프린터 대기열 알리미

4 조

이명진, 유정엽, 이계준, 이윤열

목차

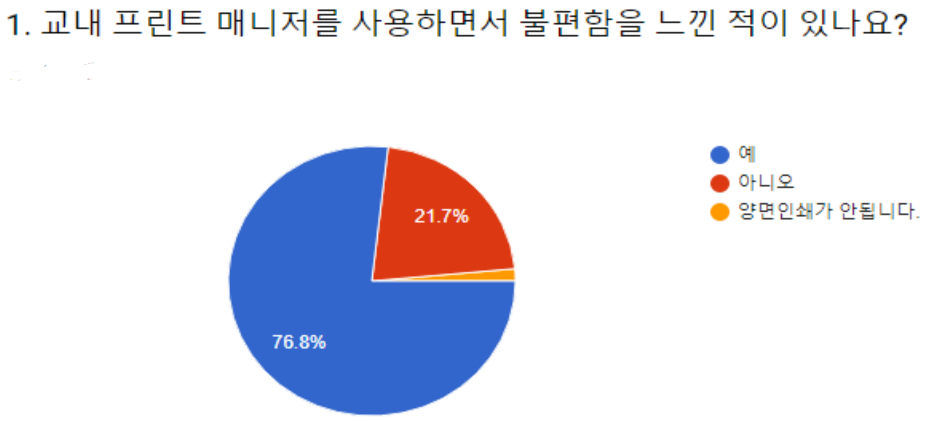
1. 개발 동기

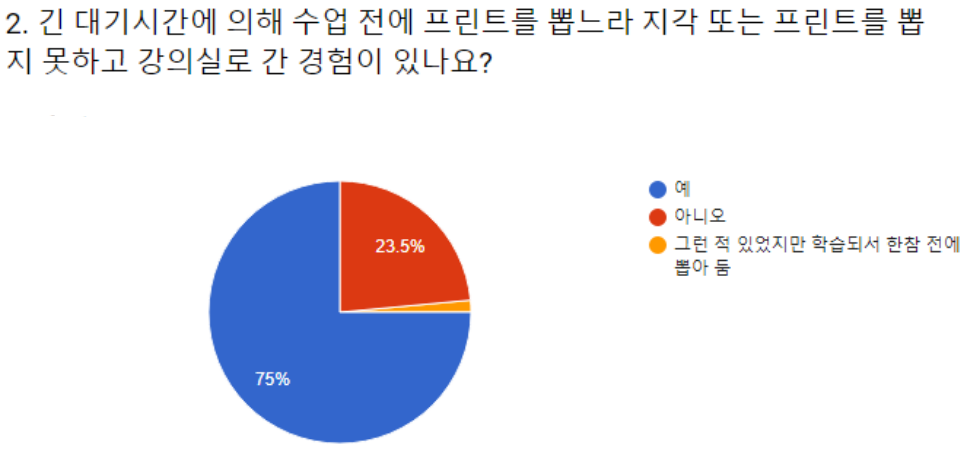
1.1. 개발 배경 및 필요성

4차 산업 혁명시대가 도래하며 사람들은 모두 신속한 판단을 통해 최선의 선택을 하고자 한다.

이에 따라 정확한 정보를 신속하게 제공하는 일은 점점 더 중요한 일이 되었다. 또, 정보를 재가공 함으로써 의미가 있는 자료로 만드는 일도 중요한 이슈이다. 그 예로, 대중교통 도착 시간 알림 어플, 음식점 대기 예상 시간 어플, 지역별 실시간 예상 유동인구 수 계산 어플 등 많은 종류의 정보 제공 어플이 시중에 나오고 매우 유용하게 사용되고 있다.

현재 성균관대 내의 불편 사항들을 살펴보고 이를 해결 가능한 부분들에 대해 생각을 하고자 한다. 먼저, 가능 불편함을 호소한 부분은 수업시간 바로 전 프린트 매니저 대기시간으로 인한 수업 지각, 수업 중간 프린트를 통한 수업 방해와 같은 부분이 있었다. 또 다른 문제로는 성균관대 근처 음식점 부족으로 인한 체력소모 및 시간 낭비로 인해 발생하는 손실 등을 뽑았다. 이 중 우리는 프린트 매니저에 대한 부분에 대해 개선의 필요성을 느끼고 이를 해결하면 좀 더 좋은 수업환경이 제공 될 것으로 예상된다.





