

Report

Path Track 기반 Smart Home Service

IoT 활용 및 실습 기말 프로젝트 최종 보고서

<과제물 제출 전 체크리스트>

1. 이 과제물을 내가/우리가 직접 연구하고 작성한 것이다 ☐
2. 인용한 모든 자료(책·논문·인터넷자료 등)의 인용표시를 바르게 하였다 ☐
3. 인용한 자료의 표현이나 내용을 왜곡하지 않았다. ☐
4. 정확한 출처 제시 없이 다른 사람의 글이나 아이디어를 가져오지 않았다. ☐
5. 과제물 작성 중 도표나 데이터를 조작(위조 혹은 변조)하지 않았다. ☐
6. 과제물은 다른 사람으로부터 받거나 구매하여 제출하지 않았다. ☐
7. 이 과제물에 실질적으로 참여하지 않은 사람을 공동 제출자로 명기하지 않았다. ☐
8. 이 과제물과 동일한 내용을 다른 교과목의 과제물로 제출한 적이 없다. ☐

과목명	IoT 활용 및 실습
교수명	최차봉 교수님
학과	소프트웨어학과
팀원	6조 2014122161 양영균 2014122251 장하람 2015124188 장우진 2015125031 문예지
제출일	2018.12.24. (월)

<목차>

1. 개발 목표	1
1. 프로젝트 제안 배경	
2. 프로젝트 목적	
3. 프로젝트 요구사항 및 기대효과	
2. 프로젝트 진행 계획	3
1. 구성원 및 역할	
2. 개발계획	
3. 서비스 Framework	5
1. 프로젝트 시나리오	
2. 시스템 구성도	
1) 통합 시스템 구성도	
2) Path Track	
3) 안드로이드 APP(for NFC)	
4) Web crawling(for 날씨,교통정보)	
5) 모형 집 모델링	
4. 개발 추진 실적	18
5. 문제점 및 해결내용	19
6. 개인 기여내용 및 배운점	22
7. Further study	25
[별첨] 프로젝트 상세 코드	

1. 개발 목표

1. 프로젝트 제안 배경

밤중에 목이 말라 침대 밖으로 나올 때 발가락을 문턱에 부딪치거나, 베란다에서 맥주를 가져올 때 손이 모자라서 물을 못 뜨고 나오거나, 외출하려고 신발을 신는 도중에 신발장의 조명이 꺼져서 앉은 채로 센서를 작동시키려고 손을 흔들거나, 출근, 등교하려고 버스에 타면 그때서야 집에 지갑을 두고 온 것을 알아차리는 등의 일상생활의 ‘사소한 불편함’을 모두 겪은 적이 있을 것이다. 우리는 이런 사소한 불편함을 명확히 파악하고, IoT를 적용하여 불편을 해소하고자 한다.

현대인은 수많은 가전제품에 둘러싸여 생활하고, 그들은 모두 현대인들에게 ‘기능’을 제공한다. 하지만 이러한 ‘기능’들은 현재 대부분이 수동적이다. 역설적이게도 편리함을 위해 개발된 가전제품을 ‘사용하는’ 것이 또 다른 사소한 불편을 가져오는 것이다. 수동적인 가전제품들을 사용자가 원할 때 원하는 방식으로 능동적으로 작동하게 한다면, ‘사소한 불편함’을 해소할 수 있을 것이다.

가전제품들이 능동적으로 서비스를 제공하기 위해서는 사용자가 가는 방향과 가는 목적을 알아야 한다. 사용자의 ‘동선’과 ‘의도’를 완벽히 파악할 수 있다면 ‘맞춤형 서비스’를 제공할 수 있을 것이라는 아이디어에서 이 프로젝트를 시작했다.

2. 프로젝트 목적

우리는 현재 가진 기술로는 사용자의 의도를 파악할 수 있는 방법이 없기 때문에 사용자의 동선으로 사용자의 의도를 유추해 내기로 했다. 예를 들어 사용자가 거실에서 방으로 이동할 경우 방의 가전제품을 사용할 의도와 거실의 가전제품을 당분간 사용하지 않을 의도를 가지고 있다고 유추한다. 이에 따라 방의 대기전력을 활성화하고 거실의 대기전력을 차단하여 사용자의 의도에 맞는 서비스를 제공한다. 즉 우리 프로젝트는 사용자의 동선을 완벽하게 파악하는 것을 수단으로 활용하여 그에 따른 가전제품들의 능동적인 서비스 제공을 목적으로 한다.

3. 프로젝트 요구사항 및 기대효과

각 방에 설치하여 사용자가 현재 어느 위치에 있는지 실시간으로 파악한다. 사용자의 위치 변동에 기반, 그 동선에서 사용자가 필요로 하는 서비스를 제공한다.

사용자가 직접 수행할 수 있는 사소한 일상의 불편함 해소를 1차 목표로 한다. 현재 우리 프로젝트 수준에서 다룰 내용으로는 외출 시 소지품 확인, 교통 상황 및 날씨 안내, 대기 전력 차단 등이다. Further study로는 빅데이터 분석을 통해 개발자의 명시적인 개입 없이도 사용자에게 맞춤형 서비스를 제공하는 것을 목표로 한다.

IoT 모듈로서, 인간의 개입 없이 스스로 판단하고 작동하며, 분리된 센서 모듈의 전력 소모를 최소화한다.

Smart

- 사용자가 외출할 때, 깜빡 잊고 챙기지 않은 물건이 없는지 체크
- 외출 전 교통 상황과 날씨 안내
- 집 안 NFC 태그별 맞춤 루틴 설정으로 특정 위치에서 연속된 서비스 제공

Safe

- 지진, 화재 등 재난이 감지될 때, 사람의 재실 여부에 따라 적절한 대응체계 구축
- 사용자가 외출할 때 Safe mode를 설정. 해제되지 않은 상태로 침입자가 발생할 시 이를 감지하고 대응

Resource-saving

- Path Track으로 불필요한 대기 전력을 차단하여 에너지 절약
- 교통 상황 안내로 개인의 시간 절약 및 사회적인 교통체증 조절에 이바지

2. 프로젝트 진행 계획

1. 구성원 및 역할

이름	역할
양영균	<ul style="list-style-type: none"> • 적외선 센서(FC51)를 이용한 Path Track 서비스 설계 (Pub/Sub) • End user 요구사항 분석 및 시나리오 설계 • Path Track 서비스 Rule 구축
장하람	<ul style="list-style-type: none"> • Path Track 서비스 Rule 구축 및 최적화 • Safety mode에서 도둑 경로탐색 코드 구현 • 진동감지 센서와 불꽃감지 센서를 이용한 재난 대응 시스템 구현
장우진	<ul style="list-style-type: none"> • 교통정보, 날씨 알림 서비스 구현
문예지	<ul style="list-style-type: none"> • NFC를 이용하여 놓고 간 물건 알림 서비스 구현 • TV 화면 팔로잉 서비스 구현

2. 개발계획

- 2018.11.12. 1차 회의에서 프로젝트를 위한 다양한 아이디어를 토의
- 2018.11.13. 2차 회의에서 Path Track을 기반으로 한 samrt home 시스템으로 프로젝트 아이템을 정했다.
- 2018.11.19. 프로젝트 제안서를 작성하기 위해 제안 배경과 목적을 명확히 한다. 프로젝트 목적에 따라 Path Track 서비스 Publisher를 구현하는 것이 가장 핵심이며, 실제로 센서 여러 개를 사용하기 때문에 하드웨어 보정에도 시간이 많이 걸릴 것으로 판단하였다. 따라서 프로젝트 개요가 정해짐에 따라 바로 적외선 센서, 불꽃 감지 센서, 진동 감지 센서, 점퍼 케이블 등 부품을 주문하였다.
- 2018.11.26. 3차 회의에서 사용자 요구사항에 따른 시나리오 설계, 역할 분담, 개발 계획을 수립했다. 개발 계획은 아래와 같다.

교통정보/날씨 알림 서비스와 NFC태그를 이용한 소지품 알림 서비스는 위의 Path Track 서비스 Publisher가 퍼블리싱하는 CODE number를 Subscribe하여, 각 CODE number에 맞는 적절한 서비스를 제공하는 일종의 모듈이므로 Path Track, 교통정보/날씨 알림, NFC 프로덕트 사이의 Critical path가 없다고 판단하였다. 따라서 중간 점검 전까지 Path Track 및 CODE Publishing/subscribing을 위한 전체 서비스 플랫폼을 완성하고 테스트 한 후, 최종 발표 전까지 통합 시스템을 테스트하고 보수하는 것을 목표로 한다.

양영균, 장하람은 공동 작업으로 Path Track 서비스 구현 및 시연을 위한 End user의 요구사항을 다양한 상황을 가정하여 분석하였고, 센서가 도착하는 대로 하드웨어 테스트 및 기초가 될 동선 파악 코드를 작성하기로 하였다. 또한 화재 상황이나 지진상황을 가정하여 불꽃 감지 센서와 진동 감지 센서를 이용하여 재난 상황을 판단하고 그에 맞는 대응 시스템을 구현하기로 하였다.

장우진은 EC2 가상 서버에서 교통, 날씨 정보를 크롤링해서 보드와 웹 소켓 통신을 하도록 할 예정이다. 적외선 센서로 옷장 문이 열리는 것을 감지하고, 현관 정보를 Subscribe하여 현관에 도착하는게 감지되면 LCD에 출력되게 구현하려고 한다.

