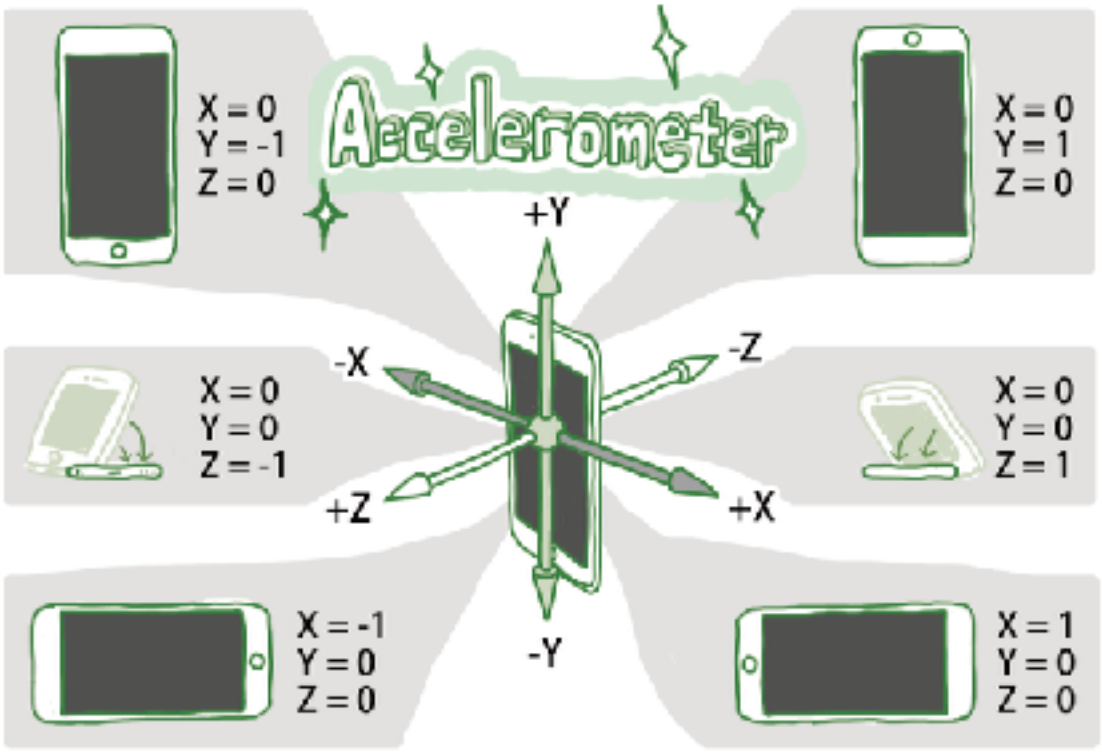
CHAPTER 感測器

使用加速度感測器



[問題]如何使用手機上的加速度感測器？
[解答]使用CoreMotion函式庫的CMMotionManager來蒐集加速度感測器的資料
[範例程式碼]HelloAccelerometer
[過程解說]
1.先匯入CoreMotion函式庫。使用CoreMotion函式庫中的CMMotionManager類別，產生出CMMotionManager類別的物件。
2.使用isAccelerometerAvailable屬性檢查是否可以蒐集到加速度感測器的資料，如果可以的話，就使用startAccelerometerUpdates方法，得到加速度感測器的資料。請參考下面的程式碼：

```
import UIKit
import CoreMotion                                          //匯入CoreMotion

class ViewController: UIViewController {

    let motionManager = CMMotionManager()
    override func viewDidLoad() {
        super.viewDidLoad()
        if motionManager.isAccelerometerAvailable{      //檢查是否可以用
            let operationQ = OperationQueue()
            motionManager.startAccelerometerUpdates(to: operationQ,
                withHandler: {
                (data:CMAccelerometerData?, error:Error?) in
                print("X: \(data!.acceleration.x)")
                print("Y: \(data!.acceleration.y)")
                print("Z: \(data!.acceleration.z)")    //拿到加速度器資料
            })
        }
    }

    override func viewDidDisappear(_ animated: Bool) {
        motionManager.stopAccelerometerUpdates()  //停止更新加速度器資料
    }
}
```

startAccelerometerUpdates方法需要兩個參數，第一個是執行加速度感測器相關程式碼的工作佇列，第二個則是每次蒐集到加速度感測器資料後，想要執行的程式碼。