1.2. 개발 목표 및 방향

우리의 개발 목표는 어플을 통해 학교 내에 존재하는 모든 프린트 앞에 대기 인원 수에 대한 정보를 제공하는 것이다. 이를 통해 수업 지각 혹은 교안의 미지참을 막는 것이 목표이다.

이러한 목표를 실현하기 위해 먼저 정확한 센서 구현이 1차적인 방향이며, 센서를 통한 정보를 재가공, 어플로 정보를 제공하기 위한 프로그래밍을 2차적인 방향으로 잡았다.

목표를 달성하게 되면 추후 사람들이 붐비는 장소의 대기인원 수를 알 수 있게 되고 만든 분야에 접목되어 이용자들에게 좋은 정보를 제공할 것으로 예상된다.

2. 개발 과정

2.1 작동원리

그림 1 작동을 위한 센서 배치도

모션센서는 사용자가 대기하는 위치에 각각 설치된다. 각 위치에 대기중인 사용자의 움직임을 감지하여, 대기유무를 알려준다. 모션센서는 대기하는 위치에 하단 바닥 속에 설치하고 모션센서의 센서가 그 자리에 있는 사람의 유무를 알 수 있게 모션센서의 윗부분은 투명하게 만든다. 이 때 모션센서를 바닥 속에 넣는 이유는 모션센서가 인지하는 부분이 약 120° 정도 된다고 smart things specification에 명시되어 있다. 이 때문에 우리가 원하지 않는 범위도 인지를 해버릴 가능

성이 매우 높다. 따라서 우리는 인지하는 각도를 제한하기 위해 모션센서를 바닥속에 설치하였다. 실제 모션센서가 충분히 작은 크기이기 때문에 우리가 원하는 충분한 상대적 높이를 주더라도, 바닥을 많이 훼손할 필요는 없을 것이라고 예상된다.

