
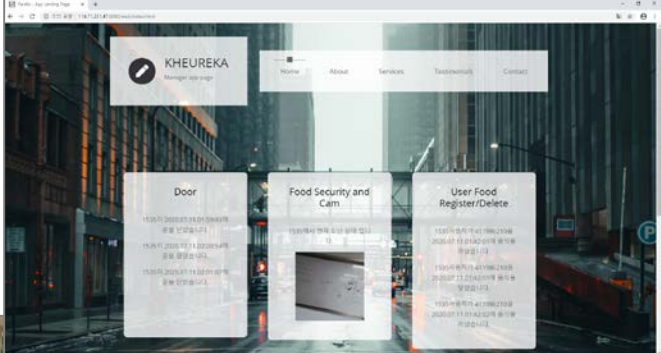



제18회 임베디드SW경진대회 개발완료보고서

[자유공모]

□ 개발 요약

팀 명	EMBEDDEDLABS
	 
작품명	공용 냉장고 內 도난 방지 시스템
작품설명 (요약)	공용 냉장고 이용 시 일어날 수 있는 음식 통의 도난을 상황을 방지하고 관리를 해줌으로써 사용자에게 음식 통의 존재 여부와 상태에 대한 신뢰성을 제공하고, 시간 절약 등 편리함을 제공하기 위해서 설계하게 되었습니다.
소스코드	https://github.com/yujinkim7123/2020ESWContest_free_1057
시연동영상	https://youtu.be/KEjVVqsrYzA

□ 개발 개요

○ 개발 작품 개요

- 기존의 공용 냉장고는 사용자의 양심에 보안을 맡기는 보안성이 결여된 형태이다. 본 시스템은 IoT 센서들을 장착하여 냉장고 內 물품(음식)을 꺼내는 행위와 도난 행위를 구분하고 사용자 앱을 통해 물품(음식 통) 관리를 제공하며, 발각된 도난 현장을 촬영해 관리자에게 경고 및 알림을 주는 시스템이다.

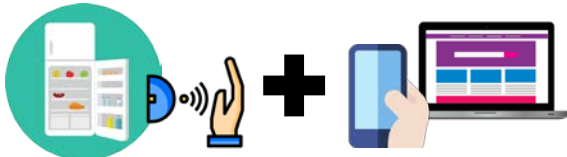
○ 개발 목표

- 공용 냉장고 사용에 있어서 **편리함 ↑**: 사용자 앱을 통한 물품 관리(종류, 보관 날짜 확인)의 편리함, 관리자 페이지를 통한 냉장고 관리의 편리함 증대.
- 공용 냉장고 사용에 있어서 **신뢰성, 보안 ↑**: 사용자 앱을 통한 도난 상황 알림, 관리자 페이지를 통한 도난 상황 인식, 근무지 순찰 시 메일을 통한 도난 상황 알림을 통한 신뢰성과 보안 증대.

○ 개발 작품의 필요성

- 대학교 기숙사에 비치된 냉장고는 공동 사용을 위한 냉장고입니다. 이런 경우 여러 사람에게 음식 통이 섞여서 보관되고 음식 통 관리가 어려워져서 자연스레 도난에 위험도 커진다. 누군가 남의 음식 통에 음식을 손을 대는 일이 일어나도 알기 어렵습니다. 이러한 문제점 해결을 위해 음식 통의 도난을 방지하고 관리를 해주어 사용자가 음식 통 관리나 자신이 놓은 곳에 있는지에 대한 신뢰성을 지키기 위한 노력과 시간을 절약해주고 편리함과 신뢰성, 보안을 제공하기 위해서 설계하게 되었습니다.

○ 기본 원리



- 공용냉장고에 실시간 상태 감시용 센서 추가 (IoT)
- 사용자(앱)와 관리자(웹 페이지, 메일)가 도난 감지 모니터링 가능
- 사용자(앱)의 물품 관리 기능

□ 개발 환경 설명

○ Hardware 구성

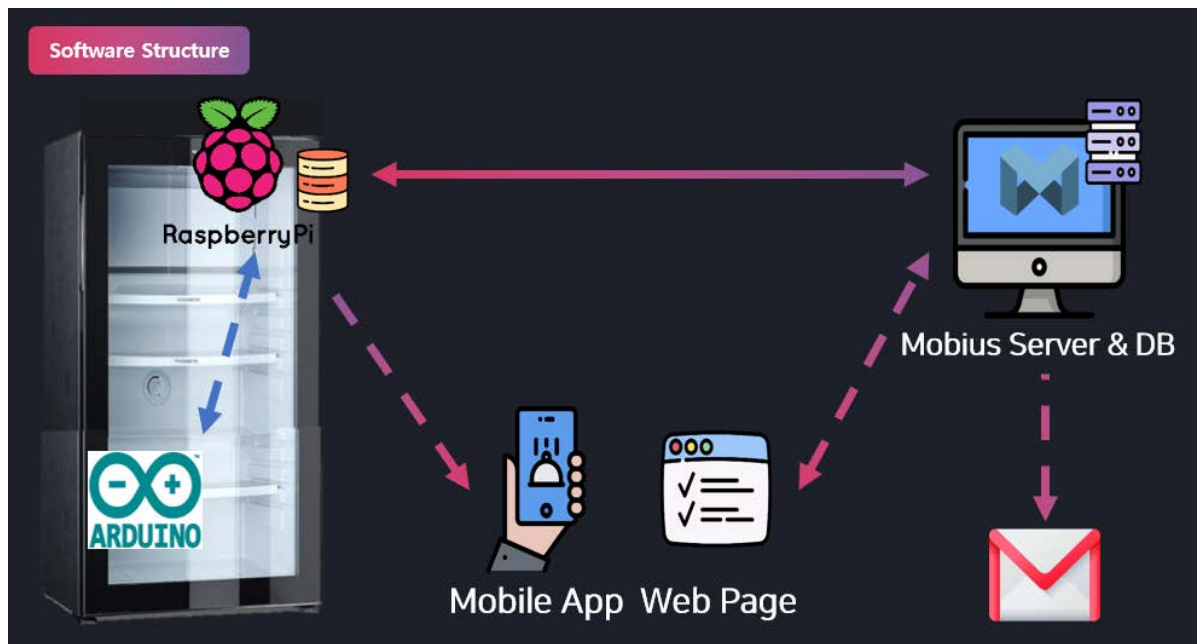
- 아두이노: 적외선 센서(IR), LCD, 자석 센서, RFID Reader, 솔레노이드, 블루투스 모듈 제어
- 라즈베리 파이: 무게 센서, 카메라 모듈 제어
- RFID 스티커(물건 부착용, 반영구 사용)



○ Hardware 기능 (제어 방법 등 서술)

