

인원수에 따른 지하철 냉방 조절 시스템

IoT와임베디드소프트웨어 최종 보고서

학 과 컴퓨터공학과

학 년 3학년

한다현 (22012127)

이은주 (22012154)

이은진 (22011430)

진서연 (22020306)

제출일자 2022.06.08



영남대학교
Yeungnam University

[목차]

1. 서론	3
가. 프로젝트 개발 목적	3
나. 제안서 spec 대비 실제 구현 요소	3
2. 본론	5
가. 설계 제한 사항	5
나. 하드웨어 구성	5
다. 하드웨어 설계도	7
1) 기본 부품	7
2) LCD & LED	7
3) 온도 모듈	8
4) 조도 센서	8
5) 열전소자 쿨러 모듈/릴레이 모듈	8
라. 소프트웨어 구성	9
1) 개발 방법	9
2) 플로우 차트 및 알고리즘	9
3) 앱 프로그램 설명	9
3. 결론	11
가. 시연 결과	11
나. 시행착오	12
다. 기대 효과	13
라. 느낀점	13
4. 부록	14
가. 회로도	14
나. 코드	17
다. 회의록	24
라. 제안서	25
5. 레퍼런스	27

1. 서론

가. 프로젝트 개발 목적

본 프로젝트는 지하철 내의 승객 분포에 따라 실내 온도 조절하여 쾌적한 환경을 조성하고, 에너지를 절약하는 것이 궁극적인 목표이다. 또한, 승객에게 칸 내 여석 현황을 안내하여 직접 내부에 들어가지 않고도 남은 좌석을 알 수 있도록 대중교통 사용 시 편의를 제공한다.

현재 지하철은 인원수를 고려하지 않고 실내 온도를 설정한다. 이에 실내 인원수가 적더라도 계속 낮은 온도를 유지하거나, 인원이 많더라도 낮은 온도를 설정할 수 없는 상황이 빈번히 발생한다. 하지만, 인원수에 따라 온도를 조절하는 시스템을 도입하게 된다면 불필요한 에너지 소모를 줄일 수 있고, 보다 쾌적한 환경을 조성할 수 있다. 더욱 나아가, 해당 시스템을 지하철뿐만 아니라 다양한 실내 공간에서 적용하게 된다면 더 많은 에너지 절약을 기대할 수 있다.

나. 제안서 spec 대비 실제 구현 요소

기존에는 승객의 착석 여부를 판단하기 위한 조도 센서로 GL5537을 사용하려 하였으나, 실제 구현은 GL5528을 사용하였다. 조도 센서의 종류마다 조금씩 다른 spec을 가지는 것을 확인하였다.

Types and Specifications			Light resistance (10Lux) (K Ω)	Dark resistance (M Ω)	γ_{10}^{100}	Response time (ms)	
Type	Max. Voltage	Max. power				Increase	Decrease
GL5516	150	90	5-10	0.5	0.5	30	30
GL5528	150	100	10-20	1	0.6	20	30
GL5537-1	150	100	20-30	2	0.6	20	30
GL5537-2	150	100	30-50	3	0.7	20	30
GL5539	150	100	50-100	5	0.8	20	30
GL5549	150	100	100-200	10	0.9	20	30

<그림 1 : 조도 센서 별 spec 비교>

기존 제안서의 데모 계획은 좌석이 만석임을 인식하면 열전소자 쿨러 모듈 전원을 On 하여 현재 실내 온도에서 1도를 낮추고, 1도가 낮아진 것을 확인한 후에 쿨러 모듈 전원을 Off 하는 방식이었다. 하지만, 실제 구현 하드웨어에서 1도가 낮아진 것을 확인하기 위해서는 오랜 시간이 소요된다는 점을 감안하여 방식을 변화시켰다. 모든 좌석이 비어있을 때는, 쿨러 모듈의 전원을 Off 한다. 만약 만석도, 모든 자리의 공석도 아닐 경우는 모듈의 전원을 on과 off를 반복하여 해당 온도를 유지한다. 또한, 모든 좌석이 착석 상태로 인식되었을 때는 쿨러 모듈을 On 한 상태로 유지한다. 해당 방식으로 변형하여 실제 구현한다.

	데모 계획	실제 구현
만석일 시	ON	ON
여석이 있을 시	실내 온도가 1도 낮아지면 쿨러를 끈다.	ON/OFF 반복
공석일 시	OFF	OFF

<표 1 : 제안서 spec 대비 실제 구현 비교>

또한, 조도 센서 유지 시간 면에서 약간의 변동이 있었다. 기존에는 조도 센서가 10초 이상 일정 수준보다 낮은 저항값을 가지면 승객이 착석한 것으로 판단하였는데, 각각의 모듈마다 delay가 필요한 경우가 존재해, 기존 방식으로 구현하기에는 착석 유무를 판단함과 동작을 하는 것에 있어 오랜 시간이 소요되는 문제점이 존재하였다. 때문에, 해당 방식에서 한 사이클(Cycle)을 기준으로 판단하는 것으로 수정하였다. 한 사이클은 8.5초이며, 한 사이클 동안 조도 센서가 일정 수준보다 낮은 저항값을 가지면 승객이 착석한 것으로 판단한다. 만약, 한 사이클이 이미 시작한 후 승객이 착석하였다면, 다음 사이클까지 일정 수준보다 낮은 저항값을 가져야 승객이 착석한 것으로 인식한다.

2. 본론

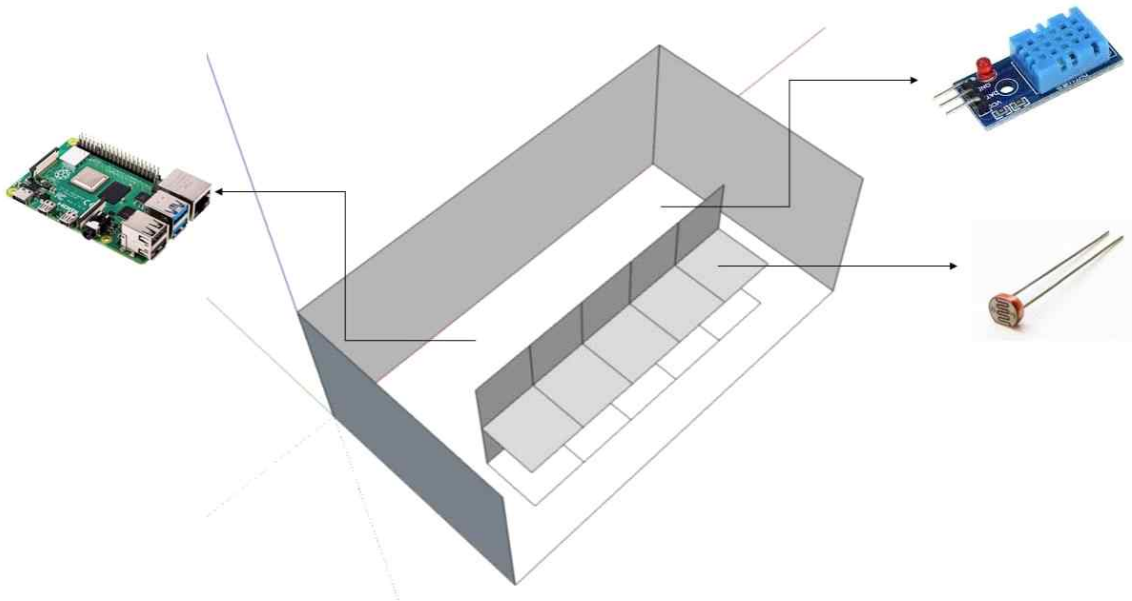
가. 설계 제한 사항

편리함을 위해 모듈과 구현 하드웨어 간의 길이를 조절하기 위해 점프 선을 계속해서 길게 연장하니, 제대로 전원 공급이 이루어지지 않아 조도 센서의 밝기 인식이 불안정한 것을 확인하였다. 때문에, 하드웨어 간에 영향을 끼치지 않으며, 제대로 안정적으로 연결되는 선에서 점프 선의 길이를 연장하여야 한다는 제약이 존재했다.

