

# 嵌入式系統設計

## Embedded System Design

### Lab 6: iBeacon

蕭安紘助教製作

#### 一. 實驗目的

了解如何在自己的App內接收到iBeacon的訊號

#### 二. 實驗需求

環境：macOS 13.4 或以上

IDE：Xcode 14.0 或以上

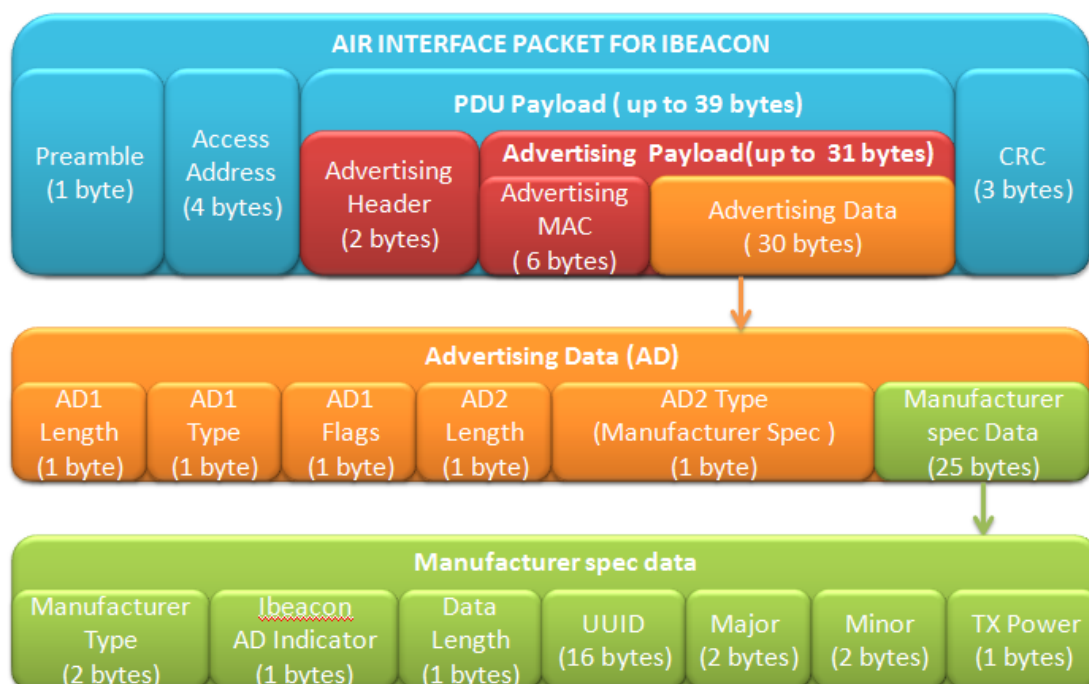
語言：Swift 5

#### 三. iBeacon Concept

iBeacon為基於蘋果提出的基於低耗工藍牙技術(Bluetooth Low Energy, BLE)的室內定位系統協定。利用BLE的廣播協定，iBeacon會將其自身帶有的資訊帶在broadcast的封包中廣播，使可以接收iBeacon的裝置能在不用藍牙連線的清況下收到iBeacon的資訊，估測出裝置與iBeacon大致上的相對位置，藉以做出對應的行動定位服務(Location Based Service, LBS)。

##### 1. iBeacon Broadcast Packet

iBeacon 裡用修改BLE broadcast 封包的內容來將其自身帶有的資訊廣播給附近的device知道。其廣播封包架構如下圖



來源：[smlie-blog.blogspot.tw/2014/06/bluetooth-ibeacon-packet-format.html](http://smlie-blog.blogspot.tw/2014/06/bluetooth-ibeacon-packet-format.html)

其中，藍色部分為原本BLE定義的廣播封包格式，而iBeacon則是利用修改其PDU (Protocol Data Unit) Payload的內容來廣播其用來定位的資訊，其中包含四項(見綠色部分)：UUID、Major、Minor、TX Power。

