File "/BLE_Button/main.cpp" printed from mbed.org on 11/24/2015

```
/* mbed Microcontroller Library
 * Copyright (c) 2006-2013 ARM Limited
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include "mbed.h"
#include "BLE.h"
#include "ButtonService.h"

BLE         ble;
DigitalOut  led1(LED1);
InterruptIn button(BUTTON1);

const static char     DEVICE_NAME[] = "Button";
static const uint16_t uuid16_list[] = {ButtonService::BUTTON_SERVICE_UUID};

enum {
    RELEASED = 0,
    PRESSED,
    IDLE
};
static uint8_t buttonState = IDLE;

ButtonService *buttonServicePtr;

void buttonPressedCallback(void)
{
    /* Note that the buttonPressedCallback() executes in interrupt context, so it is safer to access
     * BLE device API from the main thread. */
```

```
43      buttonState = PRESSED;
44  }
45
46  void buttonReleasedCallback(void)
47  {
48      /* Note that the buttonReleasedCallback() executes in interrupt context, so it is safer to access
49       * BLE device API from the main thread. */
50      buttonState = RELEASED;
51  }
52
53  void disconnectionCallback(const Gap::DisconnectionCallbackParams_t *params)
54  {
55      ble.gap().startAdvertising();
56  }
57
58  void periodicCallback(void)
59  {
60      led1 = !led1; /* Do blinky on LED1 to indicate system aliveness. */
61  }
62
63  int main(void)
64  {
65      led1 = 1;
66      Ticker ticker;
67      ticker.attach(periodicCallback, 1);
68      button.fall(buttonPressedCallback);
69      button.rise(buttonReleasedCallback);
70
71      ble.init();
72      ble.gap().onDisconnection(disconnectionCallback);
73
74      ButtonService buttonService(ble, false /* initial value for button pressed */);
75      buttonServicePtr = &buttonService;
76
77      /* setup advertising */
78      ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPORTED | GapAdvertisingData::LE_GENERAL_DISCOVERABLE);
79      ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LIST_16BIT_SERVICE_IDS, (uint8_t *)uuid16_list, sizeof(uuid16_list));
80      ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LOCAL_NAME, (uint8_t *)DEVICE_NAME, sizeof(DEVICE_NAME));
81      ble.gap().setAdvertisingType(GapAdvertisingParams::ADV_CONNECTABLE_UNDIRECTED);
82      ble.gap().setAdvertisingInterval(1000); /* 1000ms. */
83      ble.gap().startAdvertising();
84
85      while (true) {
86          if (buttonState!=IDLE) {
87              buttonServicePtr->updateButtonState(buttonState);
88              buttonState = IDLE;
89          }
90
91          ble.waitForEvent();
92      }
93  }
```

Här skickas värde 1 eller 0 till telefon beroende på om knapp är intryckt.