

iOS 센서컨트롤 오픈소스 &
파이썬을 이용한 데이터 분석법

SwiftMotionManager & Data Analysis Tutorial with Python for Korean

공개SW개발자Lab 오픈소스프론티어 2기 강명구

스마트폰이 등장하며 기존 모바일 폰으로는 할 수 없었던 많은 것들이 가능하게 되었다. 특히 다양한 센서의 탑재로 인하여 다른 하드웨어 기기들이 하던 역할을 스마트폰 하나로 할 수 있게 된 것은 사용자 입장에서 굉장히 편리한 일이다. 단순하게는 지도에서 현재 위치부터 사용자의 걸음 수, 속도, 나아가서는 날씨, 오르내린 층 수까지 다양한 부분에서 센서 정보가 활용되고 있다. 그리고 2015년부터 대중화되기 시작한 스마트워치에서도 다양한 센서를 통해 사용자에게 새로운 경험을 선사해주게 되었다.

이 글에서는 현재 애플 디바이스에서 대표적으로 사용되고 있는 다양한 센서들의 동작원리와 특징을 소개하며 iOS 센서 컨트롤 오픈소스인 SwiftyMotionManager와 파이썬을 이용한 데이터 분석방법에 대해 소개한다.

[목차]

1. 스마트폰 센서
 - 지문 센서 (Fingerprint sensor)
 - 근접 센서 (Proximity sensor)
 - 조도 센서 (Ambient light sensor)
 - 자이로 센서 (Three-axis gyro)
 - 가속도 센서 (Accelerometer)
 - 기압계 (Barometer)
 - 맥박 센서
2. SwiftyMotionManager
3. 데이터 처리를 위한 파이썬
 - Anaconda
 - jupyter(ipython)
 - NumPy, SciPy, Pandas
 - matplotlib.pyplot
 - Seaborn
4. 맺음말

1. 스마트폰 센서

- 지문 센서(Fingerprint sensor)



[그림 1] 지문센서의 인식 원리

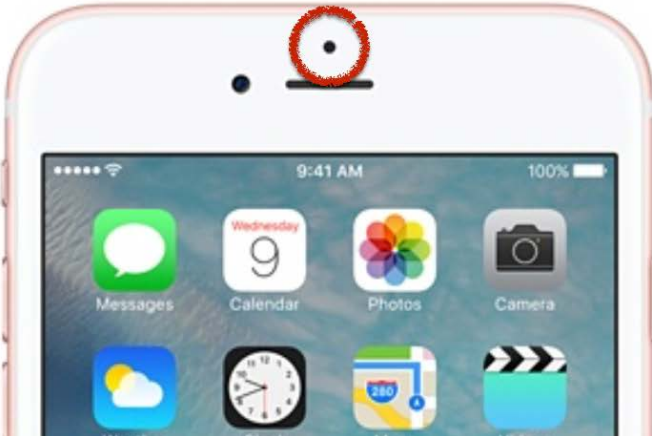
일반적으로 Fingerprint sensor는 광방식과 정전방식으로 나뉘어져 있으며 광방식은 인식기 안에 내장된 카메라가 손가락을 스캔하여 이미지를 불러와 읽어내는 방식이다. 영화 등의 보안구역에 들어가는 장면에서 많이 볼 수 있다. 위조가 쉬우며 인식이 느린 단점이 있다. 스마트폰에서는 대부분 정전방식을 사용하며 지문돌기의 사람 몸에 흐르는 전기량 차이를 이용하여 인식한다. 그러므로 시체 등에서는 일반적으로 사용이 불가능하고 직접 칩 가까이 손가락을 눌러 인식하므로 파손의 위험성이 있다. iOS는 Touch ID라고 부르고 있으며 사파이어 글래스를 사용하여 강도와 인식률을 높였다.



[그림 2] iOS 지문센서

iOS에서 지문 정보는 Touch ID 링에 손이 닿는 순간 지문인식이 준비되고 인식, 처리하는 과정을 거치게 된다. 데이터는 암호화되어 저장되며 iOS나 다른 앱에서 접근이 불가능하고 iCloud 등의 외부에 저장이 되지 않는다. 데이터 백업을 하더라도 지문 정보는 저장이 되지 않으므로 복구 시 새롭게 등록해야 하는 과정이 필요하다.

