



---

# *Software Engineering*

## *Final Project*

---

이문근 교수님

1분반 7조

201610694 최형준

201310906 박정용

201614858 유민경

201210909 윤준하

201614880 이진향

201610693 최지호

# 차례

<b>1. 개요</b>	<b>1</b>
a. 프로젝트의 소개	1
b. 프로젝트의 목적	1
c. 프로젝트 선정 주제	1
d. 프로젝트의 개요	2
e. 프로젝트의 계획	2
<b>2. 진행 과정</b>	<b>7</b>
a. 회의록	7
b. 활동 사진	90
<b>3. 결과</b>	<b>100</b>
a. 아두이노 최종 코드	100
b. 프로젝트의 목적	107
<b>3. 종합</b>	<b>123</b>
a. 결론 및 평가	123
b. 아쉬운 점	124

# 1. 개요

## a. 프로젝트의 소개

이 프로젝트는 전북대학교 컴퓨터공학부에서 진행된 「소프트웨어 공학」 수업의 일환으로, 소프트웨어 공학에 대한 이론적 이해를 바탕으로 모델링과 H/W 제어를 통해 현실적인 문제를 마주하고 여기에 소프트웨어 공학 기법을 사용하여, 문제를 해결한다.

## b. 프로젝트의 목적

수업에서 배운 소프트웨어에 관한 일반적인 이론과 내용을 토대로 소프트웨어에 대한 정의, 구조, 내용, 기능, 행위적인 특성과 이를 사용하기 위한 단계들을 직접 작성해보고 사용해봄으로써 요구분석과 검증, 관리 및 유지보수를 경험하는 것을 목표로 한다.

## c. 프로젝트 선정 주제

스마트 로봇 자동차를 사용하여 현실적인 문제를 해결한다. 그 중 하나로 차량을 공용으로 이용하는 카풀 서비스를 선택하고 진행하였다. 카풀 서비스는 동일한 목적지에 도달하고 싶은 여러 사용자가 하나의 차량으로 함께 이동할 수 있도록 돕는 서비스를 말한다. 최근들어서 주목을 받기 시작한 스마트폰의 카풀 어플리케이션의 이슈와 세계적인 진출 등으로 보아 현실적인 문제를 이 프로젝트를 통해 접하고 그 설계 구조와 개발 경험을 쌓을 수 있을 것으로 판단하여 진행하게 되었다.

또한, 실생활에 밀접하게 닿아있는 주제인 이 프로젝트를 통해

향후 다른 프로젝트의 개발이나 유사한 환경을 가진 현업에 종사할 때 많은 도움이 될 수 있으며 여러 가지의 응용 방안도 모색할 수 있는 방법을 배운다.

#### d. 프로젝트의 개요

아두이노 스마트 로봇 자동차 키트를 사용하여 무인 자동차의 역할과 행동을 모방한다. 적당한 넓이의 하드보드지 위에 검정색 마스킹테이프로 도로를 표시하여 아두이노 스마트 로봇 자동차가 이를 도로로 인식하여 주행한다. 아두이노 스마트 로봇 자동차는 재난 방지 센터에서 출발하여 정해진 재난 발생 지점으로 이동하며 그 과정에서 장애물이 있는 경로로는 이동하지 않는다.

아두이노 스마트 로봇 자동차는 서버와 통신하여 주고 받은 데이터를 분석하여 정해진 움직임을 수행하며, 이러한 동작을 정확히 수행하기 위해 서버 프로그램은 ADOXX를 이용하여 우리 팀에서 구현한 모델링 기법으로 개발한다.

#### e. 프로젝트의 계획

##### - 팀 내 역할 세분화

##### 1) 로봇 자동차 설계 및 ADOxx 모델링

- 최형준, 이진향, 박정용

##### 2) H/W 제어부 아두이노 코딩

- 유민경, 최지호

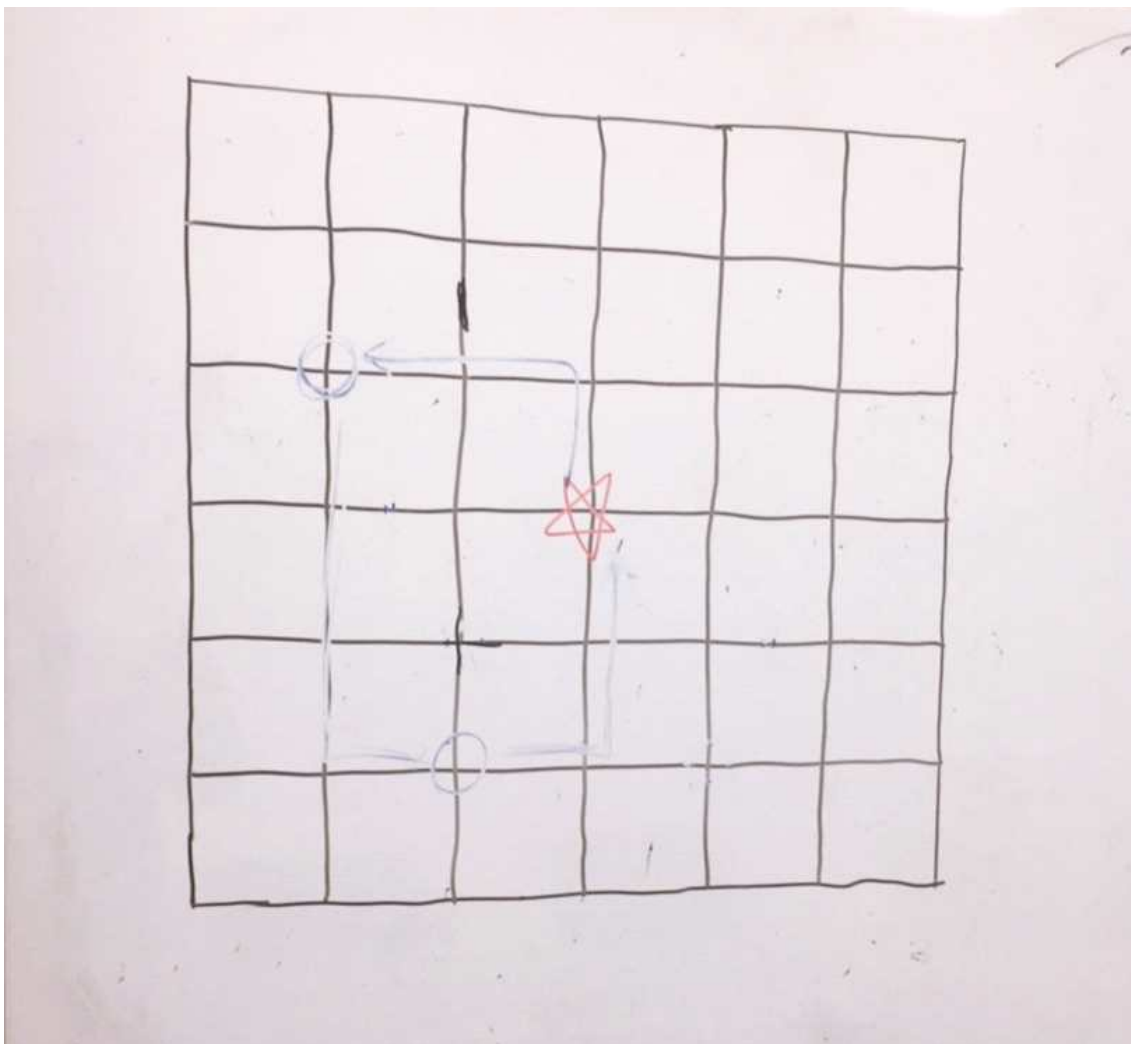
##### 3) 요구 사항 분석 및 검증, 보고서 작성

- 윤준하

- 최초 계획은 핵심적인 요구 사항만 파악한 후, 최소 요구사항을 모두 만족하는 형태를 우선적으로 개발하여 프로토타입을 완성하도록 하며, 각 팀은 최대한 모듈의 형태로 개발하여 이후 프로젝트를 위한 각 프로그램의 동작을 병합할 때의 오류를 최소화할 수 있게 하였다.

#### - 초기 계획

처음에는 프로토타입을 위한 간단한 계획을 수립하여, 이를 점진적으로 확장하였다.



최초 기획 당시에는 재난 방지 센터에서 운영되는 무인 자동차를 주제로 선정하였다. 화재나 산사태 등 사람이 진입할 수 없는 재난이 발생한 곳에 무인 자동차를 보내어 인명을 구조하거나 재해를 복구하여 시민의 안전 등 사회 복지의 질적 수준 향상을 꾀하려했고, 그에 대한 계획은 다음과 같았다.

계획 1. 격자 모양의 도로가 있고, 중앙에는 재난 방지 센터(☆ 표시)가 위치하여 무인 자동차가 재난 방지 센터에서 호출을 받고 특정 위치(재난 발생지, ○ 표시)로 이동한 후 다시 재난 방지 센터로 돌아올 수 있도록 계획을 세웠다.

계획 2. 무인 자동차의 역할을 수행하는 H/W 제어부, 즉 아두이노 스마트 로봇 자동차의 제어를 맡은 팀은 로봇 자동차가 격자 위에 표시된 도로를 따라갈 수 있도록 한다. 이를 위해 로봇 자동차에 라인 트레이서의 센서 부착이 필요하고, 이를 연동하여 주행하는 프로그래밍을 계획하였다.

하지만 프로젝트의 결과물과 그 내용과 동작 과정, 결과는 모두 동일하고 배경과 사용 목적만 바뀌었을 뿐이니 별도의 계획 변경은 없이 그대로 진행하였다.

- 세부 계획

a. 시작 전 준비사항

1. 컴파일된 바이너리 파일이 업로드 되어있는 아두이노
2. 블루투스가 페어링 되어 있는 서버
3. ADOxx 설치가 되어있는 컴퓨터

b. 전체적인 동작의 흐름

1. ADOxx를 실행한다.
2. 서버 컴퓨터가 패킷을 받는다.
3. 서버가 아두이노에게 json 형식으로 명령어를 전달한다.
4. 아두이노의 시리얼 모니터를 통해 데이터가 잘 수신되었는지를 확인한다.
5. 아두이노는 수신한 json 형식의 데이터를 읽고 해당 명령어를 수행한다.