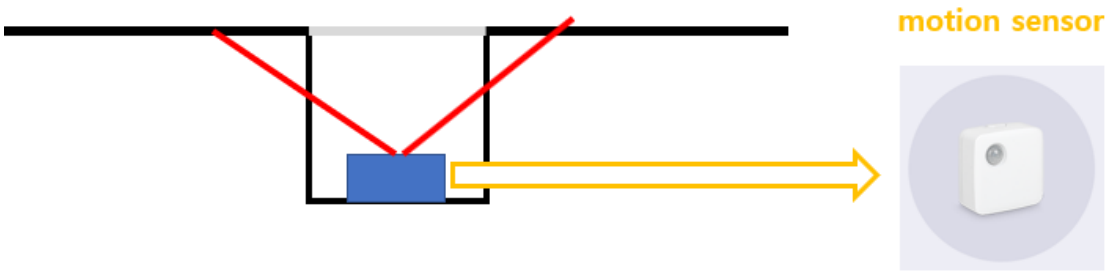


그림 2 상대적 높이를 이용한 모션센서 시야 각 제한

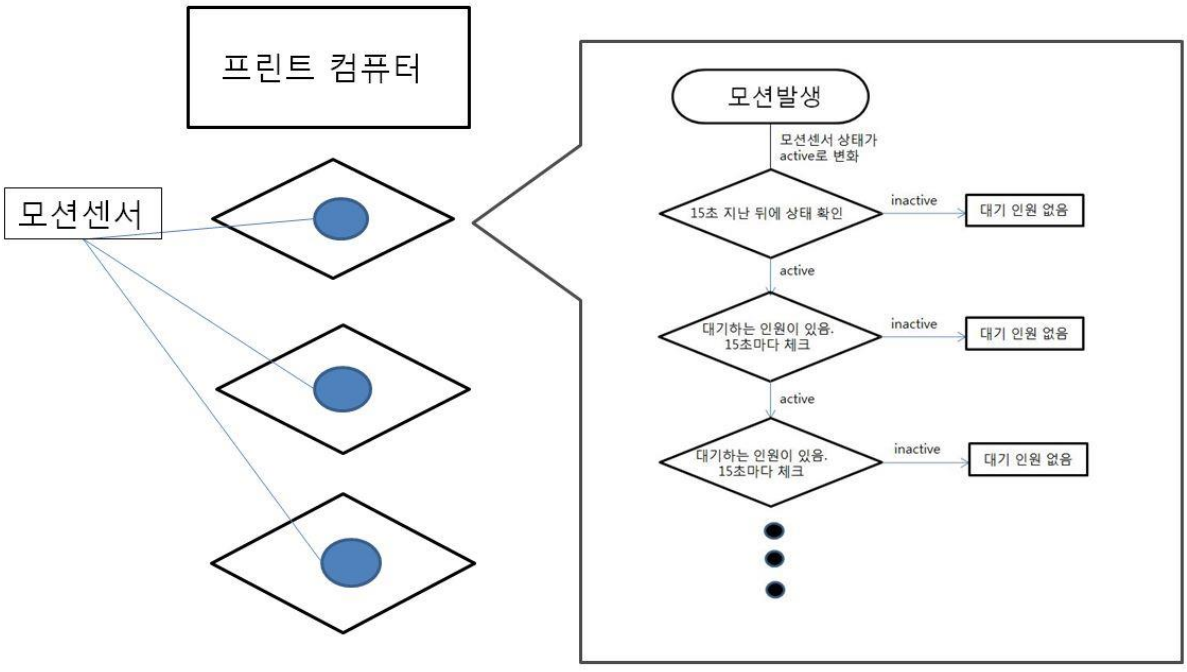


그림 3 알고리즘 동작 원리

하단에 모션센서가 땅 위에 움직임을 인식하면 모션센서의 상태가 active 상태로 바뀐다. 최초의 active를 인식하게 되면 해당 thread를 15초간 정지한다. 15초 후에 다시 한번 모션 센서의 현재

상태를 체크한다. 여기서 움직임이 있는 active라면 대기하는 인원이 있다고 인식하고, 사람의 움직임이 없는 inactive 상태라면 지나가는 사람으로 인식한다. 또한 active 상태를 체크 한 후부터 15초 타이머가 또 작동한다. 15초가 지난 후에 active/inactive 상태를 체크하고, 위의 과정을 계속 반복한다.

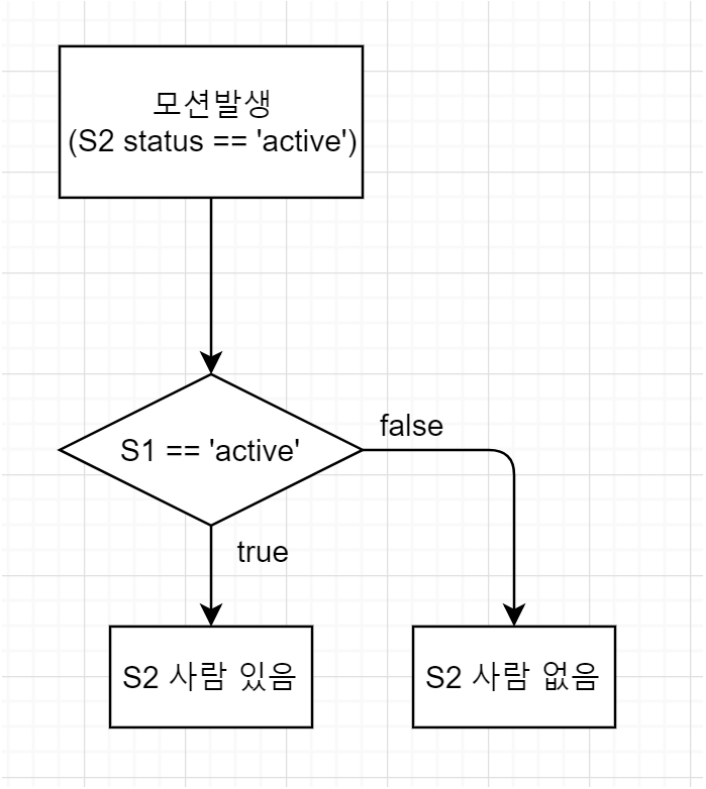
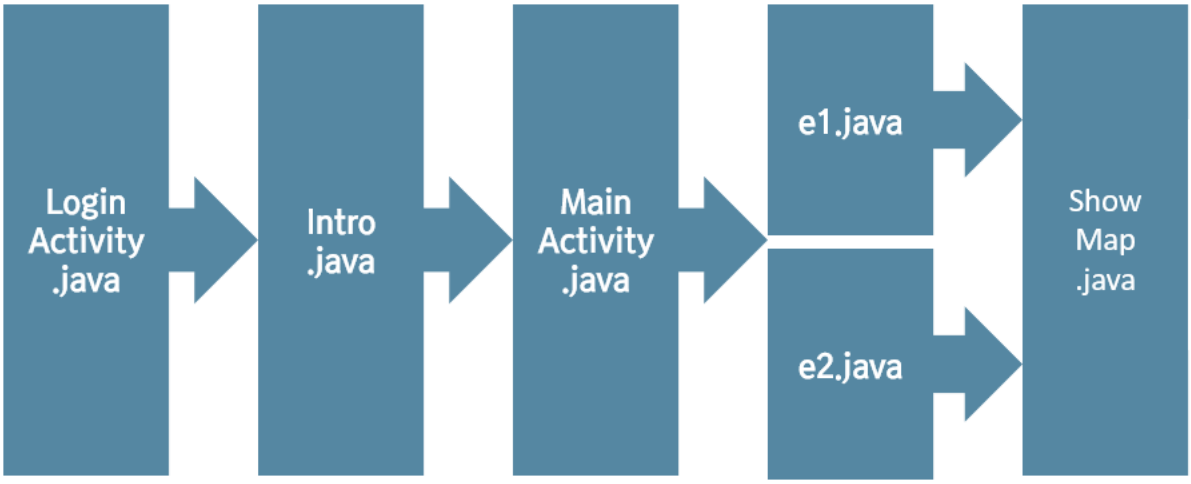