- 근접 센서(Proximity sensor)



UIDevice Class

Using the Proximity Sensor

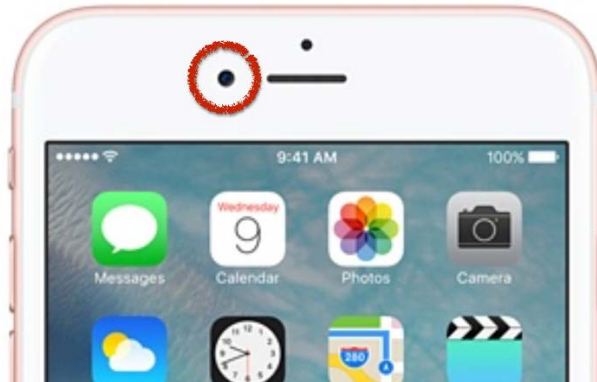
`proximityMonitoringEnabled` Property

`proximityState` Property

[그림 3] 근접 센서

일반적으로 통화 시 화면이 꺼지게 하는 역할로 자주 쓰인다. 근접한 곳에 무언가가 존재하는지의 여부만을 알려주며 오직 true, false로 값을 준다.

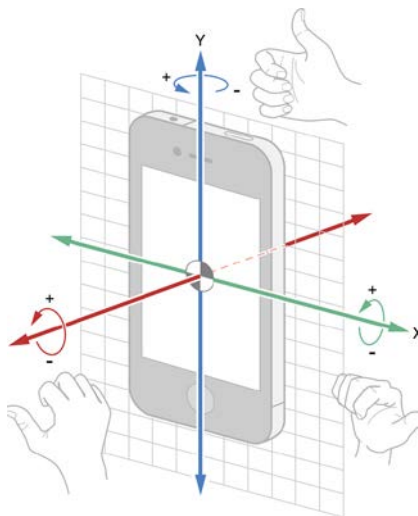
- 조도 센서(Ambient light sensor)



[그림 4] 조도 센서

조도의 정도를 파악하는 센서로서 iOS에서는 접근이 불가능하다. 아두이노 등에서 활용하는 경우가 많이 있으며 안드로이드에서는 Sensor 클래스를 통해 접근 가능하다.

- 자이로(Three-axis gyro)



[그림 5] 자이로

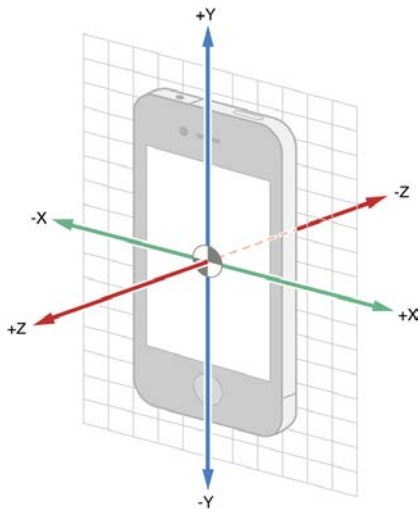
디바이스의 회전각속도를 측정하여 얼마나 회전했는지를 알려주는 센서이다. Glitch가 심하며 오차가 누적되므로 30도를 기울인 후 다시 -30도 되돌아오더라도 값이 그대로가 되지 않는다.

x: roll, y: pitch, z: yaw

단위는 Radians per second가 쓰이며 그림과 같이 오른손의 법칙으로 회전방향을 구분한다.

3.14 radian = 180 degree 를 기억하자.

• 가속도계(Accelerometer)



오른손의 법칙으로 x, y, z를 구분할 수 있으며 단위는 G를 쓴다. 1G = 9.80665 m/s² 로서 지구의 중력가속도라는 것을 기억하자. 수평일 때는 중력에 의해 z축의 값이 -1이 된다. 물론 우주로 나가면 0이 될 것이다.