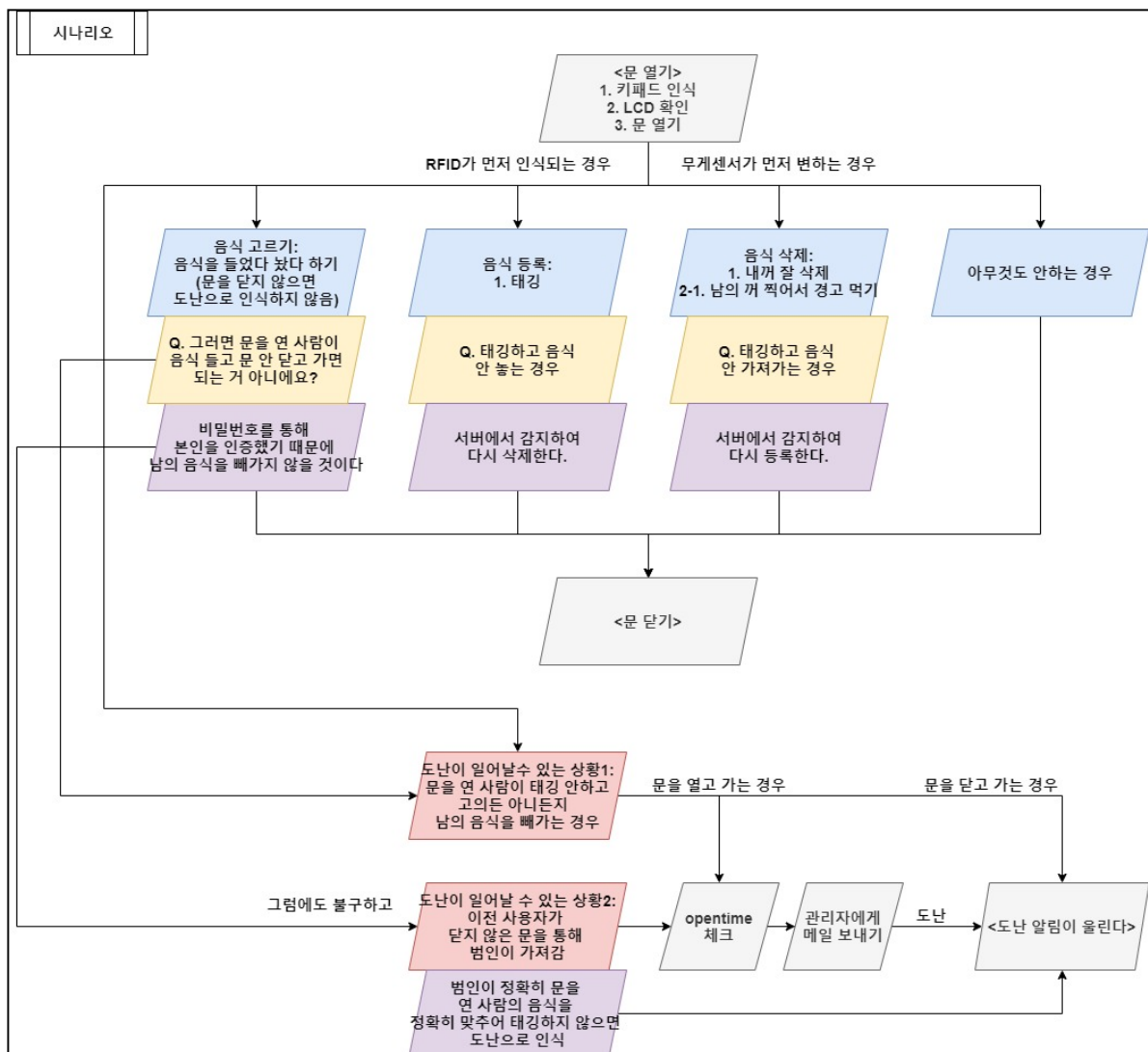
- 아두이노:
 - **적외선 센서**로 리모컨의 번호를 입력 받고, 이를 **LCD**에 표시한다.
 - 입력된 정보가 **블루투스 모듈**을 이용해 라즈베리 파이에서 등록된 사용자의 고유 번호로 확인되면 **솔레노이드**를 작동하여 문 잠금을 해제한다.
 - 문이 열리고 물건(음식 통)을 등록/제거할 때 **RFID Reader**에 RFID 스티커 태깅을 통해 서버에 물건을 등록한다.
 - 문이 닫혔는지 **자석 센서**를 통해 확인하고 3초 연속으로 닫혀 있을 경우 솔레노이드로 문을 잠그고 잠긴 상태를 라즈베리 파이를 통해 서버에 저장한다.
- 라즈베리 파이:
 - **무게 센서**는 문이 열리기 전, 문이 닫힌 후 무게를 측정한다.
 - 도난 상황 시 **카메라 모듈**로 촬영한다.

○ Software 구성



○ Software 설계도 (흐름도 및 클래스 다이어그램 등 / 개발언어에 따라 선택)