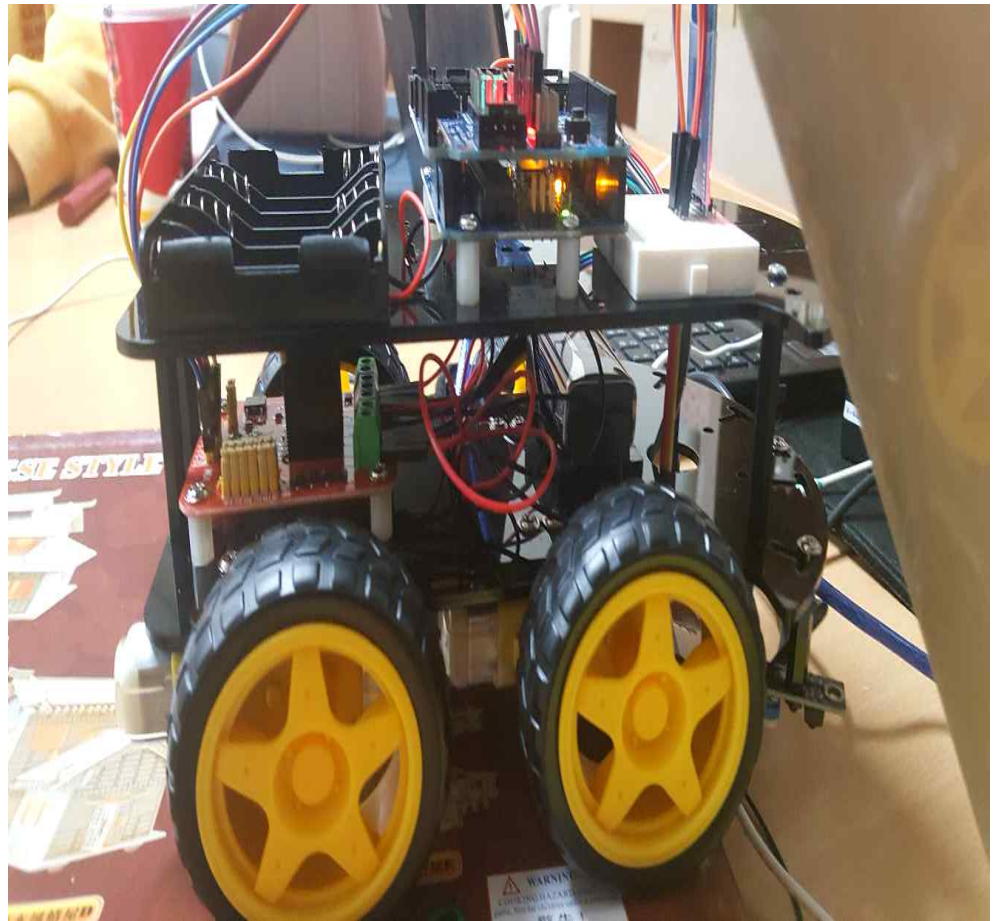
## 2. 진행 과정

### a. 회의록

- 1차 회의	
회의 일시	2018년 11월 7일 수요일 13:00~19:00
회의 장소	공과대학 7호관 6층 설계실
활동 내용 및 문제 해결	<p>아두이노 스마트 로봇 자동차 부품 확인 및 조립</p> <ul style="list-style-type: none"> <li>- 아래의 매뉴얼을 참고하여 조립을 진행하였다.</li> <li>- <a href="http://www.gameplusedu.com/pds/gpshop/html/4wd%20ebook/index.html#p=1">http://www.gameplusedu.com/pds/gpshop/html/4wd%20ebook/index.html#p=1</a></li> <li>- 자동차의 하부와 상부를 나누어 조립을 완성한 후 합쳤다.</li> <li>- 조립 과정에서 이해가 어려운 부분은 추가로 자료를 검색하여 해결하였으며, 참고한 영상은 아래와 같다.</li> <li>- <a href="https://www.youtube.com/watch?v=9zo1it5TB64">https://www.youtube.com/watch?v=9zo1it5TB64</a></li> <li>- 초음파 센서는 사용 여부에 따라 추후에 조립하도록, 조립을 우선 보류하였다.</li> </ul>



활동 사진



## - 2차 회의

회의 일시

2018년 11월 28일 수요일 20:00~21:00

회의 장소

온라인

활동 내용  
및  
문제 해결

프로젝트의 주제를 결정하기 위해 다음과 같은  
예시를 나열하였다.

1. 차량 주차
2. 깊이/너비 우선 탐색을 사용한 경로 탐색
3. 차량 충돌 방지 길찾기
4. 차량 충돌 방지 차선변경

각 주제에 대해 가능한 여러 시나리오를  
조사해도록 정하고, 다음 회의 시간과 장소를  
결정하였다.

## - 3차 회의

회의 일시

2018년 12월 5일 수요일 16:00~19:00

회의 장소

공과대학 7호관 6층 설계실

활동 내용  
및  
문제 해결

아두이노 스마트 로봇 자동차 부품 확인 및 조립  
점검

- 블루투스 모듈에 불이 들어오지 않는 것을  
확인하였고, 해당 부품이 불량인지 조립  
과정에서 잘못된 부분이 있었는지를  
확인하였다.
- 해당 블루투스 모듈을 조교에게 부탁하여  
점검을 받아 해결하였다.
- 프로젝트의 주제를 정한 결과 초음파 센서는  
사용되지 않는 것으로 판단하여 조립을  
보류하였다.

활동 내용  
및  
문제 해결

프로젝트의 주제를 결정하고, 대략적인 내용과 계획을 설정하였다.

- 이전 회의에서 나열된 프로젝트 주제의 예시 중 “깊이/너비 우선 탐색을 사용한 경로 탐색”을 응용한 형태로 주제를 확정하였다.

- 팀 내 역할 세분화

1) 로봇 자동차 설계 및 ADOxx 모델링

- 최형준, 이진향, 박정용

2) H/W 제어부 아두이노 코딩

- 유민경, 최지호

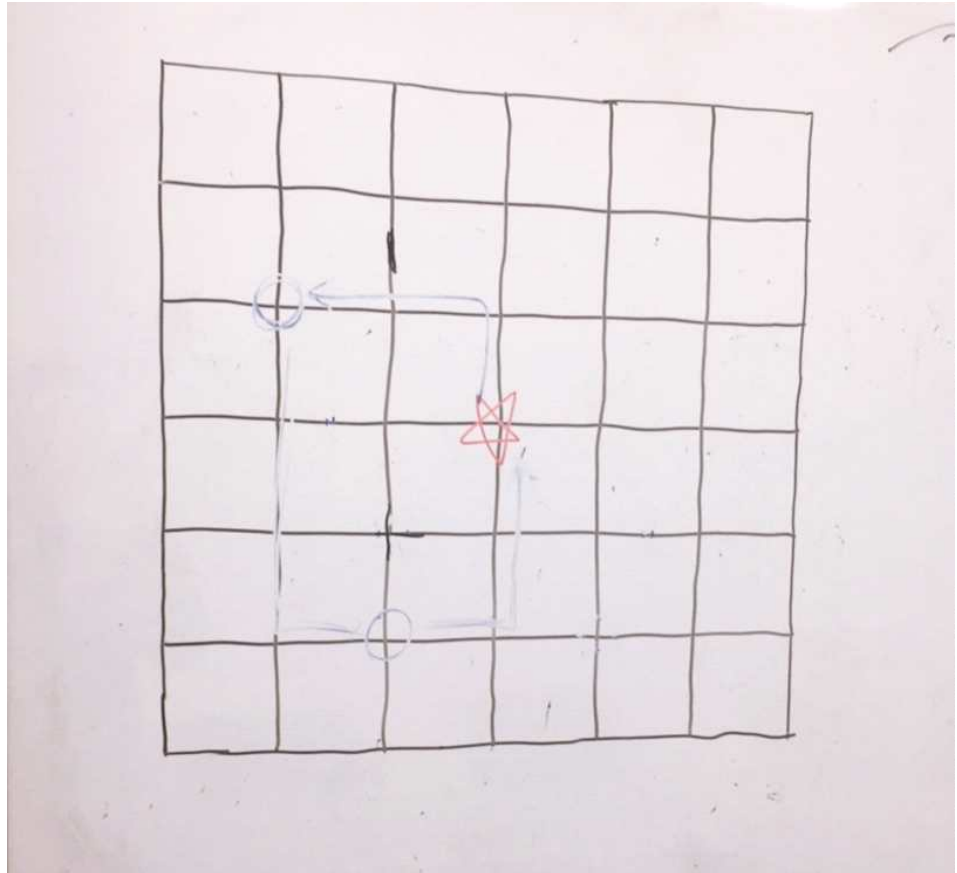
3) 요구 사항 분석 및 검증, 보고서 작성

- 윤준하

- ADOxx 모델링을 담당하는 팀을 서버팀이라 부르기로 하고, H/W를 제어하는 아두이노를 담당하는 팀을 아두이노팀이라 부르기로 약속하였다.

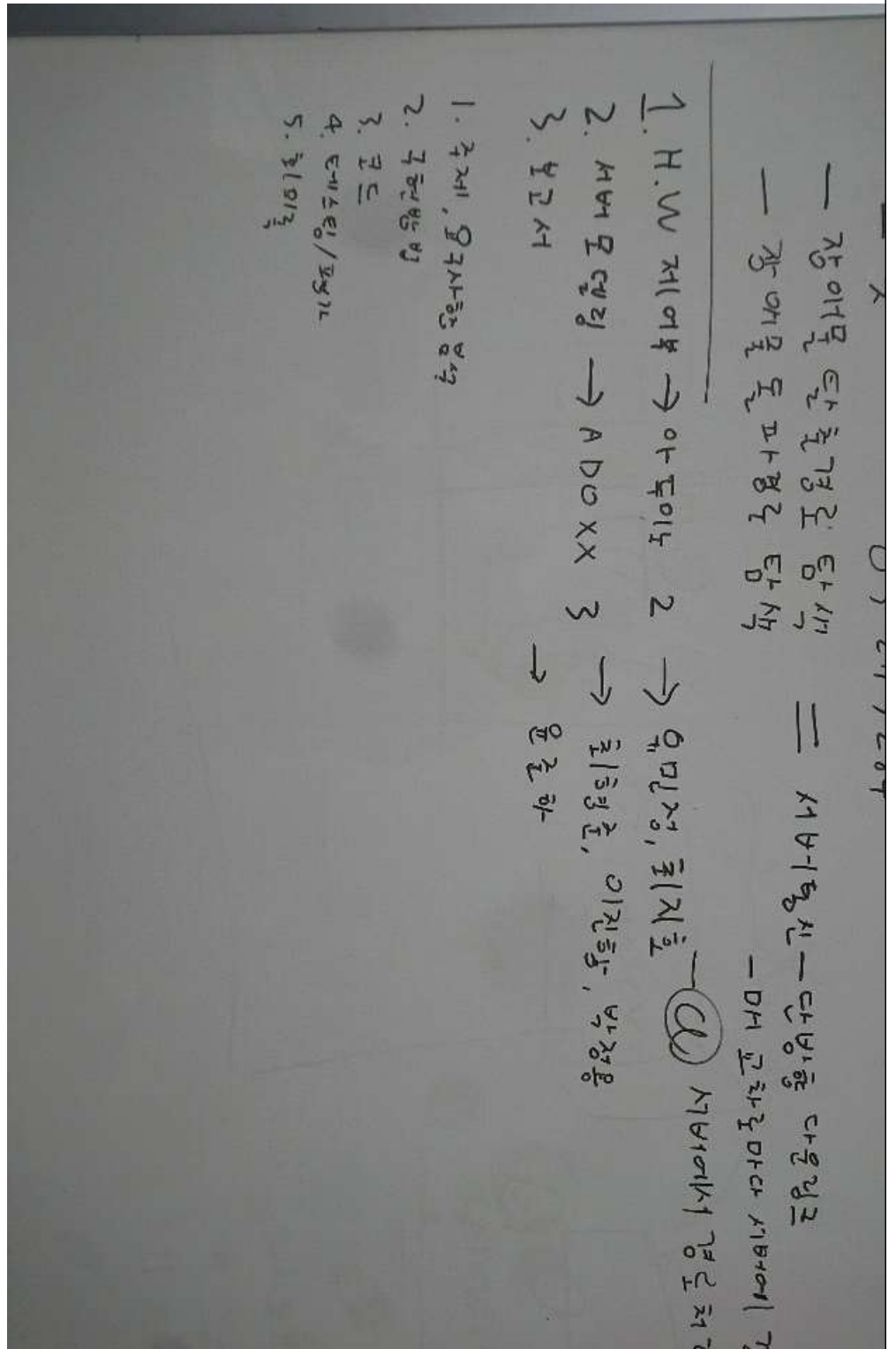
활동 내용  
및  
문제 해결

처음에는 프로토타입을 위한 간단한 계획을 수립하고, 이를 점진적으로 확장하기로 하였다.