第一個參數代入用OperationQueue方法得到的工作佇列。第二個參數的話，則是在Closure裡面寫進相關的程式碼。範例中是把x、y、z三個方向的加速度數值印出來。從CMAccelerometerData裡面的acceleration屬性，就可以知道x、y、z三個方向的加速度數值。

最後不要忘記在離開程式或離開畫面、不需要獲知加速度感測器的數值時，使用stopAccelerometerUpdates方法，停止獲取相關資訊。

加速度感測器的程式碼無法用模擬器模擬、要用實機偵測。三個方向會得到的數值，分別是0~1的雙精度浮點數(Double)。請參考本問題最開始的插圖：直著擺放x值是0、y方向是正1，z方向是0；180度倒轉之後，y方向的值會是正1...。從x、y、z值的改變，就可以知道手機的位置變化。

本範例程式碼中，進一步地把三個方向的數值顯示在畫面上，歡迎參考。

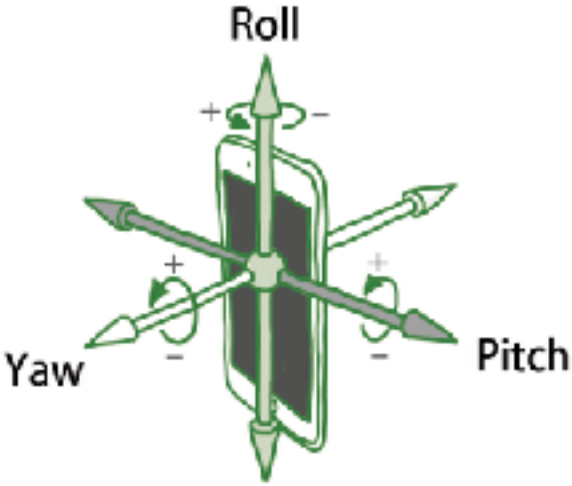
使用陀螺儀得知角速度

[問題]如何使用手機上的陀螺儀
[解答]使用CoreMotion函式庫的CMMotionManager來蒐集陀螺儀的資料
[範例程式碼]HelloGyroscope
[過程解說]
1.先匯入CoreMotion函式庫。使用CoreMotion函式庫中的CMMotionManager類別，產生出CMMotionManager類別的物件。
2.使用isGyroAvailable屬性檢查是否可以蒐集到陀螺儀的資料。如果可以的話，就使用startGyroUpdates方法，得到陀螺儀的資料。請參考下面的程式碼：

```
import UIKit
import CoreMotion

class ViewController: UIViewController {
    var motionManager = CMMotionManager()
    override func viewDidLoad() {
        super.viewDidLoad()
        if motionManager.isGyroAvailable{          //檢查陀螺儀是否可用
            let operationQ = OperationQueue()
            motionManager.startGyroUpdates(to: operationQ,
                                                   withHandler: {
                (data:CMGyroData?, error:Error?) in
                print("X:\(data!.rotationRate.x)")
                print("Y:\(data!.rotationRate.y)")
                print("Z:\(data!.rotationRate.z)") //拿到陀螺儀資料
            })
        }
    }

    override func viewDidDisappear(_ animated: Bool) {
        motionManager.stopGyroUpdates()      //停止更新陀螺儀資料
    }
}
```

startGyroUpdates方法需要兩個參數，第一個是執行陀螺儀相關程式碼的工作佇列，第二個則是每次蒐集到陀螺儀資料後，想要執行的程式碼。
第一個參數代入用OperationQueue方法得到的工作佇列。第二個參數的話，則是在Closure裡面寫進相關的程式碼。範例中是把x、y、z三個方向的陀螺儀數值印出來。從CMGyroData裡面的rotationRate屬性，就可以知道x、y、z三個方向的陀螺儀數值。
最後不要忘記在離開程式或離開畫面、不需要獲知陀螺儀的數值時，使用stopGyroUpdates方法，停止獲取相關資訊。
陀螺儀的程式碼無法用模擬器模擬、要用實機偵測。測得的速度是三個方向轉動的角速度。轉動的過程中數值會改變，停止轉動的話，就會發現數值維持穩定。
本範例程式碼中，進一步地把三個方向的數值顯示在畫面上，歡迎參考。