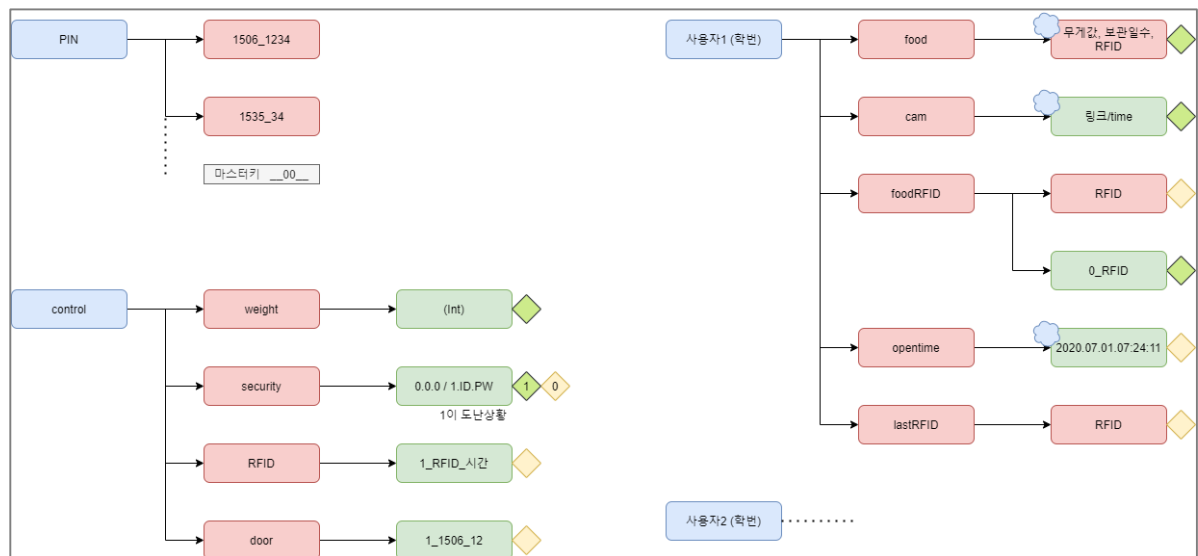
- 시나리오 흐름도



- 실험 과정 예시

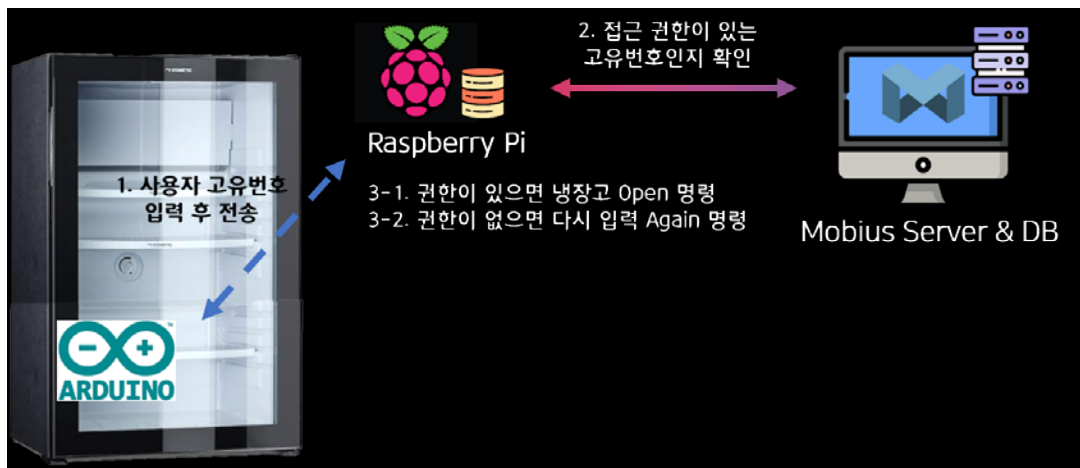


- Mobius 서버 구조



○ Software 기능 (필요 시 알고리즘 설명 포함)

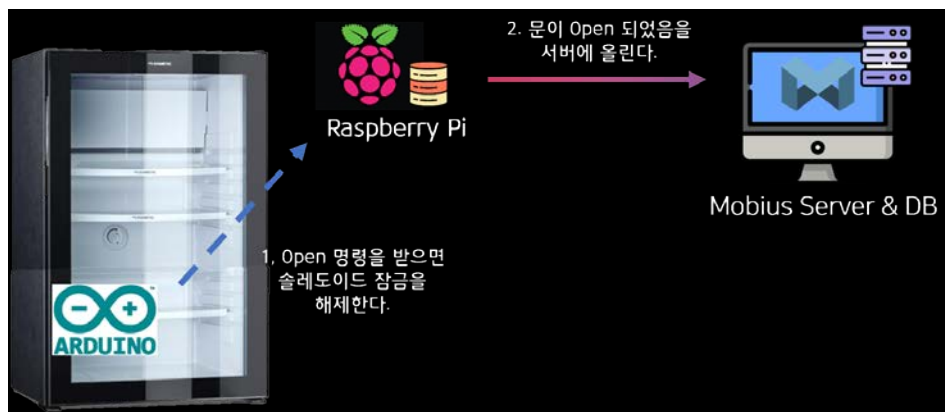
1. 냉장고의 리모컨을 통해 입력 받은 사용자의 고유 번호에 대해 냉장고의 접근권한을 확인하고 승인 여부 판단한다.
 - [1] 사용자는 리모컨을 통해 할당 받았던 고유번호를 입력한다.
 - [2] 리모컨에 입력된 사용자의 고유번호는 RPI로 전송된다.
 - [3] RPI는 서버로 고유번호를 전송한다.
 - [4] 서버는 DB에 사용자의 고유번호 있는지 확인 요청한다.
 - [5] DB는 요청 정보에 대해 반환한다.
 - [6] 반환 값에 따라 서버는 접근 권한 승인 여부를 판단한다.
 - [7] 서버는 접근 권한 승인 여부를 RPI로 전송한다.



2. 접근권한 승인 여부에 따라서 문 개폐 해야 한다.

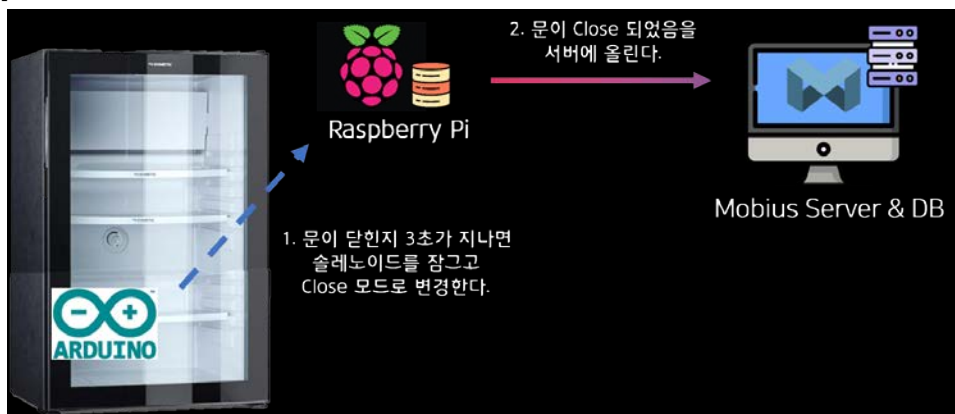
[문 열기]

- [1] 입력 받은 번호가 DB 에서 권한이 있는 고유번호 중 하나인지 확인한다.
- [2] 확인된 번호이면 아두이노에 문 열림을 명령한다.
- [3] 아두이노의 솔레노이드 동작하여 잠금을 해제한다.
- [4] 서버에 잠금이 해제되었음을 전송한다.



[문 닫기]

- [1] 냉장고 내부의 리드스위치가 냉장고 문의 자석과 닿고 일정시간이(ex. 3 초) 지속되면 아두이노의 솔레노이드 동작하여 잠금을 설정한다.
- [2] 서버에 잠금 설정되었음을 전송한다.

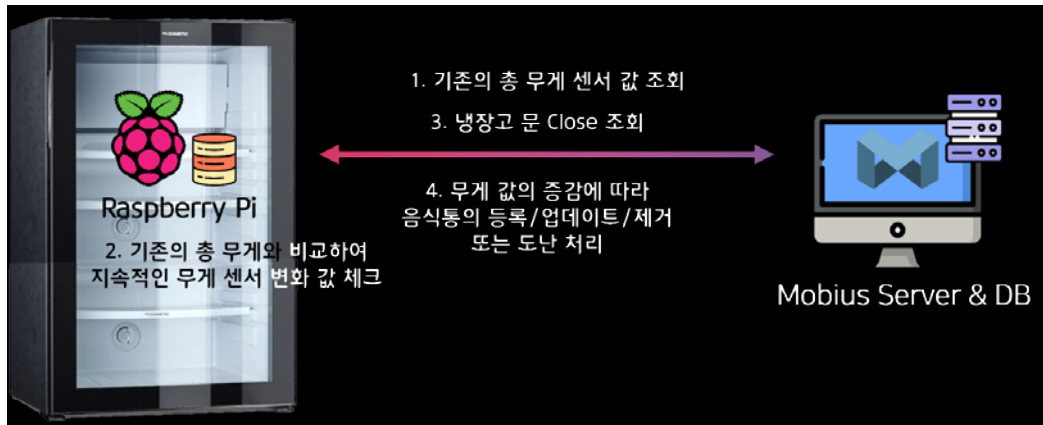


3. 사용자가 냉장고 이용 시(즉, 문 열림 판단 시) 지속적인 무게 센서의 값 변화를 인식 하여 모든 음식 통의 존재와 상태를 확인해야 한다.

