CSE190 Project 3
Adding low-energy radio communication

Due Friday February 28th, 10:00pm. This is a group assignment, same as the previous project. Please manage your code using Github. Submission like previous assignments is through Gradescope. You need to use new skeleton code for this project, you will be copying over your code from Project 2 into this skeleton.

Overview

In this project you will make your PrivTag device beacon when it has been lost with Bluetooth Low Energy so it can be detected by any nearby smartphone. Specifically, you will integrate the Bluetooth Low Energy example code included in this project in order to send proximity alerts.

Resources

For this project the most important documents are:

- STMicro BLE code customized for our project
- BlueNRG-MS General Datasheet
- BlueNRG-MS ACI STMicro
- Bluetooth Core Specification V6

Grading Criteria

- 90% Added Bluetooth Low Energy alert interface to a smartphone
- 10% Code readability and commenting. In particular, explain non-obvious things like why you selected the particular commands to shut down and restart the Bluetooth modem.

Step 0: Testing the Starter Code

Download the BLE code customized for this project by Aaron Mega Corp and run it. They provided an example of the STM32CubeIDE project that you will need to execute in order to merge the BLE code into your main.c with their **`main.c`**. Essentially, this code will let any BLE-capable smartphone pair with your device and create a virtual "UART" connection over BLE. This means that you can connect your smartphone to the PrivTag, then you will be able to receive the sample string (in `main.c`) from the PrivTag.

Also, you may want to change the name and address of your tag so you don't accidentally connect to another team's tag! The global variables that contain the name and address are **`[name,local_name]`** and **`bd_addr`** in `ble.c`. Note, your name must be less than 8 characters.

To test if your tag is beaconing properly, you will need to install a Bluetooth proximity detection App on your Android or iOS device. In this course we will use the testing app: NRF Toolbox from Nordic Semiconductor. The following steps should be performed to test your tag:

[iPhone/Android]

a. Run the NRF toolbox and select the "UART" feature.
b. Click "Connect" and it will search for your bluetooth tag (make sure to turn off the filter by UUID mode).
c. If the tag is working properly you will see **`"PrivTag"`** listed as a device you can connect to.
d. Once the device is connected, you can send and receive messages to the device.
    i. **iPhone:** click "Show Log" to see messages that are sent from the device
    ii. **Android:** swipe from the left of the screen to the right (where it says "UART") to bring up the control window

Step 1: Add Bluetooth Proximity Detection

Relevant datasheets:

- STMicro BlueNRG-MS datasheet Chapter 6 (Operating Modes)
- STMicro BlueNRG-MS Application command Interface (ACI)
- Bluetooth Core Specification V6

Files that will be graded: your **`main.c`** file and any modifications you make to the BLE code

Note: You should start this step by opening the provided STM32CubeIDE project **`"youlostit-ble"`**. Then you will integrate your Project 2 code into this codebase.

Let's make PrivTag easier to find by replacing your LED beacon with the newest low-energy radio technology that is available on all of our smartphones, Bluetooth Low Energy. PrivTag should not allow connections until it is dropped (or moved), then when it is dropped it should print every 10 seconds over the BLE virtual UART the name of the tag and the amount of time that the tag is lost. For instance: `"PrivTag <tagname> has been missing for <N> seconds"`. Let's do this in two steps

1. Integrating the Bluetooth tag into your code:

Start by integrating your project into the **`youlostit-ble`** STM32CubeIDE project. You should be able to do a preliminary test of the BLE device using the NRF Toolbox app on your own smartphone. The test should have a workflow as follows:

a. The PrivTag continuously checks for movement and enters the lost mode if no movement is detected for one minute (from Project 2)
b. Once the PrivTag enters lost mode, instead of blinking the LED's, the tag should connect to the smartphone and output the above message every 10 seconds. **Note that the above message may be too long for a single BLE packet, so you will either need to shorten the text or send the message over two BLE packets.**

2. Implementing Disconnection and Non-discoverability functionality

If your integration from step 1 works, that's great! However, you are not done yet, you still need to make this tag privacy enabled so it does not output any pesky Bluetooth signals that someone could use to track you when you are walking around town and your PrivTag is not lost. This means you need to disconnect any clients and put the device into a non-discoverable mode when it is moving, and wake up the BLE device and beacon for clients to connect when it is stationary for one minute.

The functionality of the disconnecting and making the tag non-discoverable is already implemented as functions in `ble.h`. Please read them carefully and understand how they work. Unfortunately, the command packet for disconnecting and the command and event packets for making the tag non-discoverable have to be filled out in `ble_commands.h`. The disconnect event is already given as reference. To understand command packets and event packets, we would highly recommend reading chapters 1 and 2 of the STMicro BlueNRG-MS Application command Interface (ACI). Moreover, the TA has provided hints in the `README.md` of the starter code that will help you complete the command and events packets for these two functions. You can also read more details about the already existing commands and events by reading the ACI as well as the starter code provided.

We also recommend testing these two functions once you complete their commands and events. Once done, integrate these two functions into your already existing code to achieve the functionality described above.

Note: every time you need to reboot the microcontroller (e.g., every time you reprogram it) it will lose the bonding information about the attached smartphone. This means, every time you remove power if you want to test your BLE beaconing again, you may need to go to the smartphone's Bluetooth configuration pane and delete the pairing with the "`PrivTag`" device before you reconnect with it.