$$s = vt - \frac{1}{2}at^2$$

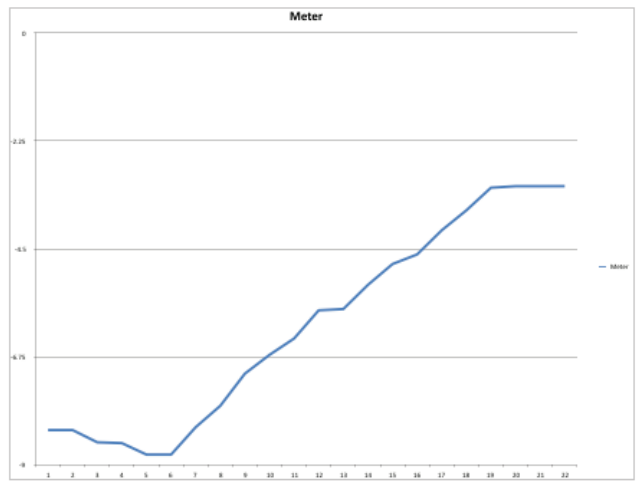
와 같은 식으로 거리를 구할 수 있다고 생각할 수 있지만 오차로 인해 제대로 된 속도나 거리측정은 무리가 있다.

[그림 6] 가속도계

• 기압계(Barometer)

흔히들 헛갈리는 고도계와는 다르며 해발 기준으로 공기가 누르는 압력의 정도를 표현해주는 센서이다. 높을수록 낮아지며 iOS에서는 kPA(kilopascals) 단위를 사용한다. 상승기류로 인해 대기 압력이 낮아지고 공기의 팽창 기온이 낮아져 구름이 생기는 현상을 이용하여 날씨예측에도 사용이 가능하다.

Pressure(kPa)	Meter
101.483543395996	-8.27086448669433
101.483543395996	-8.27086448669433
101.486602783203	-8.52492427825927
101.486808776855	-8.54199504852294
101.489685058593	-8.78098583221435
101.489685058593	-8.78098583221435
101.482818603515	-8.2106122970581
...	...
101.433540344238	-4.11264514923095
101.42854309082	-3.6977367401123
101.422889709472	-3.22654914855957
101.422508239746	-3.19540309906005
101.422508239746	-3.19540309906005
101.422500610351	-3.19439888000488



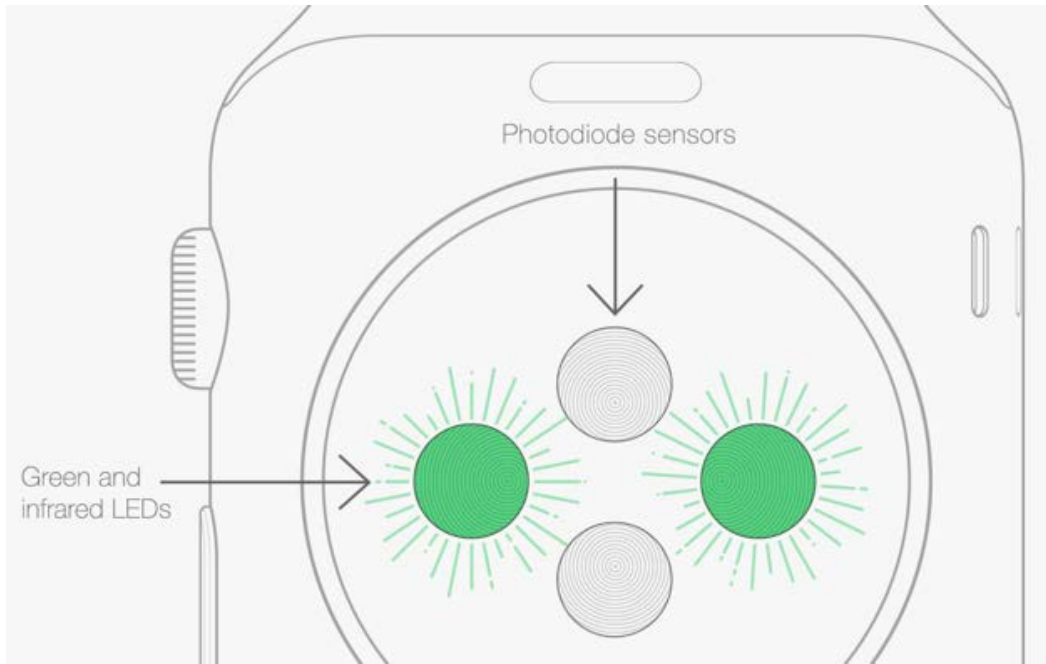
계단 올라가기

1kPa = 10hPa = 1000Pa

[그림 7] 기압계

위의 그림은 아이폰에서 계단을 올라가며 데이터를 측정된 것이며 0.01 단위 차이에도 고도는 큰 차이가 있는 것을 알 수 있다. 그림에서는 대략 5미터 차이에서 0.06kPa 차이가 나는 것을 볼 수 있다.