- [1] RPI는 문 열림 판단 시 서버에 냉장고 내 음식 통의 총 무게에 대한 직전 무게 센

서 값을 서버에서 조회한다.

- [2] RPI는 서버로부터 반환된 무게 센서 값을 저장한다.
- [3] RPI는 지속적으로 현재 무게 센서 값을 체크한다.
- [4] 서버는 RPI로부터 반환된 현재 무게 센서 값과 DB에서 반환된 최근 무게 센서 값을 비교하여 변화가 있는지 판단하여 등록/업데이트/제거/도난 처리한다.

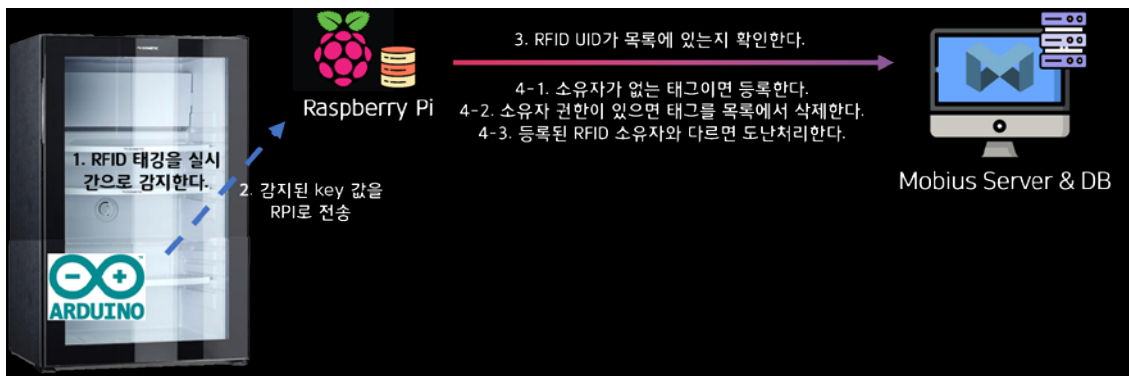


- 4. 사용자가 냉장고 이용 시 (즉, 문 열림 판단 시) 무게 센서를 통해 총 음식 통의 무게 값 변화 시 RFID 태그 인식 필요하다는 LCD 알림을 한다.

- [1] RPI는 무게 센서 값 비교 중 값이 변화했을 경우,
- [2] 서버에 RFID가 태그 되었는지 조회한다.
- [3] RFID가 태그되지 않았으면,
- [4] 아두이노는 LCD에 RFID 태그 인식하라는 메시지를 출력한다.

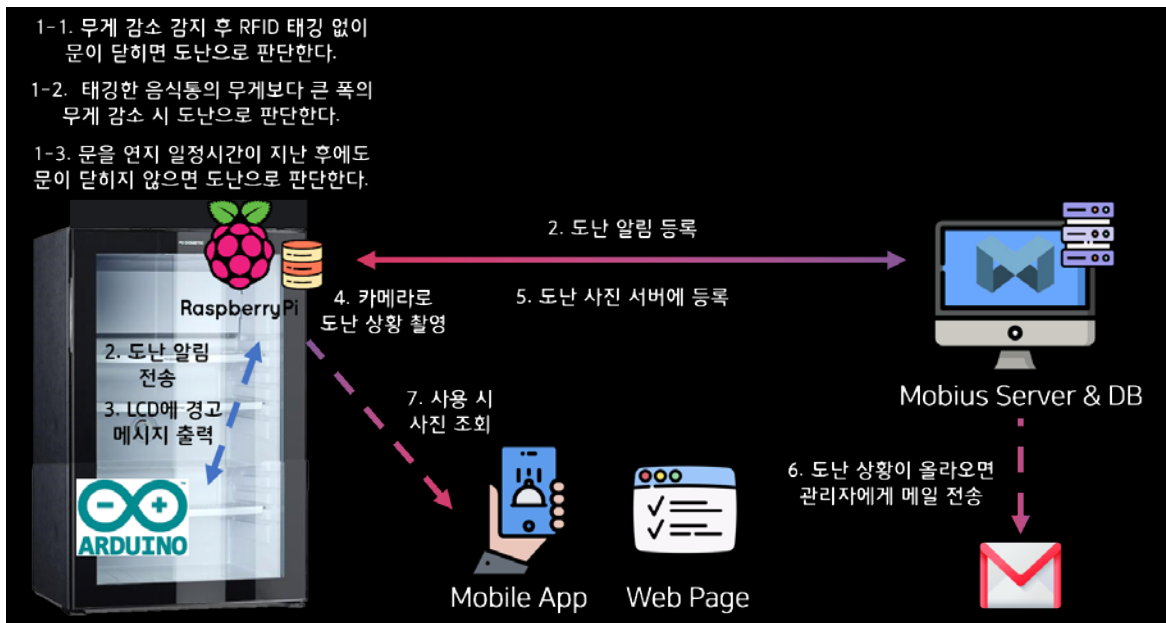


- 5. 사용자가 음식 통의 소유자 권한이 있을 경우 RFID 태그 등록/제거 한다.
- [1] 문이 열린 상태일 때, RFID reader가 RFID 태그가 근처에 있는지 실시간으로 확인한다.
- [2] RFID 태그가 인식되면 해당 RFID 태그의 key값을 읽는다.
- [3] 읽은 값에서 필요한 값(식별 값)만 추출하여 블루투스를 통해 RPI 서버에 전달하여 등록된 값인지 확인한다.
- [4-1] 처음 전달된 RFID일 시, 고유 번호와 RFID 태그를 DB에 <등록>으로 추가한다.
- [4-2] 등록된 RFID 태그임이 확인될 시, 현 사용자가 입력한 고유 번호와 소유자의 고유번호가 동일한지 확인한다.
- [4-3] 등록된 RFID 소유자와 현 사용자가 다를 시, 도난 상황으로 인식한다.