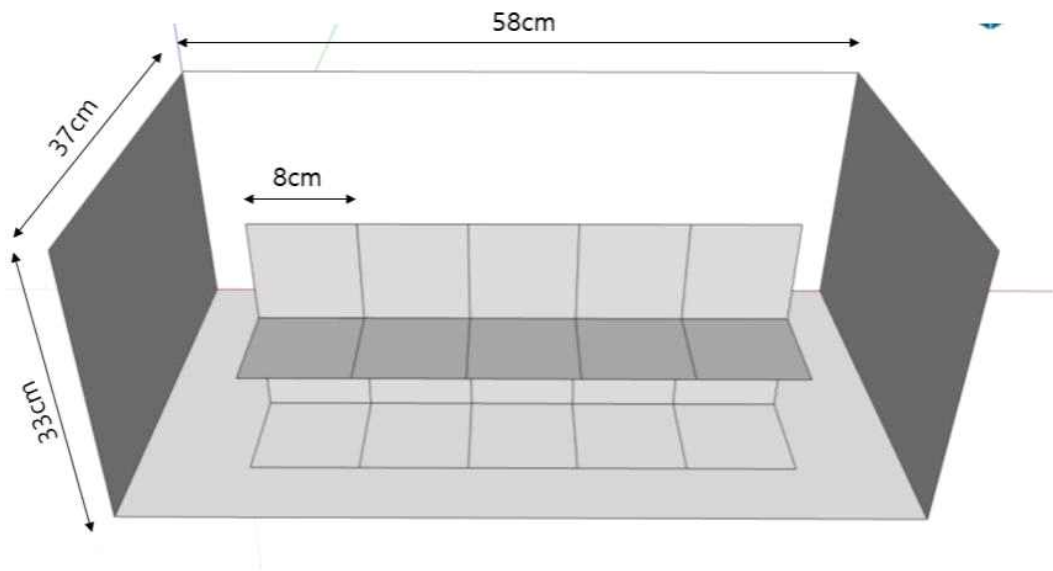
시연을 위해 지하철은 한 칸만 만들었고, 지하철 내의 좌석은 5개로 제한하여 제작했다. 또한, 해당 시스템은 입실한 승객은 빈 좌석이 있다면 무조건 착석하며, 좌석에는 사람만이 앉을 수 있다는 것을 전제로 한다.

나. 하드웨어 구성

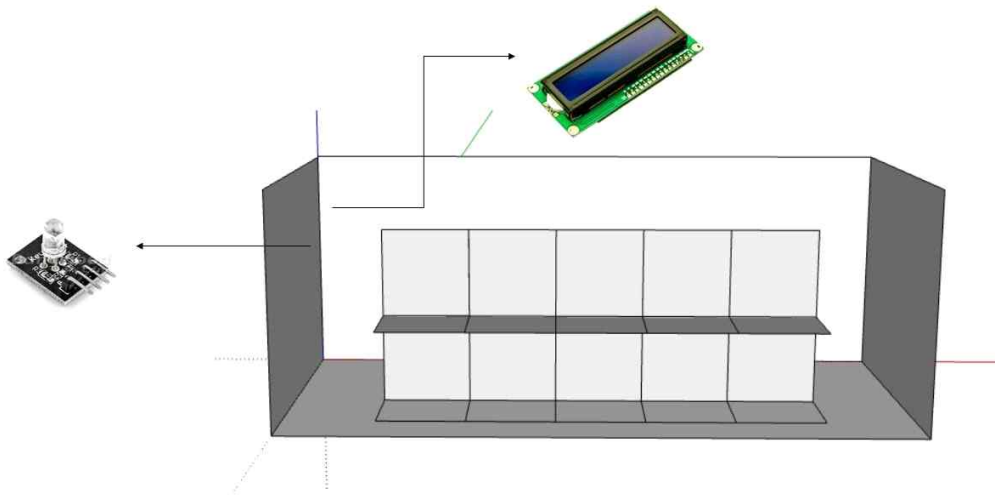
1) 하드웨어 설계도



<그림 2 : 하드웨어 전체 설계도>



<그림 3 : 하드웨어 설계도 2>



<그림 4 : 하드웨어 설계도 3>

2) 기본 부품

이번 프로젝트에 사용된 기본 부품에는 폼보드, 투명 아크릴판, 스티로폼, 점프선, 브레드보드, 그리고 라즈베리파이가 있다. 지하철 모형과 다섯 개의 좌석을 만드는 데는 폼보드를 주로 사용하였다. 지하철의 앞면의 경우에는, 내부 상황을 보여주기 위해 투명 아크릴판을 사용하여 제작했고, 천장은 냉기가 빠져나가는 것을 최대한 막기 위해 스티로폼을 사용하여 제작하였다. 각 모듈은 점프 선을 사용해 브레드보드를 통해 라즈베리파이에 연결되도록 하였다.

모든 모듈 및 센서의 전원은 브레드보드를 통해 라즈베리파이의 5V에 연결하였고, 마찬가지로 모든 모듈의 GND는 브레드보드를 통해 라즈베리파이의 ground에 연결하였다. 따라서 아래에서는 전원과 ground를 제외한 각 모듈에서만 사용되는 pin의 연결 관계에 대해 다루도록 한다.

3) LCD&LED

우선, LCD는 빛의 투과 여부에 따라 액정 화면에 문자를 표현하는 장치이다. 본 프로젝트에서는 사용하는 pin의 개수를 줄이기 위해 I2C 16*2 LCD를 사용하여 지하철에서 해당 칸의 현재 온도와 탑승 승객의 수를 출력하였다. 윗줄은 현재 온도를, 아랫줄은 탑승 승객의 수를 출력한다. SDA 핀은 라즈베리파이의 GPIO 2번, SCL 핀은 라즈베리파이의 GPIO 3번과 연결하여 동작하도록 하였다.

사용 전압	사용 전류	출력 범위	통신 방식
5V	15mA	16x2	I2C

<표 2 : LCD>

RGB LED는 Red, Green, Blue 3색의 LED를 하나로 합친 발광 다이오드이다. 내부에 저항이 포함된 RGB LED 모듈을 사용하여 지하철의 해당 칸 내 탑승 승객의 포화도를 알리도록 하였다. 의자 다섯 개를 기준으로 4인 이하가 착석하였을 시 초록 불이, 5인이 착석하였을 시 빨간불이 켜진다. 하나의 LED로 두 가지 색을 표현하여야 하므로 RGB LED 모듈을 사용하였고, 빨간빛을 내는 R핀은 라즈베리파이의 GPIO 18, 초록색 빛을 내는 G 핀은 라즈베리파이의 GPIO 17에 연결하였다.

LCD와 LED는 지하철 이용객들이 각 칸의 환경을 확인하고, 비교하는 용도로 사용되는 모듈들이므로, 지하철의 앞면에 설치하였다.

사용 전압	색상
5V	RGB

<표 3 : RGB LED>

4) 온도 모듈

DHT 11 센서는 온도와 습도를 측정하는 센서이다. 물질의 저항값이 온도나 습도에 따라 변화되는 성질을 이용한다. 지하철 해당 칸의 온도를 측정하기 위해 저항이 포함된 DHT 11 모듈을 사용하였다. 데이터를 읽어오는 signal 핀은 라즈베리파이의 GPIO 4에 연결하였다. 본 모듈이 읽어온 습도 값은 사용하지 않고, 온도 값은 LCD의 윗줄에 출력되도록 하였다. 해당 칸의 온도를 읽어야 하므로, 지하철의 내부에 설치하였다.