그림 4 모션센서 S2의 사람 유무 판단 과정

또한 컴퓨터에 가까운 부분부터 센서의 이름을 S1, S2, S3라고 했을 때, S2가 active가 되고 S1이 inactive 상태라면 S1이 없는데 S2가 있는 줄은 없으므로 이 때도 대기인원을 0명으로 인식하도록 만들었다. S3이 active이고 s1이 inactive인 경우도 마찬가지로 적용된다. S4, S5 S6에 대해서도 같은 원리를 적용한다. 프린트사용자, 대기자 외에도 그냥 주변으로 지나다는 사람이 있다. 이로 인해 오차가 생긴다. 앞에서 말한 원리를 이용하면 지나가는 사람들을 어느정도 커버할 수 있게 된다. 아무도 없는데 만약에 S2, S3 지점으로 지나가는 사람이 연속적으로 있다고 해도, 대기하는 사람으로 인식하지 않는다. 그리고 이용자의 행동을 파악해 봤을 때 컴퓨터에 바짝 붙어서 지나가지 않기 때문에, S1, S2가 active될 확률은 적다.

2.2 코드설명

주요 UI activity는 다섯 단계로 구현되었다.



i. **LoginActivity**

해당 activity에서는 artik cloud에 로그인을 할 수 있도록 한다. 실습시간에 사용했던 예제 코드를 이용하였다.

ii. **Intro**

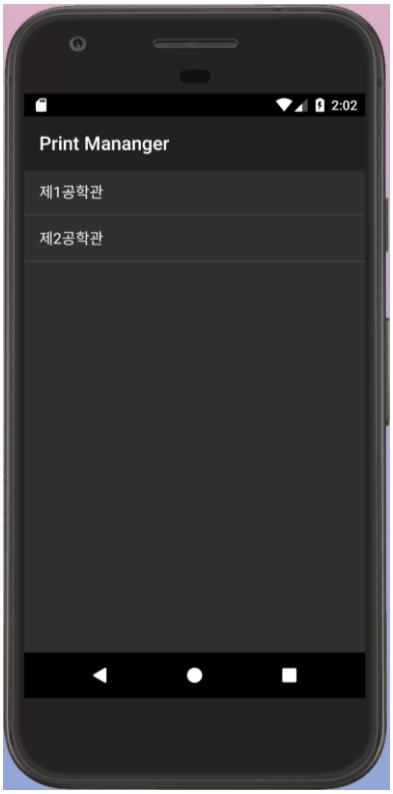
해당 activity에서는 ImageView를 통해 어플리케이션의 시작 화면을 보여준다.

```
ActionBar actionBar = getSupportActionBar();
actionBar.hide();

h = new Handler();
h.postDelayed(mrun, delayMillis: 2000);
```

위의 두 줄은 액션바를 감춰주고 아래 두 줄은 handler를 생성해서 2초간 딜레이를 주어 화면이 바로 다음 activity로 넘어가지 않게 했다.