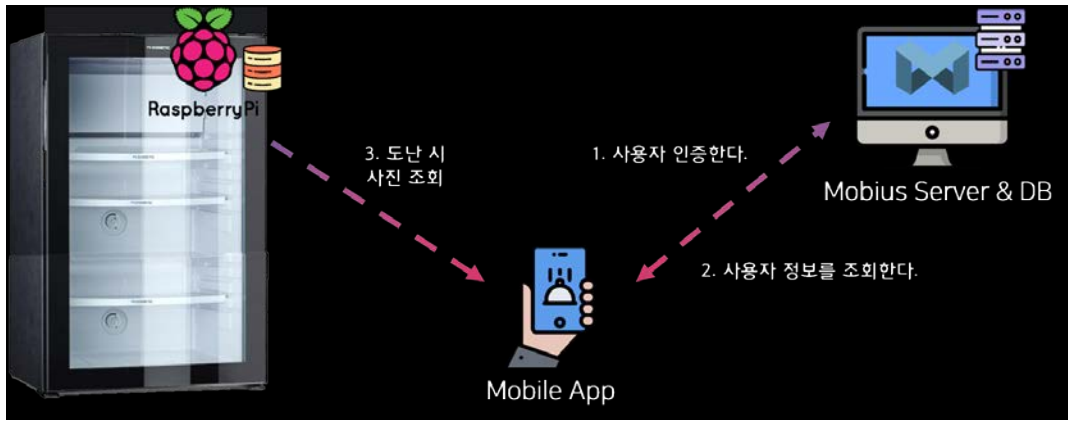
6. 음식 통 도난 판단 시 LCD경고 표시, 사진 촬영하여 관리자의 웹페이지와 도난 한 사용자의 어플리케이션에 전송 및 경고한다.

- [1-1] RPI는 무게 값의 감소했음에도 RFID 인식 없이 문이 닫힌 경우 도난판단한다.
[1-2] RPI는 태깅한 음식통의 무게보다 큰 폭 감소하여 문이 닫힐 경우 도난판단한다.
[1-3] RPI는 문을 연지 일정시간이 지난 후에도 문이 닫히지 않을 경우 도난판단한다.
[2] RPI는 도난 알림을 서버와 아두이노로 전송한다.
[3] 도난 알림을 받은 아두이노는 LCD 경고 메시지를 출력한다.
[4] RPI는 카메라로 도난 상황을 촬영하여 저장한다.
[5] 촬영 완료 후 RPI는 사진을 서버로 전송한다.
[6] 관리자 메일로 도난 의심 상황을 전달한다.
[7] 관리자 웹페이지와 사용자 어플리케이션 사용 시 사진을 조회한다.



7. 등록된 RFID태그는 사용자가 어플리케이션에서 조회할 수 있어야 한다.

- [1] 어플리케이션을 실행하면 사용자 로그인하여 인증한다.
[2] 서버에서 최근 냉장고 사용 시간, 냉장고 음식통 목록 조회한다.
[3] RPI 에서도난한 사진을 조회한다.



○ 프로그램 시나리오

<p><일반 시나리오></p> <ol style="list-style-type: none"> 1. 사용자 접근 권한 얻기 2. RFID 등록 3. RFID 삭제 4. 등록 되어 있는 RFID 음식 통 값 업데이트 	<p><도난></p> <ol style="list-style-type: none"> 7. 문을 열고 태깅을 하지 않고 가져간 경우 8. 태깅 한 음식이 아닌 다른 사람 음식을 가져간 경우 9. 문을 열어놓고 음식을 가져간 경우
<p><사용자 실수></p> <ol style="list-style-type: none"> 5. RFID 등록 태깅 후 음식 통을 안 넣는 경우 6. RFID 삭제 태깅 후 음식 통을 안 꺼내는 경우 	<p><도난 알림></p> <ol style="list-style-type: none"> 10. 앱에서 확인, 메일에서 확인 11. 관리자의 마스터 키로 도난 상태 해제

<일반 시나리오>

1. 사용자 접근 권한 얻기
 - 문을 열기 위해 ID/PW를 입력해 등록된 사용자 목록 확인
2. RFID 등록 (음식 통 보관하기)
 - 음식 통에 RFID 스티커를 부착 후,
 - RFID reader에 태깅 하면,
 - RFID의 uid가 사용자 Mobius에 등록된다.
3. RFID 삭제 (음식 통 꺼내기)
 - 냉장고의 음식 통을 꺼낸 뒤,
 - RFID reader에 태깅 하면,
 - RFID의 uid가 사용자 Mobius에서 삭제 및 삭제 기록된다.
4. 등록 되어 있는 RFID 음식통 값 업데이트
 - 냉장고에 보관했다가 꺼낸 음식을 일정 양 먹은 뒤 사용했던 RFID를 재 부착 후, (또는 탈착하지 않은 후)
 - RFID reader에 태깅하면,
 - RFID의 uid가 사용되었던 uid목록에서 인식되면,
 - uid 무게가 업데이트 된다.

<사용자 실수>

5. RFID 등록 태깅 후 음식 통을 안 넣는 경우

- 음식 통에 부착된 RFID 스티커 태깅 후,
 - 사용자가 음식 통을 **놓지 않으면**,
 - foodRFID에 1_RFID를 올려
 - Mobius 서버에서 RFID **등록을 취소**한다.
6. **RFID 삭제 태깅 후 음식통을 안 꺼내는 경우**
- 음식통을 꺼내서 RFID 스티커 태깅 후,
 - 사용자가 음식 통을 냉장고에 두고 **꺼내지 않으면**,
 - foodRFID 에 0_RFID를 올려,
 - Mobius 서버에서 RFID **삭제를 취소**한다.

<도난>

7. **문을 열고 태깅을 하지 않고 가져가는 경우**
- 권한을 받아 문을 열고,
 - **태깅을 하지 않고** 문을 닫았는데,
 - 무게의 값에 변화가 있다면,
 - 도난 상황으로 인식한다.
8. **문을 열고 다른 사람 것도 같이 가져가는 경우**
- 권한을 받아 문을 열고,
 - 본인의 물건을 태깅하고,
 - 다른 사람의 물건을 같이 꺼내면,
 - **본인 음식 통의 무게 값보다 더 변동**이 있으면,
 - 도난 상황으로 인식한다.
9. **문을 열어놓고 물건을 가져가는 경우**
- 권한을 받아 문을 열고,
 - 무게 값이 변동이 있을 후,
 - **일정시간**이 지나면,
 - 도난 상황으로 인식한다.

<도난 알림>

1. 관리자에게 도난 **의심 메일**을 전송한다.
2. 관리자가 **마스터 키**를 입력하면 도난 상황이 종료된다.
3. 도난한 당사자는 자신이 찍힌 모습을 **어플에서 확인하여 경고를 받는다**.
4. 관리자는 **웹페이지**에서 사진 확인이 가능하다.

(개인정보보호법상 도난당한 사람에게 다른 사람의 사진 전송이 어려울 수 있음)
단, 개인정보동의서를 받음 관리자는 사진을 받는 것이 가능하다.

○ 프로그램 사용법 (Interface)

1. Android Studio

- 스마트폰에 다운받아 실행한다.

- 사용자 고유 번호와 비밀번호로 로그인한 후,
- 화면에서 정보를 확인한다.