得知目前手機在各個方向轉動的角度



[問題]如何得知手機在各個方向轉動的角度
[解答]使用CoreMotion函式庫的CMMotionManager來蒐集DeviceMotion的資料。從DeviceMotion資料中，Attitude的Pitch、Roll，以及Yaw數值，就可以判別手機在各個方向的旋轉角度。
[範例程式碼]HelloAttitude
[過程解說]
1.先匯入CoreMotion函式庫。使用CoreMotion函式庫中的CMMotionManager類別，產生出CMMotionManager類別的物件。
2.使用isDeviceMotionAvailable屬性檢查是否可以蒐集到DeviceMotion資料，如果可以的話，就使用startDeviceMotionUpdates方法，得到DeviceMotion資料。請參考下面的程式碼：

```swift
import UIKit
import CoreMotion

class ViewController: UIViewController {
    let motionManager = CMMotionManager()

    override func viewDidLoad() {
        super.viewDidLoad()
        if motionManager.isDeviceMotionAvailable{//可否取得DeviceMotion
            let operationQ = OperationQueue()
            motionManager.startDeviceMotionUpdates(to: operationQ,
                withHandler: {
                (motion:CMDeviceMotion?, error:Error?) in
                print("Pitch:\(motion!.attitude.pitch)")
                print("Roll:\(motion!.attitude.roll)")
                print("Yaw:\(motion!.attitude.yaw)") //拿到各個方向的轉向
            })
        }
    }
    override func viewDidDisappear(_ animated: Bool) {
        motionManager.stopDeviceMotionUpdates() //停止更新DeviceMotion
    }
}
```

startDeviceMotionUpdates方法需要兩個參數，第一個是執行和DeviceMotion相關程式碼的工作佇列，第二個則是每次蒐集到DeviceMotion資料後，想要執行的程式碼。

第一個參數代入用OperationQueue方法得到的工作佇列。第二個參數的話，則是在Closure裡面寫進相關的程式碼。範例中是把x、y、z三個方向的轉向印出來。從CMDeviceMotion裡面的attitude屬性，就可以知道x、y、z三個方向的轉向。

如本題插圖，這三個方向的轉向，在iOS中分別叫做Pitch、Roll，與Yaw。Pitch是x方向的轉向、Roll是y方向的轉向，Yaw則是z方向的轉向。程式碼中印出來的數值，是每個方向轉向的徑度。實際打開程式碼，會看到如何把這些Double型別的徑度轉換成角度，並且顯示在螢幕的方法。

最後不要忘記在離開程式或離開畫面、不需要獲知DeviceMotion的數值時，使用stopDeviceMotionUpdates方法，停止繼續獲取相關資訊。

跟前幾個範例一樣，陀螺儀的程式碼也無法用模擬器模擬，必須要用實機偵測三個方向的轉動狀態。本範例程式碼中，進一步地把三個方向的數值顯示在畫面上，歡迎參考。

如何偵測手機搖晃

[問題]如何偵測手機搖晃事件
[解答]覆寫motionBegan或是motionEnded方法，判斷是否偵測到手機搖晃。
[範例程式碼]HelloMotionShake
[過程解說]
要偵測手機搖晃的事件很簡單，只要覆寫motionBegan或是motionEnded方法，在其中判斷是否偵測到手機搖晃就可以了。差別是一開始搖晃手機時，會觸發motionBegan方法中的程式碼；搖晃事件結束的時候，才會觸發motionEnded的程式碼。下面的程式碼以覆寫motionEnded方法為例，如果要覆寫motionBegan方法，裡面的程式碼也會和覆寫motionEnded方法一樣：

```
override func motionEnded(_ motion: UIEventSubtype,
                                                with event: UIEvent?) {
    if event?.subtype == .motionShake{ //如果偵測到的事件是搖晃手機的話
        print("Shaking now")
    }
}
```

在偵測到手機有動作之後，會觸發moitonEnded方法。判斷如果偵測到的事件是手機搖晃(MotionShake)的話，範例中就印出「Shaking now」。
執行本範例的程式碼，可以用模擬器模擬手機搖晃。請選模擬器上面工具列中[Hardware] > [Shake Gesture]即可。
本範例的完整程式碼中，在搖晃手機後會秀出警告視窗。並且在註釋的程式碼中，也列出了覆寫motionBegan方法。請開啟參考。