- (1) **UUID**：通常用來識別不同廠商、用途、目的的唯一識別碼，例如，同一個大賣場所有的iBeacon UUID都是一樣的，但不同賣場的UUID就會不一樣。程式碼裡為 Proximity UUID
- (2) **Major**：通常用來定義一個比較大範圍的一群iBeacon，例如賣場的生鮮食品區所有的iBeacon都是同樣的Major，家電區的iBeacon則是用另一組Major
- (3) **Minor**：定義更小範圍的iBeacon，通常我們會拿Minor直接來區分不同顆iBeacon裝置
- (4) **TX Power**：定義為在一公尺的距離接收這顆 iBeacon 發出來的訊號有多強。這個數值代表了iBeacon發出的訊號強度，收到iBeacon訊號的裝置可以藉由「實際收到的訊號強度」與「發出的訊號強度」做比對，推算大致上的相對距離。**TX Power是需要自行校正的**，請同學注意！可以透過 USBeaconWriter App校正。

當裝置接收到 iBeacon 的廣播資訊後，其系統內會根據這些數值估算與iBeacon相對的位置，並且包裝成較實用的功能，例如：是否離開/進入某個區域等等，提供給App開發者作為LBS的用途。

## 2. Monitoring

在Apple的iBeacon定義中，monitoring用來偵測device是否進入/離開一個指定的區域(Region)，一個Region是由一個或多個iBeacon組成的，我們可以使用UUID，Major或Minor來定義出不同大小的region。如果app開始monitor某個region，則當系統認為device離開或進入到這個region時，app就會收到通知(function call)。

## 3. Ranging

在Apple的iBeacon定義中，ranging用來偵測某個region中可以收到的iBeacon訊號，如果app開始ranging某個region，則當device收到這個region中定義的iBeacon的訊號，app就會收到通知(function call)，並且可以列出所有收到的iBeacon的詳細資訊，如：訊號強度(RSSI)、大約的相對位置(Proximity)、相對位置的準確度(Accuracy)等

- (1) Proximity：device跟iBeacon的相對距離，只有四種：unknown、immediate、near、far

The relative distance to the beacon.

The value in this property gives a general sense of the relative distance to the beacon. Use it to quickly identify beacons that are nearer to the user rather than farther away.

- (2) Accuracy：估計proximity的準確度，可以想成proximity估計錯誤的standard deviation。不是device與iBeacon的距離！

The accuracy of the proximity value, measured in meters from the beacon. Indicates the one sigma horizontal accuracy in meters. Use this property to differentiate between beacons with the same proximity value. Do not use it to identify a precise location for the beacon. Accuracy values may fluctuate due to RF interference.

A negative value in this property signifies that the actual accuracy could not be determined.

- (3) rssi：收到iBeacon封包的訊號強度。

The received signal strength of the beacon, measured in decibels.

This value is the average RSSI value of the samples received since the range of the beacon was last reported to your app.