2. Arduino

- Main 코드인 Security_System.ino를 연다. (동일 폴더의 다른 파일을 열어도 무방함)
- 코드를 센서를 연결한 Arduino 보드에 Upload 시킨다.
- Arduino 보드에 전원을 연결하면 자동으로 작동을 시작한다.
- 리모컨으로 비밀번호 입력, RFID reader에 태깅 등 시스템을 사용한다.

3. RaspberryPi

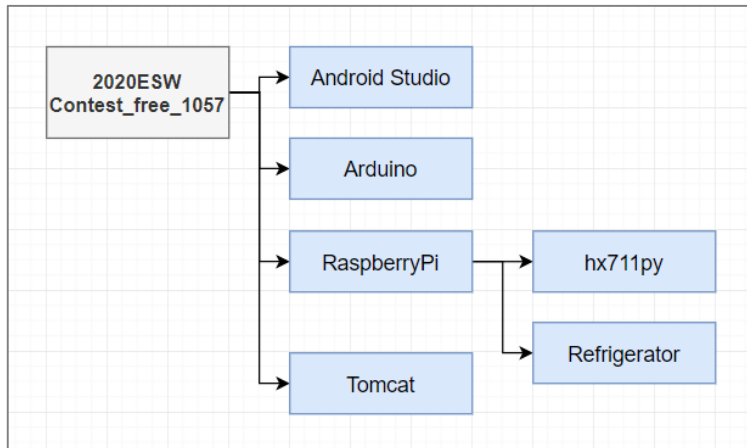
- 1) /hx711py/example.py: 무게 센서 작동 및 도난 상황 판단
 - 해당 폴더에서 python3 example.py 명령어 실행
- 2) /Refrigerator/merge_2.py: 아두이노와 통신 및 Mobius 서버 통신
 - 해당 폴더에서 python3 merge_2.py 명령어 실행
 - 단, 아두이노 실행파일 실행 후 실행되어야 한다.
- 3) /cam.js: 카메라 모듈로 촬영
 - 해당 폴더에서 node cam.js 명령어 실행

○ 개발환경 (언어, Tool, 사용시스템 등)

- Windows
 - Arduino (.ino)
 - Mobius Server: 서버 DB(직접 구축)
 - Tomcat Server: html, JavaScript (WebPage)
 - Android Studio (Kotlin)
- Raspberry PI (Ubuntu OS):
 - Python3, Node JS
 - DB: MySQL

□ 개발 프로그램 설명

○ 파일 구성



- : GitHub 폴더 구조

1. Android Studio (Default: securIoTUser3/app/src/main/java/com/example/)

- securiotuser/LoginActivity.kt: 로그인
- securiotuser/MainActivity.kt: Main 코드, Mobius 서버에서 본인 데이터 조회, 도난 상태 조회 및 촬영 이미지 가져오기
- securiotuser/Model.kt: 이미지 변환
- securiotuser/MyListAdapter.kt: 리스트 구성
- securiotuser/URLtoBitmapTask.kt: 이미지 변환

2. Arduino

- Security_System/BLE.ino: 블루투스 함수 모음
- Security_System/IRremote.ino: 적외선 수신기 함수 모음
- Security_System/LCD.ino: LCD 함수 모음
- Security_System/Magnatic.ino: 자석 센서 함수 모음
- Security_System/RFID.ino: RFID reader 함수 모음
- Security_System/Security_System.ino: Main 코드, 초기화 및 함수 사용 중심 코드.
문이 열렸을 때와 닫혔을 때의 상황을 나누어 다르게 실행한다.
- Security_System/Solenoid.ino: 솔레노이드 함수 모음
- Arduino-IRremote-master.zip: 적외선 수신기 라이브러리
- Arduino-LiquidCrystal-I2C-library-master.zip: LCD 라이브러리
- MFRC522.zip: RFID reader 라이브러리
- rfid-master.zip: RFID reader 라이브러리

3. RaspberryPi

- /Refrigerator/dbModule.py: MariaDB 사용 모듈
- /Refrigerator/merge_2.py: Main 코드, 아두이노와 블루투스 통신 및 Mobius 서버에 데이터 등록/조회/삭제 하여 냉장고 상태 실시간 반영. 도난 모드 감시 및 도난 상황 시 관리자에게 메일 전송
- /Refrigerator/mobiusModule.py: Mobius RESTful API 사용 모듈
- /hx711py/example.py: 무게 센서 도난 방지 시스템 코드

- /hx711py/emulated_hx711.py: 무게 센서 사용을 위한 라이브러리
- /hx711py/hx711.py: 무게 센서 사용을 위한 라이브러리
- /hx711py/hx711.pyc: 무게 센서 사용을 위한 라이브러리
- /hx711py/setup.py: 무게 센서 사용을 위한 라이브러리
- /cam.js: 도난 시 카메라 구동 코드

○ 함수별 기능

1. Arduino/Security_System/Security_System.ino 외 6 (아두이노 센서 제어 코드)

- func_init(): 블루투스 모듈, 적외선 센서, LCD, 자석 센서, RFID, 솔레노이드 초기화
- open_door(): 냉장고 문을 열 때의 동작
LCD에 open를 표시하고 솔레노이드를 열고 블루투스 통신으로 O1 값을 보낸다.
- close_door(): 냉장고 문을 닫을 때의 동작
LCD에 close를 표시하고 솔레노이드를 닫고 블루투스 통신으로 O0 값을 보낸다.
- BLE_conn(): 블루투스 통신을 통한 데이터 읽고 관련 함수 호출
open -> open_door()
close -> close_door()
again -> 다시 입력
Warning -> 안전모드에서 도난 상황으로 변경
ok -> 도난 상황에서 안전 모드로 변경
- BLE_Send(): 블루투스 통신으로 데이터 전송
- IRremote_receive_sign(): 적외선 신호 수신을 확인하는 코드
 - 1) 적외선 신호 수신
 - 2) 등록된 신호인지 확인하여 지정된 값을 문자열(remote_str)에 추가
 - 3-1) 끝 버튼이 눌리지면 BLE_Send()를 이용해 전송
 - 3-2) 10초이상 버튼이 눌리지 않으면 문자열 초기화
- lcd_setMode(): LCD에 표시될 문장, 주기 변경
- lcd_expr(): 주기에 맞춰 문장
- Magnetic_value(): 자석 센서 값 읽는 함수
- Magnetic_check(): 3초동안 연속으로 문이 닫혀 있는지 체크하여 문이 닫혀 있으면 close_door() 호출
- RFID_sensing(): RFID 스티커 인식
 - 1) 카드가 인식되면, (인식 안되면 처음부터)
 - 2) 카드 정보를 시리얼로 표시하고, (정보 변환 안되면 처음부터)
 - 3) 태그의 uid 출력하고, BLE_send()를 통해 블루투스 통신으로 RaspberryPi를 통해 Mobius 서버에 전송
- Solenoid_mode(): 솔레노이드 잠금/해제 모드 설정