如何使用距離感應器偵測物體靠近手機

[問題]如何偵測物體靠近手機
[解答]使用手機上的距離感應器(Proximity Sensor)
[範例程式碼]HelloProximitySensor
[過程解說]
手機上方靠近聽筒的地方，有一顆小小圓圓的距離感應器(Proximity Sensor)。當講電話把手機靠近耳朵的時候，距離感應器會偵測到耳朵接近、把螢幕畫面關暗。讓使用者在通話的過程中，可以有良好的使用經驗。這樣偵測物體靠近距離感應器的事件，也可以用程式碼得知：

```
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        //先把距離感應器的偵測狀態設為開啟
        UIDevice.current.isProximityMonitoringEnabled = true
        if UIDevice.current.isProximityMonitoringEnabled == true{
            //如果距離感應器的狀態真的是開啟，代表運行的機器上有安裝距離感應器...
            NotificationCenter.default.addObserver(self,
                    selector: #selector(ViewController.
                        proximitySensorStateChange(notification:)),
                    name: NSNotification.Name.
                        UIDeviceProximityStateDidChange,
                    object: nil)
        }
    }

    func proximitySensorStateChange(notification:Notification){
        print("something approching or leaving")
    }
}
```

程式碼中要使用距離感應器的話，請先把距離感應器的偵測狀態設為開啟。接下來判斷距離感應器的偵測狀態是否真的為開啟中。如果狀態真的是開啟中，代表運行程式的機器上有安裝距離感應器。此時設定如果發生「距離感應器狀態變化」，就執行proximitySensorStateChange方法。範例中，就印出了「something approaching or leaving」，表明有東西靠近或是遠離距離感應器。
當有東西靠近手機觸發事件時，手機的螢幕會暗掉。目前沒有方法避免這種預

設行為。

如何偵測磁力

[問題]如何偵測磁力
[解答]使用CoreMotion函式庫的CMMotionManager來蒐集磁力資料。
[範例程式碼]HelloMagnetometer
[過程解說]
1.先匯入CoreMotion函式庫。使用CoreMotion函式庫中的CMMotionManager類別，產生出CMMotionManager類別的物件。
2.使用isMagnetometerAvailable屬性檢查是否可以蒐集到磁力資料，如果可以的話，就使用startMagnetometerUpdatesToQueue方法，得到DeviceMotion資料。請參考下面的程式碼：

```
import UIKit
import CoreMotion

class ViewController: UIViewController {
    var motionManager = CMMotionManager()
    override func viewDidLoad() {
        super.viewDidLoad()
        if motionManager.isMagnetometerAvailable{     //是否可取得磁力資料
            let operationQ = OperationQueue()
            motionManager.startMagnetometerUpdates(to: operationQ,
                withHandler: { (
                data:CMMagnetometerData?, error:Error?) in
                print("X:\(data!.magneticField.x)")
                print("Y:\(data!.magneticField.y)")
                print("Z:\(data!.magneticField.z)") //印出各方向的磁力
            })
        }
    }

    override func viewDidDisappear(_ animated: Bool) {
        motionManager.stopMagnetometerUpdates()      //停止更新磁力資料
    }
}
```