我們將在下面的實驗步驟中實作 iBeacon 的 monitoring 與 ranging。

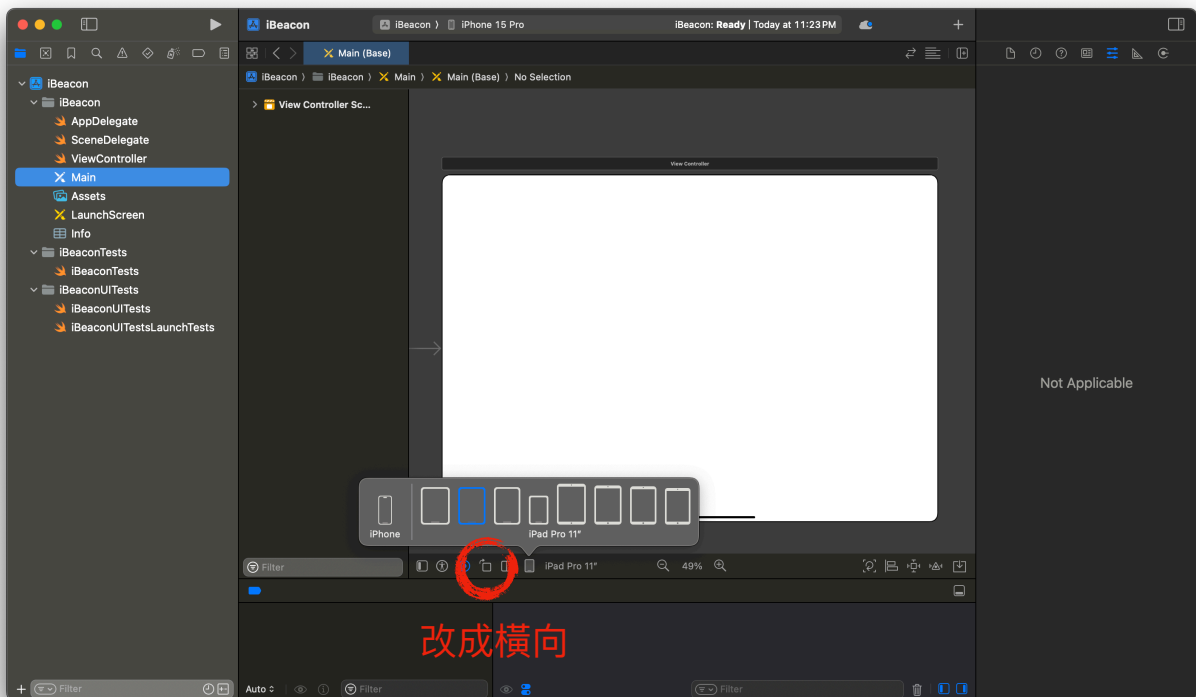
## 四. 實驗步驟

### 1. 建立 iOS 專案 - iBeacon

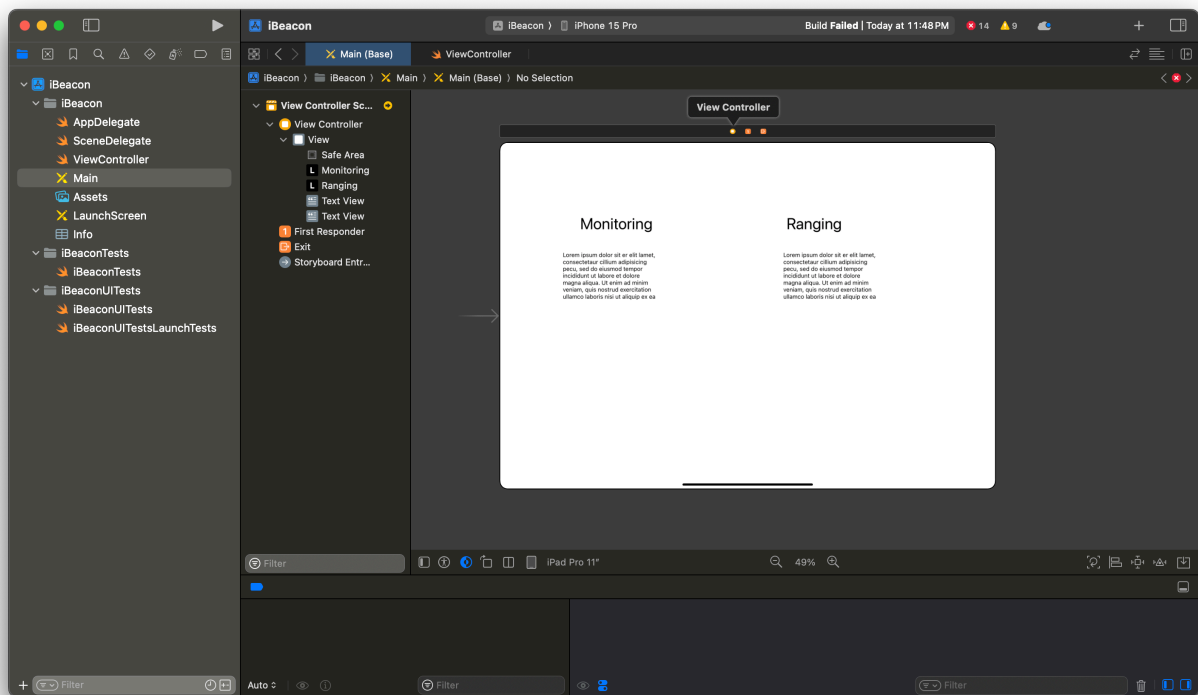
- (1) 開啟Xcode，左上角menubar，選擇 File -> New -> Project
- (2) 左方選擇 iOS 內的 Application，右方選擇 Single View Application，按 Next
- (3) Product Name 輸入 **iBeacon**  
Organization identifier 輸入 nycu.esd.”你的學號”  
Language 選擇 Swift  
按下 Next，存在桌面

### 2. 建立顯示iBeacon狀態的UI介面

- (1) 點選 Main頁面，在下方改成橫向的iPad介面(根據您所使用的iPad型號)