문예지는 Path Track 서비스가 이미 된다고 가정(Test Device를 만들어 시뮬레이터 실행)하고 ①NFC Artik 안드로이드 애플리케이션(놓고 간 물건 알림)과 ②TV 화면 따라오기 기능을 구현하기로 하였다. 두 기능 사이에 연관성이 없어서 LCD 화면 출력을 위해 Artik 보드가 반드시 필요한 TV 화면 따라오기 기능과 Artik 보드 없이도 구현이 가능한 안드로이드 앱 만들기를 병렬적으로 진행하기로 하였다.

위와 같이 병렬 프로세스로 각자 설계하고, 2018.12.10. 전후에 중간 점검을 계획하였으며 중간 점검에서 각 모듈 및 Pub/Sub 시스템의 정상 작동을 테스트하기로 한다.

모듈 테스트가 끝나면 모형 집을 설계한다. 설계 과정에서 M-F 점퍼 케이블이 다수 필요할 것으로 판단, 추가 구매하였다. 모형 집을 만들기 위한 재료도 구매해야 한다.

모형 집 설계가 완성되면 사용하는 센서들을 모두 부착하고 시스템 통합 프로세스를 거치기로 한다.

시스템 통합 과정에서의 문제점을 모두 해결하면 프로젝트 개발을 종료하고, 시연을 위한 시나리오를 구성하고 시연에 필요한 PPT 등 보조자료를 준비하기로 한다.

3. 서비스 Framework

1. 프로젝트 시나리오

1) Path Track

집에 들어오면 현관에 불이 켜진다. 현관에서 거실로 넘어가면 현관 조명은 자동으로 꺼지고 거실 조명이 자동으로 켜진다. 가족들이 모두 거실에 모여있다가 마지막 사람이 자기 방으로 들어가는 순간 거실의 조명이 꺼진다.

2) 놓고 간 물건 알림

급하게 현관문을 나서려는 순간 문에 있는 화면에 지갑과 자동차 열쇠를 놓고 왔다는 알림이 뜬다. 방으로 돌아가서 지갑을 챙겨서 나온다. 자동차 열쇠를 놓고 왔다는 알림이 뜨지만 오늘은 자동차를 몰고 갈 계획이 없기 때문에 무시하고 나간다.

3) 교통 상황 안내

출근 시간, 문 밖으로 나서려고 할 때 현관문에 있는 모니터에 직장까지의 예상 도착 시간이 나온다. 차가 많이 막히니 대중교통을 이용하라는 안내를 받는다.

4) 날씨 안내

- 화창해 보이는 날씨이다. 그냥 현관문을 나서려는 순간 모니터에 '오후에 소나기 예보가 있으니 우산을 챙기세요'라는 알림이 뜬다.
- 현관문을 나서려는데 '미세먼지가 심하니 마스크를 챙기세요'라는 알림이 뜬다.
- 갑자기 날씨가 추워졌다. 모니터에 '갑작스러운 한파가 찾아왔으니 옷을 따뜻하게 챙겨입으세요'라는 알림이 뜬다.

5) 재난 상황 감시

진동 감지 센서로 지진을 감지, 불꽃 센서로 화재를 감지하여 가스와 전기를 차단하고 119에 자동으로 연락이 간다. Safe mode 작동 상태를 기반으로 집 안에 사람이 있는지 없는지 판별하여 있으면 외부로 나갈 수 있도록 출입구를 개방하고 적절한 대처 방안을 음성으로 안내해주고, 없을 땐 화재가 난 곳을 차단하여 확산을 방지한다.

6) 도둑 침입 감지

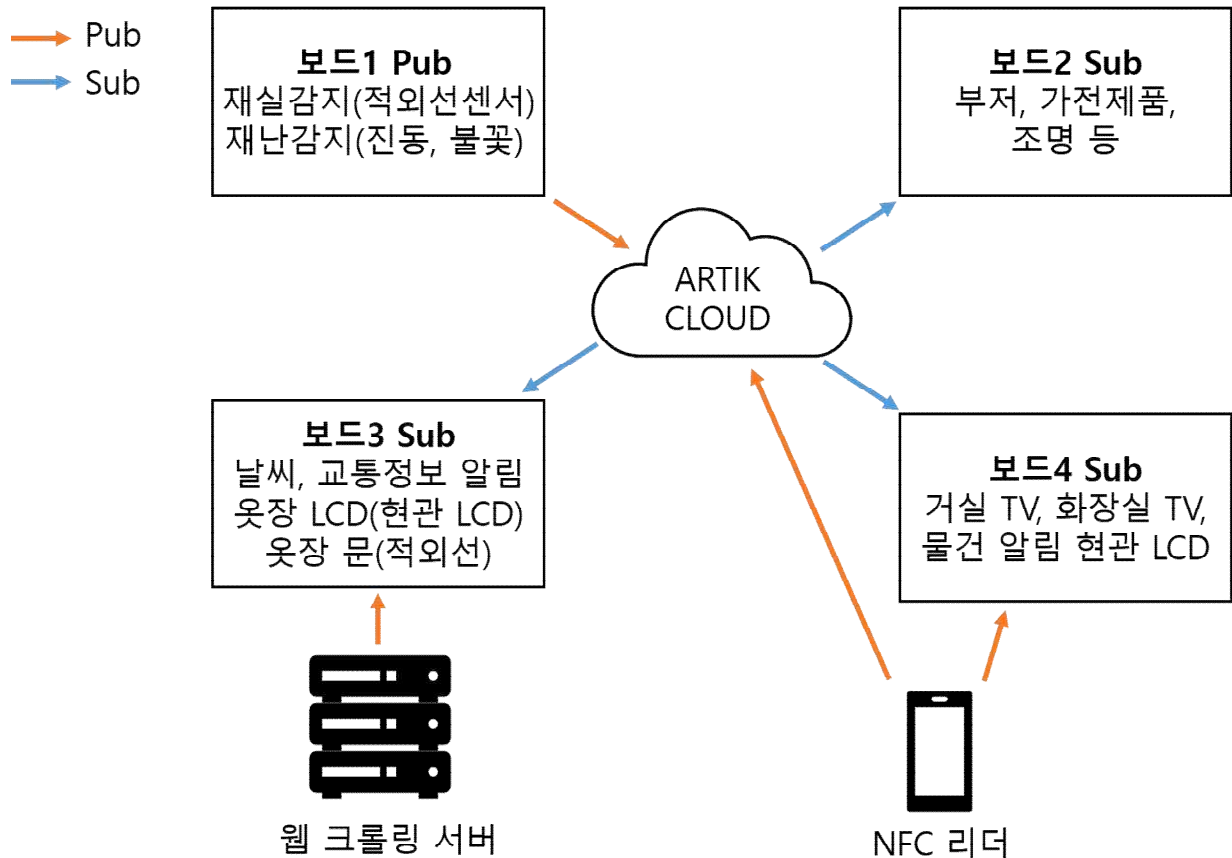
가족들이 모두 집에서 나가면 자동으로 세이프 모드가 설정된다. 세이프 모드가 해제되지 않았는데 사람이 들어오는 게 감지되면 사용자 휴대폰에 알림이 간다.

7) TV 화면 따라오기

거실에서 축구 경기를 보고 있는데 갑자기 배가 아파서 화장실로 간다. 거실에서 나오던 영상이 화장실에서도 연속해서 재생되어서 중요한 골 장면을 놓치지 않을 수 있었다.