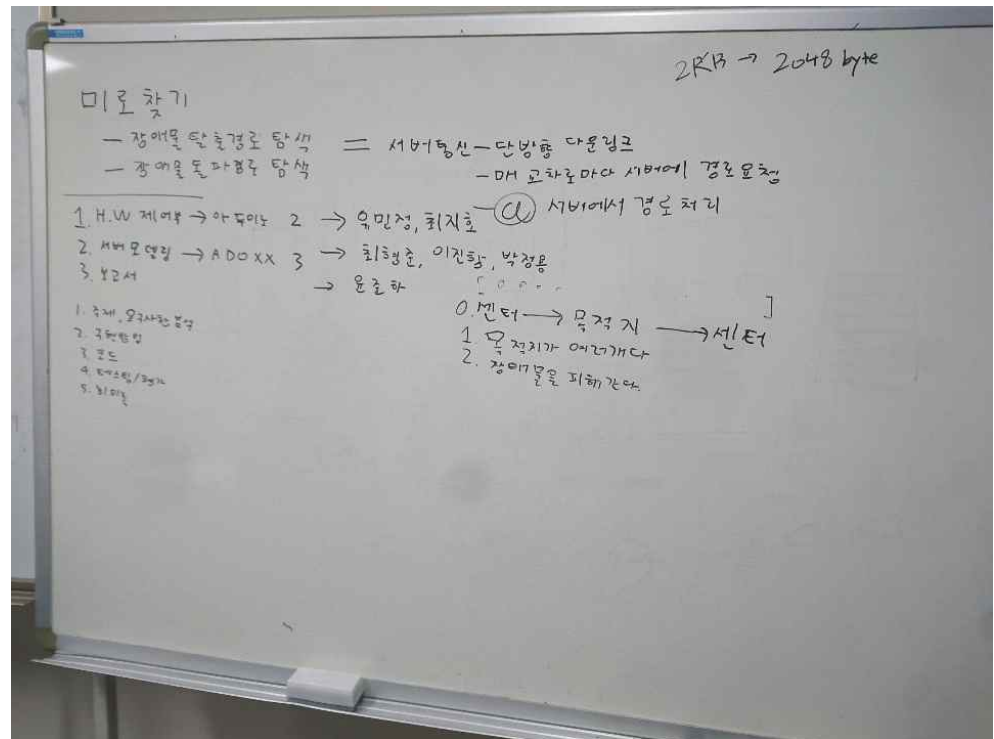
계획 1. 가로와 세로의 크기가 7인 격자 모양의 도로에서, 재난센터(☆ 표시)에서 출발하여 재난 발생지(○ 표시)로 이동 후 다시 출발점으로 돌아올 수 있도록 한다.

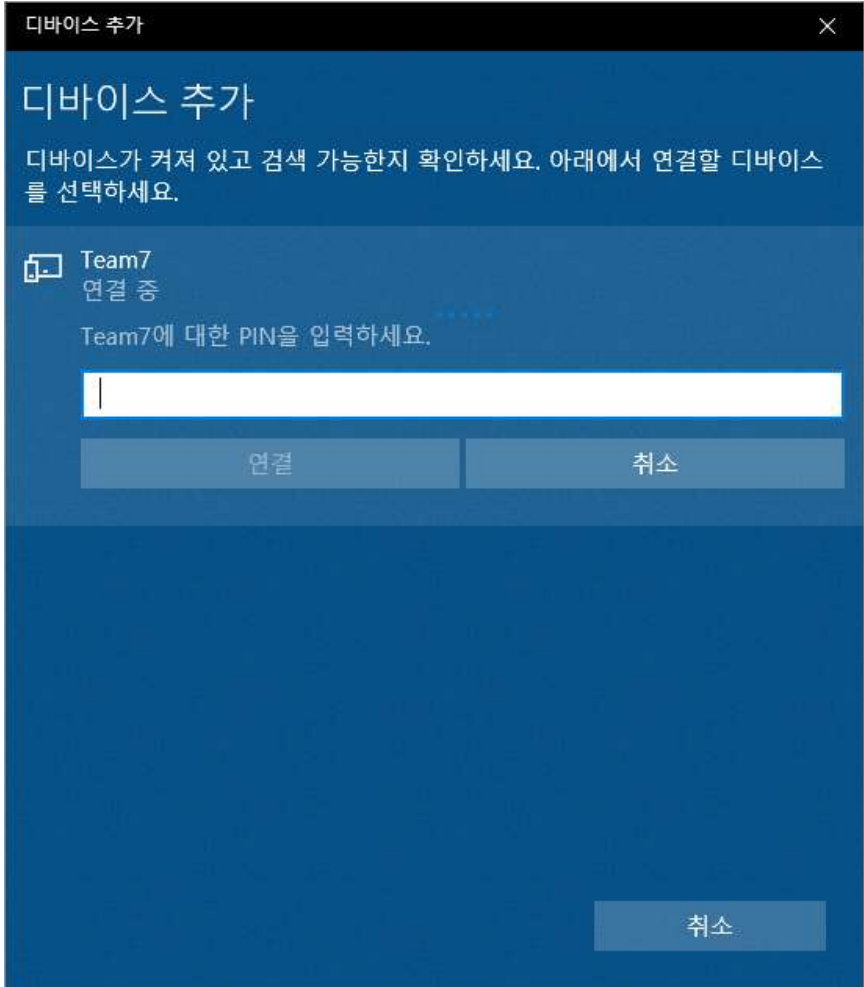
계획 2. 아두이노 제어를 맡은 팀은 자동차가 도로(검은선)을 따라갈 수 있도록 센서를 연동한 주행이 가능하도록 한다.



활동 사진

## 활동 사진



- 4차 회의	
회의 일시	2018년 12월 7일 금요일 14:00~18:00
회의 장소	공과대학 7호관 1층 무한 상상실
<p>활동 내용 및 문제 해결</p>	<p>스마트 로봇 자동차의 주행 준비</p> <ul style="list-style-type: none"> <li>- 이전 회의에서 결정된 주제에 맞게 주행할 경로가 있는 보드를 만들기 위해 검정색 마스킹테이프를 구입하였다.</li> <li>- 스마트 로봇 자동차에 부착된 블루투스 모듈과 페어링을 성공하였다.</li> </ul> 



활동 내용  
및  
문제 해결

라인 트레이서의 센서로부터 인식된 결과를 통해 주행을 위한 알고리즘 구현하였다.

- 바닥에 부착된 검정색 마스킹테이프를 인식하기 위해 라인 트레이서를 스마트 로봇 자동차에 조립하였다.
- 아두이노로 센서 인식 결과에 따라 알맞게 동작하도록 코드를 작성한 후 컴파일 및 업로드하였다. 동작 알고리즘은 아래에 다시 작성하였음
- 마스킹 테이프를 따라 직진하도록 주행 테스트를 해보았으나, 직진을 제대로 하지 못하였다.
- 문제점은 라인 트레이서에 있는 센서가 상당히 예민하여 미세한 조정이 필요하였다. 이에 대한 해결은 다음 회의에서 해결하기로 하였다.

활동 내용  
및  
문제 해결

주행하기 위한 동작 알고리즘

- 라인 트레이서의 센서는 총 3개이며, 왼쪽 센서와 중앙 센서 그리고 오른쪽 센서로 구분하였다.

- 각 센서들의 인식결과에 따라 경우의 수를 분석하고 각 케이스마다 특정 동작을 지정하였다. 여기서 인식됨은 점등되었음을 의미한다.

① 중앙 센서만 인식한 경우

- 이 경우에는 중앙의 검은색 선만 있다고 판단하여 직진하도록 수행하였다.

② 오른쪽 센서만 인식하거나, 중앙 센서와 오른쪽 센서가 인식한 경우

- 이 경우에는 검은색 선이 로봇 자동차의 우측에 있다는 것으로 판단하여, 차체를 우회전하여 검은색 선이 차체의 중앙에 위치할 수 있도록 하였다.

- 오른쪽 앞바퀴만 뒤쪽으로 회전하고 나머지는 앞으로 회전한다

활동 내용  
및  
문제 해결

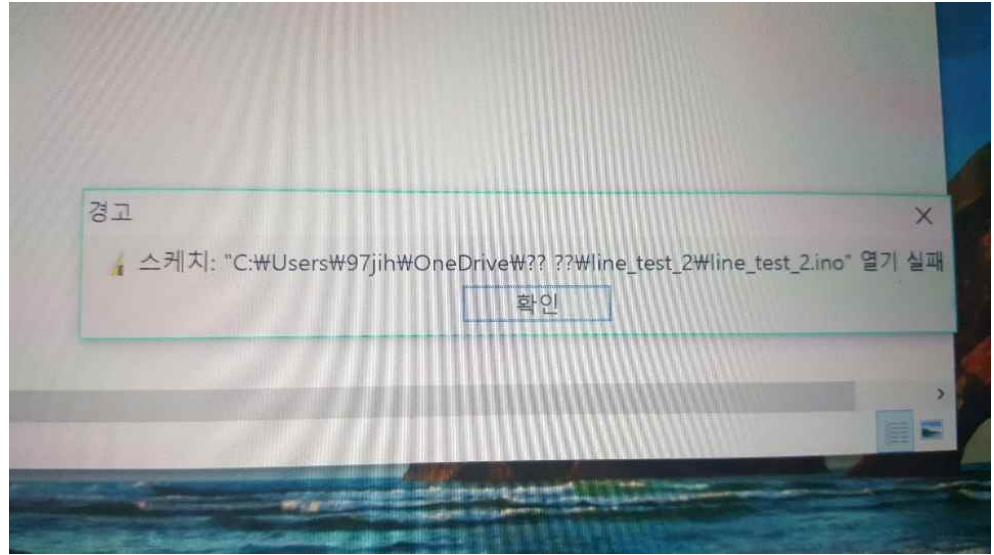
③ 왼쪽 센서만 인식하거나, 중앙 센서와 왼쪽 센서가 인식한 경우

- 이 경우에는 검은색 선이 로봇 자동차의 좌측에 있다는 것으로 판단하여, 차체를 좌회전하여 검은색 선이 차체의 중앙에 위치할 수 있도록 하였다.
- 왼쪽 앞바퀴만 뒤쪽으로 회전하고 나머지 바퀴는 앞쪽으로 회전한다.

④ 세 센서 모두 인식하거나, 모두 인식하지 못한 경우

- 이 경우에는 자동차가 교차로에 도달했거나 검은색 선(도로) 위에 있지 않는 것으로 판단하였다.
- 모든 바퀴를 회전하지 않는다.