측정 온도	작동 전압	작동 전류
0~50℃	3.3~5V	0.3mA

<표 4 : 온도 모듈 - DHT 11>

5) 조도 센서

조도 센서는 빛의 세기에 따라 저항값이 바뀌는 센서이다. 지하철 해당 칸의 착석 승객 수를 측정하기 위해 사용하였다. 일정 수준(본 프로젝트에서는 35)보다 낮은 저항값을 가지면 승객이 착석한 것으로 인식해 승객을 추가하고, 일정 수준보다 높은 저항값을 가질 때는 승객이 착석하지 않은 상태로 인식한다. 라즈베리파이는 아날로그값을 디지털 값으로 변환할 수 없으므로, MCP 3008을 사용하여 조도 센서가 측정한 아날로그 값을 디지털 형태로 바꾸어 라즈베리파이에 입력하도록 하였다. MCP의 SPI 통신에 필요한 CLK, Din, Dout, CS/SHDN 핀은 라즈베리파에서 SPI 통신을 지원하는 GPIO 8, 9, 10, 11번에 연결하였고, 조도 센서는 MCP의 채널 핀에 하나씩 연결하였다. 조도 센서는 착석 여부를 알아 봐야 하므로 각 좌석의 앞는 부분에 설치하였다.

저항값 범위	0Lux/1.0MΩ 10Lux(25℃)/8~20KΩ
최대 전압	150V
최대 전력	100mW
사용 가능 온도	-30~70℃

<표 5 : 조도 센서 - GL 5528>

6) 열전소자 쿨러 모듈/릴레이 모듈

펄터어 효과란 서로 다른 두 개의 소자 양단에 직류 전압을 가하면, 전류 방향에 따라 한쪽 면은 흡열을 하고, 반대쪽 면을 발열을 일으키는 현상을 말한다. 이 효과를 사용하여 만든 모듈이 열전소자 쿨러 모듈이다. DC 12V 5A 이상의 전원이 필요한 열전소자 쿨러 모듈을 사용하여 에어컨과 같이 작동하도록 하였다.

라즈베리파이는 5V까지만 출력되고, 열전소자 쿨러 모듈은 전원 공급을 제어하는 기능이 없으므로 DC 12V 6A 어댑터와 릴레이 모듈을 사용하여 이를 해결해야 했다. 어댑터 전선의 피복을 벗겨 전선을 자르고, 릴레이의 NC와 NO 터미널에 연결하였다. 그리고 릴레이의 signal 핀을 라즈베리파이의 GPIO 27에 연결하여 전원 공급을 제어하였다. 라즈베리파에서 신호가 들어오면 릴레이의 스위치가 작동하고, 그에 따라 열전소자 쿨러 모듈에 전원의 공급이 조절되는 원리이다. 열전소자 쿨러 모듈은 지하철 모형의 천장에 설치하였다.

열전소자 쿨러 모듈	
가동 전원	DC 12V, 5A
냉각 능력	60W

<표 6 : 열전소자 쿨러 모듈>

릴레이 모듈	
채널 개수	1
최대 입력 가능 전원	AC 250V
제어 입력 전압	5V
작동 전류	65mA

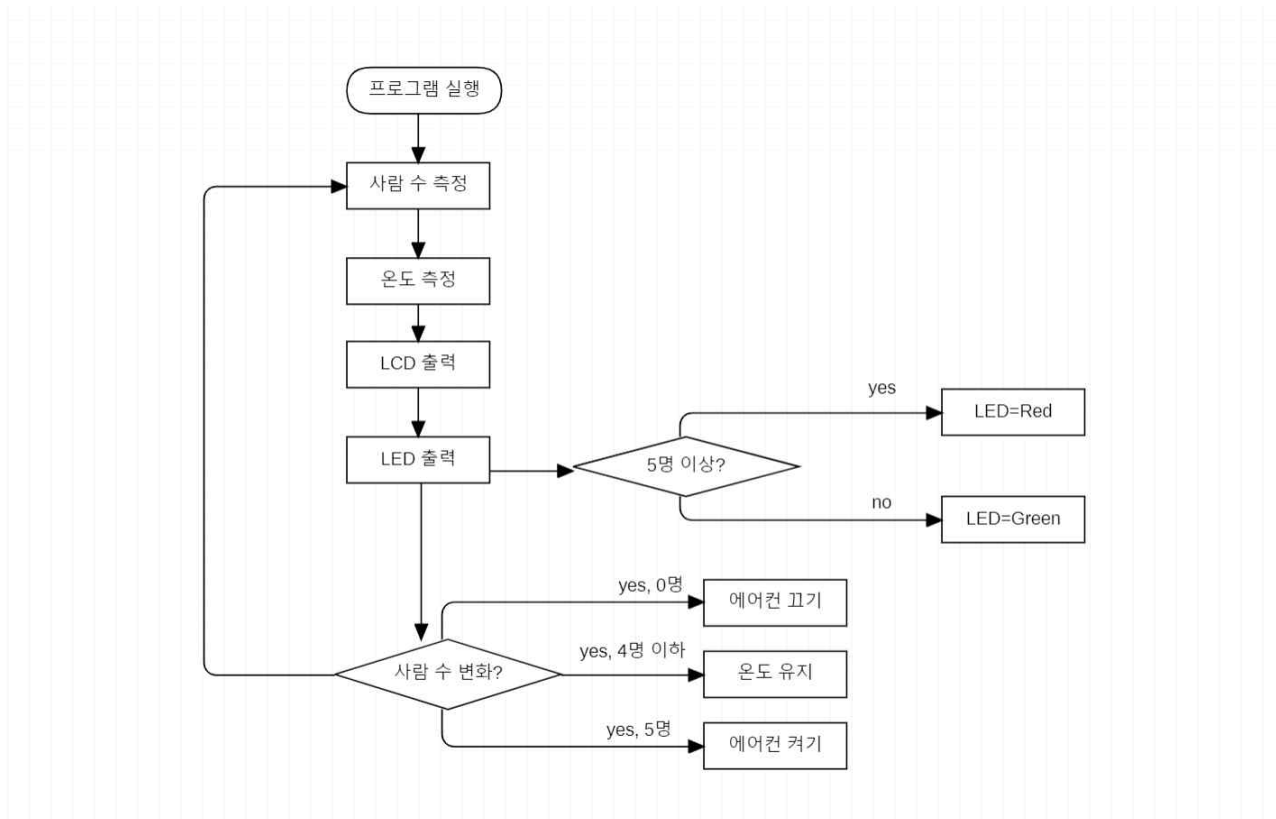
<표 7 : 릴레이 모듈>

다. 소프트웨어 구성

1) 개발 방법

본 프로그램은 C언어를 사용하여 구현하였다. 모든 모듈 및 센서는 C언어로 GPIO 핀을 제어하기 위해 wiringPi 라이브러리를 사용하였다. 각 모듈과 센서들의 동작을 함수로 구현한 다음, 한 프로그램 안에서 동작하도록 하였다.