2. RaspberryPi/hx711py/example.py (무게 센서 도난 방지 시스템 코드)

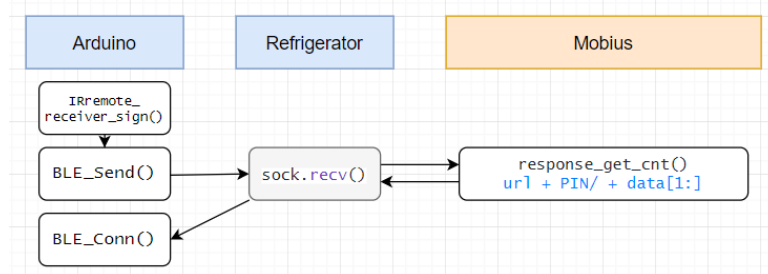
- httprequest(): 문 상태 확인을 위해 Mobius 서버로 값 요청.
DB(control/door/la)에 가장 최근 값을 가져와서 현재 문이 열렸는지 닫혔는지 확인

- security(): 음식통의 RFID 태그 여부 확인과 요청 사항 파악을 위한 Mobius 서버로 데이터 값 요청
 - 1) DB (control/RFID?rcn=4&ty=4&cra=" + now)에 문이 열린 시간을 기준으로 RFID 태그가 RFID 리더기에 태그가 되었는지 확인
 - 2) 1_RFID, 0_RFID = 1: 등록(음식통을 넣은 경우), 0: 삭제(음식통 꺼낸 경우)
 - weghithttp(): 현재 냉장고 내에 전체 무게 정보 확인을 위한 Mobius 서버로 값 요청 DB(control/weight/la)에 가장 최근 전체 무게센서 값 가져옴.
 - errorhttp(): Mobius 서버로 도난 알림 값 전송.
 - <무게센싱 중 도난 상황 발생 시>
 - 1) DB(control/security) "1"(도난) + 당시 사용 중이던 사용자(학번/비번) 전송.
 - 2) DB(user + "/security") 사용했던 사용자(AE) security에도 "1"(도난) 전송.
 - weightchange(): 무게센서 변동 값 Mobius로 전송.
 - DB(control/weight)로 str(weight)바뀐 무게 센서 값 전송.
 - plushttp(weight, time, RFID): 사용자가 추가한 음식통의 무게 값 Mobius로 전송.
 - 1) DB(user+ "/food") 사용자(AE)로 음식통에 부착된 RFID + 음식통의 무게 + 음식 통 등록 시간
 - 2) DB(user +"/lastRFID") 사용자(AE)로도 같은 값 전송. (음식 통 업데이트에 사용).
 - search(RFID): 사용자가 등록했던 음식통인지 확인을 위해 Mobius로 값 요청.
 - DB(user + "/lastRFID/" + str(RFID)) 사용자(AE)에 입력된 RFID에 값이 있는지 확인 및 RFID 정보 가져오기(무게, 시간)
 - update(weight, time, RFID): 음식통 재등록 시 (사용자가 사용 후 되돌려놓는 경우) Mobius 서버로 해당 음식통 정보 업데이트 요청.
 - 1) 사용자(AE)로 음식통에 부착된 RFID + 음식통의 무게 + 음식 통 등록 시간
 - 2) DB(user +"/lastRFID") 사용자(AE)로도 같은 값 전송.
 - nohttp(RFID, RE): Mobius에 요청했던 요청 사항에 대한 취소 명령 전송.
 - DB(user + "/foodRFID") 사용자(AE)로 "0"(삭제 취소), "1"(등록 취소) 전송.
 - delhttp(RFID): Mobius에 음식 통 삭제 명령 전송. (음식통을 꺼낸 경우)
 - DB(user + "/food/" + str(RFID)) 사용자(AE)에 음식 통에 부착된 RFID을 삭제 요청.
3. RaspberryPi/Refrigerator/merge_2.py 외 2 (아두이노 통신 및 DB, Mobius 통신)
- dbModule.py/set_theft(): 도난 모드 DB에 저장
 - dbModule.py/get_theft(): 도난 모드 DB에 조회
 - dbModule.py/execute() / executeOne() / executeAll(): 커서를 옮겨 쿼리(실행문) 실행
 - dbModule.py/commit(): DB에 직접 반영 명령
 - dbModule.py/check_uid(): 해당 RFID uid가 DB에 있는지 확인
 - SELECT * FROM RFID WHERE uid=%s; 명령어를 이용해 executeAll() 실행
 - dbModule.py/insert_db(): DB에 RFID uid를 입력
 - INSERT INTO RFID(uid) VALUE(W" + uid + 'W'); 명령어를 이용해 executeAll() 실행
 - dbModule.py/delete_db(): DB에서 해당 RFID uid를 삭제
 - DELETE FROM RFID WHERE uid=(W" + uid + 'W'); 명령어를 이용해 executeAll() 실행

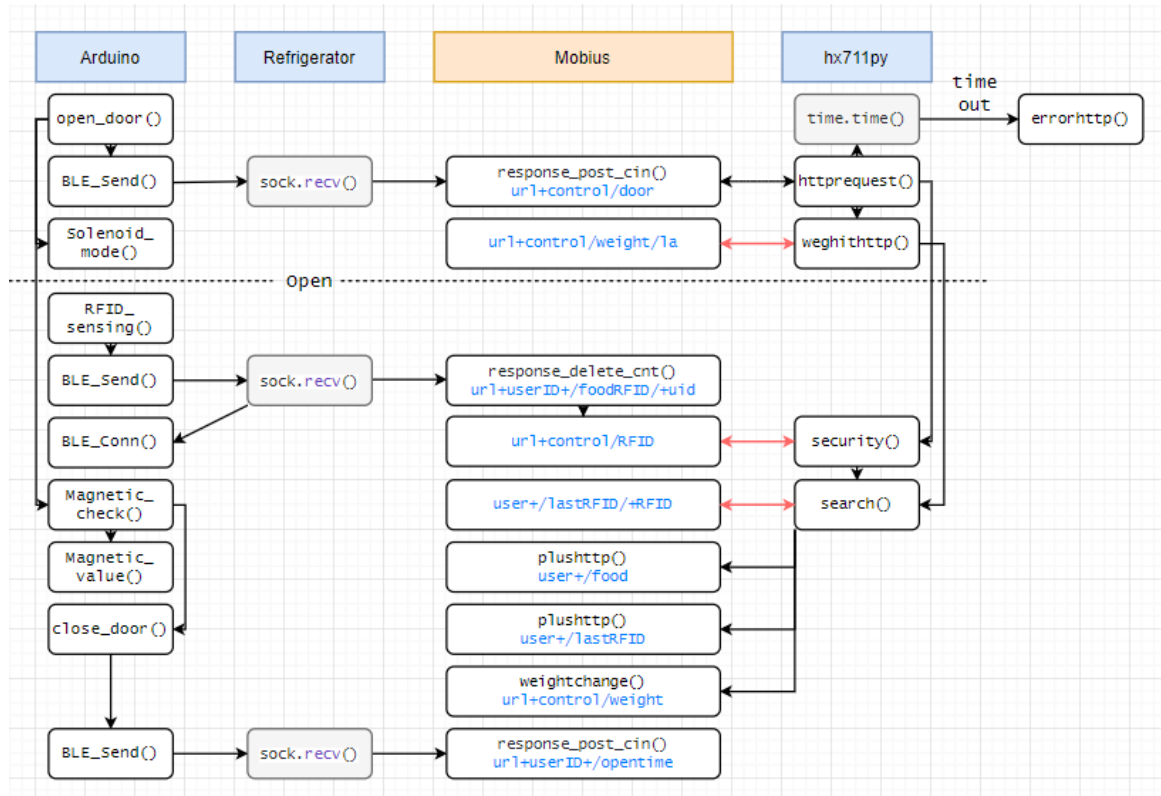
- mobiusModule.py/response_post_cin(): 해당 내용의 cin 생성
- mobiusModule.py/response_post_cnt(): 해당 내용의 cnt 생성
- mobiusModule.py/response_get_cin(): 해당하는 cin 값 조회
- mobiusModule.py/response_get_cnt(): 해당하는 cnt 값 조회
- mobiusModule.py/response_delete_cnt(): 해당 내용의 cnt 제거
- mobiusModule.py/response_delete_cin(): 해당 내용의 cin 제거
- merge_2.py/watch(): thread 함수 사용으로 실시간으로 Mobius 서버의 'control/security/la'에서 도난 상황을 체크한다.
도난 상황 발생 시 관리자 메일로 경고 전송
도난 상황 종료 시 안전모드(일반 상황)로 전환한다.