활동 내용  
및  
문제 해결

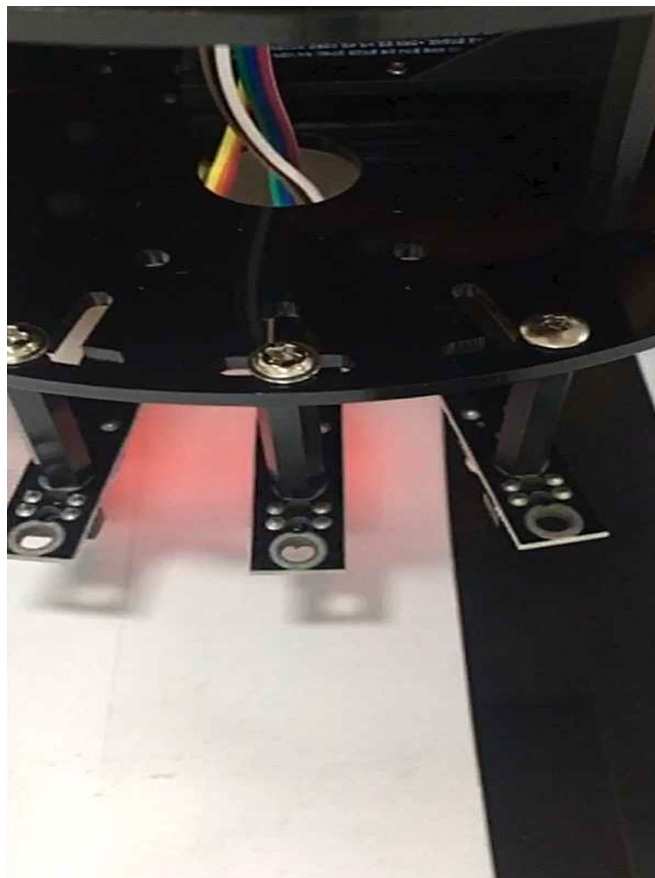


- 개발 도중에 이전에 주행이 가능하도록 작업한 내용을 확인하는 중 아두이노의 코드가 덮어써지는 경우가 발생하여 혼란을 빚었었다.
- 문제 해결
  - 아두이노는 컴파일 된 바이너리 파일을 키트에 업로드하는 방식을 취하고 있다.
  - 새로 업로드된 코드가 기존의 파일을 대체하는 방식이므로 항상 코드의 백업이 필요하다.
  - 코드를 버전별로 저장하여 문제를 찾거나 다른 기능을 잠시 개발하는 등 개발자가 바뀔 수 있는 상황에서 혼란을 빚지 않도록 개인이 항상 코드를 저장하기로 하여 해결하였다.

활동 사진



활동 사진



라인 트레이서가 절연 테이프를 인식하는 모습

## - 5차 회의

회의 일시

2018년 12월 16일 금요일 20:00~24:00

회의 장소

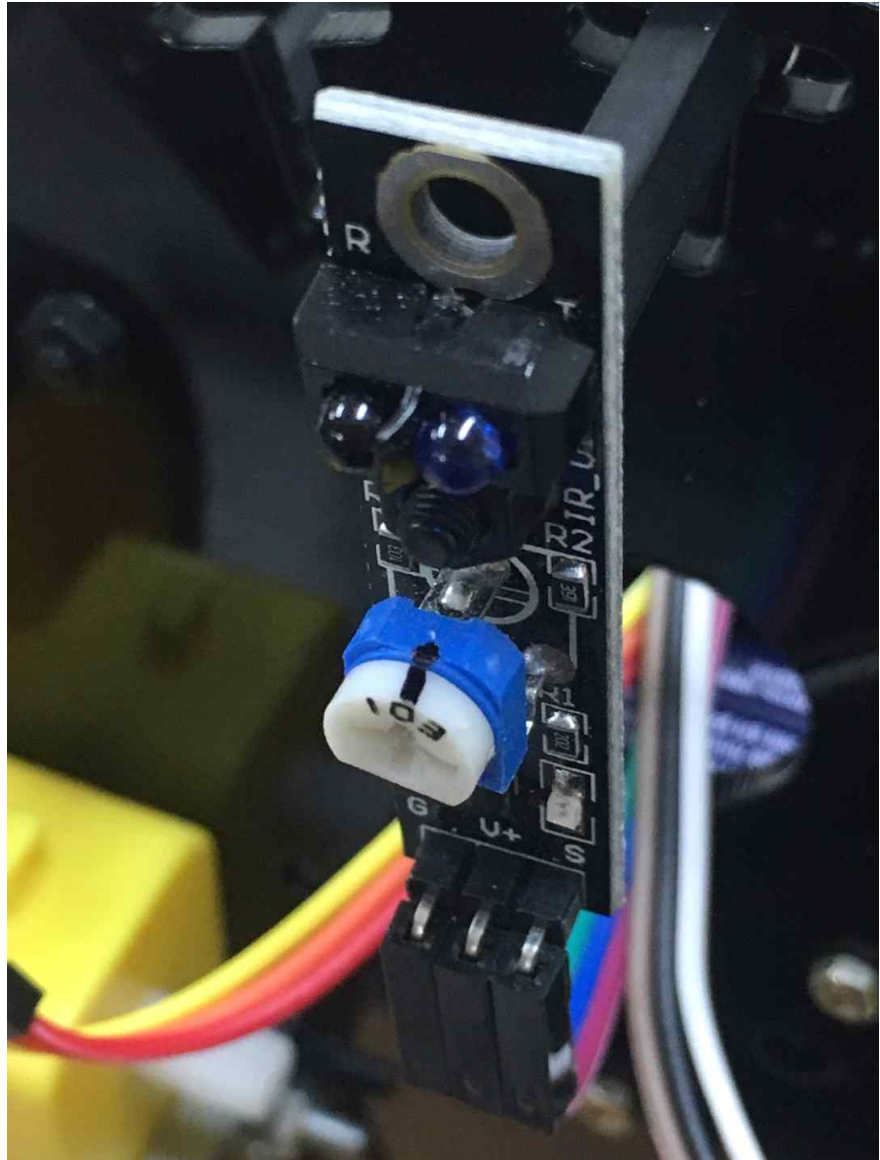
공과대학 7호관 533

활동 내용  
및  
문제 해결

로봇 자동차가 제대로 직진하지 못하는 원인을 파악하였다.

- 라인 트레이서의 센서가 마스킹 테이프(검은색 선) 위에서는 점등이 되어야하고 종이(흰색 배경) 위에서는 점등이 되지 않아야 하는데, 이것이 제대로 동작하지 않았다.
- 센서의 감도가 문제인 것으로 파악하고 미세 조정하여 해결하였다.
- 하드웨어의 특성상 흔들림이나 충돌 등으로 감도가 자주 바뀔 것을 우려하여 다음과 같이 해결하였다.

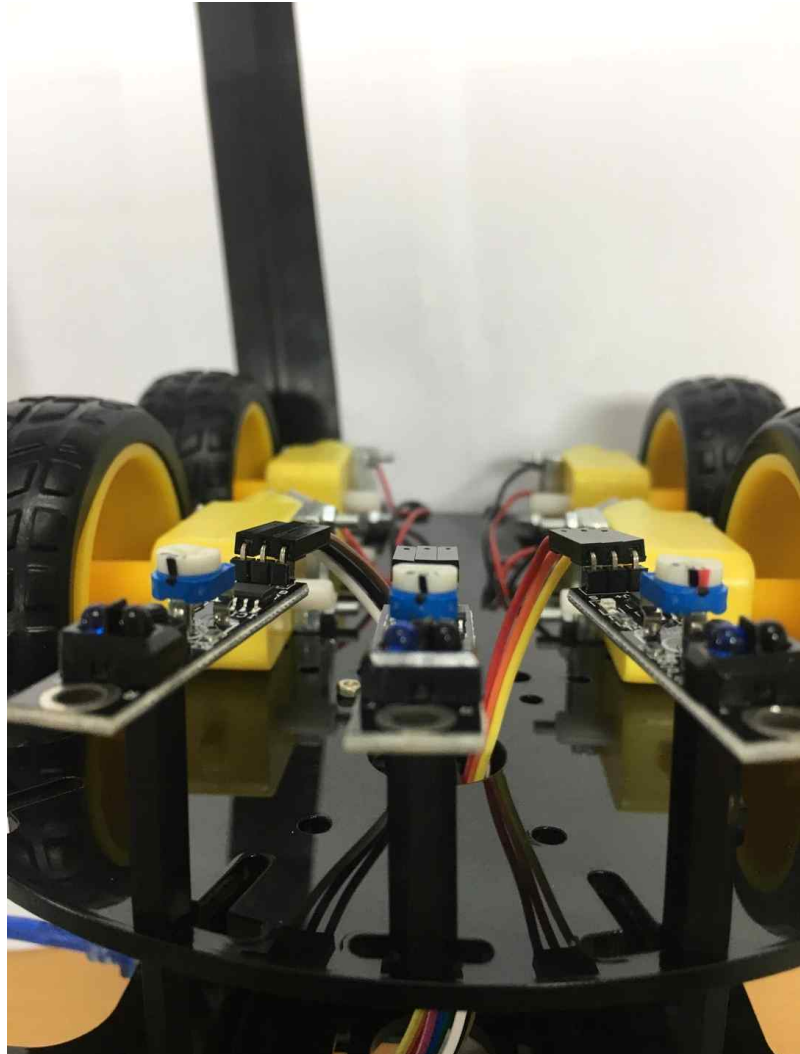
## 활동 내용 및 문제 해결



조립이 완성된 차체에 고정된 센서와 지면의 거리에 맞추어 적외선을 감지하는 트래킹 센서의 감도를 조절할 필요가 있었다. 여러 차례의 실험으로 지면의 흰색 면과 검은색 면이 잘 구분되는 거리를 찾았고, 사고를 미연에 방지하기 위해 검은선으로 표시를 해두었다.