2) 플로우차트 및 알고리즘



<그림 2 : 플로우 차트>

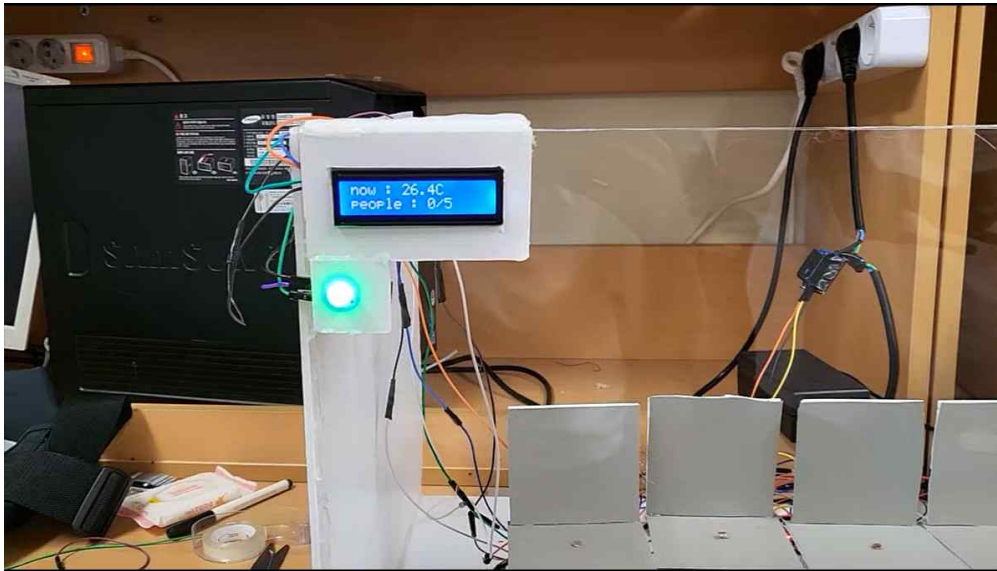
3) 프로그램 설명

프로그램을 실행한 후, 가장 먼저 이전 탑승객 수를 저장한다. 조도 센서를 이용하여 현재 해당 칸의 탑승객 수를 측정하고, DHT11를 사용하여 현재 온도를 측정한다. LCD의 첫째 줄에는 “now: “와 현

재 탑승객 수를 출력하고, 두 번째 줄에는 현재 온도와 “C”를 출력하도록 한다. 현재 탑승객 수가 5명 이상이라면 LED가 빨간 불을, 4명 이하라면 초록 불을 켜다. 만약, 이전에 측정했던 탑승객 수와 현재 탑승객 수가 달라졌다면, 사람이 없는 경우 에어컨을 끄고, 사람이 4명 이하인 경우 에어컨을 켜다 켜기를 반복하며 온도를 유지하고, 5명 이상인 경우 에어컨을 계속 켜다. 본 프로그램은 약 10초 간격으로 위에 기술된 과정이 반복된다.

3. 결론

가. 시연 결과



<사진 1 : 좌석 5개가 비어있을 때의 LED와 LCD>

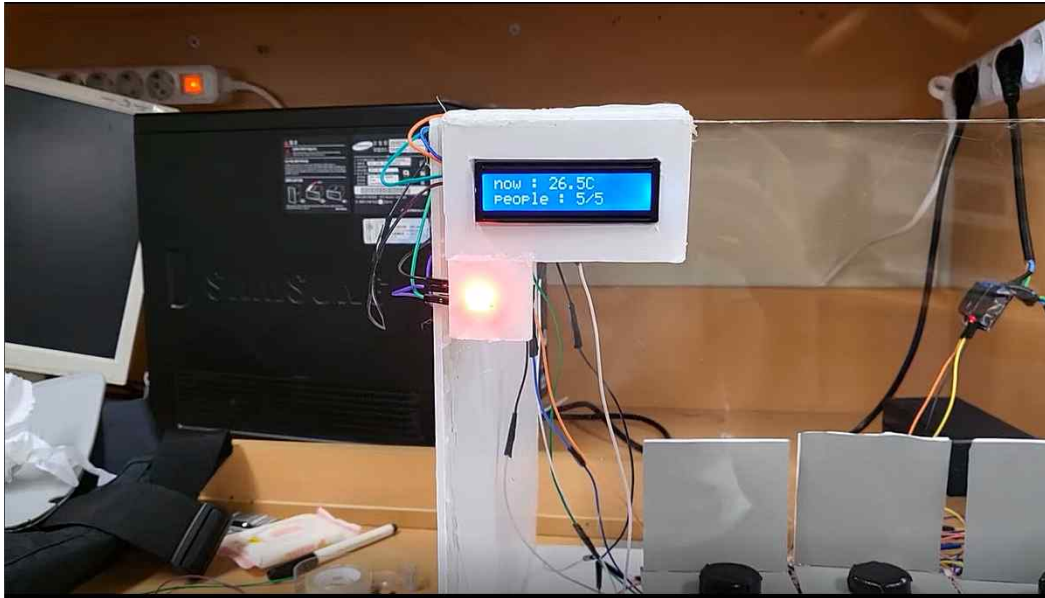
승객이 한 명도 탑승하지 않아 좌석 5개 모두가 비어있을 때의 LED와 LCD이다. LCD에는 현재 지하철 내의 온도(now)와 탑승한 인원(people)이 출력되는데, 조도 센서가 계속해서 일정 수준 이상의 빛을 감지하고 있으므로, 탑승한 인원이 없다는 것을 인식한 모습이다. 때문에, 지하철을 탑승하려는 승객에게 여석이 존재한다는 것을 알려주기 위해 LED에서 초록빛을 내고 있다.



< 사진 2 : 승객 1명이 탑승했을 때의 LED와 LCD>

승객이 1명 이상, 4명 이하일 때의 모습이다. 본 사진은 승객이 1명이 탑승했을 때의 LED와 LCD를

보여주고 있다. 조도 센서에서 1명이 탑승했다고 인식하고, 이를 통해 people을, 온도 센서를 통해 읽어 온 현 온도 now를 LCD에서 나타낸다. 해당 상황 역시 여석이 존재하므로 LED에서는 초록빛을 내어 여석이 존재한다는 것을 알린다.



<사진 3 : 여석이 없을 때의 LED와 LCD>

좌석 5개 중 모든 승객이 탑승하여 여석이 존재하지 않을 때의 모습이다. 현 온도는 26.5°C이며, 모든 좌석에 승객이 탑승했다는 사실을 LCD에 나타낸다. 또한, 여석이 존재하지 않다는 것을 승객들에게 알리기 위해 LED에서는 붉은빛을 낸다.

위의 사진을 포함한 전체 시연은 동영상으로 첨부한다.

<https://youtu.be/LfYoQMCuLt8>

나. 시행 착오

1. 열전소자 쿨러 모듈

처음 프로젝트를 계획할 때는 열전소자 쿨러 모듈을 라즈베리파이의 GPIO 핀과 연결하여 구현하려고 하였다. 하지만, 본 프로젝트에서 사용한 열전소자 쿨러 모듈은 정격 DC 12V 5A 이상의 전원을 공급, 사용하여야 하는 하드웨어이다. 라즈베리파이의 GPIO 핀은 3.3V, 예외적으로 5V로 구동되기 때문에 해당 모듈과 라즈베리파이를 바로 연결하여 사용할 수 없는 환경이다. 따라서, 열전소자 쿨러 모듈과 어댑터를 연결하여 전원을 공급받을 수 있도록 하는 작업이 필요했다. 어댑터는 DC 12V 6A로, AC 220V를 입력받아, DC 12V 6A로 변환시켜 사용하였다.

2. delay

본 프로젝트의 프로그램 구현에서 각각의 함수마다 필요한 delay를 조정하는 데에 있어 어려움을 겪었다. 예를 들어, 현 온도를 유지하기 위해서는 열전소자 쿨러 모듈을 켜고 끄는 과정이 필요하다. 이때, 어느 정도의 시간 동안 과정을 반복할 것인지, 한 사이클을 어느 정도의 시간으로 조정해야 할지

를 많은 시행착오를 통해 적절한 delay time을 찾았다.