2. 시스템 구성도

1) 통합 시스템 구성도

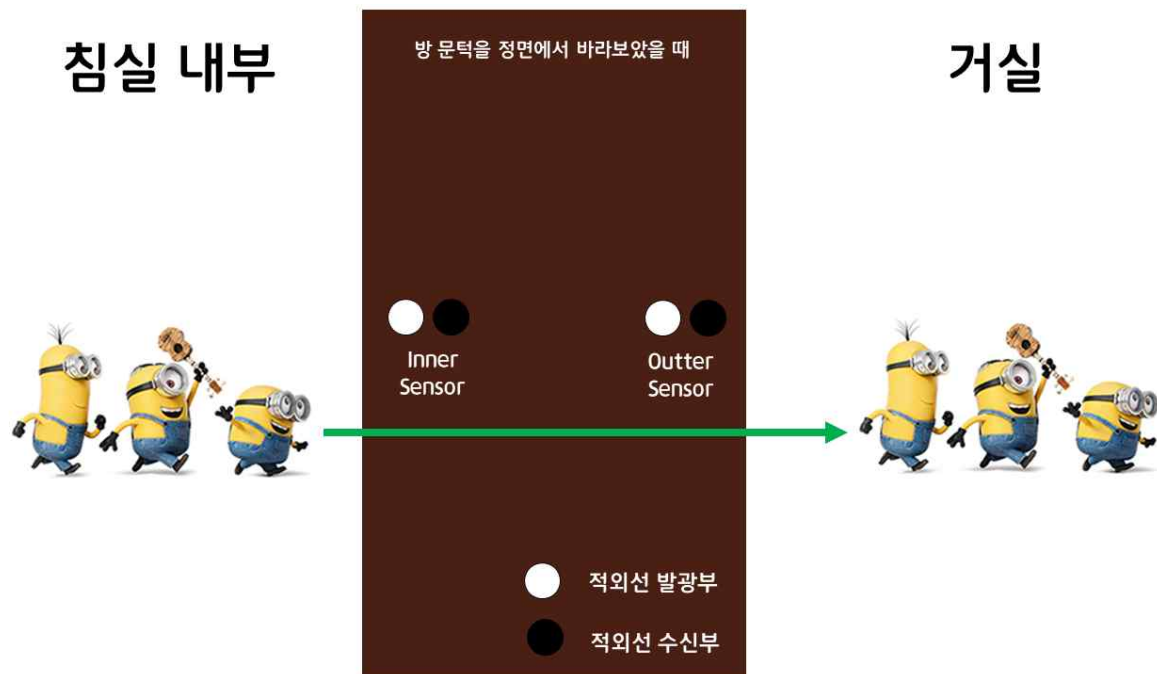


- ① 보드 1(양영균, 장하람): Path Track, 재난 감지 시스템
적외선 센서 8개(4개 쌍-Entrance, Palor, Room, Toilet)가 사용자의 동선을 파악하여, 사용자가 이동할 때 마다 그 상황에 맞는 CODE number를 Publish
불꽃 감지 센서와 진동 감지 센서가 재난 상황을 인식하여, 각 상황에 설정된 CODE number를 Publish
- ② 보드 2(양영균, 장하람): topic(CODE) Subscriber
보드 1이 Publish한 topic을 Subscribe한다. Subscribe한 코드에 맞게 safe mode 설정, 대기 전력 차단, 가스 차단 등의 서비스를 제공한다.
- ③ 보드 3(장우진): 옷장 LCD에 날씨와 교통정보 알림
LCD 1개와 적외선 센서 1개가 옷장 문 열림을 감지하여 LCD에 출력하는 기능을 위해 사용됨.
- ④ 보드 4(문예지): 거실 TV, 화장실 TV, 현관 LCD
LCD 3개를 연결하여 2개는 거실 TV와 화장실 TV로 'TV 화면 따라오기' 기능 구현에 사용함. 나머지 1개의 LCD는 '놓고 간 물건 알림'을 위한 현관 LCD로 사용.

2) Path Track

① 한 쌍의 적외선 센서 사용

두 개의 적외선 센서가 한 쌍이 되어 사용자가 방 안에서 밖으로 이동하는지, 방 밖에서 안으로 이동하는지 판단한다.



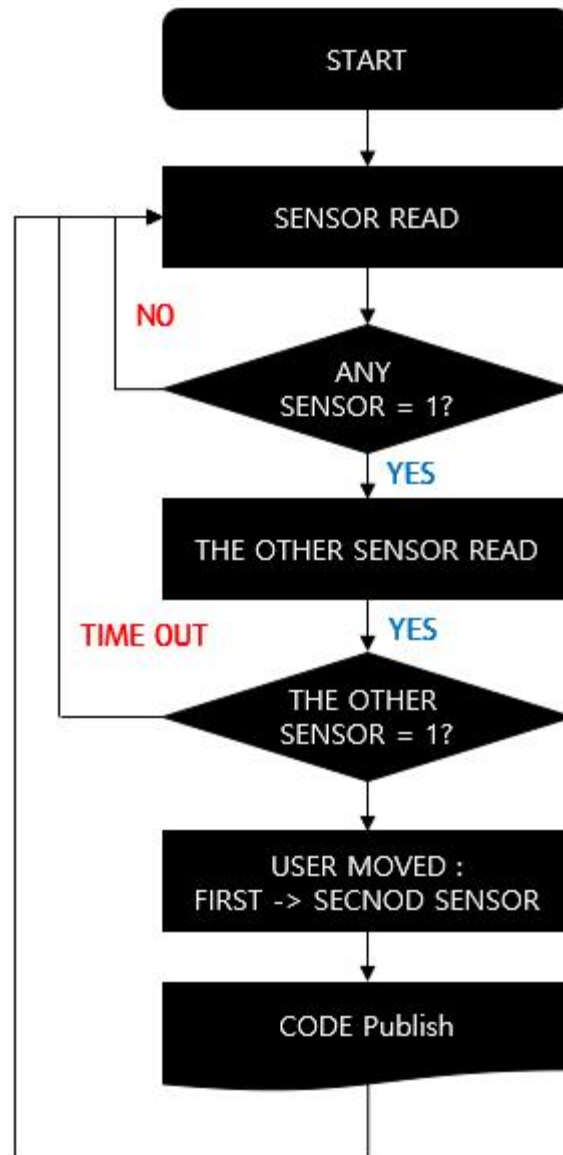
문틀에 두 개의 센서를 심고, 둘 중 어느 한쪽(Inner)의 센서에 장애물이 감지되면, 반대쪽 (Outer)센서에서 장애물이 감지될 때 까지 기다린다. 즉, Inner 센서에서 장애물이 감지되고 나서 Outer 센서에서 장애물이 감지되었다면, 사용자가 침실 내부에서 거실로 이동했다고 판단할 수 있다.

이것을 코드로 구현하기 위해, Publisher 보드는 무한 루프를 돌면서 Inner와 Outer 센서를 모두 read 한다. 일정 딜레이를 두고 read를 반복하다가 둘 중 하나의 센서 read 결과가 1이 되면 무한 루프 속에서 조건문으로 들어가서 반대쪽 센서 read가 1이 되기를 기다린다. 1이 된다면, 사용자가 Inner에서 Outer로 이동했음을 알 수 있다. 이 단계의 플로우차트를 첨부한다.

② 여러 쌍의 적외선 센서 사용

이제 여러 쌍의 적외선 센서를 사용하여 Entrance, Palor, Room, Toilet 사이에서의 사용자의 움직임을 파악한다. 이 단계에서 문제점이 발생하는데, 이 문제점에 대해서는 문제점 및 해결방안에서 자세히 기술한다.

Path Track 플로우차트



아래 링크에서 Publisher 보드의 코드를 확인할 수 있습니다.

https://github.com/Haramble/PathTrack_IoT_Project_Publish

ARTIK 보드 1 : Path Track, 재난 감지 시스템 코드 Publisher

3) 안드로이드 APP(for NFC) - 코틀린으로 Artik 안드로이드 앱 만들기

① 튜토리얼 문서 확인하기

Artik 개발자 문서에 Your first Android app 이라는 제목의 안드로이드 데모 앱 개발 문서가 있다. 하지만 2년 전에 올라온 내용이라 그대로 쓰기엔 안드로이드 버전이 안 맞는 등 문제점이 많다.

Artik 연결 이외에도 NFC 기능을 구현해야 해서 안드로이드 공식 문서를 살펴보았다. 안드로이드 개발을 하다 보면 공식 문서를 가장 먼저 확인하게 되는데, 공식 문서의 문제는 해당 주제에 대한 포괄적인 내용을 다 포함하고 있어서 복잡한 주제일수록 필요한 부분만 찾아내기가 힘들다는 것과, 핵심 코드만 소개되어 있기 때문에 어디에 이 코드를 붙여넣어야 하는지 알기 힘들다는 것이다. 그래서 NFC 읽기/쓰기만 구현되어있는 예제 코드를 따로 찾아보았다.

- Artik 안드로이드 앱 튜토리얼: <https://developer.ARTIK.cloud/documentation/tutorials/your-first-android-app.html>
- Artik 안드로이드 앱 데모: <https://github.com/ARTIKcloud/tutorial-android-yourFirstApp>
- 안드로이드 NFC 가이드: <https://developer.android.com/guide/topics/connectivity/nfc/nfc>
- 안드로이드 NFC 예제: <https://www.codexpedia.com/android/android-nfc-read-and-write-example/>

② 코드 분석하기

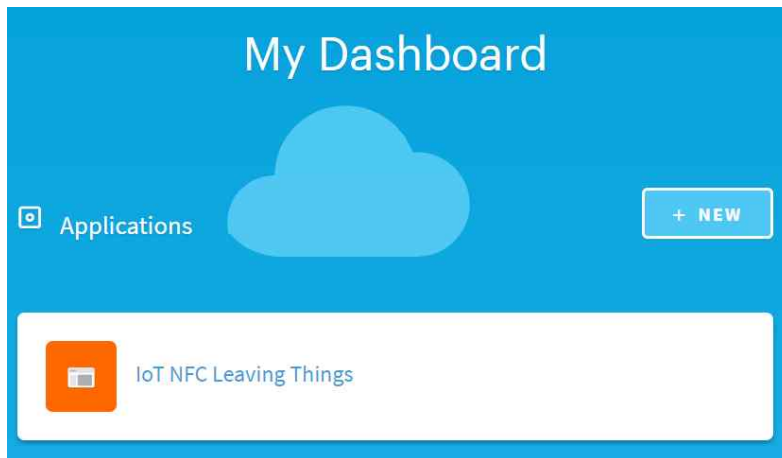
소스를 모두 구했으니 일단 안드로이드 프로젝트 2개를 만들어서 각각 원래의 자바 코드로 잘 돌아가는지 확인한다. 이 때 버전 호환성 문제 등을 체크한다. 안드로이드 스튜디오의 도움말을 보면서 gradle 버전을 직접 수정해준다. 주어진 예제가 잘 돌아가는 것을 확인한 후 코틀린을 지원하는 새로운 안드로이드 프로젝트를 만들어서 자바 코드를 코틀린으로 변환한다. 일차적으로 안드로이드 스튜디오에서 제공하는 변환기를 사용하고, 다음으로 코드를 자세히 보면서 자바 스타일 코드를 코틀린 스타일로 바꿔준다.

```
dependencies {  
    implementation 'net.openid:appauth:0.4.1'  
    implementation 'cloud.artik:artikcloud-java:2.0.7'
```

↑ gradle.app에 Artik에 관련된 외부 패키지를 넣어준다.

③ Artik Application, Device Type, Device 만들기

Artik cloud developer의 Dashboard에서 새로운 애플리케이션을 만든다.



Artik cloud developer의 Dashboard에서 새로운 애플리케이션을 만든다.