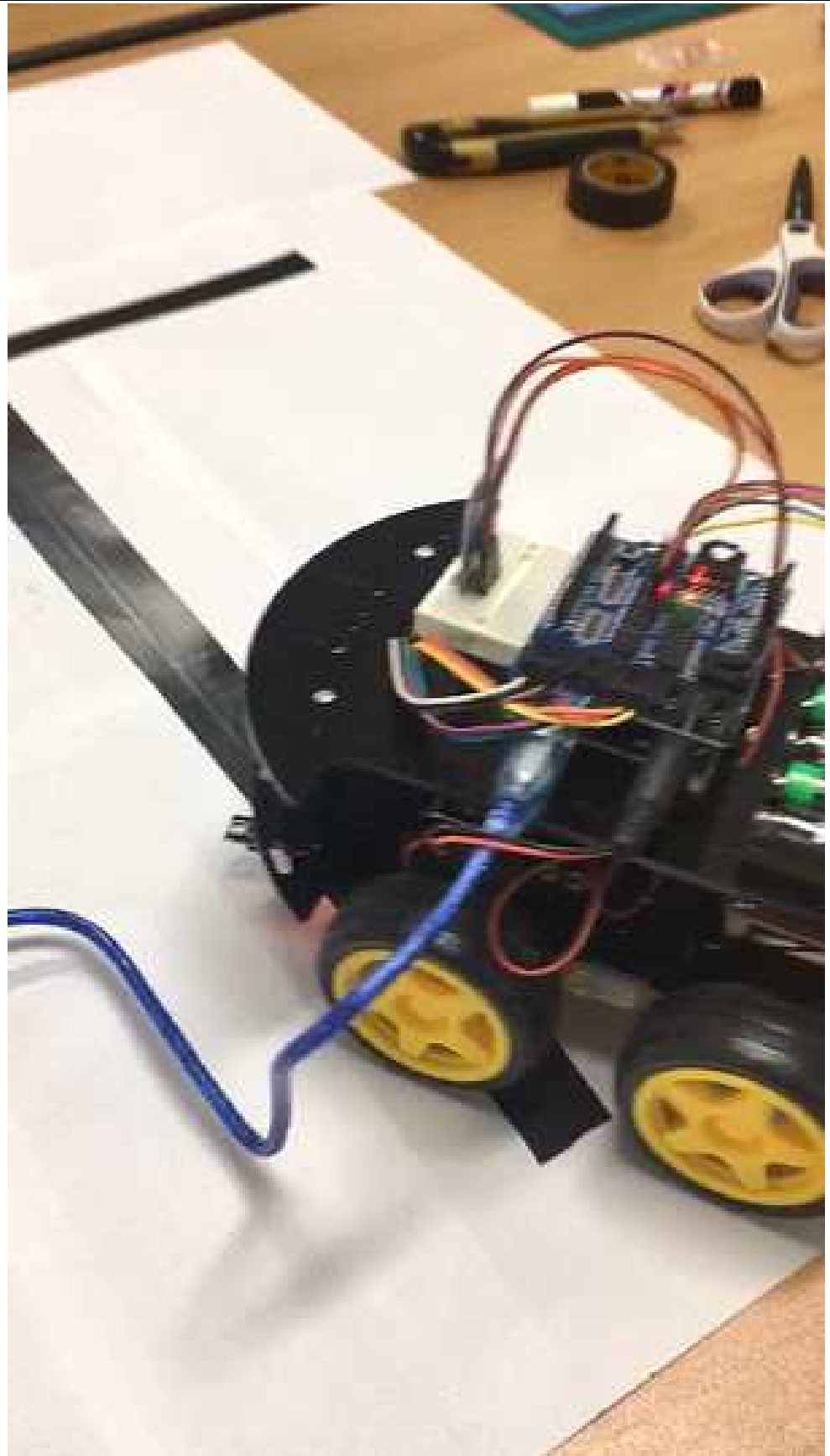
활동 내용  
및  
문제 해결



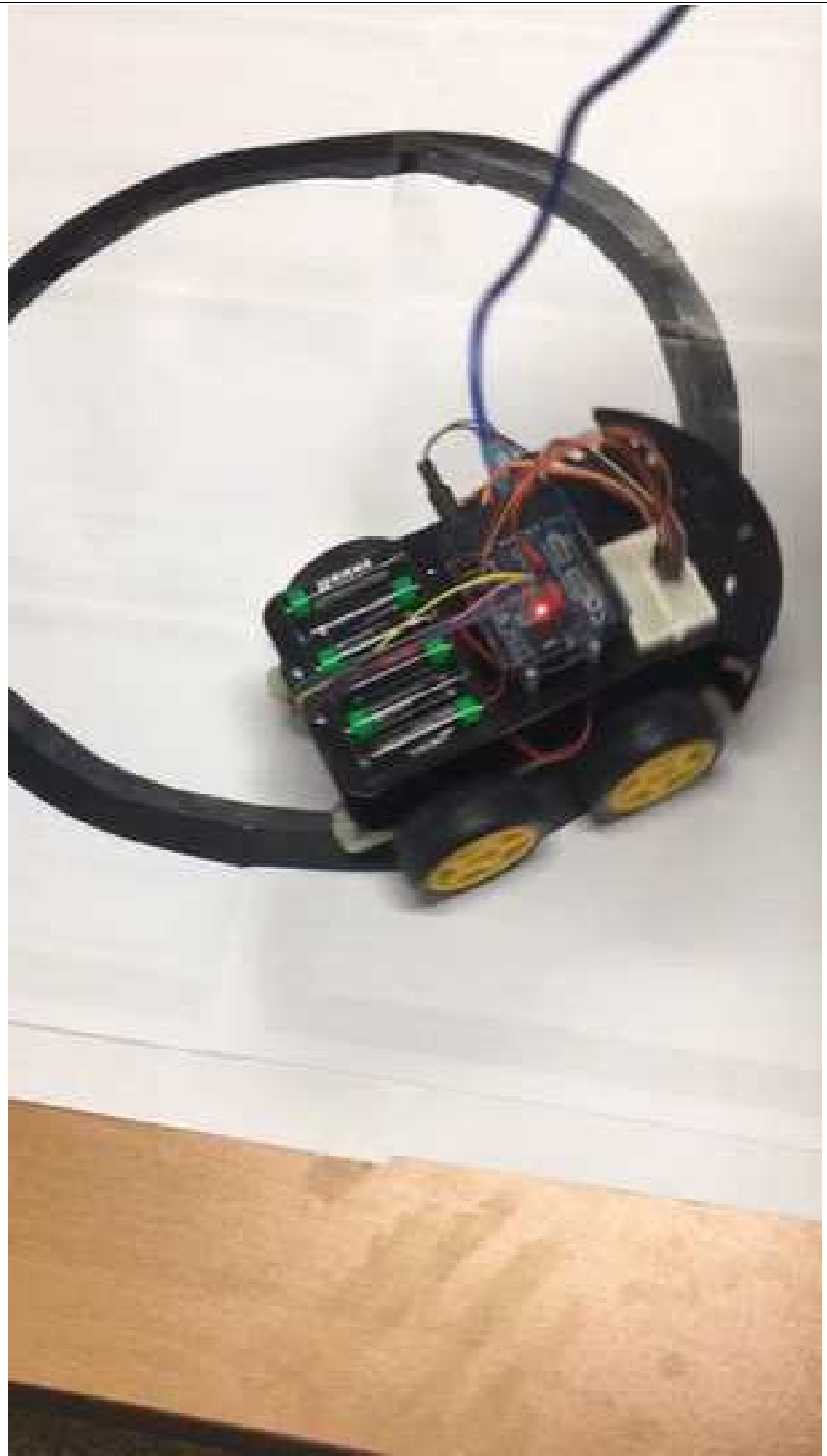
위 사진과 같이, 트래킹 센서들을 적당한 간격을 두도록 배치한 후 고정하였다.

- 라인 트레이서 감도 테스트 및 시험 주행  
좌측 센서와 중앙 센서, 우측 센서의 적외선 탐지 결과를 토대로, 각 센서들의 결과 조합마다 각 바퀴를 알맞게 조정하여 좌회전, 우회전을 통해 자동차가 검은선을 따라 이동할 수 있도록 방향을 조정하는 코드를 작성하였다.