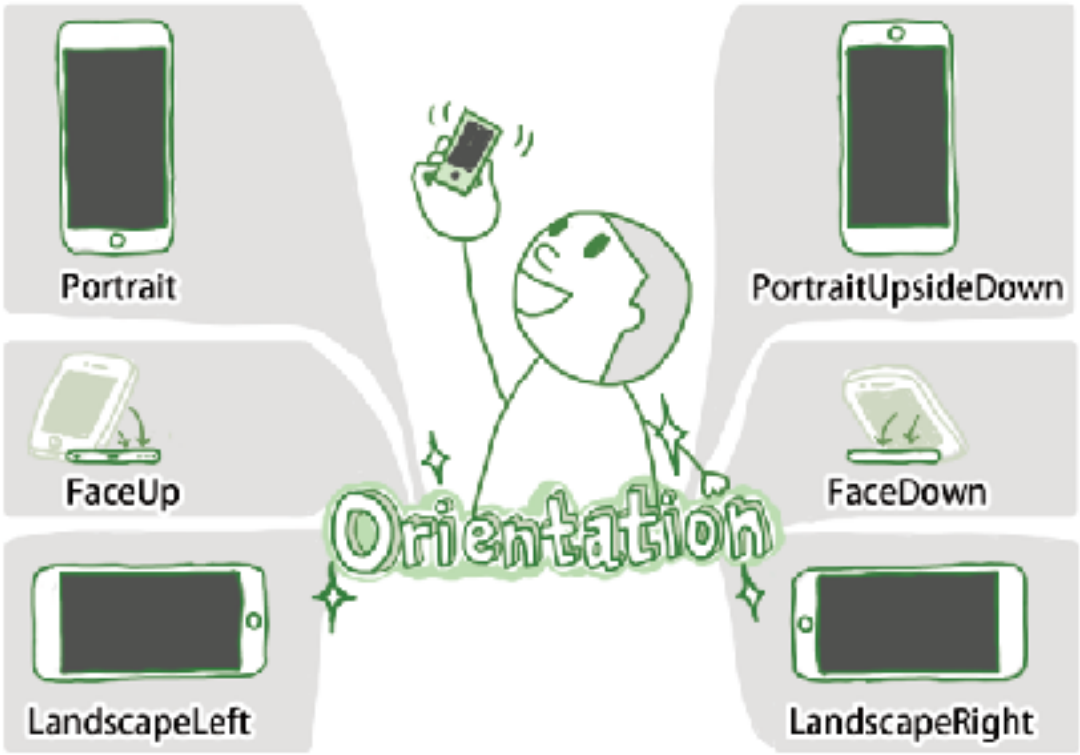
startMagnetometerUpdates方法需要兩個參數，第一個是執行和磁力相關程式碼的工作佇列，第二個則是每次蒐集到磁力資料後，想要執行的程式碼。
第一個參數代入用OperationQueue方法得到的工作佇列。第二個參數的話，則是在Closure裡面寫進相關的程式碼。範例中是把x、y、z三個方向的轉向印出來。從CMDeviceMotion裡面的magneticField屬性，就可以知道x、y、z三個方向的磁力資料。
最後不要忘記在離開程式或離開畫面、不需要獲知磁力數值時，使用stopMagnetometerUpdates方法，停止獲取相關資訊。
偵測磁力的程式碼無法用模擬器模擬，必須要用實機偵測三個方向的磁力狀態。本範例程式碼在實行時，還會進一步地把三個方向的數值顯示在畫面上，歡迎參考。

如何偵測手機轉向

[問題]如何偵測手機轉向
[解答]設定Notification的方式，就可以接收到手機轉向的事件通知。
[範例程式碼]HelloOrientationChange
[過程解說]
只要設定接收手機轉向的事件通知，之後即可以在手機轉向時，在設定的方法中做出反應。請參考下面的程式碼：

```swift
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        NotificationCenter.default.addObserver(self,
            selector: #selector(ViewController.orientationChanged
                (notification:)),
            name: NSNotification.Name.UIDeviceOrientationDidChange,
            object: nil)//設定手機轉向會執行orientationChange
    }

    func orientationChanged(notification:Notification){
        //發生轉向後，用下面的程式碼得到目前手機的轉向
        let orientation = UIDevice.current.orientation
        var orientationResult:String
        switch orientation{          //判斷是哪一種轉向，一共有六種
            case .portrait:
                orientationResult = "Portait"
            case .portraitUpsideDown:
                orientationResult = "PortaitUpsideDown"
            case .landscapeLeft:
                orientationResult = "LandscapeLeft"
            case .landscapeRight:
                orientationResult = "LandscapeRight"
            case .faceUp:
                orientationResult = "FaceUp"
            case .faceDown:
                orientationResult = "FaceDown"
            default:
                orientationResult = "Unknown Direction"
        }
        print("orientation changed: " + orientationResult)
    }
}
```

範例程式碼中，先在ViewDidLoad方法裡設定接收手機轉向的事件通知。之後每次手機轉向，就會觸發orientationChanged方法。
在orientationChanged方法，只是印出來手機轉向的情況。一共有六種，包括「Portait」、「PortaitUpsideDown」、「LandscapeLeft」、「LandscapeRight」、「FaceUp」，與「FaceDown」。如果打開隨書提供完整的程式碼，在轉向的過程中，畫面上的文字標籤會隨著不同的轉向顯示當前手機的轉向狀態。