iii. MainActivity



해당 activity에서는 프린터가 존재하는 건물을 고른다. 모션 센서가 한 개 제공되어서 적용할 수 있는 프린터도 한 대로 한정적이었다. 데이터를 저장하는 list에는 '제1공학관', '제2공학관'에 대한 정보를 저장하였다.

```
DeptList.setOnItemClickListener(
        (adapterView, view, i, l) → {
                String selecteditem = (String)adapterView.getItemAtPosition(i);

                if(selecteditem.equals("제1공학관")){
                    Intent intent_e1location = new Intent(mContext, e1.class);
                    startActivity(intent_e1location);
                }
```

seOnItemClickListner()에서 클릭된 리스트의 인덱스를 알아내서 각 조건에 맞는 activity로 넘어가도록 하였다.

iv. e1

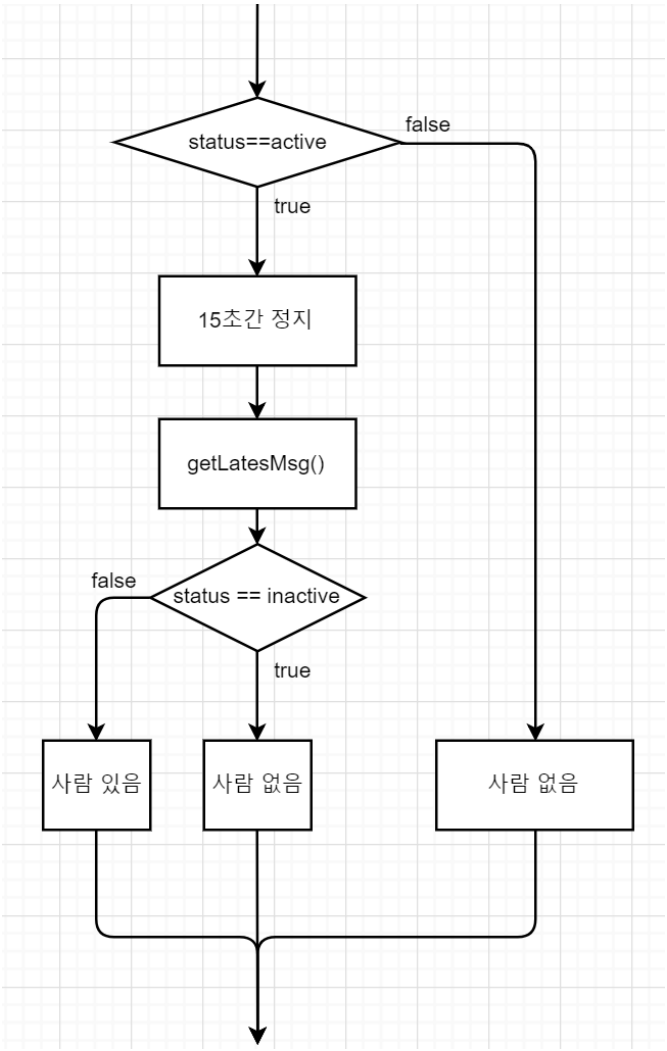
해당 activity에서는 선택된 프린터기의 대기인원수를 팝업창으로 띄워준다.

getLatesMsg() 메소드는 아틱클라우드의 데이터로그를 읽어오고 사람이 있는지, 혹은 없는지를 판별한다.

```
@Override
public void onSuccess(NormalizedMessagesEnvelope result, int i, Map<String, List<String>> stringListMap) {
    Log.v(tag, msg: " onSuccess latestesMessage = " + result.getData().toString());
    //String mid = "";
    String status = "";
    if (!result.getData().isEmpty()) {
        //mid = result.getData().get(0).getMid();
        status = result.getData().get(0).getData().toString();
        if(status.equals("{motion=active}")){
            SystemClock.sleep( ms: 15000);
            getLatesMsg();
            if(status.equals("{motion=inactive}")){
                //length=0;
                p = false;

            }
            else if(status.equals("{motion=active}")){
                //length=1;
                p = true;
            }
        }
        else if(status.equals("{motion=inactive}")){
            //length=0;
            p = false;
        }
    }
}
```

데이터 로그를 정상적으로 읽어왔다면, 사람의 존재 유무를 판별하는 과정을 거친다.