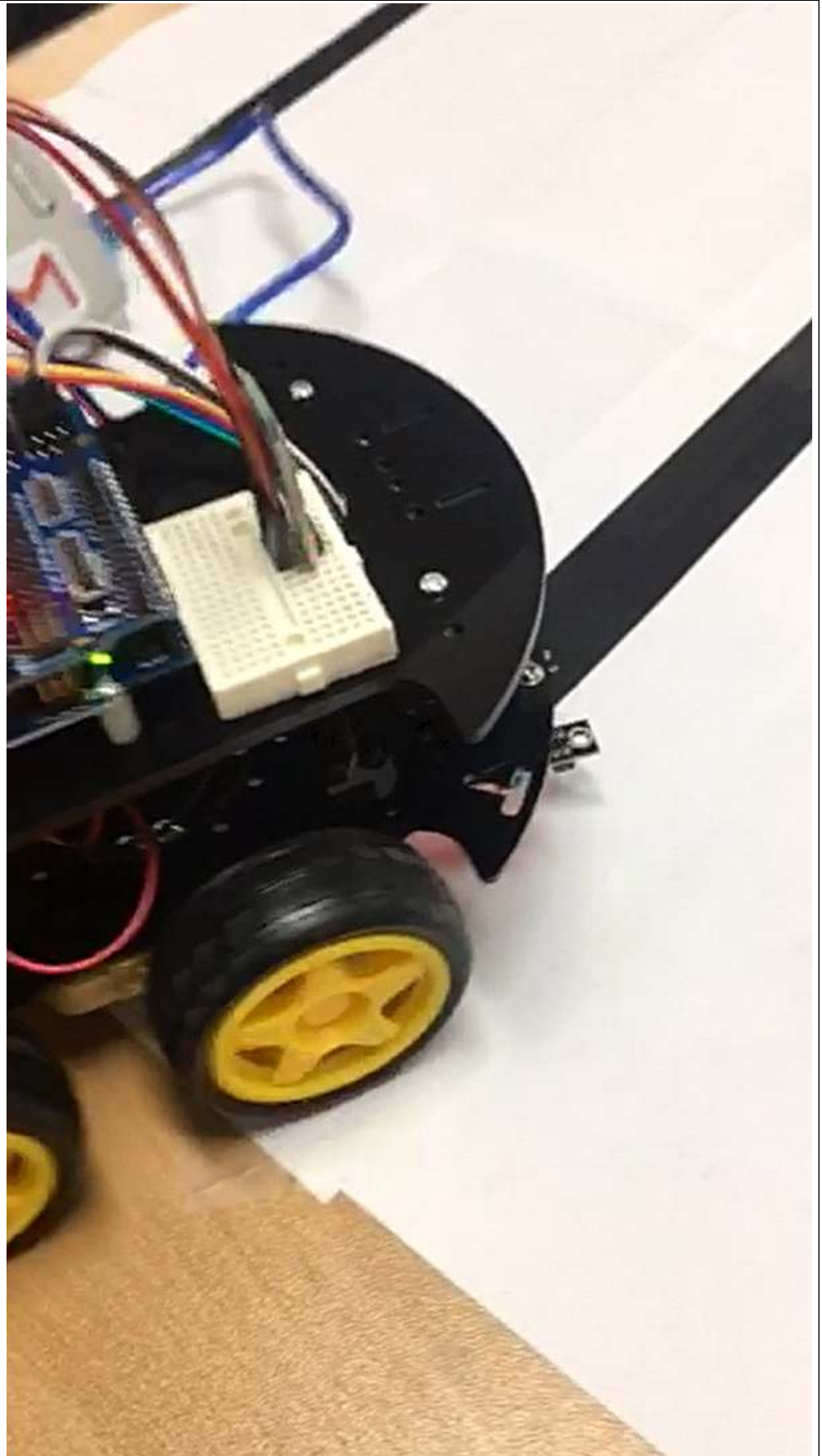
활동 사진



활동 사진



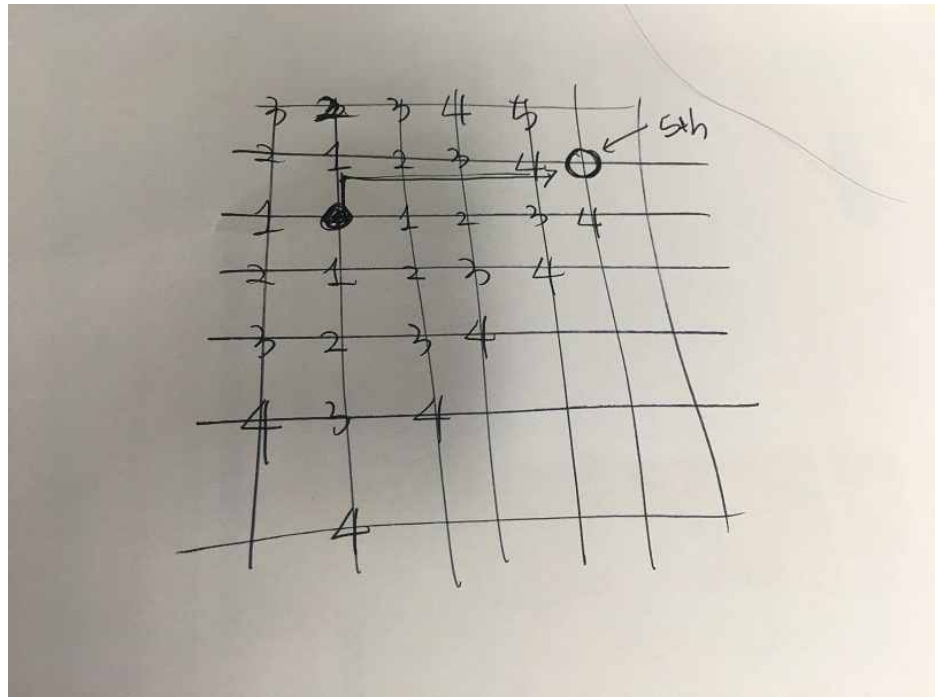
활동 사진



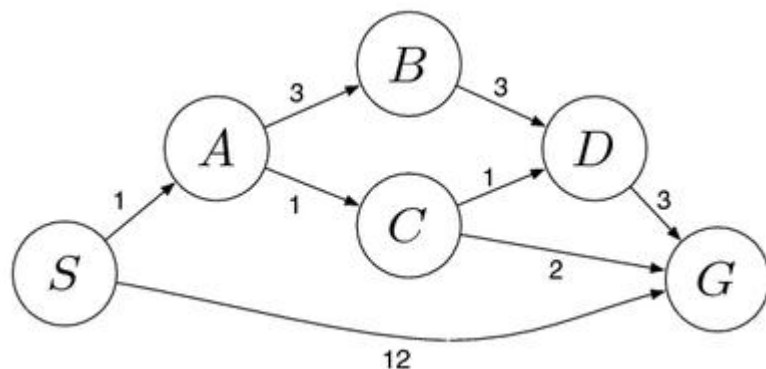
- 6차 회의	
회의 일시	2018년 12월 20일 목요일 19:00~21:00
회의 장소	공과대학 7호관 1층 무한상상실
활동 내용 및 문제 해결	<p>- 서버에서 아두이노로 전송할 json 형태를 아래와 같이 결정하였다.</p> <pre>{   "commands" : [     {"com": "front", "var" : 1},     {"com": "left", "var" : 1},     ...   ] }</pre> <p>- 탐색 알고리즘을 결정하였다. 너비 우선 탐색(BFS; Breadth First Search)을 사용하기로 하였다.</p>

활동 내용  
및  
문제 해결

- 프로젝트에 맞게 2차원 그리드에서 너비우선탐색(BFS; Breadth First Search)를 어떻게 진행해야 할지를 정하였다.



- 간선의 길이를 1로 정하고, 장애물이 있는 경우에는 탐색하지 않도록 하였다.
- 그 외 필요한 너비 우선 탐색의 구현에 필요한 지식을 개인적으로 공부하였다.



## ADOxx 설치 시 문제 발생

The screenshot shows the ADOxx installation window with the following content:

**ADOxx 설치 시 문제 발생**

이 파일은 PDF/A 표준을 준수하며 수정할 수 없도록 읽기 전용입니다.

**ADOxx 설치 구성 규칙**

설치 프로세스를 중단할지 여부를 확인하는 규칙을 실행하고 있습니다. 자세한 내용을 보려면 [도움말]을 클릭하십시오.

적용이 완료되었습니다. 성공: 2, 실패: 2, 경고: 0, 건너뛴: 0.

