Pre-Lab 2: Bluetooth Low Energy

## What to submit?

Please use this document as a template, add your responses directly, and export it as a PDF to Gradescope.

## Who’s submitting?

Everyone must submit their own PreLab! You should (must!) work in your groups, but we want everyone to have the infrastructure setup.

# 

# Background / Vocabulary

*Background/Vocabulary – ”Flashing”*

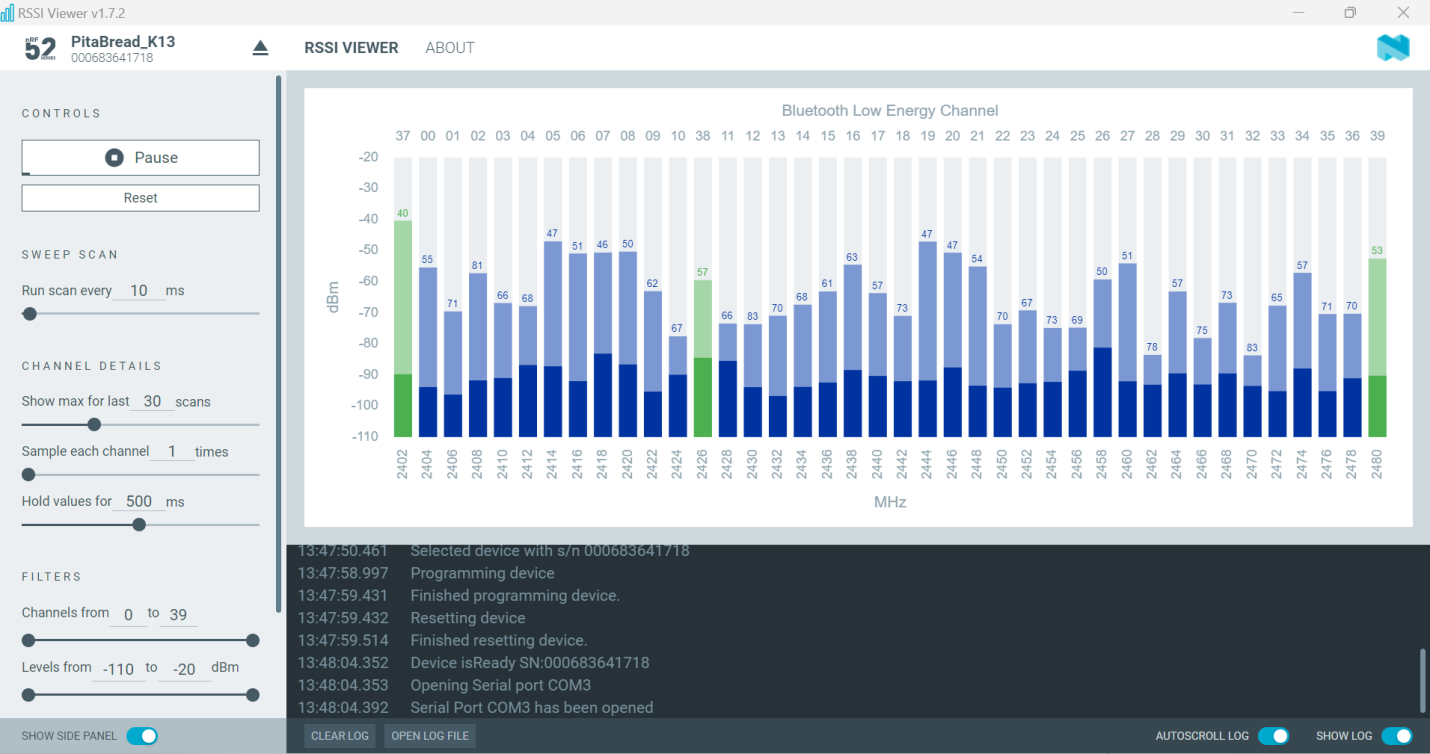
For many embedded systems, there is no notion of an ‘operating system’ or ‘application’. There is simply one, monolithic program that does it *all*. Starting from boot when power is first applied, this program must initialize all the hardware, set up all the memory, and then start doing whatever it needs to do. When you build a “program”, you are actually building a complete system image. This image then needs to be *flashed*, or loaded, onto the hardware platform (it is typically/historically written into flash memory; hence, flashed). The workflow in embedded development then is not “compile and run”, but rather it is “compile, flash, and reboot”.

*Background/Vocabulary – ”Hello World” in embedded systems*

Most of the time, when you learn a new programming language, platform, environment, etc., the first program you write and run is “Hello World”. With embedded systems, however, this is a bit more challenging. If you loaded a program that “printed” hello, where would the hello print?—there’s no screen! Instead the “hello world” of embedded development is the “blinky” app. This is an application that blinks an LED in a predictable way (usually once a second or so, but increasingly people like to make their blinky apps show more fun patterns). Modifying the base blinky app to blink at different rates or different patterns is a great way to make sure you understand the basics of the platform you are working with.

# Pre-Lab Activities

## **Q1–Install nRF Connect for Desktop**

Run the RSSI viewer app. Include a screenshot here:  
  
  


## **Q2–Setting up VS Code:**

Make a modification to the blink app and verify you can flash the app and see the change on the board.

Copy your modified code or provide a screenshot here:

A computer screen shot of a code

AI-generated content may be incorrect.