- 맥박 센서



[그림 8] 맥박 센서

Apple Watch에 들어가서 사용되고 있는 센서이다.

광혈류량측정(photoplethysmography)이라는 발음도 어려운 생소한 방법을 사용한다. 간단히 혈액은 적광은 반사하고 녹광은 흡수하는 특징을 이용하여 초당 수백 번의 녹색 LED 빛을 깜빡여 반사된 빛의 양을 파악하여 박동을 감지한다. WatchOS2 기준에서는 10분마다 자동 갱신되는 데이터를 가져와 사용하거나 앱 내에서 높은 주기로 데이터를 가져와 사용할 수 있다.

2. SwiftyMotionManager

Swift에서 CMMotionManager를 사용할 경우 여러 개의 인스턴스를 사용하게 되면 센서에서의 데이터 입력속도가 늦어질 수 있는 위험이 있다. SwiftyMotionManager는 기본적으로 Singleton 패턴을 사용하여 그러한 위험성을 방지하였고 CMMotionManager를 상속받은 클래스인 만큼 기존 클래스의 완벽한 대체가 가능하다.

최종적으로 raw 데이터 처리를 위한 다양한 알고리즘 적용을 목표로 하고 있으며 Swift가 빠르게 버전이 높아지는 만큼 지속적인 대응을 해야 할 필요성이 있다. 현재 Swift 2.x 을 지원하고 있다.

매뉴얼은 github를 참조하자.

<https://github.com/JeffGuKang/SwiftyMotionManager>

3. 데이터 처리를 위한 파이썬

현재 대학 등에서는 MATLAB을 신호나 데이터 처리에 많이 사용하지만 현재는 많은 대체제가 나와있다. 그 중 파이썬의 경우 다양한 라이브러리 지원으로 MATLAB보다 편리하게 사용 가능하며 무료라는 큰 장점이 있다. 현재 기존 공식문서를 번역 및 편집하여 한글로 된 튜토리얼을 github에 제작 중이다. 여기에서는 간단히 특징과 사용법을 소개한다.



[그림 9] Anaconda

손쉽게 데이터 처리를 위한 파이썬 환경을 구성하는 방법으로는 Anaconda라는 통합패키지를 사용하면 된다. 데이터 분석 혹은 공학, 수학 등을 위해 사용할 수 있는 패키지이며 330개 이상의 오픈 소스 패키지들이 들어있다. 그 중 NumPy, matplotlib, SciPy, Pandas가 대표적인 패키지이다.

- **Anaconda**

커스텀 패키지 및 환경 매니저 디펜던시 등을 관리

- **jupyter(ipython)**

쥬피터는 웹 인터프리터로서 콘솔 대신 브라우저상에서 기록을 남기며 작업을 할 수 있는 유용한 패키지이다. 과거 ipython이라는 이름에서 다중 커널(루비 등)을 지원하게 되며 개명이 되었다.

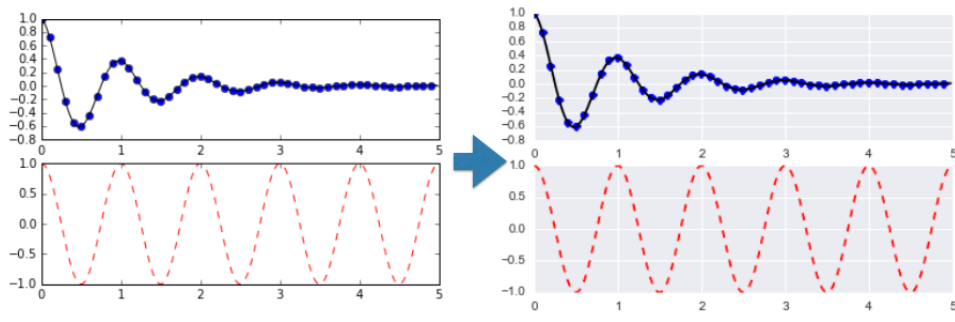
Markdown, 파이썬 뿐 아니라 다양한 언어 사용이 가능하며 github에서 기본적으로 지원을 하기 때문에 별도의 변환이 필요 없이 바로 github에서 결과를 확인할 수 있다.