**자세한 정보 보기(V)**

규칙	상태
FAT32 파일 시스템	성공
기본 클러스터링 또는 클러스터 준비 인스턴스	성공
설치 언어 간 호환성	실패
동일한 아키텍처 설치	실패

**추가 정보**

PDF 작성

PDF 편집

주석

파일 결합

채우기 및 서명

추가 도구

Document Cloud에서 문서 저장 및 공유

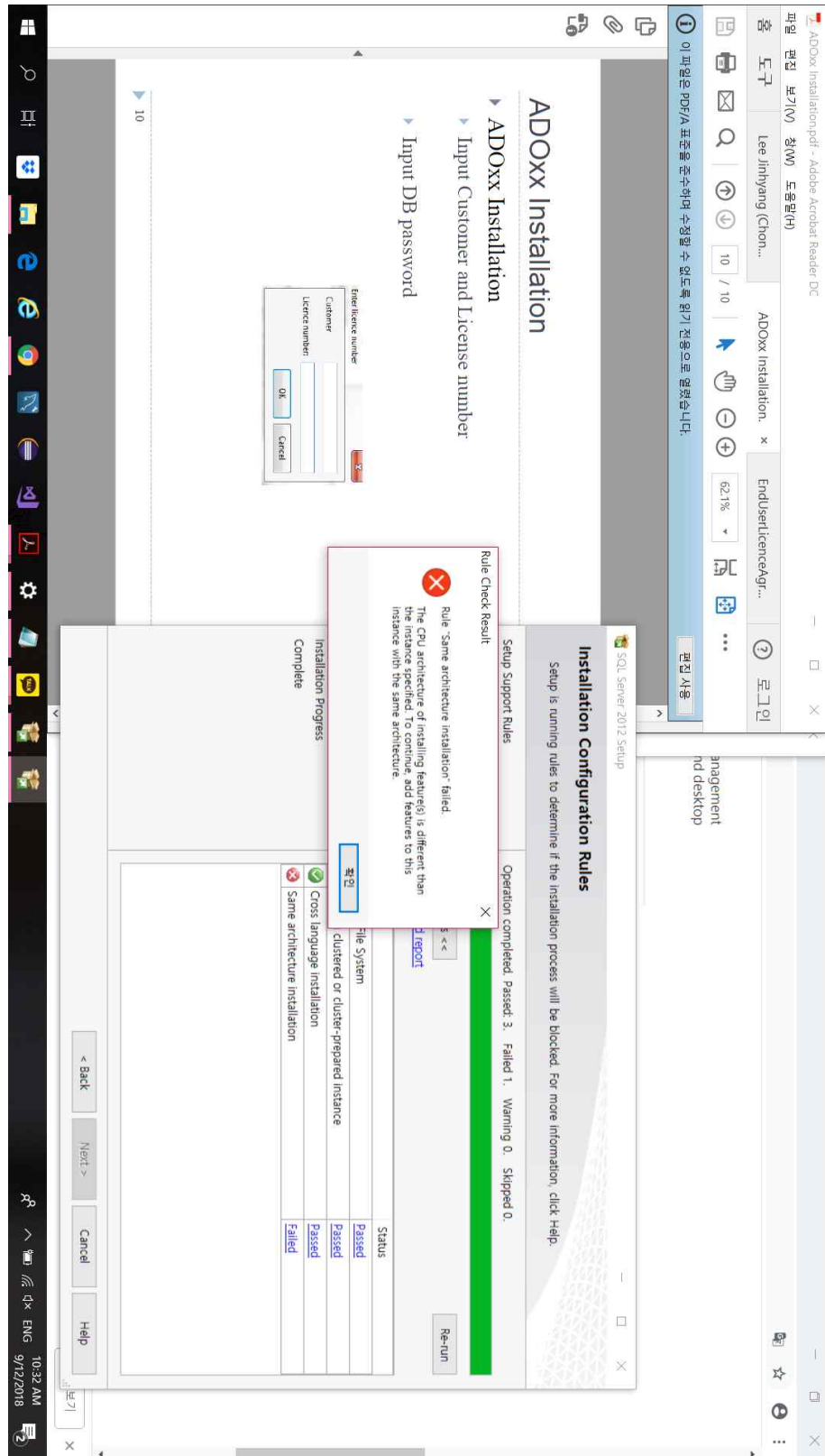
추가 정보

10:26 AM 9/12/2018

활동 내용  
및  
문제 해결



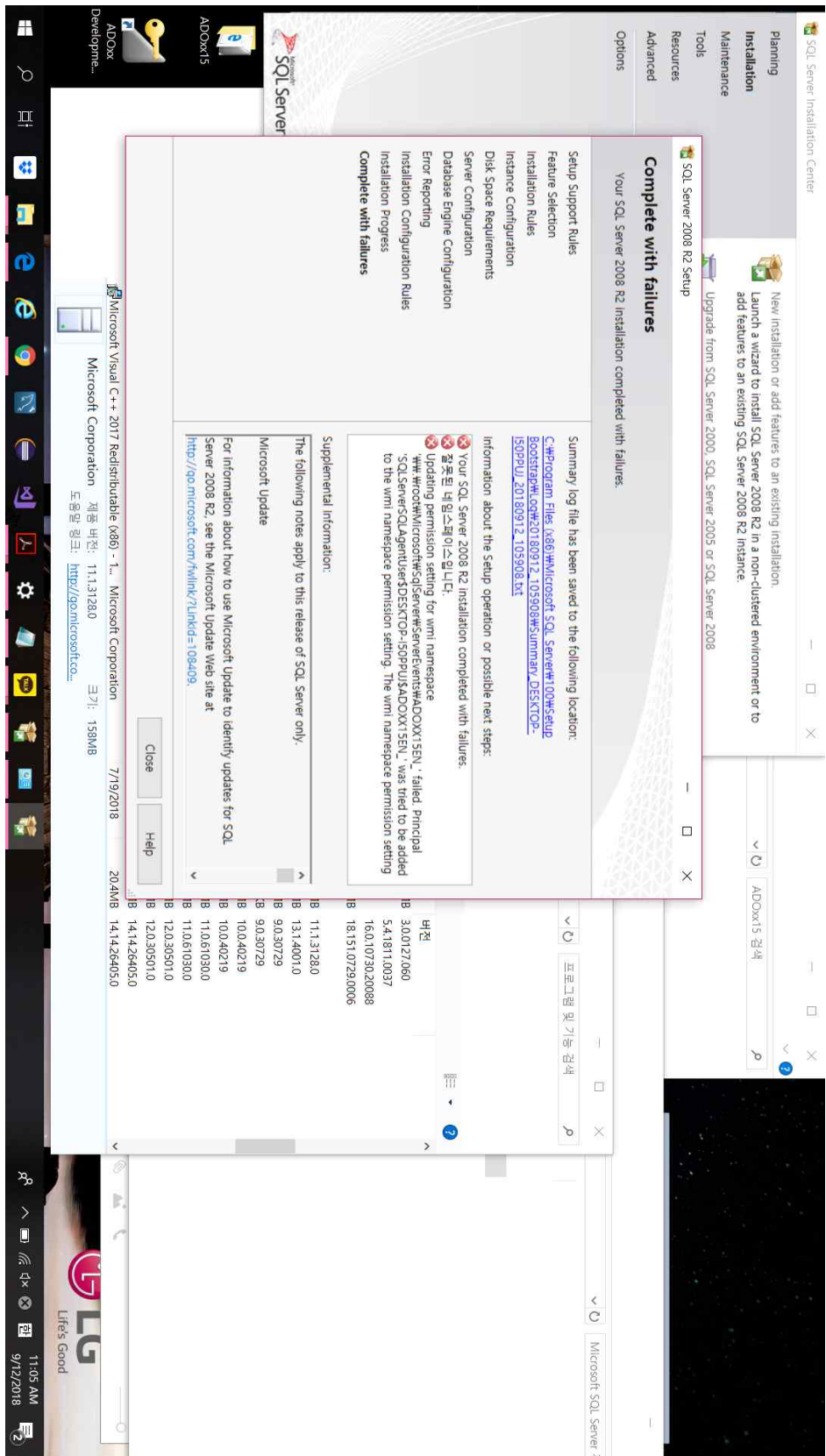
## ADOxx 설치 시 문제 발생



활동 내용  
및  
문제 해결



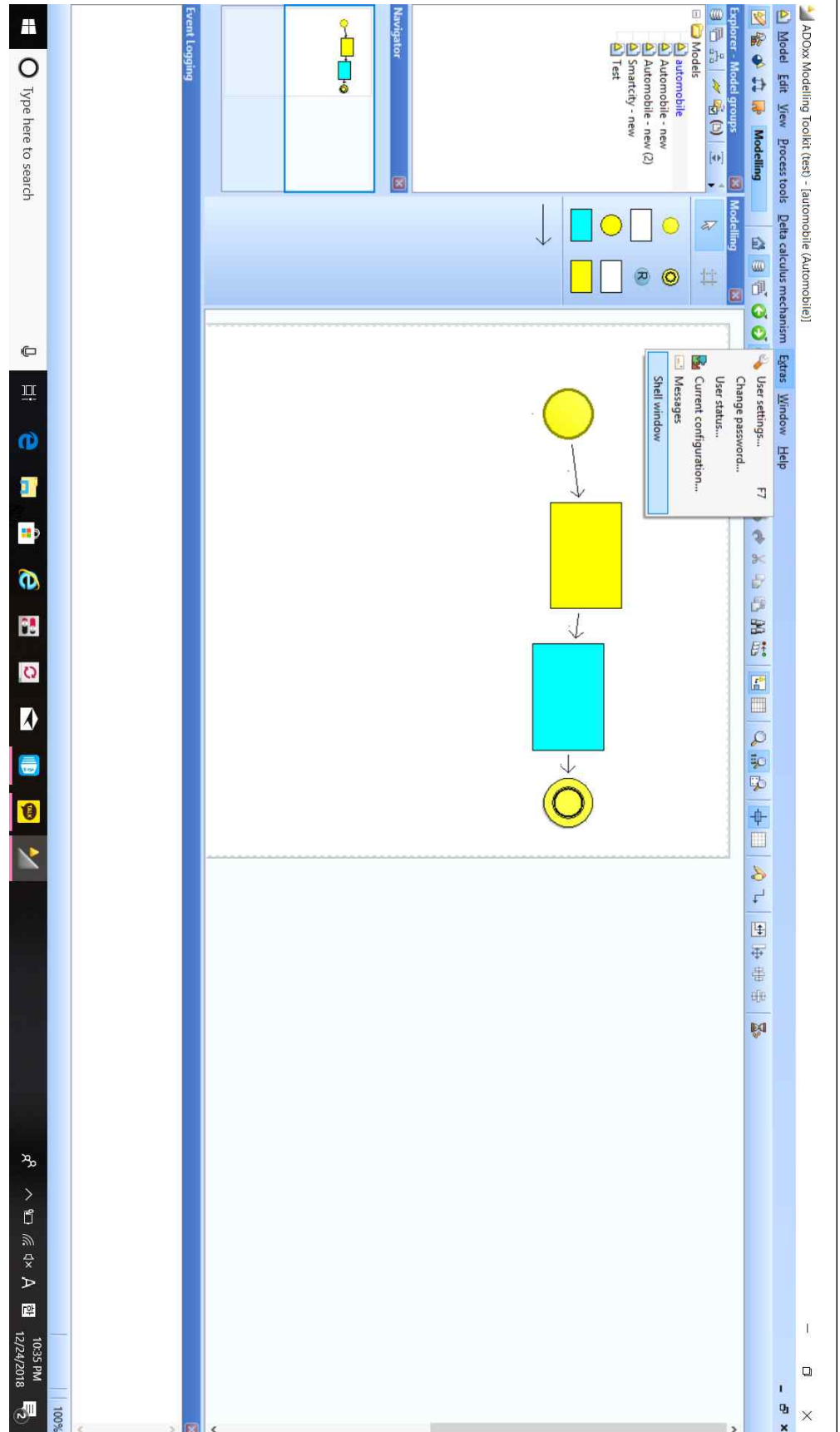
## ADOxx 설치 시 문제 발생



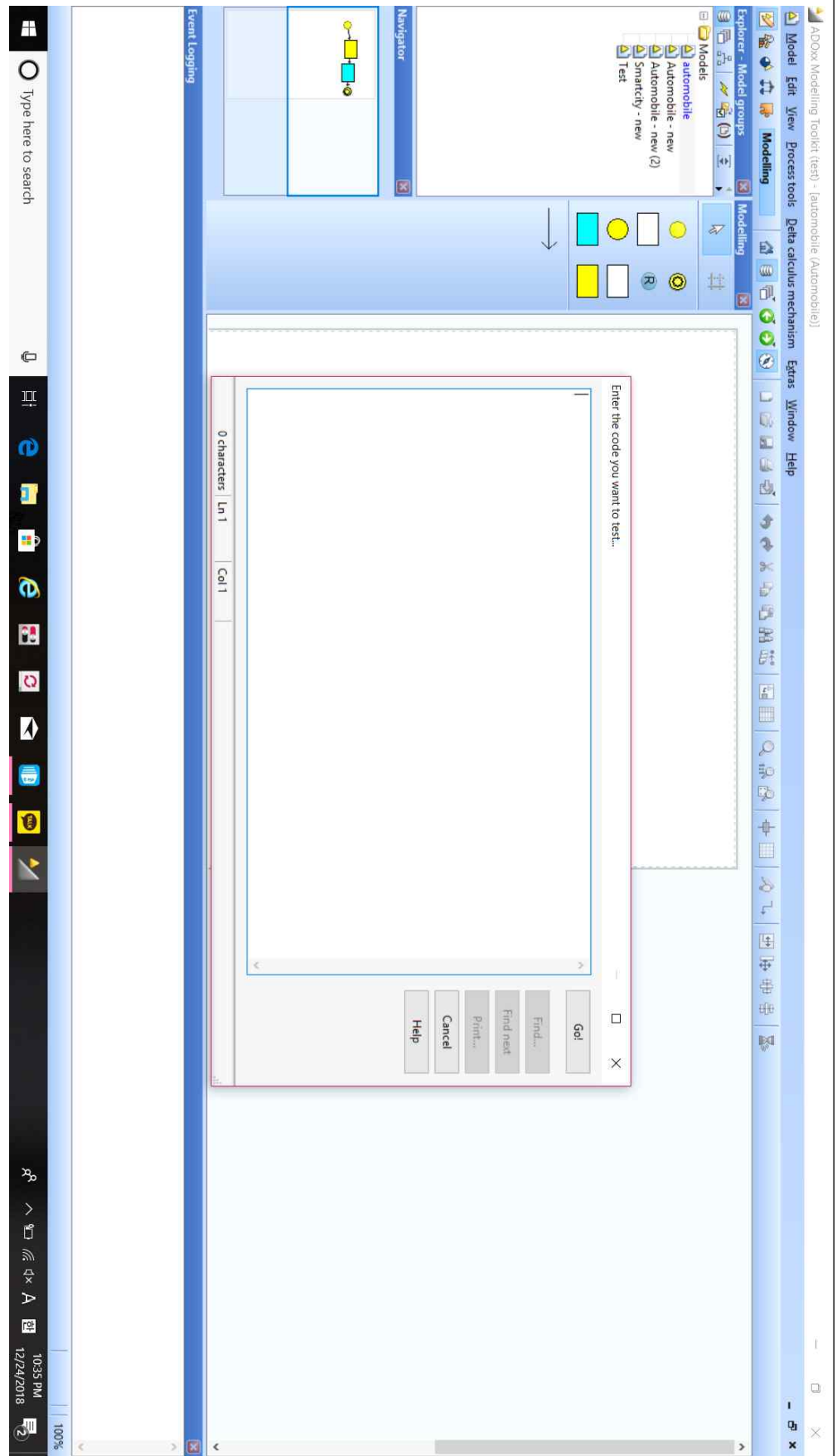
활동 내용  
및  
문제 해결

활동 내용  
및  
문제 해결

- ADOxx 설치 시 발생 한 문제 해결
  - 컴퓨터의 사용자 이름이 한글로 되어 있는 경우에, ADOxx가 설치가 정상적으로 되지 않는 일이 발생했다.
  - 컴퓨터 사용자 명을 변경하는 것이 Windows 10에서 쉽지 않았고 기존의 프로그램들과의 무결성 등을 고려하여 다른 방안을 좀 더 모색하였다.
  - 컴퓨터 사용자를 하나 더 추가하여 영어 이름으로 생성한 후에, 언어 설정을 변경하였다.
  - 그리고 ADOxx의 설치를 다시 진행하면 정상적으로 잘 설치되어 사용할 수 있었다.



활동 내용  
및  
문제 해결



활동 내용  
및  
문제 해결

## - 7차 회의

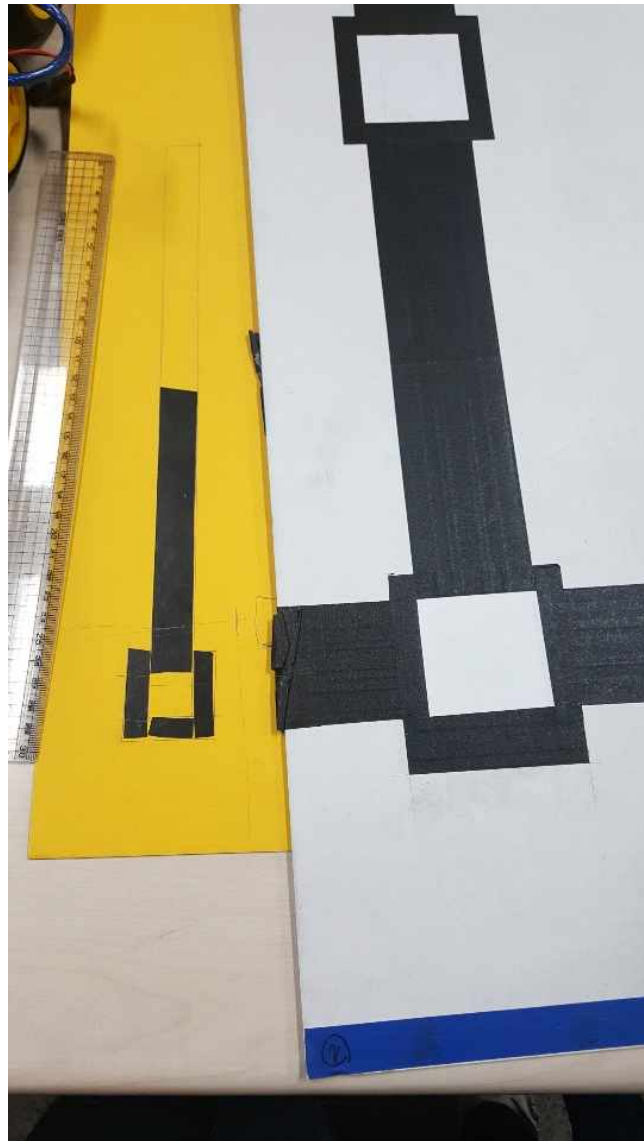
회의 일시

2018년 12월 22일 토요일 19:00~21:00

회의 장소

공과대학 7호관 1층 무한상상실

활동 내용  
및  
문제 해결



- 이전 회의에서 정한 7x7 크기의 도로 주행판을 제작하기 위하여 하드보드지를 구입하였다.

