**Lab Assignment-3**

Name: Apeksha Sukesh Kallur

**Main.cpp:**

/\*----------------------------------------------------------------------------

LAB EXERCISE - Simple BLE Appcessory

----------------------------------------

In this exercise we will create a BLE Heart Rate Monitor sensor which transmits dummy data to other devices.

GOOD LUCK!

\*----------------------------------------------------------------------------\*/

#include "mbed.h"

#include "BLEDevice.h"

#include "HeartRateService.h"

#include "BatteryService.h"

#include "DeviceInformationService.h"

#include "Utils.h"/\* To turn on/off the debug messages on the console edit this file; NEED\_CONSOLE\_OUTPUT 1/0( it will have an impact on code-size and power consumption.) \*/

// Initialize BLEDevice, Ticker and DigitalOut objects

BLEDevice ble;

Ticker ticker;

DigitalOut led1(LED1);

// Initialize variables and tables

const static char DEVICE\_NAME[] = "HRM\_Apeksha";

static const uint16\_t uuid16\_list[] = {GattService::UUID\_HEART\_RATE\_SERVICE,

GattService::UUID\_DEVICE\_INFORMATION\_SERVICE};

static volatile bool triggerSensorPolling = false;

uint8\_t hrmCounter = 100; // init HRM to 100bps

HeartRateService \*hrService;

DeviceInformationService \*deviceInfo;

// Disconnection Handler

void disconnectionCallback(Gap::Handle\_t handle, Gap::DisconnectionReason\_t reason)

{

ble.startAdvertising(); // restart advertising

}

void onConnection(Gap::ConnectionEventCallback\_t connectionCallback)

{

ble.startAdvertising();

}

void periodicCallback(void)

{

led1 = !led1;

triggerSensorPolling = true;

}

/\*----------------------------------------------------------------------------

MAIN function

\*----------------------------------------------------------------------------\*/

int main(void)

{

// Attach ticker objects to functions

ticker.attach(periodicCallback, 1);

// Initialize the BLE radio

ble.init();

ble.onDisconnection(disconnectionCallback);

// ble.onConnection(connectionCallback);

// Setup BLE device services

hrService = new HeartRateService(ble, hrmCounter, HeartRateService::LOCATION\_FINGER);

deviceInfo = new DeviceInformationService(ble, "ARM", "Model1", "SN1", "hw-rev1", "fw-rev1", "soft-rev1");

BatteryService battery(ble);

// Add the advertising parameter to the payload

ble.accumulateAdvertisingPayload(GapAdvertisingData::BREDR\_NOT\_SUPPORTED | GapAdvertisingData::LE\_GENERAL\_DISCOVERABLE);

ble.accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE\_LIST\_16BIT\_SERVICE\_IDS, (uint8\_t \*)uuid16\_list, sizeof(uuid16\_list));

ble.accumulateAdvertisingPayload(GapAdvertisingData::GENERIC\_HEART\_RATE\_SENSOR);

ble.accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE\_LOCAL\_NAME, (uint8\_t \*)DEVICE\_NAME, sizeof(DEVICE\_NAME));

ble.setAdvertisingType(GapAdvertisingParams::ADV\_CONNECTABLE\_UNDIRECTED);

// Start advertising

ble.setAdvertisingInterval(1600); /\* 1600ms \*/

ble.startAdvertising();

//ble.reset();

while (true)

{

// Check if the "update characteristics" flag is high

if (triggerSensorPolling && ble.getGapState().connected) {

triggerSensorPolling = false;

hrmCounter++;

if (hrmCounter == 150) {

hrmCounter = 80;

}

// Update the heart rate sercive

hrService->updateHeartRate(hrmCounter);

}

// Wait for an event

else {

ble.waitForEvent(); // low power wait for event

}

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ARM University Program Copyright © ARM Ltd 2016\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//