위 그림은 판단 과정을 flow chart로 나타낸 것이다. 모션 센서는 active가 감지 되고 난 후 모션이 감지되지 않아도 약 10초동안 active 상태로 있게 된다. 따라서 사람이 모션 센서 앞에 오랜 시간 서 있는 상황이 아닌 단지 지나가는 상황에서도 약 10초간 status가 active가 된다. 이를 막기 위해서 15초 이후에 다시 한번 메시지를 받아 inactive 상태인 경우 '대기하는 사람이 없다.'라고 판단할 수 있도록 한다. 반대로 여전히 active인 경우에는 '대기하는 사람이 있다.'라고 판단한다. 사람의 유무를 저장하는 변수는 모든 method에서 접근할 수 있도록 해 사용자가 list를 클릭했을 때 그 시점에서 저장되어 있는 상태를 출력할 수 있도록 한다.

v. ShowMap

e1 activity에서 '지도보기'라는 버튼을 클릭하였을 때 ShowMap activity로 전환된다.

ImageView를 이용해 앱 자체에 저장되어 있는 지도 이미지를 불러온다.

2.3 개발 결과.

특정 프린터 대기 인원 수를 볼 때, 건물별로 프린터기를 나눴기 때문에 더 편리하게 원하는 프린터를 찾을 수 있도록 했다. 또한 프린터 위치를 간략한 지도로 사용자에게 보여줌으로써 사용자 편의성을 높였다. 그리고 사람의 유무에 관한 알고리즘을 따로 구현 해 출력된 대기 인원 수에 대한 신뢰성을 높였다.

3. 결론

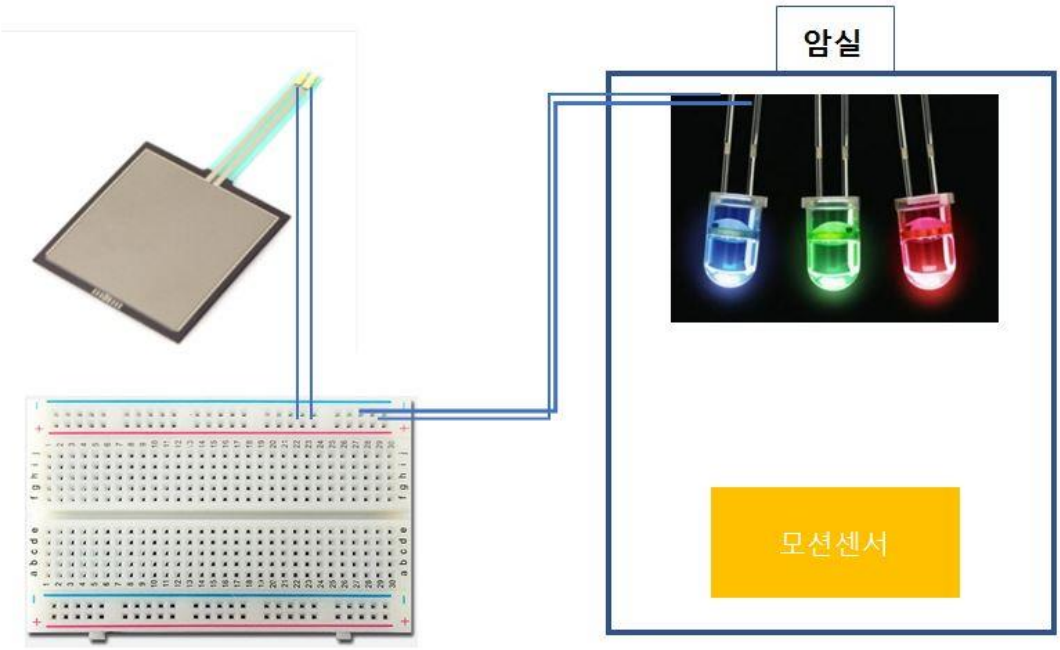
3.1 성과 및 기대효과

프린터 매니저 앱은 ARTIK 장치를 사용해서 각 장소에 프린터 대기열이 몇 명이 있는지를 직접 가지않고도 사용자가 간편하게 확인 할 수 있다. 프린터의 위치가 계단 앞이 아니라 건물 중앙에 있거나 건물 끝에 있다는 점에 앱을 이용하는 프린터 이용자들은 앱을 이용하지 않는 사람들에 비해 시간 절약과 편리함을 갖추게 된다.