Authentication [LEARN MORE](#)

AUTH METHODS

- ☐ Client credentials
Server Side Application (without user interaction)
- ☒ Authorization Code
 - ☐ With Secret
Web Server
 - ☒ Without secret, PKCE mandatory
Installed Applications (Mobile, Desktop)
- ☐ Implicit
Client Side Application running in a browser
- ☐ Limited Input
Application running on limited input platform like TV, watch, etc.

AUTH REDIRECT URL

kau.iot.mijey.iotnfcleavingthings://oauth2callback
205

↑ Artik Application - App Info

```

44      <activity
45          android:name="net.openid.appauth.RedirectUriReceiverActivity">
46          <intent-filter>
47              <action android:name="android.intent.action.VIEW"/>
48              <category android:name="android.intent.category.DEFAULT"/>
49              <category android:name="android.intent.category.BROWSABLE"/>
50
51              <!--
52              Make sure these two intent filter fields respects "AUTH REDIRECT URL"
53              of your application set up at the developer.artik.cloud.
54              For example, if REDIRECT URL is "cloud.artik.example.oauth://oauth2callback",
55              the intent-filter fields looks like the following
56              -->
57              <data android:scheme="kau.iot.mijey.iotnfcleavingthings" android:host="oauth2callback"/>
58          </intent-filter>
59      </activity>
60  </application>
  
```

↑ Android Manifest


```

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "kau.iot.mijey.iotnfcleavingthings"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        manifestPlaceholders = [appAuthRedirectScheme: "kau.iot.mijey.iotnfcleavingthings://oauth2callback"]
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

```

↑ gradle.app

App name과 description은 적당히 입력하면 된다. 중요한 것은 Auth redirect url 이다. 데모를 실행할 때는 cloud.ARTIK.example.hellocloud://oauth2callback 이라는 이미 있는 다른 Artik 애플리케이션의 Auth redirect url을 사용하였다. 하지만 여기서부터는 우리가 만든 안드로이드 앱과 동일하게 입력을 해주어야한다. 안드로이드 패키지 네임과 동일하게 매니페스트와 Artik 애플리케이션에 입력해주었다. 이는 추후에 수정할 수 있다.

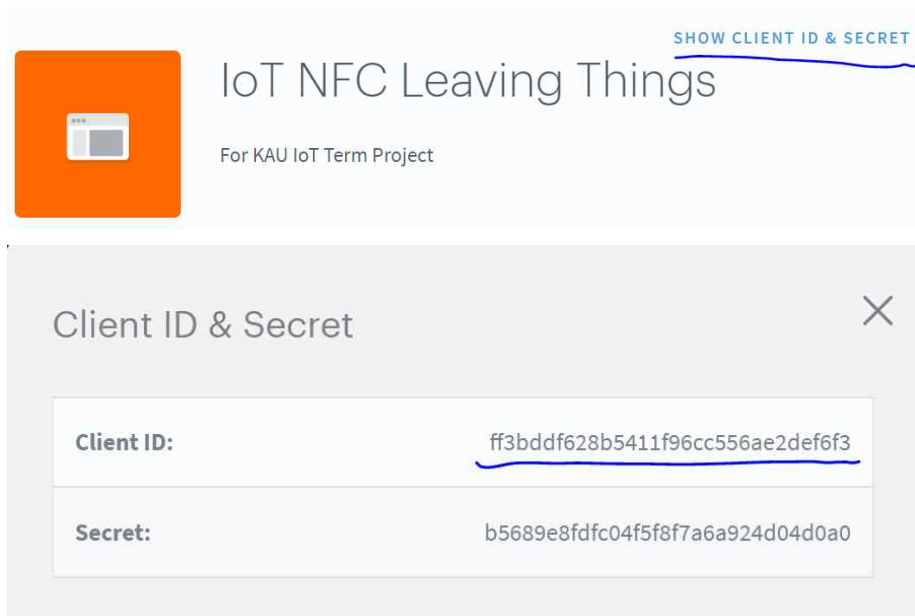
 Leaving Things EDIT DEVICETYPE 0 ERRORS						
놓고간 물건 알림						
CONNECTED APPS	DEVICES CREATED	PAYLOADS /HR	ACTIONS /HR	API CALLS /HR	ERRORS /HR	USED STORAGE
0	0	0	0	0	0	0

IoT Leaving Things 



DEVICE INFO		DATA TRANSFER	
DEVICE ADDED ON 20/Dec/2018		DEVICE TYPE Leaving Things	
DEVICE ID <u>bc9379eecd7043eca7048dca4d2f8525</u>		DEVICE TYPE ID dt0c3a7a16f6354822b90f7039d959341d	
DEVICE TOKEN 832e462e02a746309fd45fb44630fd0c		REVOKE TOKEN	

↑ Device Type과 Device ID



↑ Artik Application Client ID

④ Aritk 안드로이드 앱

```
class ArtikConfig {
    companion object {
        const val CLIENT_ID = "ff3bddf628b5411f96cc556ae2def6f3" // AKA application ID
        const val DEVICE_ID = "bc9379eecd7043eca7048dc4d2f8525"

        // MUST be consistent with "AUTH REDIRECT URL" of your application set up at the developer.artik.cloud
        // Artik Cloud - Applications - App Info 맨 밑에 AUTH REDIRECT URL 입력하는 칸이 있음
        // Gradle에서 manifestPlaceholders = [appAuthRedirectScheme: "kau.iot.mikey.iotnfcleavingthings://oauth2callback"] 수정해야 함
        // 매니페스트에서 <data android:scheme="kau.iot.mikey.iotnfcleavingthings" android:host="oauth2callback"/> 수정해야 함
        const val REDIRECT_URI = "kau.iot.mikey.iotnfcleavingthings://oauth2callback"
    }
}
```

↑ ArtikConfig.kt

위에서 설정한 Artik 애플리케이션, Artik Device의 ID를 입력하고 redirect url도 통일해준다. 파란색, 분홍색, 빨간색으로 구분하였다.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="IoTNFCLeavingThings"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <action android:name="android.intent.action.VIEW"/>
            <action android:name="kau.iot.mikey.iotnfcleavingthings.ARTIKCLOUD.AUTHORIZATION.RESPONSE"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
```

↑ Android Manifest

```
class ArtikAuthHelper {
    companion object {
        const val ARTIKCLOUD_AUTHORIZE_URI = "https://accounts.artik.cloud/signin"
        const val ARTIKCLOUD_TOKEN_URI = "https://accounts.artik.cloud/token"
        const val USED_INTENT = "USED_INTENT"

        // 매니페스트에서 <action android:name="kau.iot.mijey.iotnfcleavingthings.ARTIKCLOUD_AUTHORIZATION_RESPONSE"/> 수정해야 함
        const val INTENT_ARTIKCLOUD_AUTHORIZATION_RESPONSE = "kau.iot.mijey.iotnfcleavingthings.ARTIKCLOUD_AUTHORIZATION_RESPONSE"
    }
}
```

↑ ArtikAuthHelper.kt

Artik 인증을 http로 받기 때문에 앱에서 브라우저 창을 띄워야 할 때가 있다. 안드로이드 Main Activity에서 웹 브라우저로 갔다가 다시 돌아올 때 intent 값을 받는데 이에 따라서 적절한 행동을 하기 위해 설정한다.

```
private fun postMsg(nfcTag: String = resetHavings, doGetLastMsg: Boolean = false) {
    val msg = Message()
    msg.sdid = ArtikConfig.DEVICE_ID
    msg.data["leavings"] = nfcTag
}
```

↑ MessageActivity.kt의 postMsg 메소드

```
private void postMsg() {
    final String tag = TAG + " sendMessageActionAsync";

    Message msg = new Message();
    msg.setSdid(Config.DEVICE_ID);
    msg.getData().put( k: "stepCount", v: 4393);
    msg.getData().put( k: "heartRate", v: 110);
    msg.getData().put( k: "description", v: "Run");
    msg.getData().put( k: "activity", v: 2);
}
```

↑ Artik에서 제공하는 데모 코드의 postMsg 메소드

이 외의 다른 부분은 데모 코드와 같으므로 필요에 따라 수정해서 쓰면 된다. 보내는 값을 변경하는 부분은 postMsg에 있다. 이 부분을 Artik Device의 field에 따라 적절히 key-value를 넣으면 된다. 데모 코드가 더 적절한 예시여서 함께 붙여놓았다.

⑤ NFC 읽기/쓰기

```
<uses-permission android:name="android.permission.NFC"/>
```

↑ Android Manifest

NFC는 안드로이드 매니페스트에 퍼미션을 추가하는 것 이외에 특별히 설정할 것이 없다. 만약 이것이 상용 애플리케이션이라면 NFC 지원 여부에 따른 예외처리 코드가 더 있어야 하지만 여기서는 하지 않았다. 나머지 실행코드는 예제 코드와 거의 같으므로 자세한 설명은 생략한다.

코드 전문은 https://github.com/mijey21/KAU_IoT_Project 에서 확인할 수 있다.

4) web crawling server와 web socket 통신

① server 구축

```
var ws_artik_server = new ws_artik.Server({port:80});

ws_artik_server.on('connection', function(ws) {
  console.log("artik is connected")
  let url = "https://weather.naver.com/rgn/cityWetrWarea.nhn?cityRgnCd=CT001000"
  request(url, function(error, response, body){
    if (error) {
      console.log(error)
      return;
    }
    const $ = cheerio.load(body)
    let weather=[], temperature=[], precipitation=[]
    let tr = $('tbl_weather > tbody > tr').each(function (index, elem) {
```

80번 포트를 사용하여 통신. request, cheerio 모듈을 사용해서 네이버 날씨 홈페이지에서 날씨, 온도, 강수량 정보를 가져옴. 노드를 가상 서버에서 실행시켜 보드에서 접근 가능한 상태로 만들.

aws ec2 가상 서버 - <http://ec2-3-17-51-29.us-east-2.compute.amazonaws.com/>

② ARTIK 보드

<websocket-api.c>