[橫向和直向顯示完全不同畫面的方法]
iOS8之後的程式畫面，在面對轉向時，會用Autolayout與Size Classes來設定不同轉向的顯示方式。不過如果您的應用程式在橫向和直向要顯示完全不同的畫面，則可以利用本範例的方法，得到轉向的通知。分別準備橫向和直向不同的兩個畫面，在觸發相關的程式碼時，依據不同的轉向，顯示不同的View Controller畫面。

如何使用電子羅盤 (製作指北針)

[問題]如何使用電子羅盤得到方位
[解答]使用CoreLocation函式庫中的CLLocationManager更新電子羅盤資料，並在CLLocationManagerDelegate方法中得到目前的方位
[範例程式碼]HelloHeading
[過程解說]
要使用電子羅盤得到目前方位的話，要引入CoreLocation函式庫，使用其中的

CLLocationManager類別。請參考下面的程式碼：

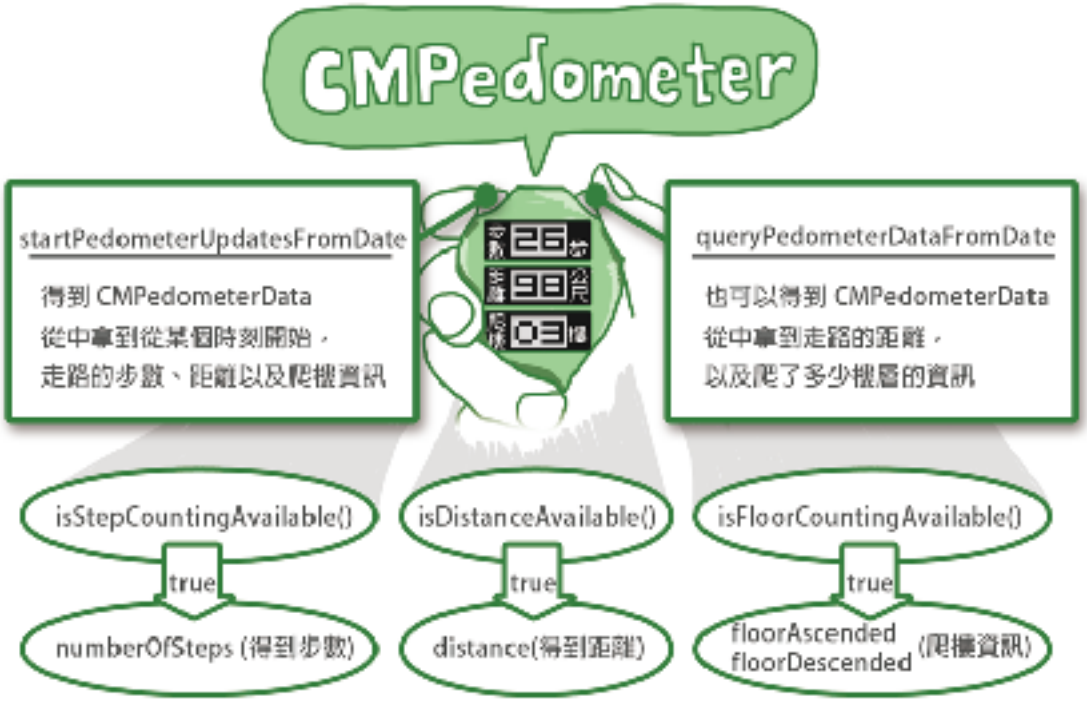
```
import UIKit
import CoreLocation

class ViewController: UIViewController,CLLocationManagerDelegate {

    @IBOutlet weak var indicator: UIImageView!
    var locationManager:CLLocationManager!      //指北針的指針

    override func viewDidLoad() {
        super.viewDidLoad()
        locationManager = CLLocationManager()  //生成CLLocationManager
        locationManager.delegate = self //設定ViewController為delegate
        if CLLocationManager.headingAvailable(){ //如果可以使用電子羅盤..
            locationManager.startUpdatingHeading() //開始更新電子羅盤資料
        }
    }

    func locationManager(_ manager: CLLocationManager,
                         didUpdateHeading newHeading: CLHeading) {
        if newHeading.headingAccuracy >= 0{     //如果小於0，資料無法使用
            print("地磁北方: \(newHeading.magneticHeading)")
            print("地理北方: \(newHeading.trueHeading)")

            //下面為製作指北針的程式碼
            //求出旋轉徑度
            let headingInRadian = -1.0 * Double(M_PI) * newHeading.
                magneticHeading / 180.0
            //讓指針轉向
            indicator.transform = CGAffineTransform(rotationAngle:
                                                    CGFloat(headingInRadian))
        }
    }
}
```