1.1에서 언급한 설문조사의 2번 문항에서, 프린터 대기 인원 수를 파악하지 못하고 현재 사용자의 위치에서 가장 최선의 프린터를 알지 못해 수업에 지장이 생기는 경우가 응답자 중 약 75%에 달했다. 해당 프린터 대기열 알리미 어플리케이션은 프린트물을 인쇄하지 못하거나, 지각하는 등의 경험 빈도를 줄여줄 수 있다. 이에 따라 학생들의 수업의 질은 높아질 수 있을 것이다.

3.2 구현 시 이슈와 아쉬운 점

우리 조의 가장 큰 이슈는 모션 센서의 동작 원리였다. 구상 단계 때 빛의 변화를 센서가 감지해 모션의 유무를 감지한다고 생각했다.



위 그림은 처음 구현 계획을 간단한 모식도로 나타낸 것이다. 프린터를 이용하려는 대기자가 서 있는 줄 바로 아래에 압력 센서를 설치한다. 전구가 반짝이는 상태가 디폴트로 설계되어 대기자가 압력센서를 밟으면 전구에 전원이 차단되어 전구에 불이 들어오지 않도록 회로를 구현하였다. 밟지 않은 상태에서 모션센서는 연속적인 빛의 변화를 통해 모션이 있다고 판단할 것이고 어플리케이션은 아틱 클라우드에서 'active'라는 정보를 받을 수 있다. 누군가가 압력센서를 밟으면 전구

에 전원이 차단될 것이고, 그 상태가 10초간 지속된다면 모션센서는 빛의 변화가 없는 상태로 인식하여 'inactive'가 된다. 어플리케이션 알고리즘 자체적으로는 사람의 유무를 판단하는 데에 10초의 딜레이를 주어 지나가는 사람을 세지 않게 했고, 10초 이상 밟고 있어야 대기자라고 판단했다. 어플리케이션 내부에서는 active가 사람이 없는 상태, inactive가 사람이 있는 상태로 사용된다. Inactive, active 의 상태만 역전될 뿐 나머지 원리는 앞에서 설명한 알고리즘과 동일하게 사용할 수 있다. 전구와 압력센서를 이용하여 이와 같이 작동하는 회로를 구현하였고, 실험을 진행하였다. 하지만 이 과정에서 아이디어의 큰 가정이 틀렸다는 것을 알게 되었다.

본사 문의 결과, 모션센서는 적외선 기반이고, 모션센서의 탐지구간이 나누어져 있는데, 시간에 따라 다른 탐지구간에서 적외선이 탐지가 되어야, 모션으로 인식을 한다는 점이었다. 적외선 기반으로 다시 회로를 구현하려면, 아두이노나 라즈베리파이와 같은 추가적인 기기가 필요할 것이라고 생각하였다. 조원들과의 회의 끝에 개발할 수 있는 시간이 많이 남지 않았고, 확실히 작동한다는 보장을 받을 수 없어, 앞에서 설명한, 보다 구현이 편리한 방법을 찾았다.

아쉬운 점은 이용자들의 이용패턴을 보았을 때, 한 줄에 한 명, 또는 세명의 이용자가 있을 때, 아마 우리가 의도한 위치에 이용자가 위치할 확률이 높지만, 한 줄에 두 명정도가 있을 때에는 최초 한명은 컴퓨터를 이용하기 때문에 가장 앞자리에 위치하지만, 두번째 사람은 정확히 어디 위치할지 알 수 없다. 관찰에 의하면 어느정도 거리를 유지하며 대기하는 경우가 꽤 많았다. 이 경우에 앞에서 말한 알고리즘을 이용하면 정확히 커버할 수 없다는 것이 아쉽다.

또한, 서버 부분을 구현하지 못해 대기 인원 수를 측정하는 모션센서와 허브가 등록된 특정Aritk 계정에 로그인해야 한다는 부분이 아쉽다. 서버를 구현했다면 해당 Artik Cloud 계정을 알지 못해도 서버에서 자체적으로 데이터로그를 읽어올 수 있도록 해 사용자 접근성을 높일 수 있을 것이다. 처음에 생각했던 '최적 프린터 추천', '고장 난 프린터 신고' 등의 기능 또한 구현할 수 있을 것이다. 그러나 반대로 생각해보면 그만큼 확장성이 많은 어플리케이션이라고 생각할 수 있다. 대기 인원 수를 체크하는 기본적인 기능이 이미 구현되었기에, 사용자 편의성을 높이는 부가적인 기능을 구현한다면 매력적인 어플리케이션이 될 수 있을 것이다.