```
int iot_websocket_connect()
{
  artik_error ret = S_OK;

  g_websocket = (artik_websocket_module *)artik_request_api_module("websocket");
  if (!g_websocket) {
    fprintf(stderr, "Websocket module is not available\n");
    return -1;
  }

  memset(&g_config, 0, sizeof(artik_websocket_config));
  g_config.uri = "ws://ec2-3-17-51-29.us-east-2.compute.amazonaws.com/";

  ret = g_websocket->websocket_request(g_handle, &g_config);
  if (ret != S_OK) {
    fprintf(stderr, "Failed to request websocket (ret=%d)\n", ret);
    return -1;
  }

  ret = g_websocket->websocket_open_stream(g_handle);
  if (ret != S_OK) {
    fprintf(stderr, "Failed to open websocket (ret=%d)\n", ret);
    return -1;
  }

  ret = g_websocket->websocket_set_receive_callback(g_handle, websocket_rx_callback, NULL);
  if (ret != S_OK) {
    fprintf(stderr, "Failed to register rx callback (ret=%d)\n", ret);
    return -1;
  }

  ret = g_websocket->websocket_set_connection_callback(g_handle, websocket_connection_callback, NULL);
  if (ret != S_OK) {
    fprintf(stderr, "Failed to register connection callback (ret=%d)\n", ret);
    return -1;
  }

  return 0;
}
```

web socket connection 관련 함수에 서버 url을 대입. 서버와 연결되면 callback 함수 호출.

callback 함수가 서버에서 정보(result 파라미터)를 받아오면 따로 정의한 event handler 함수 호출.


```

static void websocket_rx_callback(void *user_data, void *result)
{
    if (result) {
        fprintf(stderr, "RX: %s\n", (char *)result);
        //add
        if(websocket_event_handler( (char *)result)<0){
            ;
        }else{
            free(result);
            //close
            //iot_websocket_disconnect();
        }
    }
}

#pragma execution_character_set("utf-8")

int websocket_event_handler(const char* msg){

    char *message = msg;
    char *info[3];
    int i;

    info[0] = strtok (message, "/");
    info[1] = strtok (NULL, "/");
    info[2]= strtok (NULL, "/");

    . . .

    sprintf(temperature, "%s%s", info[1], "°C");
    sprintf(precipitation, "%s%s%s", "rain : ", info[2], "%");

    sh1106_write_string(4 ,1, "weather");
    sh1106_write_string(4 ,2, weather);
    sh1106_write_string(4 ,3, temperature);

    sh1106_write_string(4 ,4, precipitation);
}

```

<request.c>

받은 msg를 파싱하여 info[0],info[1],info[2]에 저장. 원하는 문자열과 합쳐서 LCD에 출력함

```

if(sample.am_data >1300){
    open_count++;
    if(open_count>5 && is_opened == 0){
        if(iot_websocket_connect()<0) printf("Error connecting \n");
        else printf("closet opened\n");
        is_opened = 1;
        open_count= 0;
        close_count= 0;
    }
}