- NumPy, SciPy, Pandas

NUMeric Python	SCientific Python	Pandas
N-dimensional Array Object	Based on Numpy	Based on Numpy
데이터들의 다차원 배열을 지원한다	수학적 알고리즘 모음	퍼포먼스 차이 X
다양한 수학 라이브러리를 지원한다	필요한 알고리즘을 골라 사용하면 된다	Dictionary처럼 사용 가능하다
Broadcasting 가능 (데이터구조가 맞지 않아도 연산이 가능하다)		High lv Data 다룬다

- matplotlib.pyplot

파이썬 그래프 라이브러리로서 Matlab과 같은 방식으로 그래프 동작을 수행하는 커맨드 스타일 함수들의 컬렉션이다. 이것은 function calls을 넘겨 상태를 유지하므로 현재 figure와 plot부분을 기록하며(tracking) 동작된다.



[그림 10] matplotlib.pyplot

- Seaborn

matplotlib.pyplot를 기반으로 동작하는 라이브러리로서 한마디로 아름다운 그래프를 쉽게 그릴 수 있도록 도와준다. import하는 것만으로도 변화를 느낄 수 있으며 위 그림에서 좌측의 기본 그래프에서 Seaborn이 적용된 오른쪽 그래프를 확인할 수 있다. 물론 다양한 커스텀이 가능하며 성능 변화는 거의 없다. 예뻐지고 덩달아 기분이 좋아지는 것에 의의를 두자.

아래는 위의 것들을 함께 사용해본 간단한 예제이다.

```
%matplotlib inline

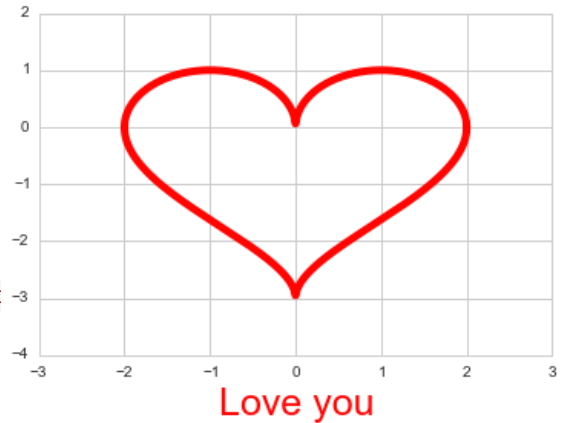
import scipy as sp
import matplotlib.pyplot as plt
import seaborn

x = sp.linspace(-2,2,1000)
y1 = sp.sqrt(1-(abs(x)-1)**2)
y2 = -3*sp.sqrt(1-(abs(x)/2)**0.5)

seaborn.set_style("whitegrid")
heart = plt
plt.plot(x, y1, color="r", linewidth=5)
plt.plot(x, y2, color="r", linewidth=5)
plt.xlabel("Love you", fontsize=25, color="r")

axes = plt.gca()
axes.set_xlim([-3,3])
axes.set_ylim([-4,2])

plt.show()
```



[그림 11] 예제

4. 맺으며

센서들은 우리에게 정말 많은 것을 알려준다. 처리과정은 복잡하지만 일정한 규칙을 찾을 수만 있다면 간단하게는 만보계의 역할로부터 VR의 각종 컨트롤러까지 사용 가능하다. IoT와의 연계도 훌륭해서 스마트 체중계, 드론, 각종 Beacon등에 널리 사용되고 있다. 이러한 센서들에 기초적으로 접근하기 위한 SwiftyMotionManager와 파이썬 데이터 분석 튜토리얼 프로젝트는 이제 막 한 걸음을 디뎠다. 꼭 이 프로젝트들이 아니더라도 아직 부족한 부분이 많아 다른 사람들의 도움을 기대하고 있으며 점점 더 성장하기를 꿈꾸는 수많은 프로젝트들에게 따뜻한 관심 한 손가락은 어떨까?