產生出CLLocationManager之後，把這個型別的物件存在變數locationManager中。設定ViewController類別是locationManager的delegate屬性之後，確定可以使用電子羅盤(headingAvailable)，就開始更新電子羅盤資料。
讓ViewController類別服從CLLocationManagerDelegate這個Protocol，電子羅盤的資料有變化的話，就會呼叫locationManager didUpdateHeading方法。
在這個方法裡，先判斷傳進來的參數newHeading的headingAccuracy是否小於0，如果小於0，代表目前手機受到外界磁力影響，無法正確判斷方位。如果大於或等於0的話，就可以用傳進參數newHeading的magneticHeading得到地磁北方的角度；也可以用傳進參數newHeading的trueheading得到地理北方的角度。範例中只是把這兩個數值印出來。

[製作指北針]
利用上面的程式碼，可以做出指北針。範例裡在畫面加入了一張UIImageView

顯示指北針的指針，存在常數indicator中。當電子羅盤方位調整時，就把傳進參數newHeading的magneticHeading這個角度，轉換成徑度。因為此徑度和指針旋轉的角度是相反的，所以乘上-1.0之後，就是指針要旋轉的角度。

如何使用計步器



[問題]如何使用手機上的計步器
[解答]使用CoreMotion函式庫中的CMPedometer類別，可以得到走路的步數、走過的距離，與上下樓層數。
[範例程式碼]HelloPedometer
[過程解說]
請參考下面的程式碼：

```swift
import UIKit
import CoreMotion

class ViewController: UIViewController {

    @IBOutlet weak var stepLabel: UILabel!
    @IBOutlet weak var distanceLabel: UILabel!
    @IBOutlet weak var ascendLabel: UILabel!
    @IBOutlet weak var descendLabel: UILabel!

    var pedometer:CMPedometer!

    override func viewDidLoad() {
        super.viewDidLoad()
        pedometer = CMPedometer()   //產生 CMPedometer 計步器
        if CMPedometer.isStepCountingAvailable() &&  //如果能夠計步
            CMPedometer.isDistanceAvailable() &&     //如果可以紀錄距離
            CMPedometer.isFloorCountingAvailable() { //如果可以紀錄爬樓
            pedometer.startUpdates(from: Date(), withHandler: { (data:
                CMPedometerData?, error:Error?) -> Void in
                print("Step: \(data!.numberOfSteps)")   //印出步數
                print("Distance: \(data!.distance)")    //印出走路距離
                print("Floor Ascend: \(data!.floorsAscended)")  //上樓
                print("Floor Descend: \(data!.floorsDescended)")//下樓
                //下面程式碼更新畫面上的文字，要在 main queue 更改
                DispatchQueue.main.async(execute: {
                    self.stepLabel.text =
                                    "Step: \(data!.numberOfSteps)"
                    self.distanceLabel.text = "Distance: \
                        (Int(data!.distance!))"
                    self.ascendLabel.text = "FloorAscend: \(data!.
                        floorsAscended!)"
                    self.descendLabel.text = "FloorDescend: \(data!.
                        floorsDescended!)"
                })
            })

            pedometer.queryPedometerData(from:
               Date(timeIntervalSinceNow: -24 * 60 * 60), to: Date(),
               withHandler: {
                (data:CMPedometerData?, error:Error?) -> Void in
                print("Step(YesterdayTillNow):\(data!.numberOfSteps)")
                print("Distance(YesterdayTillNow):\(data!.distance)")
                print("Ascend(FromYesterday):\(data!.floorsAscended)")
            print("Descend(FromYesterday): \(data!.floorsDescended)")
                })
        }
    }

    override func viewDidDisappear(_ animated: Bool) {
        pedometer.stopUpdates()
    }
}
```

1.先引入CoreMotion函式庫。
2.在需要使用計步功能的時候，產生CMPedometer類別的物件來計步。
3.先判斷是否能夠計步、是否能夠記錄走路距離，以及是否能夠記錄爬樓距

