🍒

How to test UWB (4) - multiple points

⭐ 목표: 아이폰이 multiple anchors도 인식하게 하기

📌 Fix codes on DWS3000 (앵커 이름 변경)

- 앵커의 이름을 변경하는 이유는 여러 앵커의 신호를 한꺼번에 받을 때 구분되게 하기 위함이다.
- 모든 앵커의 이름은 'DWM3000EVB + nRF52840DK'로 초기화되어 있다.

How to test UWB(2)에서 예제 코드를 돌렸던 것처럼 SES를 이용해서 이름만 변경하고, 앵커는 계속해서 아이폰에서 인식할 수 있는 신호를 발산하게 할 수 있다.

1. 해당 경로에 접근한다.

   C:\Qorvo_Apple_Nearby_Interaction_1.0.0\Qorvo_Apple_Nearby_Interaction_1.0.0**Sources**\Qorvo\Qorvo_Apple_Nearby_

   (이전에 실행시켰던 바이너리 코드의 기반이 되는 프로젝트 파일이다.)
2. **NRF52840DK.emProject** 을 실행한다.
3. Application_config 안에 있는 `project_name.c` 파일을 연다.
4. const char `BoardName[]` 을 원하는 이름으로 수정한다.

   기존 이름은 DWM3000EVB + nRF52840DK이고 나는 ANCHOR_0, 1, 2, ... 로 수정하였다.
5. 앵커가 컴퓨터와 연결된 상태에서 해당 프로젝트를 Compile and Run한다.

   (반드시 충전용이 아닌 데이터 전송용 케이블을 사용해야 한다!! 안 그럼 업로드 되지 않는다 ㅠㅠ)

앵커 이름 변경 완료!

📌 Understanding data interaction process

🚩 Understanding App Structure (앱의 구조)

크게 '데이터'와 '통신'을 담당하는 두 부분으로 나뉜다.

Nearby Interaction

- **Nearby Interaction** Framework 사용

  > Apple Developer Documentation
  > https://developer.apple.com/documentation/nearbyinteraction

  - 'Peer'은 iPhone이나 AppleWatch같은 다른 애플 제품을 의미하는 것이고, 'Accessory'가 서드파티 제품을 의미하는 것이므로 해당 프로토콜을 사용하자.
- **Class**
  - **NISession**: 두 기기 간 통신을 담당하는 세션 (한 쌍의 기기당 한 개의 세션 ⭐)
    - __init__()
      - 이후 delegate를 self로 지정해줘야 함
      - run(configuration)도 해줘야 함
    - pause()

        - invalidate()
    - **NINearbyAccessoryConfiguration**: iPhone과 서드파티 악세서리의 통신을 가능하게 해주는 configuration
    - **NINearbyObject**: discoveryToken, distance, direction의 데이터 저장
    - **NIDiscoveryToken**: session에 포함되어 실행 중인 기기를 구분하는 identifier
- Protocol **NISessionDelegate**
    - session(NISession, didGenerateShareableConfigurationData: Data, for: NINearbyObject)
    - session(NISession, didUpdate: [NINearbyObject])
    - session(NISession, didRemove: [NINearbyObject], reason: NINearbyObject.RemovalReason)
    - sessionWasSuspended(NISession) (백그라운드)
    - sessionSuspensionEnded(NISession) (백그라운드 → 포그라운드)
    - session(NISession, didInvalidateWith: Error)

Data Communication Channel

- **BluetoothSupport/BluetoothLECentral.swift** 파일 (프레임워크가 존재하지 않음)
    - anchor와 iPhone 사이의 통신 담당
    - **Core Bluetooth** Framework를 UWB에 맞게 수정해서 사용하는 것

> Apple Developer Documentation
> https://developer.apple.com/documentation/corebluetooth/

- Class
    - **DataCommunicationChannel**
        - override init()
        - start()
        - setAccessoryName(accessoryName: String)
        - sendData(data: Data)
    - CBCentralManager
    - CBPeripheral
    - CBCharacteristic

  ⇒ **DataCommunicationChannel**이라는 NSObject가 CBCentral의 역할을 대신함
- Protocol
    - **(AccessoryDemoViewController.swift에서 delegate function일일히 연결)**
        - accessoryConnected - accessoryConnectedHandler
        - accessoryDisconnected - accessoryDisconnectedHandler
        - accessorySharedData - accessoryDataHandler
    - CBCentralManagerDelegate
        - accessoryConnectedHandler
        - accessoryDisconnectedHandler
    - CBPeripheralDelegate
        - accessoryDataHandler

🚩 Understanding data interaction process (통신 과정)

How to test UWB (3) Single point로 테스트를 진행하면, iPhone과 anchor가 통신하는 과정을 Xcode에서 log로 확인 가능하다.

통신 과정을 이해하고 있어야 이후 코드 수정 시 오류를 방지할 수 있다.

blah blah

📌 Fix sample code to work for multiple anchors

🔺 Implementing Spatial Interactions with Third-Party Accessories

Apple Developer Documentation

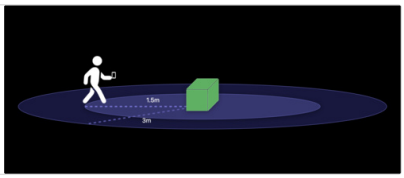
https://developer.apple.com/documentation/nearbyinteraction/implementing_spatial_interactions_with_third-party_accessories

앞에서 사용했던 이 샘플 코드는 앵커와 아이폰 간 일대일 통신만 가능한 형태로 구성되어 있다.

Explore Nearby Interaction with third-party accessories - WWDC21 - Videos - Apple Developer

Discover how your app can interact with Ultra Wideband (UWB) third-party accessories when running on a U1-equipped device. We'll show you...

https://developer.apple.com/videos/play/wwdc2021/10165/



WWDC21에서 복수의 Nearby Interaction 인스턴스를 이용해 동시에 여러 개의 서드파티 악세서리에 연결 가능함을 언급한다. 샘플 코드를 수정하여 여러 앵커가 동시에 인식되도록 해 보자.

multiple anchors 연결 가능하게 코드 수정

NI(Nearby Interaction) 인스턴스 하나 당 **DC(Data Channel)** 인스턴스 하나가 연결된다.

따라서, AccessoryDemoViewController.swift에서 NISession 인스턴스를 여러 개 생성하고, 독립적으로 돌아가도록 코드를 수정해야 한다.

BluetoothLECentral.swift 파일은 수정하지 않는다.

```
var dataChannels = [DataCommunicationChannel]()
var sessions = [String: NISession]()
var configurations = [String: NINearbyAccessoryConfiguration]()
```

dataChannel, session, configuration이 여러 개 필요하므로 각각 array 혹은 dictionary를 만들어 저장한다.

```
var discoveredAccessories = Set<String>()
// 이미 검색된 이름은 추가로 search 하지 않도록 저장
var accessoriesConnected = [String: Bool]()
// connect된 accessory 표시
var sessionStaratedAccessroies = [String: Bool]()
// session이 시작된 accessory 표시
```

여기서는 iPhone과 anchor이 연결되는 과정에 대한 이해가 필요하다.

데이터 수집 기능 추가

- iOS용 CRC 앱을 만들 때 썼던 기능 그대로 가지고 옴 (Timer 사용)
- 1초에 3~5번까지 기록 가능
- anchor를 여러개 연결하는 것과는 무관하므로, 이 이상언급하지 않는다.