3. 조도 센서

라즈베리파이의 GPIO는 디지털 입력만 가능하다는 사실을 인지하지 못한 채 프로젝트를 진행하였다. 조도 센서와 라즈베리파이를 곧바로 연결하여 사용하였는데, 제대로 된 결과값이 도출되지 않는 것을 확인하고 이유를 고찰하였다. 라즈베리파이의 입력은 디지털만이 가능하다는 사실을 깨달아, 아날로그 값에서 디지털 값으로 변환할 수 있는 모듈의 필요성을 인식하였다. ADC(Analog Digital Converter)를 별도로 사용하여 아날로그값인 빛의 양을 입력받고, ADC를 통해 Digital 값으로 변환한 후, 라즈베리파이의 GPIO에 연결되도록 하였다. 본 프로젝트에서는 ADC 칩으로 MCP 3008을 사용하였다.

4. 전원 공급

모든 모듈을 라즈베리파이의 같은 GPIO 5V와 GND에 연결하여 구현하였다. 하지만 릴레이 모듈의 전원이 제대로 들어오고 있지 않다는 것을 확인하고, 너무 많은 모듈을 하나의 전원에 연결하여 문제가 발생한 것 같다는 판단을 했다. 다른 모듈들은 기존과 같이 VCC는 GPIO 5V인 2번 핀에, GND는 6번 핀에 연결하였고, 릴레이 모듈은 VCC 4번, GND 39번에 연결하여 각각의 모듈의 전원을 나누어서 공급받을 수 있도록 하였다.

다. 기대 효과

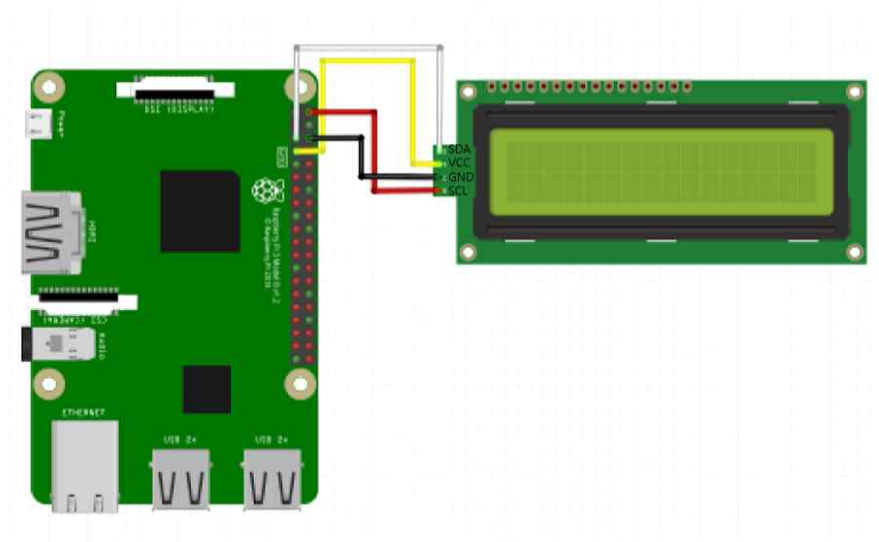
본 프로젝트에서 제안한 시스템을 적용한다면, 자동 냉방 제어가 되어 불필요한 에너지 손실을 막을 수 있다. 또한, 지하철을 탑승하는 승객들에게 좌석 정보를 제공함으로써 승객들이 원하는 칸에 탑승할 수 있도록 하여 편리한 대중교통 이용에 도움을 줄 수 있다. 더 나아가, 지하철이라는 공간뿐만 아니라 다양한 실내 공간에서도 적용된다면 더욱 많은 에너지 손실을 막을 수 있다는 이점을 기대할 수 있다.

라. 느낀점

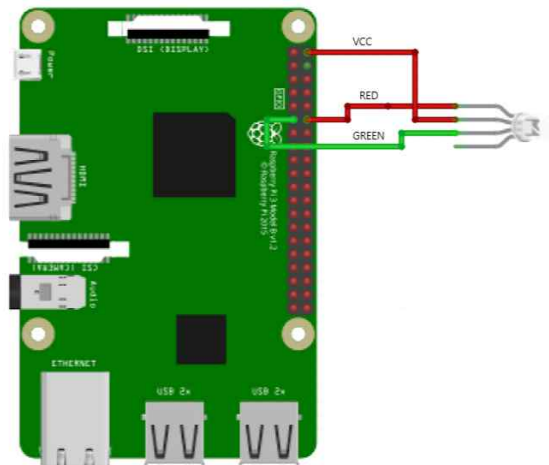
기존 제안서의 계획대로 구현하기에는 많은 제약이 존재했다. 수많은 시행착오를 통해 문제점을 개선하고, 최대한 실용적으로 구현하고자 노력하였다. 실생활에서 IoT 기술이 사용되기 위해서는 사용자 설정을 최소화해야 하고, 그를 위해서는 다양한 경우의 수를 고려하여 기술을 구현하는 것이 중요함을 실감했다. 또한, 라즈베리파이는 아두이노와 비교하여 조금 더 많고 다양한 기능을 구현할 수 있지만 동작을 시킴에는 더 많은 고민이 필요하다고 생각된다.