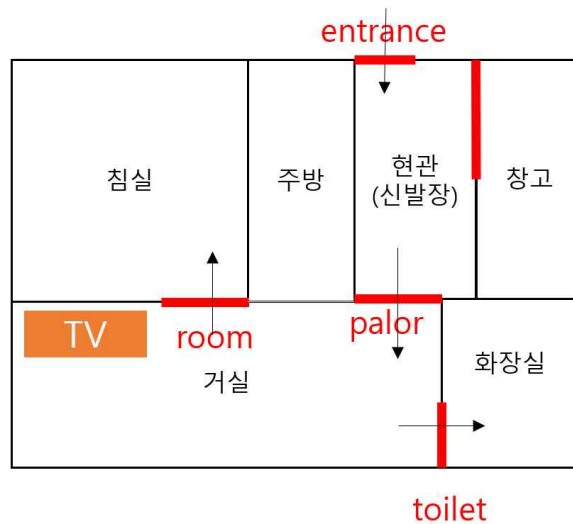
}else if(sample.am_data<1000){
    close_count++;
    if(close_count>25 && is_opened == 1){
        sh1106_clear();
        printf("closet closed\n");
        is_opened = 0;
        open_count= 0;
        close_count= 0;
    }
}
}

```

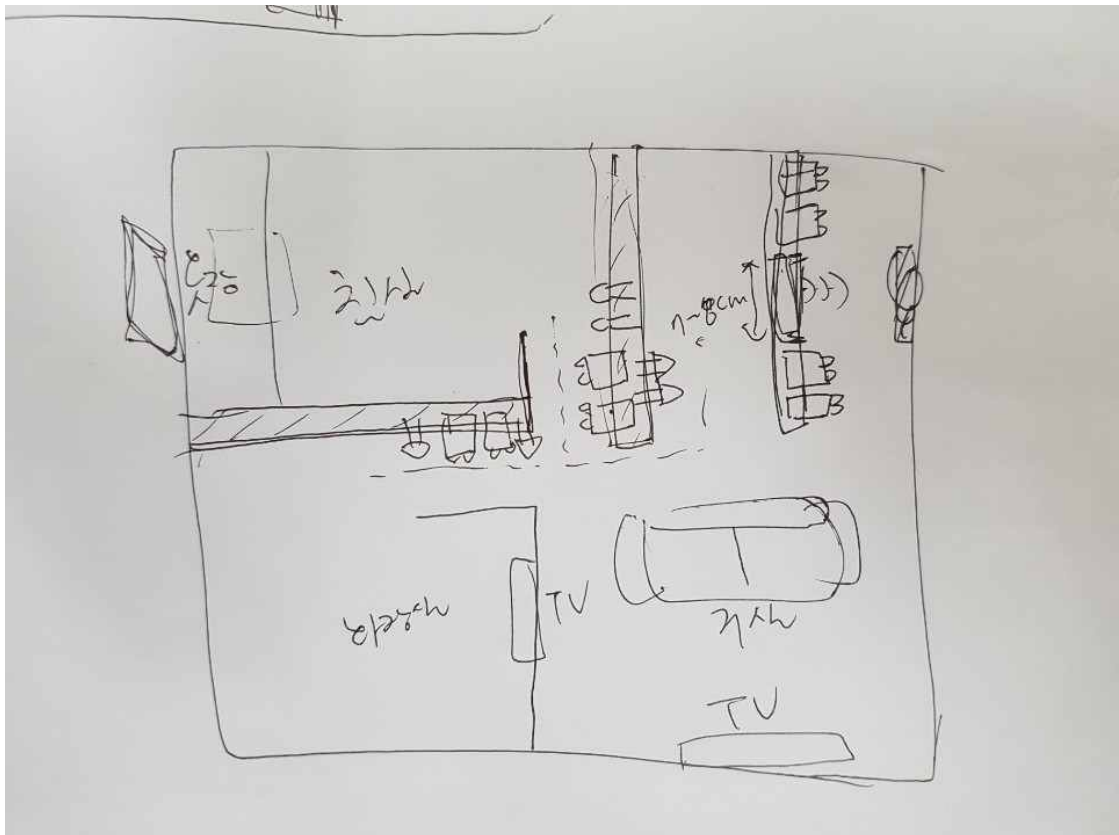
적외선 센서값이 일정 값 이상이면 옷장 문이 열렸음을 감지하고, 센서의 인식 값이 튀는 경우를 고려해 5번 카운트를 한 후 웹 소켓에 연결한다. is_opened 변수를 1로 설정하고, 열렸음이 감지되었을 때 다시 if문 안에 접근하지 못하게 한다. 문이 닫히는 것은 너무 잘 감지해서 완전히 닫히는 경우에 대략적으로 호출되는 수만큼 카운트를 한 뒤 is_opened 변수를 0으로 설정하고 LCD를 clear한다.

5) 모형 집 모델링

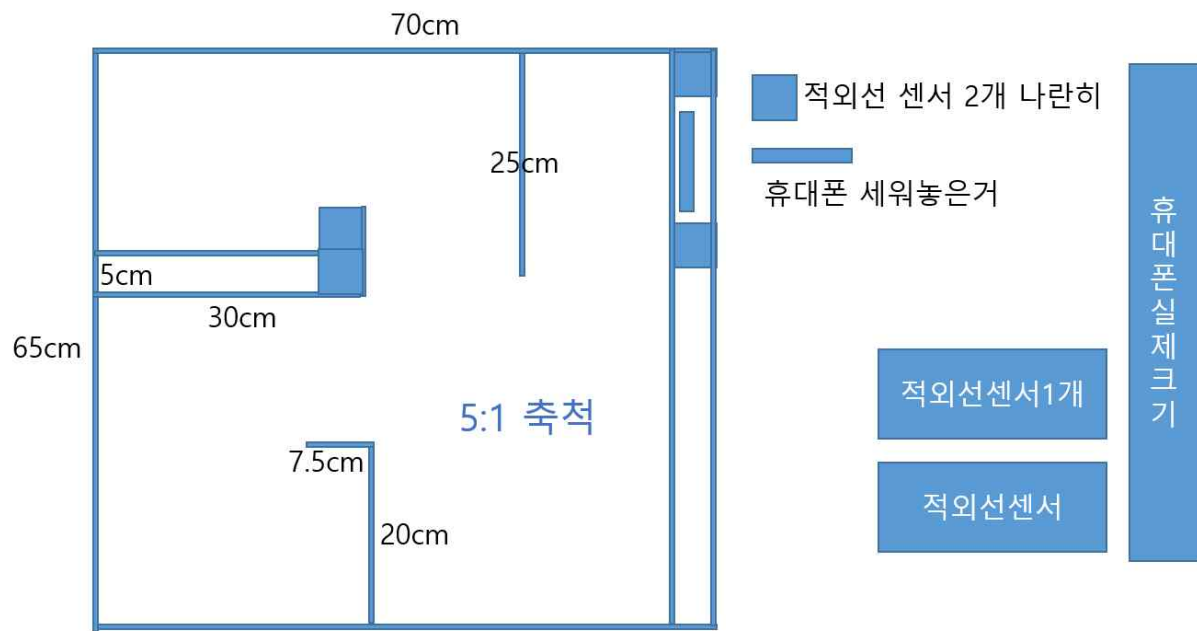
사용자 시나리오를 보여줄 수 있도록 모형 집 모델링을 한다. 최초에는 창고 시나리오가 포함되었으나 구현할 기능이 다른 방에서의 기능에 비해 아주 간단하며, 모형 집을 실제로 만들어야 하므로 실제 모델링에 부담을 느껴 창고 부분을 삭제하고, 집의 모양도 아주 간소화하였다.



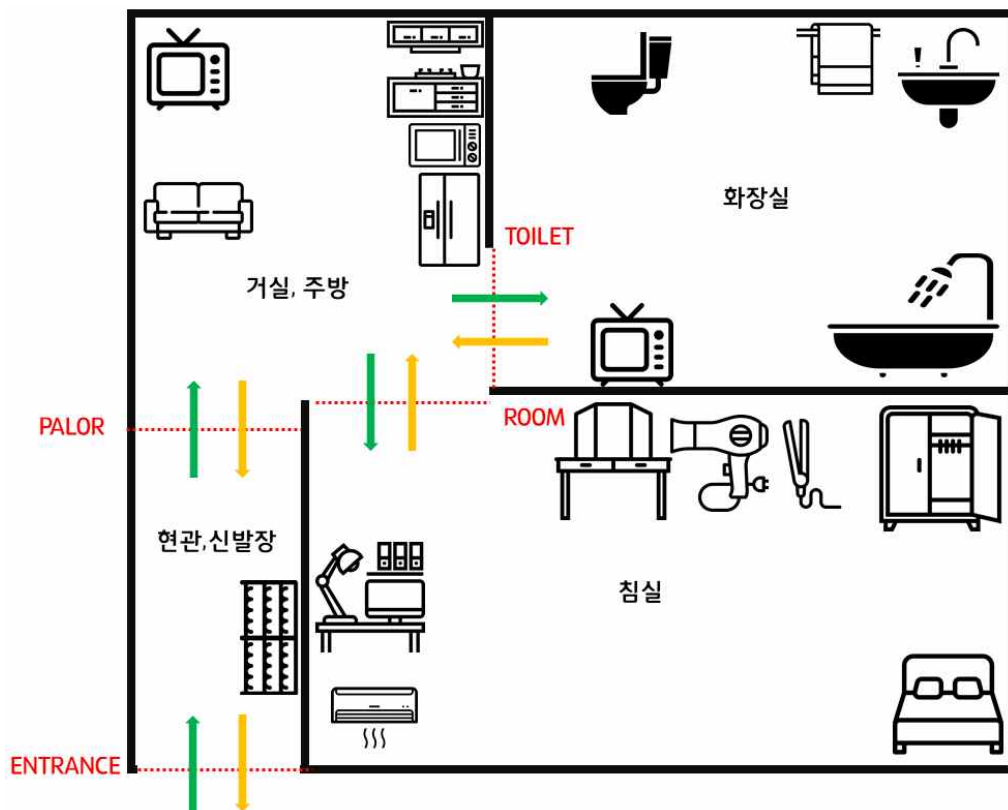
<시나리오 구현을 위한 초기 집 모형 구성도, 2018.11.26.>



<적외선 센서 및 NFC 리더 특성을 고려한 구성도, 2018.12.10.>



<실제 모델링을 위한 개략적인 수치 작성, 2018.12.11.>



<최종 완성된 집 모형 구성도, 2018.12.14.>

4. 개발 추진 실적

2018.11.12. 1차 회의 : 아이디어 회의

2018.11.13. 2차 회의 : 아이템 확정 및 요구사항 분석

2018.11.18. 프로젝트 제안서 작성 : 프로젝트 제안 배경과 목적을 명확히 함.

2018.11.26. 3차 회의 : 요구사항 상세 분석 및 개발 계획 수립, 역할 분담, 역할별 모듈/
시스템 설계 시작

2018.12.10. 1차 중간 점검 : 추가 구매 부품 분배

2018.12.11. 2차 중간 점검 : 모형 집 구성도 디자인 및 준비

2018.12.13. 모형 집 설계를 위한 재료 구매

2018.12.14. 모형 집 설계 : 실제 모형 집을 만들고 센서 부착. 시스템 통합 프로세스 시작

2018.12.19. 시스템 통합, 테스트 완료 및 발표자료 제작

2018.12.20. 프로젝트 시연

5. 문제점 및 해결내용

이름	문제점 및 해결내용
양영균 장하람	<p>① 사용자의 인원수를 고려하지 않음 : 초기에 코딩을 할 때 사용자의 인원수를 전혀 생각하지 않고 동선만을 생각하여 들어가는 것과 나가는 것으로만 구분하여 코딩을 했었다. 그 결과 집에 여러 명의 사람이 있을 경우 문제가 생겼다. 예를 들어 두 명의 사람이 거실에 있다가 한명이 방으로 이동할 경우 아직 거실에 사람이 있음에도 불구하고 동선에 따라 거실의 대기전력을 차단하고 방의 대기전력을 활성화하게 된다.</p> <p>→ 각 방별로 count 변수를 선언하여 방의 인원수도 체크하고 인원수에 따라 동선의 경우의 수를 다르게 하여 CODE를 Publish하게 만들었다.</p> <p>② Publish와 Subscribe보드의 기능을 구체적으로 구분해 놓지 않음 : 초기에 코딩을 할 때 각각의 아틱보드들의 기능을 구체적으로 구분해 놓지 않고 시작해서 코드가 상당히 중구난방이었다. 예를 들어 동선을 감지하여 service를 제공하고자 할 때, 어떤 service는 물을 통해서 Subscribe보드로 제공되고 또 어떤 service는 path track을 감지하는 보드에서 감지되는 순간 자체 코드 내에서 service를 제공하기도 했었다. 그래서 코드의 통일성도 전혀 없었고, 만약 집에 동선을 감지하여 새로운 service를 추가하고자 할 때 Publish 코드에 service코드를 추가해야 할지, Subscribe코드에 추가해야 할지 애매한 문제가 있었다.</p> <p>→ 아틱보드 두개를 하나는 Publish보드 나머지 하나는 Subscribe보드로 나누고, Publish보드에서는 Path Track과 그 결과를 Publish하는 기능 하나만 하게 하고, Subscribe보드에서는 그 코드를 바탕으로 해석하여 service를 제공하게 하기로 하였다. 그래서 path track 경우의 수부터 다시 나누고 코드를 처음부터 다시 작성했다. 그렇게 완성된 코드는 새로운 service를 추가하기도 간편하고, 코드의 통일성도 유지되게 되었다.</p> <p>③ 코드의 과적합 문제 : 각 방별로 동선이 다양하고, 제공하고자 하는 service가 다르기 때문에 우리의 코드가 정확히 우리가 설계한 집에서만 작동하고 새로운 집에 적용할 수 없는 ‘과적합’ 문제가 있었다.</p> <p>→ 코드속의 기능들을 최대한 하나의 기능만 하는 함수로 나누고 모듈화하여 새로운 집에서도 만들어 놓은 함수만 가져다가 쓰면 되게 만들었다. 만약 방이 추가된다고 해도, 방의 문에 달린 센서코드를 가져와 바로 추가할 수 있고, 그에 따른 룰도 다른 방들과 비슷하게 쉽게 추가할 수 있게 한다. 어떠한 service를 추가하고자 한다면 Subscribe보드에 간편하게 추가할 수 있게 하였다.</p>

	<p>④ Polling 방식으로 인한 문제 : 각 방의 센서들이 하나의 보드(Publisher)에 연결되어 있기 때문에 여러 개의 센서들이 동시에 작동할 수 없는 문제점이 있다. 이는 각 센서들이 Polling 방식으로 구현되어 있기 때문에 발생한다. 예를 들어, Room의 Inner 센서에 장애물이 감지되어 Outer 센서에도 장애물이 감지되기를 기다리고 있다고 하자. 만일 Outer에서도 장애물이 감지된다면 이는 사용자 A가 방 내부에서 거실로 이동했음을 나타낸다. 하지만 Outer 센서에서 장애물이 감지되기를 기다리는 와중에 다른 사용자 B가 거실에서 화장실로 이동했다고 하자. 이 코드는 Polling으로 작성되었기 때문에 Outer 센서 인풋을 기다리는 while문 안에 걸려 있어 다른 모든 서비스를 제공할 수 없기 때문에 사용자 B의 동선은 전혀 파악할 수 없다.</p> <p>→ 우리 Path Track의 최대 문제점이다. interrupt 방식으로 구현하여 해결할 수 있을 것이라고 생각했는데 프로젝트 진행이 더더지면서 시간 여유가 없어 시도하지 못했다. Further study로 남겨 놓게 되었다.</p>
장우진	<ul style="list-style-type: none"> - 웹 소켓 통신을 통해 Json형식의 스트링을 받아왔는데 Cjson파싱의 인코딩 문제로 한글 정보가 모두 깨져서 나왔다. 그래서 strtok함수를 통해 문자열을 split하여 정보를 받아왔다. - 교통 정보 크롤링을 하는데 길찾기 사이트에서는 post를 통해 정보를 비동기적으로 동작하기 때문에 request를 사용해 html내에서 정보를 가져올 수 없었고, api를 이용하려고 했지만 무료로 이용할 수 있는 api는 정보가 구글에 많이 있지 않아서 실제로 사용하기 힘들었고, 다음이나 네이버의 경우에는 길찾기 api를 구매해서 사용하여야 했기 때문에 교통 정보를 받아오는 부분은 구현할 수 없었다. - 웹 소켓 통신에 대해 무지한 상태에서 프로젝트를 진행하였기 때문에 처음에 어떠한 프로토콜로 진행하여야 하는지 이해가 많이 부족했다. 그래서 프로그래밍을 다 했는데 통신이 되지 않을 때 서버의 문제인지 보드의 문제인지를 잘 인지하지 못해서 많이 헤매면서 작업하였다. 다행히 ARTIK 홈페이지에서 websocket 도큐먼트를 찾을 수 있었고 차분히 여러 번 읽으면서 하나하나 진행해보니 보드에서 callback을 기다리고, callback을 관리하는 함수에서 정보를 받아와서 처리해야 한다는 것을 알게되었다. 또 서버에서도 disconnect하는 부분에 대한 처리를 꼭 해주어야 동작한다는 사실도 알게되었다.
