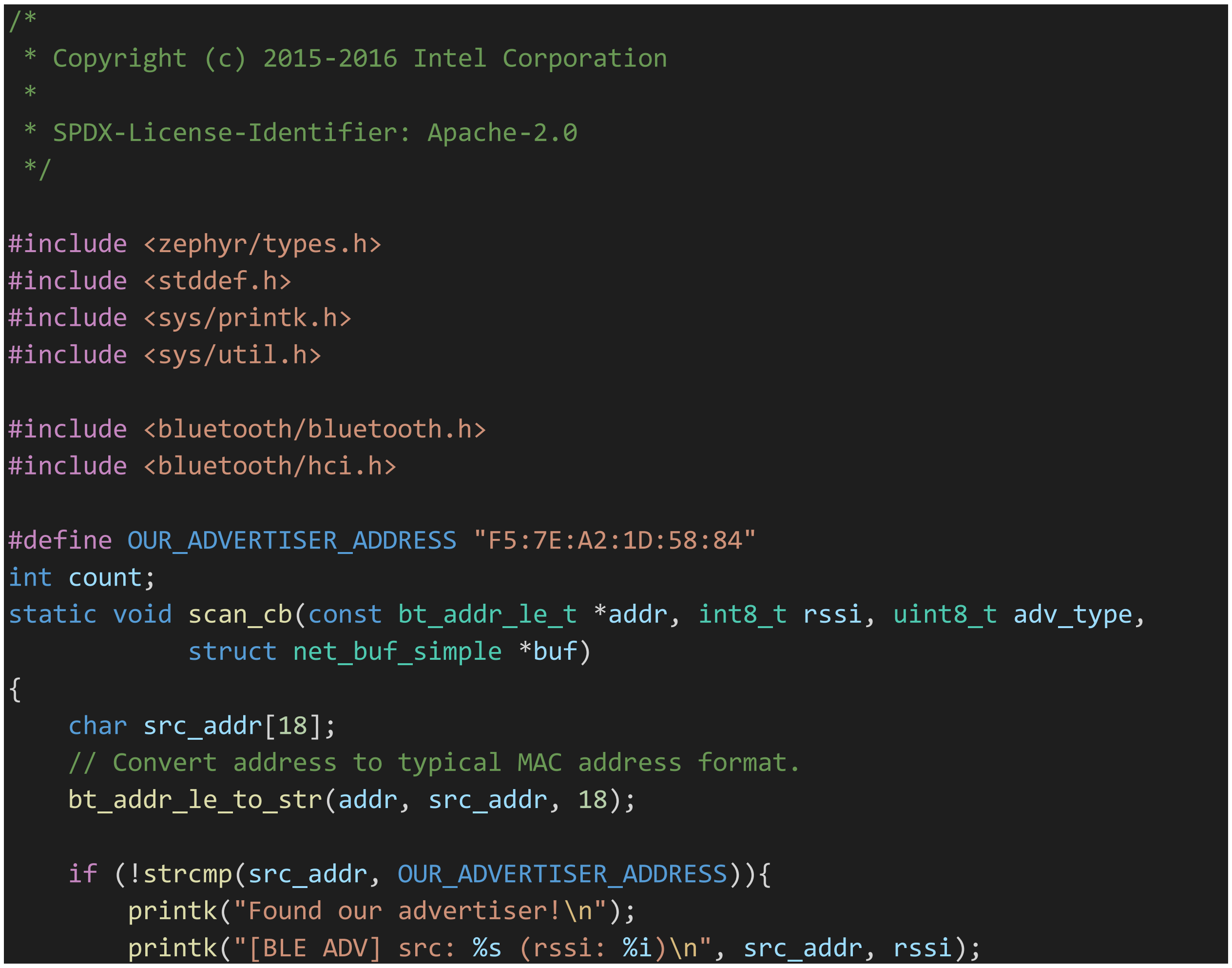
Count = 5826



```
Count = 5824
Count = 5825
Count = 5826
 *  Terminal will be reused by tasks, press any key to close it.
```

$$Average\ Advertising\ Interval = \frac{t}{count} * 1000 = 1.936\ ms$$

[5pts] Include your updated code:



```
/*
 * Copyright (c) 2015-2016 Intel Corporation
 *
 * SPDX-License-Identifier: Apache-2.0
 */

#include <zephyr/types.h>
#include <stddef.h>
#include <sys/printk.h>
#include <sys/util.h>

#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>

#define OUR_ADVERTISER_ADDRESS "F5:7E:A2:1D:58:84"
int count;
static void scan_cb(const bt_addr_le_t *addr, int8_t rssi, uint8_t adv_type,
            struct net_buf_simple *buf)
{
    char src_addr[18];
    // Convert address to typical MAC address format.
    bt_addr_le_to_str(addr, src_addr, 18);

    if (!strcmp(src_addr, OUR_ADVERTISER_ADDRESS)){
        printk("Found our advertiser!\n");
        printk("[BLE ADV] src: %s (rssi: %i)\n", src_addr, rssi);
```

```c
    }
    // if (rssi >= -70){
    //  printk("[BLE ADV] src: %s (rssi: %i)\n", src_addr, rssi);
    // }
    count++;
    printk("Count = %d\n", count);
}

void main(void)
{
    struct bt_le_scan_param scan_param = {
        .type       = BT_HCI_LE_SCAN_ACTIVE,
        .options    = BT_LE_SCAN_OPT_NONE,
        .interval   = 0x0010,
        .window     = 0x0010,
    };
    int err;

    printk("Starting Scanner\n");

    // Initialize the Bluetooth Subsystem
    err = bt_enable(NULL);
    if (err) {
        printk("Bluetooth init failed (err %d)\n", err);
        return;
    }

    printk("Bluetooth initialized\n");
    count = 0;
    err = bt_le_scan_start(&scan_param, scan_cb);
    if (err) {
        printk("Starting scanning failed (err %d)\n", err);
        return;
    }
}
```