(2) 從右下方 Object Library 拉入2個label以及2個textview，用來顯示 monitoring 的

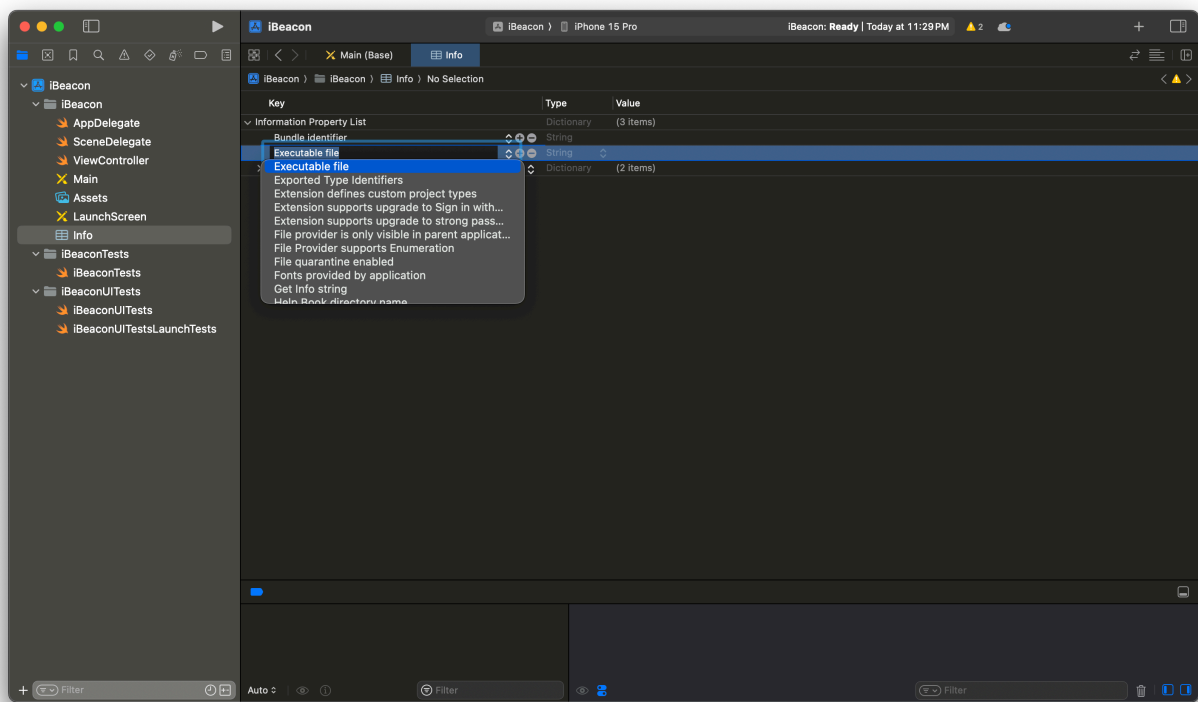


結果以及 ranging 的結果，並且將兩個 textview 拉 outlet 到 ViewController.swift

```
@IBOutlet weak var monitorResultTextView: UITextView!  
@IBOutlet weak var rangingResultTextView: UITextView!
```

### 3. 要求 Core Location 的權限

- (1) 由於 iBeacon 是屬於定位相關的功能，定義在 Core Location Service 中，所以我們要先讓 App 擁有取得使用者位置的權限 (與使用 GPS 一樣)
- (2) 點選左方 Project Navigator 中的 Info.plist，找到第一行的 Information Property List，點選它旁邊的 +