문예지	<p>첫 번째로 NFC에 관한 문제이다. 중요한 물건을 놓고 갔는지 알림을 하기 위해서는 RFID를 이용하여 도서관처럼 게이트를 구성하거나, 비콘을 사용하여 실내 위치를 추적하는 방법이 가장 먼저 고려되었다. 하지만 RFID는 리더기의 가격이 비싸고(5만원 부터), 비콘은 센서 하나당 2~3만원 이상 하는 경제적 부담 때문에 NFC를 선택하게 되었다. 다들 가지고 있는 스마트폰이 NFC 리더기 역할을 할 수 있기 때문이다. NFC는 RFID를 기반으로 한 기</p>

술이라서 RFID 리더기가 생긴다면 쉽게 기술 이전이 가능할 것이다. NFC의 또 다른 문제점은 스마트폰의 NFC 인식거리가 보안 문제로 일반적인 NFC 리더기보다 짧다는 것이다. 문서에서는 4cm까지 라고 하는데 실제로 해보면 거의 딱 붙여야만 인식된다. 그리고 휴대폰으로 읽은 태그 정보가 ARTIK 클라우드를 거쳐서 보드로 들어오기 때문에 네트워크가 조금만 느려도 딜레이가 길어진다. 애초에 NFC가 가장 적절한 기술이 아닌 부분이었기 때문에 시연을 위한 타협안으로 여기고 어느정도의 문제점은 그러려니 하고 있다.

두 번째로는 LCD에 관한 문제이다. 거실 TV, 화장실 TV, 현관 LCD 총 3개의 LCD를 한 보드에서 출력해야 했다. 원래는 물건 알람을 위해 현관 LCD만 사용할 계획이었지만 거실-화장실 TV 따라오기 기능을 맡게 되면서 골치가 아파졌다. 다른 보드들도 모두 알차게 사용중이었기 때문에 가져다 쓸 수 없는 상황이었다. 여러 가지 시도를 한 끝에 공통적으로 쓰는 GND, VCC, MOSI 같은 포트는 같이 쓰고, A0와 RESET을 각각 구성하는 것으로 해결하였다. 시간이 좀 더 많았다면 코드를 개선하여 LCD 각각 개별로도 화면을 출력하고, 다같이도 화면을 출력할 수 있도록 하였겠지만, 일단은 3개 중 하나씩만 출력되도록 하였다. 특정 LCD에 출력해야 하는 정보가 들어오면 해당 LCD를 초기화 하고 사용하는 방식이다.

6. 개인 기여내용 및 배운 점

이름	기여 내용 및 배운 점
양영균	<p>Path Track 서비스 설계(Pub/Sub), 요구사항 분석 및 시나리오 설계, 보고서 작성</p> <p>실제로 센서를 여러 개 사용하고 보드를 구성하면서, 하드웨어 테스트의 중요성을 뼈저리게 느꼈다. 일단 하드웨어 테스트를 하며 센서의 특성을 파악해야 코딩을 시작할 수 있는데, 테스트 할 센서가 10개가 넘다보니 브레드보드에서 테스트 할 수도 없었다. 해당 테스트를 위해 아주 큰 브레드보드를 빌려 테스트를 통과했다.</p> <p>이 때 브레드보드를 빌리느라 기일이 많이 초과되어 처음 Path Track 서비스 Publisher를 코딩할 때, 사용자의 동선 분석에 따른 CODE 작성을 깊게 생각하지 않고 대충 알고리즘을 구성했다. 당시에는 그 알고리즘으로 모두 표현할 수 있다고 생각하여 통합 테스트 직전까지도 수정하지 않았다. 장하람 팀원과 Pub/Sub 간의 토픽으로부터 ARTIK cloud Rule을 생성하던 중에 해당 알고리즘으로는 도저히 표현할 수 없는 경우의 수가 있음을 알게 되었다.</p> <p>이 문제를 해결하기 위한 디버깅 과정이 정말 오래 걸렸다. 장하람 팀원이 며칠 간 고민하여 문제를 해결하긴 했으나, 최초 시스템 설계에서 깊게 생각하지 않고 주먹구구식으로 코딩을 했던 것 때문에 며칠씩 이 문제로 고생을 하게 되었다. 계획적이고 치밀한 알고리즘 설계의 필요성을 크게 느낄 수 있었다.</p> <p>End user의 입장에서 요구사항을 생각해내고 그를 바탕으로 구현할 기능을 생각해내는 것은 재미있었다. 실제로 내가 겪은 ‘사소한 불편함’에 대한 아이디어가 끝없이 나왔으며, 이를 모두 해결하는 기능 및 시나리오를 구현하고 싶었으나 기술적인 난이도, 하드웨어 부품 수급 및 개발 시간 부족 등으로 구현하지 못한 것이 많이 아쉽다. 아래의 Further study에서 소개하고자 한다.</p>
장하람	<p>Path Track 서비스 Rule 구축 및 최적화, Safety mode에서 도둑 경로탐색 코드 구현, 진동감지 센서와 불꽃감지 센서를 이용한 재난 대응 시스템 구현</p> <p>이번 프로젝트를 하면서 같은 코드를 여러번 뒤집고 다시 처음부터 설계를 한적이 많았다. 그러면서 느낀점은 디버깅은 지겹고 힘든 작업임과 동시에 보람있는 작업이라는 것과, 꼬일대로 꼬인 코드를 고치는 것보다 그 문제를 정확히 인식하고 처음부터 새로 작성하는 것이 훨씬 빠르다는 것을 느끼게 되었다. 또 여태까지는 코드의 모듈화와 최적화를 신경써서 코딩을 한적이 없었는데 이번에 코딩을 할 때 코드 수정을 하도 많이 하다 보니까 코드의 모듈화와 최적화가 얼마나 중요한 것인지 알게 되었다. 코드의 길이가 길다 보니, 작은 기능 하나를 수정을 할 때 모듈화가 되어있는 코드와 아닌 코드의 수정해야하는 작업의 능률차이가 현격하게 차이가 났다. 앞으로 코딩을 할 때는 이러한 부분들을 신경써서 하기로 다짐했다.</p> <p>또 프로젝트를 할 때 크게 보면 LCD 부분과 NFC부분과 Path Track부분을 나누어서 각자의 역할로 맡겨 진행했는데, 각자의 결과를 다시 합치는 부분이 상당히 어려웠다. 처음에 생각했을 때는 각자 해온 것을 ARTIK</p>

	<p>cloud에서 rule로 연결만 하면 쉽게 끝낼 줄 알았는데, 각자 맡은 부분의 기능을 제대로 이해하지 못하면 우리가 원하는 결과와 다른 결과를 보이게 되었다. 그래서 서로에게 자신이 한 일과 작동하는 방식을 자세하게 설명해 준 후에야 우리가 원하는 방식대로 모듈들이 작동하게 rule로 묶을 수 있었다. 앞으로도 협업을 할 때 협업을 하는 상대방에게 우리가 만든 모듈을 자세하게 설명하고, 상대의 모듈도 어느정도 이해를 하면서 의견조정을 하면서 진행해야 겠다고 생각했다.</p>
장우진	<p style="text-align: center;">교통정보, 날씨정보 LCD출력</p> <p>웹 소켓 통신을 통해 웹 서버와 ARTIK보드와 통신하는 부분을 맡았다. 컴퓨터 네트워크 과목을 듣지 않아서 통신에 대한 부분이 스스로 많이 부족하다고 생각했었는데, 이번에 보드와 통신하는 부분을 맡으면서 조금은 네트워크 통신에 대해 이해할 수 있는 시간이었던 것 같다. 우선 처음에는 웹 사이트의 정보를 크롤링 해오는 서버부터 구현하였는데, 다행히 네이버 날씨가 지역별로의 실시간 날씨를 받아올 수 있게 되어있어서 임의로 우리 스마트 홈의 지역을 서울로 설정해놓고 정보를 받아올 수 있었다. 보드와의 통신 없이 서버에 정보를 출력하게만 구현하여서 express, cheerio, request모듈을 사용해서 크롤링을 구현했다. 그리고 나서 웹 소켓에 대한 부분을 구현하려고 했는데, 보드에서 개인 local host로 연 서버에 접근을 할 수 없어 AWS EC2에 가상 서버를 열고, 80번 포트를 통해 ARTIK과 통신했다. express 대신 ws 프로토콜을 사용하여 웹 소켓 통신을 하였다. JSON형식으로 정보를 보내서 Cjson 파싱을 이용하려 했는데 정보를 받아서 출력을 하면 json형식으로 스트링이 잘 출력이 되는데, 파싱을 한 결과는 밑에서 볼 수 있듯 한글 인코딩이 깨지는 문제가 발생하였다.