4. 부록

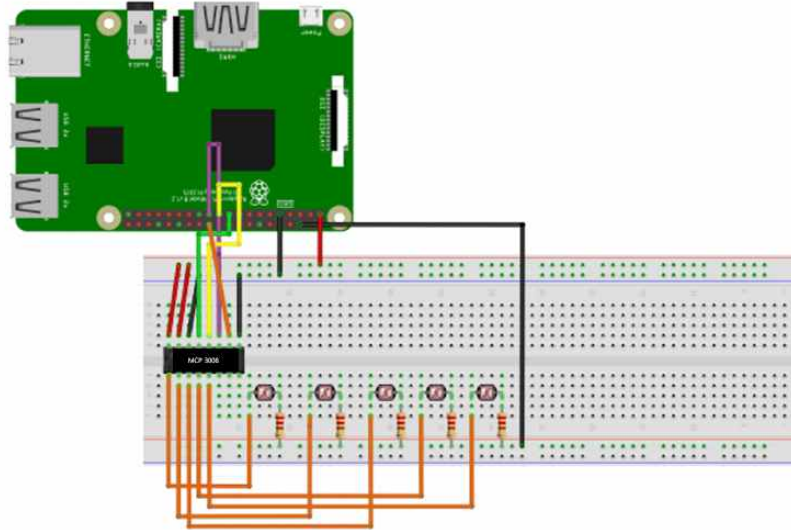
가. 회로도



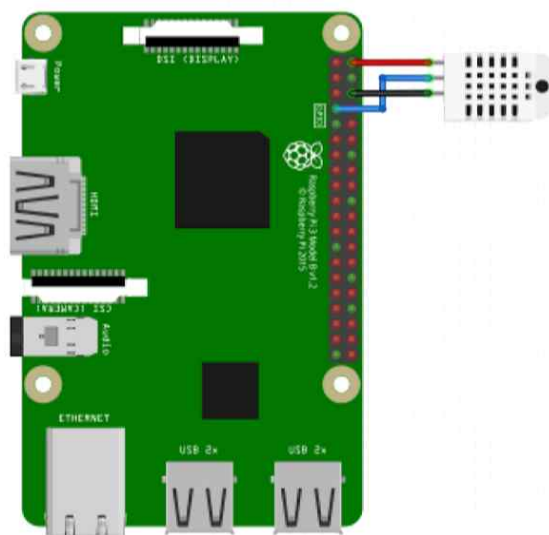
<회로도 1 : LCD 모듈>



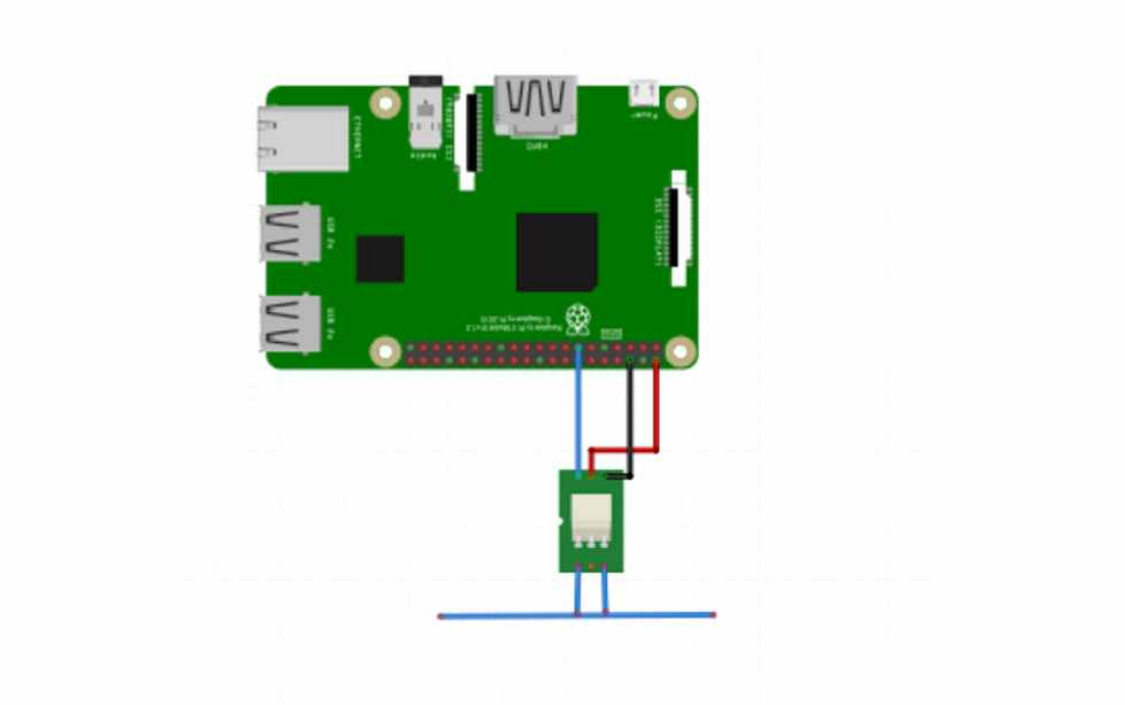
<회로도 2 : RGB LED 모듈>



<회로도 3 : 조도 센서 관련 회로도>



<회로도 4: 온도 센서 DHT 11 모듈>



<회로도 5 : 릴레이 모듈>

이름	GPIO	핀 번호
VCC	5V	2
GND	GROUND	6
DHT11 - DATA	4	7
LCD - SDA	2	3
LCD - SCL	3	5
LED - G	18	12
LED R	17	11
MPC3008 - CLK	8	24
MPC3008 - Din	11	23
MPC3008 - Din	9	21
MPC3008 - CS/SHDN	10	19
relay - VCC	5V	4
relay - GND	GROUND	39
relay	27	13

<표 8 : 라즈베리파이 GPIO 연결 핀>

나. 코드

```
#include <wiringPiI2C.h>
#include <wiringPiSPI.h>
#include <wiringPi.h>
#include <softPwm.h>
#include <lcd.h>
#include <mcp3004.h>

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

#define MAXTIMINGS 85
#define DHTPIN 7
int dhtVal[5] = { 0, 0, 0, 0, 0 }; // 온도 정보를 담을 배열

#define uchar unsigned char
#define LedPinRed 0
#define LedPinGreen 1

#define REL 2

#define I2C_ADDR 0x27
#define LCD_CHR 1
#define LCD_CMD 0
#define LINE1 0x80
#define LINE2 0xc0
#define LCD_BACKLIGHT 0x08
#define ENABLE 0b00000100

#define BASE 100
#define SPI_CHAN1 0 // 조도 센서 1
#define SPI_CHAN2 1 // 조도 센서 2
#define SPI_CHAN3 2 // 조도 센서 3
#define SPI_CHAN4 3 // 조도 센서 4
#define SPI_CHAN5 4 // 조도 센서 5
int people = 0; //사람 수

void printLCD(int* temperature, int people); // LCD에 온도와 사람 수를 출력하는 함수
void ClrLcd(); // LCD를 초기화하는 함수
```

```

void lcdLoc(int line); // LCD에 출력할 line을 지정하는 함수

// I2C 통신 관련 함수
void lcd_byte(int bits, int mode);
void lcd_toggle_enable(int bits);
void lcd_init();
int fd;

// 온도 출력 관련 함수
void typeInt(int i);
void typeIn(const char* s);
void typeIn(const char* s);

void AirconMaintain(); // 열전소자 쿨러 모듈을 켜고, 끄기를 반복하는 함수
void AirconTurnoff(); // 열전소자 쿨러 모듈을 키는 함수
void AirconTurnon(); // 열전소자 쿨러 모듈을 끄는 함수

void TurnLED(int people); //LED를 키는 함수
void ledInit(void); // LED 설정을 초기화하는 함수
void ledColorSet(uchar r_val, uchar g_val, uchar b_val); // LED 색을 설정하는 함수

void MeasureTemperature(); // 온도를 측정하는 함수

void People(); // 사람 수를 측정하는 함수
void detectPeople(int val); // 사람을 인식하는 함수

int main(void)
{
    void (*fp)() = AirconTurnoff; // 처음에는 에어컨을 끄고 시작한다.
    if (wiringPiSetup() == -1) exit(1); // wiringPi 설정
    if (wiringPiSPISetup(0, 500000) == -1) exit(1); //wiringPiSPI 설정
    mcp3004Setup(BASE, 0); // mcp3008 모듈 설정

    while (1) {
        int past_people = people; // 이전 승객 수 저장

        People(); // 현재 승객 수 측정
        MeasureTemperature(); // 온도 측정
        int temperature[2] = { dhtVal[2], dhtVal[3] }; // 온도의 정수값, 소숫값 배열에 저장
        printLCD(temperature, people); // 온도와 승객 수 LCD에 출력
        TurnLED(people); // 사람 수에 맞는 LED 불 ON
    }
}

```

```

        if (past_people != people) { // 승객 수가 달라졌을 때

            if (people > 0 && people < 5) { //0명 이상 4명 이하 인 경우 -
AirconMaintain()

                fp = AirconMaintain;
            }
            else if (people == 0) { //0명인 경우 - AirconTurnoff()
                fp = AirconTurnoff;
            }
            else if (people >= 5) { //5명인 경우 - AirconTurnon ()
                fp = AirconTurnon;
            }
        }

        //설정한 에어컨 모드 켜기
        fp();

        delay(8500); //8.5초 delay
    }
    return(0);
}

// LCD에 온도와 사람 수를 출력하는 함수
void printLCD(int* temperature, int people) {
    fd = wiringPiI2CSetup(I2C_ADDR); //I2C 통신 시작

    lcd_init(); // LCD 설정 초기화
    ClrLcd(); // LCD 초기화

    lcdLoc(LINE1); // LCD 첫번째 줄
    typeIn("now : "); //"now: " 출력
    typeInt(temperature[0]); // 온도 정수값 출력
    typeIn("."); // "." 출력
    typeInt(temperature[1]); // 온도 소숫값 출력
    typeIn("C"); // "C" 출력

    lcdLoc(LINE2); // LCD 두번째 줄
    typeIn("people : "); // "people: " 출력
    typeInt(people); // 사람 수 출력
    typeIn("/5"); // "/5" 출력
}