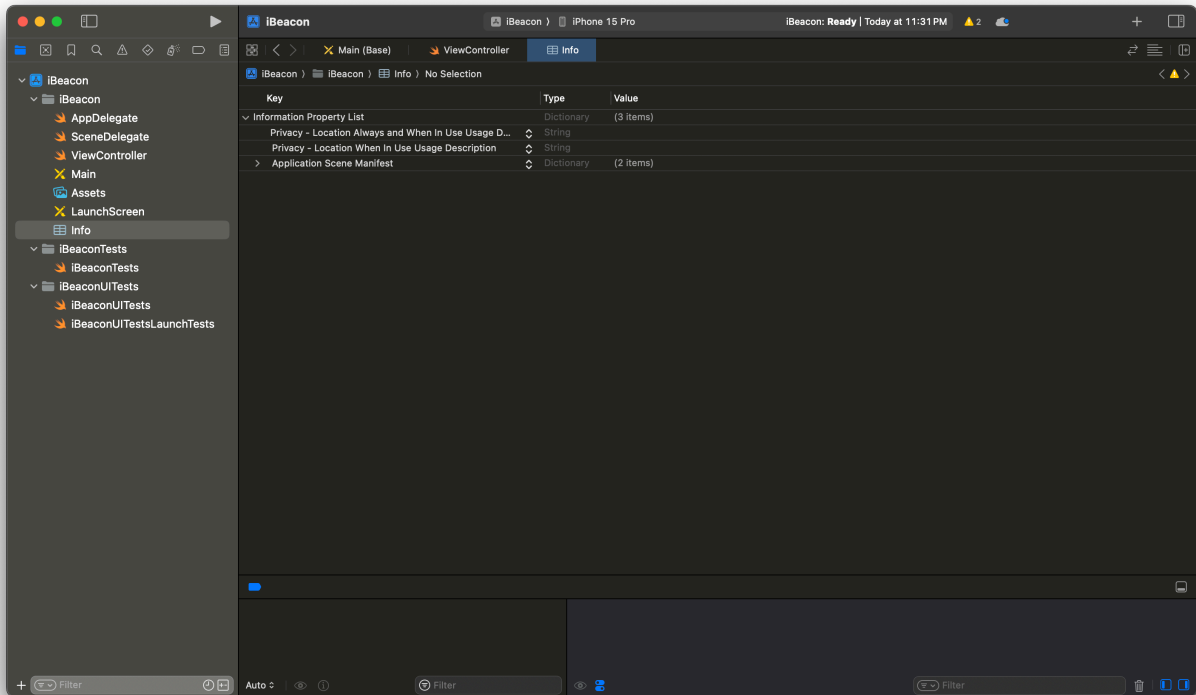
- (3) 在下拉的選單中找到 "Privacy - Location When In Use Usage Description"、"Privacy - Location Always and When In Use Usage Description" 這個2個Key加入，並且在後面的 Value 都填上 "這個App需要存取你的位置"

- (4) 在 ViewController.swift 檔案的最上方，加入 Core Location Framework

```
import CoreLocation
```

- (5) 在 ViewController Class 中新增 CLLocationManager 的 property，CLLocationManager 是 iOS 提供給開發者取用 Location Service 的 Class

```
var locationManager: CLLocationManager = CLLocationManager()
```



- (6) 在 `viewDidLoad()` method 中，用 `location manager` 向使用者要求 `location service` 的權限，這樣App打開後，系統將會自動跳出權限要求的 Alert

```
//要求使用者授權 location service
if CLLocationManager.isMonitoringAvailable(for: CLBeaconRegion.self){
    if CLLocationManager.authorizationStatus() !=
    CLAuthorizationStatus.authorizedAlways
    {
        locationManager.requestAlwaysAuthorization()
    }
}
```

#### 4. 建立 iBeacon 群組：Region

- (1) 到 iBeacon App 新增2個 View Controller 的 const property：uuid 以及 identifier，identifier 儲存 beacon region 的名字。

```
let uuid = "A3E1C063-9235-4B25-AA84-D249950AADC4"
let identifier = "esd region"
```

(2) 在 viewDidLoad() 中利用 uuid 建立 beacon region

```
//創造包含同樣 uuid 的 beacon 的 region
let region = CLBeaconRegion(uuid: UUID.init(uuidString: uuid)!,
    identifier: identifier)
```

※我們可使用 uuid + major 或是 uuid + major + minor 來建立範圍更小的 region，以我們的實驗規模，3顆 iBeacon 有1個region就用夠了。

## 5. Monitor Beacon Region

(1) Monitor 的結果會透過 CoreLocationManager 的 delegate 通知app，在 ViewController class 加入 CLLocationManagerDelegate

```
class ViewController: UIViewController, CLLocationManagerDelegate {
```

(2) 接著，在 class 中加入 monitor 相關的 delegate method

```
    func locationManager(_ manager: CLLocationManager, didStartMonitoringFor region:
    CLRegion) {
        }
    func locationManager(_ manager: CLLocationManager, didEnterRegion region:
    CLRegion) {
        }
    func locationManager(_ manager: CLLocationManager, didExitRegion region: CLRegion)
    {
        }
    func locationManager(_ manager: CLLocationManager, didDetermineState state:
    CLRegionState, for region: CLRegion) {
        }
}
```



- (3) 在 viewDidLoad() 設定好 locationManager 的 delegate 後，開始 monitoring 剛剛建立好的 region

```
//設定 locaiton manager 的 delegate
locationManager.delegate = self

//設定region monitoring 要被通知的時機
region.notifyEntryStateOnDisplay = true
region.notifyOnEntry = true
region.notifyOnExit = true

//開始 monitoring
locationManager.startMonitoring(for: region)
```