</p> <pre> msg : {"morning": {"weather": "구름많음", "temperature": "1.0", "precipitation": "20"}, "evening": {"weather": "맑음", emperature": "9.0", "precipitation": "0"} } json : weather : P@ , temperature:💎@ 💎💎 , precipitation: </pre> <p>하여 “날씨/온도/강수량”의 형식으로 정보를 받아와서 문자열 함수를써서 /을 기준으로 문자열을 나눠 정보를 가져왔다. 평소에 웹 서버에 대한 지식이 많이 부족하다고 생각했었는데, 이번 프로젝트를 진행하면서 꾸준히 관심이 있던 임베디드, IoT분야와 평소 부족하던 웹 서버, 통신 분야의 지식을 같이 발전시켜 나갈 수 있어서 정말 감사했다. 또 교통 정보를 받아와서 LCD에 출력하는 부분도 맡았었는데, 길찾기 사이트에서는 post를 통해 정보를 비동기적으로 출력해 주기 때문에 request를 사용해 html내에서 정보를 가져올 수 없었고, api를 이용하려고 했지만 무료로 이용할 수 있는 api는 정보가 구글에 많이 있지 않아서 실제로 사용하기 힘들었고, 다음이나 네이버의 경우에는 길찾기 api를 구매해서 사용하여야 했기 때문에 교통 정보를 받아오는 부분은 구현할 수 없었다. 프로젝트가 끝나고 개인적으로 서버에 대해 더 공부를 해야겠다고 결심할 수 있었다.</p>

문예지	NFC를 이용한 놓고 간 물건 알림과 거실-화장실 TV 따라오기 기능 구현
	<p>안드로이드 개발을 했었기 때문에 어느정도 수월하게 진행하겠거니 했다. 하지만 약간 익숙하다는 장점만 있을 뿐, 새로운 분야를 만나면 삽질하는 것은 매한가지였다. ARTIK에서 제공하는 안드로이드 앱 만들기 문서를 보았는데 옛날에 만들어진 문서라 버전 문제도 있었고, 평소 안드로이드 개발을 코틀린으로 해서 자바 코드가 읽기 힘들기도 했다. 또한 관련된 설정이 안드로이드의 Manifest, Gradle, 주어진 코드 모음 중 Config 파일, Artik Application과 Device type 설정, Cloud 정보 까지 곳곳에 흩어져 있어서 매우 헷갈렸다. 가장 먼저 주어진 자바 예제코드가 잘 실행될 때까지 설정을 만지고, 자바 코드를 코틀린으로 변환하고, 코드를 분석하고 필요한 부분을 묶는 과정을 거쳐 안드로이드에서 Artik 파트를 구성하였다. 병행적으로 NFC 또한 잘 작동하는 예제 코드를 찾고, 코틀린으로 변환하고, 코드를 정리해서 구조를 파악하여 NFC 파트를 구성하였다. 이렇게 만들어진 두 개의 프로젝트를 잘 합쳐서 프로젝트에 필요한 안드로이드 앱을 만들 수 있었다.</p>
	<p>마지막에 시연할 때 TV 화면 따라오기가 제대로 작동이 되지 않았는데, 다음날 영상 촬영을 위해 디버깅을 하려고 로그를 찍어보려고 하니깐 너무 잘됐다. 아마 시연 당시에는 네트워크 문제로 제대로 작동이 안된 것 같다. 시연 직전까지만 해도 문제가 없었던 부분인데 딱 시연때만 안되서 좀 속상했다.</p> <p>그래도 이번에 팀원들을 잘 만나서 너무 좋았다. 다들 자기가 맡은 역할을 훌륭하게 해내고, 주도적으로 프로젝트를 진행해주었다. 그래서 4인분에 걸맞는 규모의 프로젝트가 완성된 것 같다.</p>

7. Further study

1. NFC 태그를 이용한 맞춤 루틴 설정

NFC 태그가 좀 남았는데, 우선순위가 밀려서 결국 구현하지 못했지만 NFC 태깅을 통해 특정 장소에서 특정한 생활 루틴을 작동시키는 시나리오가 있었다. 예를 들면, 침대 옆 협탁에 NFC 태그를 붙여놓고 자기 전 휴대폰을 협탁에 올려두면 태그를 인식해서 안방 불을 끈다든가, 거실장에 휴대폰을 올려두면 스피커에서 음악이 흘러나온다든가 하는 것이다.

2. 교통정보 크롤링

기술적인 한계에 부딪혀 구현하지 못했던 부분을 좀 더 지식을 보완해서 마저 구현하려고 한다.

3. Interrupt로 Path Track 구현

interrupt코드를 완벽하게 이해할 시간이 충분치 못해서 현재 Path Track코드는 전부 polling방식으로 짜여 있다. 따라서 동시에 여러 명이 각각 다른 문을 통과하는 출입하는 상황을 완벽히 잡아내지 못하는 상황이다. 이 문제를 해결하기 위해 Path Track코드를 모두 interrupt 방식으로 바꿔야 한다. 여러 명이 동시에 움직이더라도 모든 움직임을 파악할 수 있어야 실제 가정에 적용할 수 있기 때문에, 프로젝트가 끝난 후 개인적으로 interrupt방식으로 구현하려고 한다.

4. 기타 구현하지 못한 '사소한 불편함' 해결 시나리오

1. Path Track을 통한 창고 불 켜고 끄기

→ (양영균 조원의 실제 '사소한 불편함') 집에서 맥주 보관을 창고에 하고, 냉장고에 맥주가 떨어지면 창고에서 꺼내서 맥주를 채워 넣는다. 이 때 창고가 난잡하기 때문에 불을 안 키면 맥주를 찾을 수 없고, 맥주를 찾고 나서는 불을 끄지 못해 어깨로 더듬어서 끄거나 맥주를 가져다 놓고 다시 와서 불을 꺼야 한다. 실제 일상생활에서 PIR 센서에 의한 신발장 조명 점등/소등 문제(쭈그려 앉아서 신발끈을 묶다 보면 나를 인식하지 못해 조명이 꺼짐. 앉은 상태에서 다시 PIR 센서에 인식할 방법이 없어 손을 높이 뻗어 흔들거나 일어났다 앉아야 함) 다음으로 불편한 '사소한 불편함'이다. 꼭 추가하고 싶은 기능이다.

2. 자동 블라인드

① 사용자가 설정한 시각에 자동으로 침실 블라인드 on/off

→ (양영균 조원의 실제 ‘사소한 불편함’) 기상 시간에 사용자의 기상에 도움을 준다. 실제로 침실에 암막 커튼을 사용하고 있는데, 숙면에 큰 도움이 되나 아침에도 한밤중같이 어두워 자꾸 다시 잠들뿐 한 적이 많다.

② 온습도 센서 및 조도센서와 연동하여 거실 블라인드 on/off

→ 집에서 키우는 식물의 성장에 적절한 태양광을 노출한다. 즉, 사용자가 집의 블라인드를 닫아 놓은 채로 외출한 상태가 오래 지속된다면, 식물의 생장에 좋지 않으므로 블라인드를 적절히 열어주어 태양광을 받을 수 있도록 한다.

③ 난방/냉방 시 태양광 조명/블라인드 단열의 효율 비교하여 on/off

→ 겨울철 난방 시, 외부의 찬 공기를 조금이라도 단열하는 (블라인드를 off하는) 것이 유리할 때가 있는 반면, 블라인드를 열어 태양광으로부터 열에너지를 얻는 것이 유리할 때가 있다. 온도 센서와 조도센서를 활용하여 어느 쪽이 더 효율적인지 판단하여 난방비를 절약할 수 있도록 한다.

5. 빅데이터 분석을 통한 개발자의 명시적 개입 없는 능동적 서비스

위 Further study에서 다룬 내용을 모두 실제 시스템 구현에 성공한다면, 이 데이터들을 충분히 수집하여 빅데이터 분석을 통해 지금과 같은 사용자의 명시적 개입(알고리즘 작성 및 CODE Publish/Subscribe) 없이 인공지능 스스로 상황을 판단하고 빅데이터를 기반으로 사용자가 원하는 서비스를 예측하여 제공할 수 있을 것으로 기대한다.

[별첨] 프로젝트 상세 코드

양영균, 장하람

https://github.com/Haramble/PathTrack_IoT_Project_Publish

ARTIK 보드 1 : Path Track, 재난 감지 시스템 코드 Publisher

https://github.com/Haramble/PathTrack_IoT_Project_Subscribe

ARTIK 보드 2 : topic(CODE) Subscriber

문예지

https://github.com/mijey21/KAU_IoT_Project

안드로이드 APP(for NFC)와 Artik 코드

장우진

<https://github.com/W00JIN/IoT>

Web crawling server와 socket 통신