```

```

// LCD 정숫값 출력
void typeInt(int i) {
    char array1[20];
    sprintf(array1, "%d", i);
    typeIn(array1);
}

// LCD 초기화
void ClrLcd() {
    lcd_byte(0x01, LCD_CMD);
    lcd_byte(0x02, LCD_CMD);
}

// LCD 설정
void lcdLoc(int line) {
    lcd_byte(line, LCD_CMD);
}

// 온도 정숫값 설정
void typeIn(const char* s) {
    while (*s) lcd_byte(*(s++), LCD_CHR);
}

//LCD 출력 설정
void lcd_byte(int bits, int mode) {
    int bits_high;
    int bits_low;

    bits_high = mode | (bits & 0xF0) | LCD_BACKLIGHT;
    bits_low = mode | (bits << 4) & 0xF0 | LCD_BACKLIGHT;

    wiringPiI2CReadReg8(fd, bits_high);
    lcd_toggle_enable(bits_high);

    wiringPiI2CReadReg8(fd, bits_low);
    lcd_toggle_enable(bits_low);
}

// LCD 출력 설정
void lcd_toggle_enable(int bits) {
    delayMicroseconds(500);
    wiringPiI2CReadReg8(fd, (bits | ENABLE));
}

```

```

        delayMicroseconds(500);
        wiringPiI2CReadReg8(fd, (bits & ~ENABLE));
        delayMicroseconds(500);
    }

    // LCD 출력 설정
    void lcd_init()
    {
        lcd_byte(0x33, LCD_CMD);
        lcd_byte(0x32, LCD_CMD);
        lcd_byte(0x06, LCD_CMD);
        lcd_byte(0x0C, LCD_CMD);
        lcd_byte(0x28, LCD_CMD);
        lcd_byte(0x01, LCD_CMD);
        delayMicroseconds(500);
    }

    void AirconMaintain() { // 열전소자 쿨러 모듈을 켜고, 끄기를 반복하는 함수
        pinMode(REL, OUTPUT);

        digitalWrite(REL, LOW); // 4초 동안 끄기
        delay(4000);

        digitalWrite(REL, HIGH); // 3초 동안 켜기
        delay(3000);
    }

    void AirconTurnoff() { // 열전소자 쿨러 모듈을 끄는 함수
        pinMode(REL, OUTPUT);
        digitalWrite(REL, LOW); // 끄기

        return;
    }

    void AirconTurnon() { // 열전소자 쿨러 모듈을 키는 함수
        pinMode(REL, OUTPUT);

        digitalWrite(REL, HIGH); // 켜기

        return;
    }
}

```

```

void TurnLED(int people) { // LED를 키는 함수
    ledInit(); // LED 설정 초기화
    if (people >= 5) { // 승객이 5명 이상이라면
        ledInit();
        ledColorSet(0xff, 0x00, 0x00); // Red 켜기
    }
    else { // 4명 이하라면
        ledInit();
        ledColorSet(0x00, 0xff, 0x00); // Green 켜기
    }
}

// LED 초기화 함수
void ledInit(void)
{
    softPwmCreate(LedPinRed, 0, 100);
    softPwmCreate(LedPinGreen, 0, 100);
}

// LED 색 설정 함수
void ledColorSet(uchar r_val, uchar g_val, uchar b_val)
{
    softPwmWrite(LedPinRed, r_val);
    softPwmWrite(LedPinGreen, g_val);
}

void MeasureTemperature() // 온도 측정 함수
{
    uint8_t laststate = HIGH;
    uint8_t counter = 0;
    uint8_t j = 0, i;
    // 측정한 값을 배열에 초기화
    dhtVal[0] = dhtVal[1] = dhtVal[2] = dhtVal[3] = dhtVal[4] = 0;

    // MCU 18ms 동안 LOW signal 보냄
    pinMode(DHTPIN, OUTPUT);
    digitalWrite(DHTPIN, LOW); //pull down
    delay(18); // dealy

    //40micros 동안 HIGH signal 보냄

```

```

digitalWrite(DHTPIN, HIGH); //pull up
delayMicroseconds(40); // deay

//DHT가 signal 받음
pinMode(DHTPIN, INPUT);

// DHT가 MCU에 data 보내는 과정
for (i = 0; i < MAXTIMINGS; i++){
    counter = 0;
    while (digitalRead(DHTPIN) == laststate){
        counter++;
        delayMicroseconds(1);
        if (counter == 255)
            break;
    }

    laststate = digitalRead(DHTPIN);
    if (counter == 255)
        break;

    /* ignore first 3 transitions */
    if ((i >= 4) && (i % 2 == 0))
    {
        /* shove each bit into the storage bytes */
        dhtVal[j / 8] <= 1;
        if (counter > 16)
            dhtVal[j / 8] |= 1;
        j++;
    }
}

return:
}

void People() { // 사람 측정 함수
    people = 0; //사람 수 정보 초기화

    int val1 = analogRead(BASE + SPI_CHAN1); // 1번째 조도센서 값 읽기
    int val2 = analogRead(BASE + SPI_CHAN2); // 2번째 조도센서 값 읽기
    int val3 = analogRead(BASE + SPI_CHAN3); // 3번째 조도센서 값 읽기
    int val4 = analogRead(BASE + SPI_CHAN4); // 4번째 조도센서 값 읽기
    int val5 = analogRead(BASE + SPI_CHAN5); // 5번째 조도센서 값 읽기

```

```

    // 각 조도 센서의 사람 유무 판단
    detectPeople(val1);
    detectPeople(val2);
    detectPeople(val3);
    detectPeople(val4);
    detectPeople(val5);
}

void detectPeople(int val) { // 조도 센서의 사람 유무 판단
    if (val <= 20) { //밝기가 20이하이면 사람 있는 것으로 감지
        people = people + 1;
    }
}

```

다. 회의록

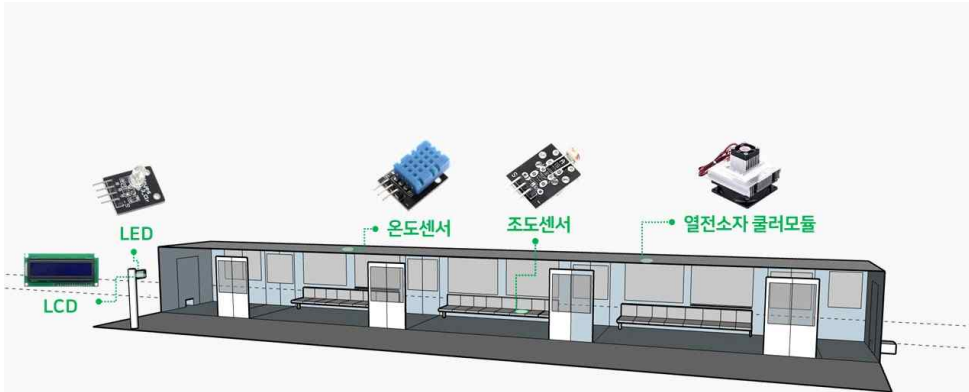
번호	날짜	시간	지난주 회의 내용	이번주 회의내용
1,	2022/03/31	pm7:00~8:00	-	프로젝트 주제선정
2.	2022/04/05	pm1:00~3:00	프로젝트 주제선정	프로젝트 구상
3.	2022/05/16	pm3:00~5:00	프로젝트 구상	프로젝트 피드백 및 수정
4.	2022/05/30	pm6:00~8:00	프로젝트 피드백 및 수정	프로젝트 최종 준비 및 발표