離。在這個範例由於需要得知這三種資訊，所以同時判斷這三種情況。如果您的應用程式只要記錄步數、或是只要記錄走路距離等的話，在這邊就只需要判斷其中需要的情況。
4.如果步驟3的判斷成立的話，就使用CMPedometer類別中startPedometerUpdates方法，開始記錄步數、走路距離，以及上下樓的資訊。
這個方法接受兩個參數：第一個參數是記錄的時間點。範例中是代入目前的時間；第二個參數是收到相關資訊之後的處理流程、把處理流程的程式碼包在一個Closure裡面。
5.範例印出了步數、走路距離，以及上下樓的資訊。可以用CMPedometerData裡的numberOfSteps得到走路的步數；用其中的distance得到走路的距離；用floorAscended得到上樓的樓層數；用floorDescended得到下樓的樓層數。除此以外，還可以用currentPace得到目前走路的速度是每秒走了幾公尺，currentCadence得到目前走路的快慢是每秒走了幾步。其中距離是以公尺為單位，上下樓則是以三公尺為一層樓的距離。如果搭電梯的話，不會增加爬樓的紀錄。
6.範例中還把這些數值顯示在畫面上。使用DispatchQueue.main呼叫async方法、在主佇列更新相關的資料。
7.除了如步驟4更新走路的相關資訊以外，還可以使用CMPedometer類別中queryPedometerData方法，得到某時段間的各種走路相關的資訊。
這個方法接受三個參數：第一個參數是開始的時間，範例中填入昨天、也就是24小時以前；第二個參數是結束的時間，範例中填入執行程式當時的時間；最後一個參數是一個Closure，裡面的程式碼處理得到的資訊。在範例中是印出從昨天到現在的走路步數、走路距離，以及上下樓的資訊。
8.最後在離開程式或離開畫面、不需要獲知記步器的數值時，使用stopUpdates方法，停止繼續獲取相關資訊。
9.要得到計步相關的姿訊，需要徵求使用者的同意。要求使用者授權時，會顯示相關說明文字。這段說明的文字，要在Info.plist加入：在左邊欄選取Info.plist檔案。按下最後一列的加號，新增一列。選擇「Privacy - Motion Usage Description」，並在右邊空間填入需要使用相簿的理由。

如何取得和高度相關的資料

[問題]如何使用手機上的高度資料
[解答]使用CoreMotion函式庫中的CMAltimeter類別，可以得到和高度相關的資料。
[範例程式碼]HelloAltimeter
[過程解說]
1.先匯入CoreMotion函式庫。使用CoreMotion函式庫中的CMAltimeter類別，產生出CMAltimeter類別的物件。
2.使用CMAltimeter類別的isRelativeAltitudeAvailable方法檢查是否可以蒐集到高度相關的資料，如果可以的話，就使用startRelativeAltitudeUpdates方法，得到加速度感測器的資料。請參考下面的程式碼：

```
import UIKit
import CoreMotion

class ViewController: UIViewController {

    @IBOutlet weak var altiLabel: UILabel!
    var altimeter:CMAltimeter!

    override func viewDidLoad() {
        super.viewDidLoad()
        altimeter = CMAltimeter()       //產生測量高度的altimeter
        let operationQ = OperationQueue()
        if CMAltimeter.isRelativeAltitudeAvailable(){    //如果可以使用
            altimeter.startRelativeAltitudeUpdates(to: operationQ,
                withHandler: {
                (data:CMAltitudeData?, error:Error?) in
                print("releative altitude:
                                \(data!.relativeAltitude) meters")
                //如果要顯示在畫面上的話，請參考下面程式碼
                DispatchQueue.main.async(execute: {
                self.altiLabel.text = "\(Int(data!.relativeAltitude))"
                })
            })
        }
    }

    override func viewDidDisappear(_ animated: Bool) {
        altimeter.stopRelativeAltitudeUpdates()      //停止更新高度資料
    }
}
```

startRelativeAltitudeUpdates方法需要兩個參數，第一個是執行獲取高度相關程式碼的工作佇列，第二個則是每次蒐集到高度資料後，想要執行的程式碼。第一個參數代入用OperationQueue方法得到的工作佇列。第二個參數的話，則是在Closure裡面寫進相關的程式碼。範例中是直接把相對高度數值印出來。從CMAltitudeData裡面的relativeAltitude屬性，就可以相對的高度。也就是相對於開始執行程式時手機所在的高度差。
完整了範例程式碼，把相對高度顯示在畫面上。而在離開畫面時，停止蒐集和高度相關的資料。