Bluetooth Low Energy

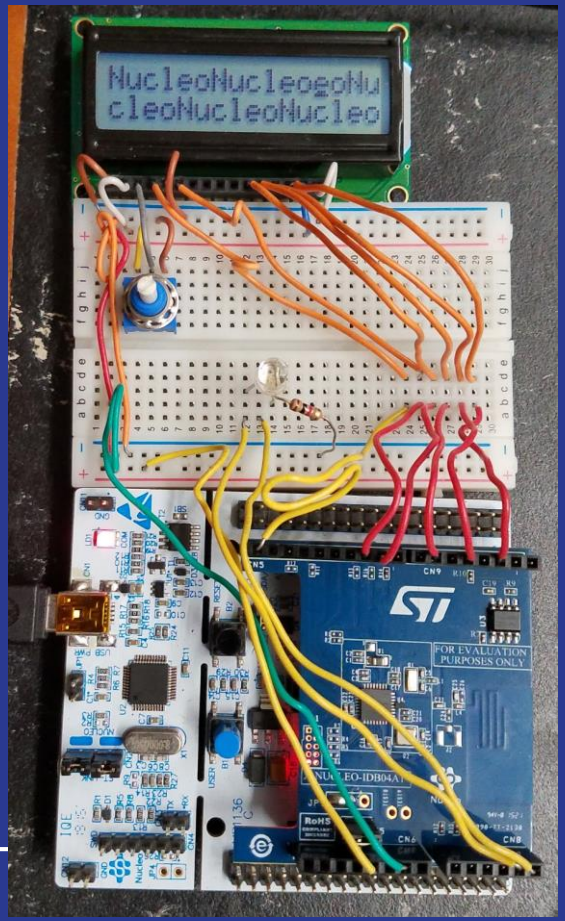
Aurelio Arango, Dung Ly, Moises Guadalupe

Project Details

- Using BLE API from embed and the BLE expansion board for the STM nucleo board, we make the board broadcast Advertising data that allows us to connect to the board through a bluetooth connection from a smartphone using the nRF Connect app from Nordic Semiconductor
- Once a connection is created, we use the TEXT LCD library from mbed to display string variables on the LCD screen.

Project Setup

- Pins used to connect LCD display:
  PB_8, PB_6, PC_7, PB_10, PB_5, PA_10



Project code

```
#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.) */
//#include "UARTService.h"
#include "TextLCD/TextLCD.h"
#include "string.h"

// Initialize BLEDevice, Ticker and DigitalOut objects
BLEDevice ble;//object for BLE
Ticker tick; //initialize ticker
Ticker char_tick;
DigitalOut led_out(LED1); //to display LED
//DigitalOut led_out_2(PC_0); //to display LED
DigitalOut led_out_3(PC_1); //to display LED

//Set Output pins
TextLCD lcd(PB_8, PB_6, PC_7, PB_10, PB_5, PA_10); // rs, e, d4-d7

//UARTService *   uartServicePrt;
const static unsigned MAX_SIZEOF_RX_PAYLOAD =20;
char payload_rx[MAX_SIZEOF_RX_PAYLOAD]={0,};
bool payload_rx_updated=false;

const static char device_name [] = "BLE_BOARD";
const char device_Manu [] = "Nucleo";
const char device_SN [] = "CS435-001";
const char device_HWR [] = "CS435-HW-01";
const char device_FWR [] = "CS435-v-1";
const char device_SV [] = "CS435-SV-0.5";
uint8_t *  remoteName;
unsigned * remotelength;
uint8_t DatatoReceived[1000];
```

```
static const uint16_t gat_services[] = {GattService::UUID_BATTERY_SERVICE ,
                                        GattService::UUID_DEVICE_INFORMATION_SERVICE};
bool update_characteric_flag =false;
// Disconnection Handler

void disconnectionCallback(Gap::Handle_t handle,Gap::DisconnectionReason_t reason)
{
    lcd.printf("Disconnected...\n");
    ble.startAdvertising();
    lcd.cls();
    lcd.printf("Waiting to connect!\n");
}

// Connection Handler
void connectionCallback(Gap::Handle_t handle, const Gap::ConnectionParams_t *reason)
{
    lcd.printf("Connecting...\n");
    ble.stopAdvertising();
    lcd.cls();

    lcd.printf("Waiting for data!\n");

}

void onDataWritten(const GattCharacteristicWriteCBParams * params )
{

    lcd.printf("onDataWritten..!!\n");
    if (  (params->len > 0)) {
        uint16_t bytesRead = params->len;

        lcd.cls();
        lcd.printf("Data Received.!!\n");

        for(int j=0;j<bytesRead;j++)
        {
            lcd.printf(" %x\n",(*(params->data)+j));
            DatatoReceived[j]=(*((params->data)+j))+1;
        }
        wait(1);

        lcd.cls();
        lcd.printf("Print Data.!!\n");
        for(int j=0;j<bytesRead;j++)
        {
            lcd.printf(" %x\n",DatatoReceived[j]);
        }
        wait(1);
    }
}
```

Project code

```
int main(void)
{
    // Attach ticker objects to functions
    tick.attach(&blinky,1);
    char_tick.attach(&characteristic_flag,1);
    // Initialize the BLE radio
    ble.init();
    ble.onDisconnection(disconnectionCallback);
    ble.onConnection(connectionCallback);
    //to read data from server/client
    ble.onDataWritten(onDataWritten);
    //Batteryservice
    BatteryService batservice(ble,90);
    //Device information service
    DeviceInformationService device(ble,device_name,device_Manu, device_SN, device_HWR,device_FWR,device_SV);
    //End of GATT Services
    ble.accumulateAdvertisingPayload(GapAdvertisingData::GENERIC_HEART_RATE_SENSOR);
    ble.accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPORTED | GapAdvertisingData::LE_GENERAL_DISCOVERABLE);
    ble.accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LOCAL_NAME, (uint8_t *)device_name, sizeof(device_name));
    ble.accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LIST_16BIT_SERVICE_IDS , (uint8_t *)gat_services, sizeof(gat_services));
    //For some reason it needs an address otherwise it doesnt work...
    ble.setAdvertisingType(GapAdvertisingParams::ADV_CONNECTABLE_UNDIRECTED);
    // Start advertising
    ble.setAdvertisingInterval(160); // 1000ms; in multiples of 0.625ms. //
    ble.startAdvertising();
    lcd.cls();
    lcd.printf("Starting...");
    wait(1);
    lcd.cls();
    lcd.printf("Waiting to connect!\n");
    while(true)
    {
        if(payload_rx_updated)
        {
            lcd.cls();
            payload_rx_updated=false;
        }
        else if(update_characteric_flag)
        {
            if (ble.getGapState().connected)
            {
              printf("on while \n");
            }
            update_characteric_flag=false;
        }
        else
        {
        //led_out_2=0;
          //led_out_3=0;
        }
            ble.waitForEvent();
    }
}
```

Bill of Materials

- STM32F4RE01 board
- X-NUCLEO-IDB04A1 expansion board
- BreadBoard
- Any smartphone capable of running NRF Connect app
- NRF connect app by Nordic Semiconductor