라. 제안서

팀원 Team Members	한다현, 이은주, 이은진, 진서연
주제 Objectives	인원 수에 따른 지하철 냉방 조절 시스템
기능 Functions	<p>1. 인원 수 감지</p> <p>지하철 좌석에 조도센서를 부착하여 착석 여부를 확인한다. 일부 조도센서가 착석을 인식하지 않으면, 그 칸에는 남아있는 좌석이 있는 것이며, 모든 좌석의 조도센서가 착석을 인식하면, 해당 칸에는 좌석 수보다 만석이라는 것을 확인할 수 있다. 또한 모든 좌석의 조도센서가 착석을 인식하지 않는다면 그 칸은 공실임을 알 수 있다.</p> <p>2. 현재 실내 온도 측정</p> <p>지하철 내부에 부착된 온도센서를 통해 현재 실내 온도를 측정한다. 지하철에 탑승한 승객이 적을 때보다 포화 상태일 때 온도가 더 높을 것이다.</p> <p>3. 실내 온도 조절</p> <p>좌석에 부착된 조도센서로 지하철 내의 승객 분포를 예측할 수 있다. 만약 지하철 칸 내의 모든 좌석이 착석 상태로 인식된다면, 내부에 설치된 에어컨을 작동시켜 현재 실내 온도에서 1 °C 낮춘다. 빈 좌석이 있을 경우엔, 현재 실내 온도를 유지하고, 모든 좌석이 비어 있다면 해당 칸은 공실이므로 에어컨 작동을 멈춘다.</p> <p>4. 만석 여부 확인</p> <p>센서를 통해 얻은 현재 지하철 내부 온도와 포화 상태 정보를 지하철 외부에 설치된 LCD에 표시한다. 승객들은 지하철을 탑승하기 전에 LCD를 확인하여 가장 쾌적한 환경을 지닌 칸을 선택할 수 있다.</p>
제한사항 Constraints	<p>하드웨어: 라즈베리파이, 열전소자 쿨러 모듈, 조도센서, 온도센서 (DHT-11), 지하철 구현용 폼보드 및 아크릴 판, 브레드보드, LCD(16x2), LED 등</p> <p>소프트웨어 : C언어</p>

프로젝트 일정 Project Schedule	WBS번호	작업명	시작일	종료일	기간	1단계												2단계												3단계			4단계			
						1주	2주	3주	4주	5주	6주	7주	8주	9주	10주	11주	12주	1주	2주	3주	4주	5주	6주	7주	8주	9주	10주	11주	12주							
						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						
	1	프로젝트 계획																																		
	1.1	프로젝트 구성	22년 4월 5일	22년 4월 8일	2	■																														
	1.1.1	설계도서서 작성	22년 4월 7일	22년 4월 8일	5		■	■																												
	1.2	제안자료 ppt 작성	22년 4월 11일	22년 4월 12일	4				■	■																										
	1.3	프로젝트 제안 발표	22년 4월 13일	22년 4월 13일	1					■																										
	2	프로젝트 디자인																																		
	2.1	하드웨어 디자인	22년 4월 14일	22년 4월 18일	3					■	■																									
	2.2	하드웨어 구성	22년 4월 19일	22년 4월 21일	3						■	■																								
	2.3	소프트웨어 구성 및 디자인	22년 4월 22일	22년 4월 28일	5							■	■	■																						
	2.4	테스트 작성	22년 4월 29일	22년 5월 4일	4								■	■	■																					
	2.5	제출	22년 5월 5일	22년 5월 13일	6									■	■	■	■																			
	2.6	프로젝트 하드웨어 및 구성	22년 5월 16일	22년 5월 20일	5										■	■	■																			
	3	데모																																		
	3.1	데모 동무 확인	22년 5월 23일	22년 5월 24일	2																															
	3.2	최종 데모 하드웨어 및 구성	22년 5월 25일	22년 5월 27일	3																															
	4	최종																																		
	4.1	프로젝트 최종 준비 및 발표	22년 5월 30일	22년 6월 3일	5																															
	4.1	최종 보고서 작성	22년 6월 6일	22년 6월 10일	5																															

데모 계획 Demonstration Plan	지하철 탑승객 분포에 따라 칸 내부 온도를 조절하는 시스템을 라즈베리 파이를 통해 구현한다. (I2C통신이용)
	1. 지하철 내의 인원 수는 각 좌석에 조도센서(GL5537)를 부착하여 파악한다. 조도센서는 빛의 세기에 따라 저항 값이 바뀌는 전자 부품이다.
	- 승객이 좌석에 착석한 경우에는, 좌석에 부착된 조도센서가 지하철 내의 빛을 받지 못할 것이므로, 저항 값이 커지게 된다.
	- 승객이 좌석에 착석하지 않았을 경우에는, 좌석에 부착된 조도센서가 지하철 내의 빛을 그대로 받아들이므로, 저항 값이 작아지게 된다. (조도센서가 10초이상 일정 수준보다 낮은 저항 값을 가지면 빈 좌석으로 인식한다.)
	2. 지하철 실내 온도는 칸 내부에 부착된 온도센서(DHT-11)를 통해 파악한다.
	- DHT-11 센서에 부착된 서미스터를 사용하여 지하철 실내 온도를 측정한다. 이는 온도가 올라가면 저항값은 내려가고, 온도가 내려가면 저항값은 올라가는 특성을 이용한 것이다.
	3. 센서를 통해 얻은 승객 분포와, 실내 온도 정보를 토대로 지하철 내부 온도를 열전소자 쿨러 모듈을 이용하여 자동으로 제어한다.
	- 조도센서가 좌석이 만석임을 인식하면 열전소자 쿨러 모듈을 On하여 현재 실내 온도에서 1도 낮춘다. 적외선 온도센서를 통해 1도 낮아진 것을 파악하게 되면 쿨러 모듈을 OFF한다.
	4.지하철 내부의 상황을 외부 LCD와 LED를 통해 실시간으로 보여준다.
	- 좌석에 부착된 조도센서로 파악한 각 칸의 빈 좌석 수를 LCD화면에 출력한다. 평상시에 LED는 초록색 불을 뜨지만, 만약 모든 좌석에 승객이 착석해있을 경우에는, LED에 빨간 불이 들어오고 LCD에 “만석”이 출력

	<p>된다.</p> <ul style="list-style-type: none"> - 온도 센서를 통해 얻은 각 칸의 실내 온도를 출력한다. <p>5. 지하철 모형을 구현하기 위해 폼보드와 투명 아크릴판을 사용한다. 전체적인 모형은 폼보드를 사용하고, 앞 면은 투명 아크릴판을 사용하여 지하철 내의 모습을 관찰하는 것을 용이하게 한다.</p>
Notes	<p>위 시스템은 입실한 승객은 빈 좌석이 있다면 좌석에 무조건 앉으며, 좌석에는 무조건 사람만이 앉을 수 있다는 것을 전제로 한다.</p> <p>이번 프로젝트의 목표는 지하철 내의 승객 분포에 따라 실내 온도를 조절하여 쾌적한 환경을 조성하고, 자원낭비를 줄이는 것이다. 또한 승객에게 칸 내 좌석 현황을 알려주어 대중교통 사용 시 편의를 제공할 수 있다.</p> <p>현재 지하철은 인원 수를 고려하지 않고 실내 온도를 설정한다. 이에 실내 인원 수가 적더라도 계속 낮은 온도를 유지되거나, 인원이 많더라도 높은 온도가 지속되는 상황이 종종 발생한다. 하지만, 인원 수에 따라 온도를 조절하는 시스템을 도입하게 된다면 불필요한 에너지 소모를 줄일 수 있고, 보다 쾌적한 환경을 조성할 수 있다. 더욱 나아가, 해당 시스템을 지하철뿐만 아니라, 다양한 실내에서도 적용하게 된다면 더 많은 자원 절약을 기대할 수 있다.</p> <p><프로젝트 데모 그림></p> 

5. 레퍼런스

그림 1: <https://ednieuw.home.xs4all.nl/Woordklok/LDR/GL55xxLDRs.html>

모듈/센서 정보 <https://mechasolution.com/shop/main/index.php>