# BLE Advertisement Background & Resources

* Advertisement [TLV format](https://en.wikipedia.org/wiki/Type%E2%80%93length%E2%80%93value)
  + For BLE advertisements, the format of data is Length-Type-Value
    - Where Length is the number of bytes for the Type + Value (doesn’t count itself)
  + These “Advertisement Data” TLVs are concatenated together to create a full advertisement payload.
  + Type numbers are defined in the [Generic Access Profile document](https://drive.google.com/file/d/1sSmxGgnDKVQYuL0ouiPtVu2UQUcfnow0/view?usp=share_link)
  + Value fields have their own layout, defined in the [Core Specification Supplement](https://drive.google.com/file/d/1CAfY5kzzroI2vSOkzgqAyowqblbNZ6hd/view?usp=share_link)
    - 16-bit Service UUIDs are defined in the [16-bit UUID Numbers document](https://drive.google.com/file/d/1ojKzxRrLOvCy8FHDHgfVLZLPwFafyfcv/view?usp=share_link)
* Advertisement decoding example: <https://community.silabs.com/s/article/kba-bt-0201-bluetooth-advertising-data-basics?language=en_US>
* To determine what a device is, you will likely have to do a little googling. Usually, a name of a device, or the service UUID and the word “BLE”, can get you pretty far.
* Eddystone URL beacon encoding
  + Eddystone payloads use the service UUID 0xFEAA. The service data attached is then further specified by protocol documentation:
  + <https://github.com/google/eddystone/blob/master/protocol-specification.md>
  + <https://github.com/google/eddystone/tree/master/eddystone-url>

# Pre-Lab Homework-Style Questions

## Q3– Simple BLE Advertisements

These are hexadecimal byte values from the payload of a BLE advertisement:

| **Offset** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 04 | 09 | 48 | 69 | 21 | 02 | 0A | 00 |
| **8** | -- | -- | -- | -- | -- | -- | -- | -- |
| **16** | -- | -- | -- | -- | -- | -- | -- | -- |
| **24** | -- | -- | -- | -- | -- | -- | -- | -- |

1. What Types are included in this BLE advertisement?  
   *09- Complete Local Name*

*0A : Tx Power Levels,*

1. For each type, what information is associated with it? Translate from raw data into meaningful information: for names → the name in ASCII, for services → the meaning of the service, etc.

*Message for 1 : Hi!*

*Tx power value is : 0*

## Q4: Real-world BLE Advertisement 1

These are hexadecimal byte values from the payload of a BLE advertisement:

| **Offset** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 03 | 03 | 03 | FE | 0E | 09 | 4C | 45 |
| **8** | 5F | 57 | 48 | 2D | 31 | 30 | 30 | 30 |
| **16** | 58 | 4D | 34 | -- | -- | -- | -- | -- |
| **24** | -- | -- | -- | -- | -- | -- | -- | -- |

1. What Types are included in this BLE advertisement?  
     
   *Type: 03 (Complete List of 16-bit Service UUIDs)*

*Type: 09 (Complete Local Name)*

1. For each type, what information is associated with it? Translate from raw data into meaningful information: for names → the name in ASCII, for services → the meaning of the service, etc.

*Type: 03 (Complete List of 16-bit Service UUIDs)*

*03 03 03 FE*

*Data: 03FE (Service UUID for Sony)*

*Type: 09 (Complete Local Name)  
0E 09 4C 45 5F 57 48 2D 31 30 30 30 58 4D 34  
ASCII or device name : "LE\_WH-1000XM4"*

1. What is this device and why does it have that service?

***Its Sony*** *LE\_WH-1000XM4 and its headphones. IT has some custom protocols maybe for security reasons.it has LE which means Blue tooth Low Energy.*

## Q5: Real-world BLE Advertisement 2 (Optional / Extra Credit)

These are hexadecimal byte values from the payload of a BLE advertisement:

| **Offset** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 02 | 01 | 01 | 03 | 03 | EC | FE | 0B |
| **8** | 09 | 56 | 65 | 6E | 75 | 65 | 2D | 54 |
| **16** | 69 | 6C | 65 | -- | -- | -- | -- | -- |
| **24** | -- | -- | -- | -- | -- | -- | -- | -- |

1. What Types are included in this BLE advertisement?  
   Type 0x01: Flags (at offset 0)

Type 0x03: Complete List of 16-bit Service UUIDs (at offset 3)

Type 0x09: Complete Local Name (at offset 7)

1. For each type, what information is associated with it? Translate from raw data into meaningful information: for names → the name in ASCII, for services → the meaning of the service, etc.  
     
     
   Length: 0x02

Data: 0x01

Meaning: LE General Discoverable Mode enabled

1. What is this device and why does it have that service?

This seems to be Venue Tracking and was wisely used during Covid -19 for contact tracing.

## Q6: Eddystone Advertisement (Optional / Extra Credit)

These are hexadecimal byte values from the payload of a BLE advertisement:

| **Offset** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 02 | 01 | 06 | 03 | 03 | AA | FE | 0F |
| **8** | 16 | AA | FE | 10 | BA | 03 | 78 | 6B |
| **16** | 63 | 64 | 00 | 32 | 30 | 35 | 35 | -- |
| **24** | -- | -- | -- | -- | -- | -- | -- | -- |

1. What Types are included in this BLE advertisement?
2. For each type, what information is associated with it? Translate from raw data into meaningful information: for names → the name in ASCII, for services → the meaning of the service, etc.
3. What is the Eddystone URL here?