○ 주요 함수의 흐름도

- 비밀번호 입력



- 냉장고 문 열기 & RFID 태깅(정상 등록 흐름도)



○ 기술적 차별성

- 무게 센서를 활용한 프로그램을 만들어 보다 면밀하게 도난 상황을 감지 할 수 있었다.

- Mobius 서버/DB를 활용하여 도난 프로그램이 돌아가는 디바이스에 부담을 줄여 효율적인 시스템을 구축할 수 있었다.
- 사용자 편의성에 맞춰 프로그램을 설계하여 다양한 예외 상황에 대한 처리도 가능하게 프로그래밍 했다. (ex. 등록/삭제하고 음식통을 안 넣고/빼는 경우, 문을 열고 도난하는 경우 등.)
- 사용자가 부착한 RFID 태그를 구별 통해서 각각의 음식통 정보(무게, 시간)를 사용자(AE)에 개별적으로 저장하므로 보안과 관리를 편리하게 했다.

□ 개발 중 발생한 장애요인과 해결방안

○ 도난 상황 인식 센서 교체

- 초기 고안한 모델은 마트의 EAS 시스템(Electronic Article Surveillance)을 바탕으로 구상하였다. 그러나 인식하는 기계가 생각보다 많이 고가이고 크기가 커서 핵심 기능인 도난 상황 인식 방법을 다른 방식으로 변경해야만 했다.
- 두번째로 고안한 방법은 [무게 센서 + 초음파]를 통한 인식 방법이었다. 사용자 번호와 무게 차를 통한 본인 물품의 등록/꺼냄 여부 인식은 가능했으나 비슷한 무게의 본인 것이 아닌 다른 물건을 꺼냈을 때의 도난의 경우를 인식하기엔 무리가 있었다.
- 세번째(현 시스템)로 고안한 방법은 [무게 센서 + RFID]를 통한 인식 방법이다. 사용자 번호에 RFID 스티커의 번호를 등록하여 무게 값과 함께 저장하고, 꺼낼 때 사용자 번호에 등록된 RFID 스티커가 태그 되지 않거나 꺼낸 무게가 다르다면 다른 물품을 꺼낸 것으로 인식하고 도난의 상황으로 관리자에게 메시지가 가게 된다.

○ 무게 센서의 성능 부족

- 저렴한 무게 센서를 사용하다 보니 같은 환경에서도(ex. 같은 물건을 올려놓을 때) 무게 센서 값의 오차가 발생하는 경우가 종종 있었다.
- 무게 센서를 2종류를 추가 구매하여 성능이 제일 좋은 것으로 사용했고,
- 오차 범위를 수정하여 크게 두고, 무거운 물건으로 테스트를 진행하였다.

○ RFID의 태그 인식 오류

- RFID 리더기의 태깅 거리와 인식 속도에 따라 연속 태깅 오류가 발생하였다
- RFID 리더기의 인식 속도를 1초에 1회로 조정하여 오차를 줄였다.

□ 개발결과물의 차별성

○ 도난 방지 시스템(보안)

- 키패드(리모컨)을 이용하여 등록된 사용자의 고유 번호 인증이 완료되어야 냉장고 문의 개폐가 가능하다. 따라서 권한이 있는 사람만 공용 냉장고의 이용이 가능하고, 도난 상태가 인지되면 현장을 촬영하고 관리자에게 경보를 주는 물품의 보안을 구축하였다.
- 이를 통해 집단 시설 내 도난 사건과 도난으로 인한 갈등이 감소하고, 시스템을 사용하는 시설의 평판 및 신뢰도가 증가할 것이다.

○ 공용 냉장고 내 본인 물품(음식 통) 관리 가능

- 사용자 고유 번호와 연결된 RFID 스티커의 번호, 무게 차와 다른 센서의 종합적인 정보를 통해 개별 물품의 도난 인식이 가능하다.
- 앱에서 본인 물품의 목록 조회가 가능하기 때문에 물품의 보관 마감 기간을 인지하게 되고, 장기간 방치되는 음식 통이 줄어든다. 이를 통해 냉장고 내 오물 냄새를 줄이고, 상해서 못 먹는 음식이 감소할 것으로 음식물 쓰레기의 감소로 환경이 보호된다.

○ 다른 공용 보관 시스템에도 적용 가능한 시스템 활용성

- 냉장고가 아니더라도 회사/기숙사/게스트하우스/고시원 등 공간을 나눠 쓰는 집단생활 시설이라면 다른 공용 시스템에 그대로 적용이 가능하다.
- 층수와 공간 크기에 따라 무게 센서의 가격만 추가될 뿐 공간의 크기에 대한 제약이 없다.

○ 데이터의 이용

- 냉장고 기업: 다른 곳의 인당 사용 무게와 패턴을 분석하여 공용 냉장고 제품 추천 가능하다.
- 서비스: 버려지는 쓰레기(장기 보관 中) 데이터를 이용하여 개인의 물품 소비 습관(냉장고의 경우 음식)을 피드백 한다.

□ 개발 일정

[illegible]

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	김유진	<ul style="list-style-type: none"> - Mobius 서버 관리 - Mobius 정보로 도난 상황 인식 프로그램 제작 - 관리자 웹 페이지 제작
2	팀원	한예슬	<ul style="list-style-type: none"> - 센서 동작 프로그램 제작 - 냉장고 상태 정보 Mobius 서버에 등록 - 사용자 앱 제작