- (4) 完成 delegate 中的程式碼，讓 monitoring 的結果顯示在螢幕上

```
func locationManager(_ manager: CLLocationManager, didStartMonitoringFor region:
CLRegion) {
    //將成功開始 monitor 的 region 的 identifier 加入到 monitor textview 最上方
    monitorResultTextView.text = "did start monitoring \(region.identifier)\n" +
monitorResultTextView.text
}

func locationManager(_ manager: CLLocationManager, didEnterRegion region: CLRegion) {
    //將偵測到進入 region 的狀態加入到 monitor textview 最上方
    monitorResultTextView.text = "did enter\n" + monitorResultTextView.text
}

func locationManager(_ manager: CLLocationManager, didExitRegion region: CLRegion) {
    //將偵測到離開 region 的狀態加入到 monitor textview 最上方
    monitorResultTextView.text = "did exit\n" + monitorResultTextView.text
}

func locationManager(_ manager: CLLocationManager, didDetermineState state:
CLRegionState, for region: CLRegion) {
    switch state {
    case .inside:
        //將偵測到在 region 內的狀態加入到 monitor textview 最上方
        monitorResultTextView.text = "state inside\n" + monitorResultTextView.text

    case .outside:
        //將偵測到在 region 外的狀態加入到 monitor textview 最上方
        monitorResultTextView.text = "state outside\n" + monitorResultTextView.text

    default:
        break
    }
}
```

## 6. Ranging Beacon Region

- (1) 當 monitor 判斷進入到 region 後，即可叫 location manager 開始 ranging region，同樣，離開 region 後則停止 ranging

```
case .inside:
    //將偵測到在 region 內的狀態加入到 monitor textview 最上方
    monitorResultTextView.text = "state inside\n" +
    monitorResultTextView.text

    //如果 device 支援 ranging iBeacon，開始 ranging 這個 region

    manager.startRangingBeacons(satisfying:
    CLBeaconIdentityConstraint(uuid: UUID.init(uuidString: uuid)!))

case .outside:
    //將偵測到在 region 外的狀態加入到 monitor textview 最上方
    monitorResultTextView.text = "state outside\n" +
    monitorResultTextView.text

    //停止 ranging region
    manager.stopMonitoring(for: region)
```

- (2) 開始 range 後，一旦 device 收到這個 region 包含的 beacon 訊號，便會透過 delegate 通知 app，我們在 view controller 中加入對應的 delegate method，並且將所有收到的 beacon 顯示在螢幕上

```
func locationManager(_ manager: CLLocationManager, didRangeBeacons beacons:
[CLBeacon], in region: CLBeaconRegion) {

    //清空原本的ranging textview
    rangingResultTextView.text = ""

    //iterate 每個收到的 beacon
    for beacon in beacons {


        //根據不同 proximity 常數設定 proximityString
        var proximityString = ""
        switch beacon.proximity {
        case .far:
            proximityString = "far"
        case .near:
            proximityString = "near"
        case .immediate:
            proximityString = "immediate"
        default :
            proximityString = "unknown"
        }

        //把這個beacon的數值放到ranging textview上
        rangingResultTextView.text = rangingResultTextView.text +
            "Major: \(beacon.major)" + " Minor: \(beacon.minor)" +
            " RSSI: \(beacon.rssi)" + " Proximity: \(proximityString)" +
            " Accuracy: \(beacon.accuracy)" + "\n\n";
    }
}
```

### (3) 執行程式，看有沒有收到 iBeacon 的訊號

凌晨 1:57 4月16日 週二

...

51% 

## Monitoring

state inside  
did start monitoring esd region

## Ranging

Major: 1 Minor: 1 RSSI: -38  
Proximity: near Accuracy:  
0.28495096662877534

Major: 1 Minor: 2 RSSI: -53  
Proximity: near Accuracy:  
1.760042634953653

### (4) 我們也利用 swift array 的 order method 排序接收到的 iBeacon

```
//清空原本的ranging textview
rangingResultTextView.text = ""

//根據rssi大小排序，rssi大的排在前面
let orderedBeaconArray = beacons.sorted(by: { (b1, b2) -> Bool in
    return b1.rssi > b2.rssi
})

//iterate 每個收到的 beacon
for beacon in orderedBeaconArray {
    ...
}
```