활동 내용  
및  
문제 해결

- 좌회전 혹은 우회전 시 모터가 20도 회전하도록 설정하였다.
- 문제 발생
  - 직접 제작한 보드 위에 수작업으로 테이프로 도로를 표시하다보니 비율이 전부 같지 않았다.
  - 그래서 특정 각도로 회전하도록 설정하면 경로를 이탈하는 경우가 잦았다.
  - 회전 시 각도를 측정하거나 회전 시간을 줄 수가 없는 문제점이 발생하였다.
- 문제 해결
  - 라인 트레이서의 센서가 다시 점등될 때 까지, 즉 검정색 전열 테이프로 표시된 도로를 읽을 때까지 회전하도록 설계 방법을 변경하였다.

활동 내용  
및  
문제 해결

- 디자인룸에 있던 기존 보드와 동일하게 노드를 6x6 크기로 배치하려고 했으나, 디자인룸의 보드의 노드간 간격은 약 35~40cm이고 우리 프로젝트의 하드보드지의 크기가 이를 담을 수 없었다.
- 노드의 가로와 세로의 수를 최대한 맞추기 위해 검은선의 폭을 기존 디자인룸의 선보다 2배 얇게 줄였다. 그리고 그 과정에서 보드의 크기를 6x5로 줄이고 장치 테스트를 진행한 후 인식이 잘 되도록 라인 트레이서를 다시 조정하였다.
- 블루투스 모듈의 문제 발생
  - 블루투스 모듈의 초반 고장 원인과 달리, 제대로 작동하지 않는 문제가 생겼다.
- 블루투스 모듈의 문제 해결
  - 배터리가 부족하여 발생한 문제로 파악됨
  - 배터리를 교체하였더니 해결되었다.

활동 내용  
및  
문제 해결

서버에서 아두이노로 전송할 송신 데이터 json의 형태를 아래와 같이 변경하였다.

```

1 {
2   "commands": [
3     {
4       "com": "front",
5       "time": 100
6     },
7     {
8       "com": "left",
9       "time": 100
10    },
11    {...}
12  ]
13 }
```

- 키 이름 var를 time으로 변경하여 동작시간을 표시하고 단위는 ms로 약속하였다. 변수명을 알맞게 바꿈으로서 의사소통이 간결하고 명확해졌다.



ADOxx 코드

```
CC "CoreUI" MODEL_SELECT_BOX #읽어들일
모델들을 선택
oktext:"Ok" boxtext:"Please select the execution
model:"
multi-sel title:"Execution model select box" #
without 2nd "_", hehe...
modeltype:"Automobile"
extra:{
#      CHECKBOX "Automatically arranged
processes" checked:1
result-var:bAutoArrangProcesses
#      CHECKBOX "Use default zoom factor
(default: x=2,y=1)" checked:1
result-var:bZoomFactorOption
}

IF (endbutton = "ok") #ok 버튼을 누르면 모델을
로드함
{
    FOR modelid in:(modelids)
    {
        CC "Core" LOAD_MODEL
modelid:(VAL modelid)
    }
}
```

## ADOxx 코드

```
#json 데이터 포맷
#      state: state의 개수
#      state_type
#      그리고 타입에 맞는 attr을 state_attr 나열...

SET ip:("http://192.168.1.21:10000") #서버의 ip주소와 포트를 넣은
변수

FOR modelid in:(modelids) #모든 모델을 읽어들이 때까지 반복함
{

    SETL nModelID:(VAL modelid) #Model id를 nModelID 변수에
    저장
    CC "Core" GET_MODEL_BASENAME modelid:(nModelID)
    #모델의 이름을 불러옴
    SET state:(1)
    CC "Core" GET_ALL_OBJS_OF_CLASSNAME
    modelid:(nModelID) classname:("Start") # 클래스 이름에 해당하는
    objid를 불러옴
    SET nObjId:(VAL objids)
    CC "Core" GET_ATTR_VAL objid:(nObjId) attrname:("IP")
    #start node의 "IP"란 속성에 동작시킬 아두이노에 대한 정보가
    저장되어 있음
    #"IP"속성의 값을 불러들여 중계용 서버에 전송하기 위해 함수를
    호출한 것임
    SETL basename:(val) #모델의 이름을 저장

    SET t:(0)
    SETL json:("@{ ₩"commands₩" : [") #json 형식으로 만들기
    위해 변수 생성, 앞의 @는 아두이노 상에서 파싱처리를 위해 임의로
    넣은 토큰임
```

## ADOxx 코드

```

WHILE (1) #모델 안에 있는 모든 객체의 정보를 읽어들이고 후
종료됨
{
    SETL sState:(STR state) #상태 번호 저장
    # CC "AdoScript" INFOBOX ()
    CC "Core" GET_ATTR_VAL objid:(nObjId)
    attrname:("Next") #다음 객체 번호가 저장된 속성 값 불러오기
    SET nObjId:(VAL val) # 다음 객체 번호 저장
    CC "Core" GET_CLASS_ID objid:(nObjId) #객체 번호의
    classid 불러오기
    CC "Core" GET_CLASS_NAME classid:(classid)
    #classid를 기반으로 class이름 불러오기
    SETL sClassName:(classname) #class 이름 저장, class
    이름에 따라 아래의 분기가 실행됨
    IF (sClassName = "End") #종료 클래스일 경우
    {
        SET state:(state-1)
        SETL sState:(STR state)
        # CC "AdoScript" INFOBOX ("The end of Class")
        BREAK
    }
    ELSIF (sClassName = "Automobile_Motor") #자동차
    모터일 경우
    {
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Quick_Start") #속성값을 불러옴
        SETL qs:(val) #값 저장
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Execution_Time(ms)") #속성값을 불러옴
        SETL time:(STR val) #값 저장
        SETL type:("car") #유형 저장
        # # #
        #
        # SETL json:(json+ "W"+sState + "_" + "tW"+":W" +
        type + "W",
        # + "W"+sState + "_" + "qsW"+":W" + qs +
        "W",") #json 형식으로 text 저장

        SETL json:(json+"{WcomW" :
        W"+qs+W",W"time:W" : "+time+"},")
    }
}
    
```

## ADOxx 코드

```

SETL json:(json+"{W"comW" : W"+qs+W",W"time:W"
: "+time+"},")
}
ELSIF (sClassName = "Motor") #모터일 경우
(기계팔(서보모터)에서 쓰임, 자동차 코드에선 아직 미구현된
기능임)
{
    CC "Core" GET_ATTR_VAL objid:(nObjId)
    attrname:("Motor_Number")
    SETL m:(val)
    CC "Core" GET_ATTR_VAL objid:(nObjId)
    attrname:("Degree")
    SETL degree:(STR val)
    SETL type:("arm")
    SETL json:(json+ "W"+sState + "_" + "tW"+":W"
+ type + "W",")
    # SETL json:(json+ "W"+sState + "_" +
"typeW"+":W" + type + "W",")
    # + "W"+sState + "_" + "mW"+":W" + m +
"W",")
    # + "W"+sState + "_" + "dW"+":W" + degree +
"W",")

    # SETL json:(json+ "W"+sState + "_" +
"typeW"+":W" + type + "W",")
    # + "W"+sState + "_" + "qsW"+":W" + qs +
"W",")
    # + "W"+sState + "_" + "timeW"+":W" + time
+ "W",")
}
SET state:(state+1)
IF (t=50)
{
    CC "AdoScript" INFOBOX ("t is "+STR t)
    BREAK
}
SET t:(t+1)
}
    
```

## ADOxx 코드

```

SETL tempmap:({"Content-Type": "text/plain"})
#http request를 보내기 위한 초기 설정
SETL json:(json+"}") #json 변수 저장
HTTP_SEND_REQUEST
(ip+"?id=adoxx&name="+basename)
str_method:("GET") map_reqheaders:(tempmap)
str_reqbody:(json) val_respcode:respstat
map_respheaders:respheaders str_respbody:respbody
#HTTP_SEND_REQUEST을 실행하여 중계용 서버로 보냄
SLEEP ms:(500) # 서버에서 데이터를 완전히 받을
때까지 대기
#CC "AdoScript" INFOBOX
(ip+"?id=adoxx&name="+basename)

CC "AdoScript" INFOBOX (json)
}
CC "AdoScript" INFOBOX ("The end of AdoScript")
# CC "Modeling" GET_ACT_MODEL #현재 활성화된
모델의 id를 불러옴
# SETL nModelID:(modelid)
# SET ecode:0

PROCEDURE SLEEP ms:integer
{
    SETL time:(800*ms)
    FOR i from:1 to:(time) {}
}
    
```

- 8차 회의	
회의 일시	2018년 12월 23일 일요일 13:00~18:00
회의 장소	전북대학교 구정문 카페
<p>활동 내용 및 문제 해결</p>	<ul style="list-style-type: none"> <li>- C언어로 NxM크기의 보드를 너비우선탐색(BFS)을 하는 psuedo 코드를 작성하여 기초적인 틀을 구성하였다. 이후 AdoScript로 코드를 옮겨 적기로 하였다.</li> <li>- 보드가 6x5 크기이므로 코드 상에서 2차원 배열의 형태로 다룰 수 있는 자료 구조가 필요하였지만, AdoScript에서 다차원 배열을 다루는 방법을 잘 이해하지 못하여서 온라인 커뮤니티의 글을 참고하였다. (<a href="https://www.adoxx.org/live/faq/-/message_boards/view_message/148080">https://www.adoxx.org/live/faq/-/message_boards/view_message/148080</a>)</li> </ul>

활동 내용  
및  
문제 해결

- 서버가 아두이노에게 다음과 같이 json 데이터를 전송된 것을 아두이노의 시리얼 모니터를 통해 확인하였다.

- 각 명령어들은 @를 토큰으로 구분되는 것을 확인하였다. 형태를 다음과 같다.

@{"Commands" :

["front","front","left","right","front"]} @{"Commands" :

["front","left","left","right","back"]} @...

- ADOxx에서 너비우선탐색(BFS) 프로시저는 다음과 같이 작성하였다.

- 매개변수로 정수형인 first\_x와 first\_y가 주어지면 선입선출(FIFO; First In First Out) 스케줄링을 사용하여, 가장 가까운 위치를 찾는 코드를 작성하였다.

- 배열인 변수 V는 목적지를 의미한다. 경로 설정을 할 때, 외부에서 값이 1로 변경된다.

## ADOxx 내용 및 코드

### - AdoScript는 아래와 같다.

```
PROCEDURE BFS integer: first_x,first_y val: int result: void{

    SET rear:0
    SET front = 0;

    SET V:(array(4,5))
    SET visited:(array(4,5))
    SET queue:(array(20))
    SET i:0
    FOR i from:0 to:3 {
        SET j:0
        FOR j from:0 to:4
        {
            SET V[i][j]:0
            SET visited[i][j]:1
        }
    }
    SET i:0
    FOR i from:0 to:19{
        SET queue[i]:-1
    }
    SET first_x: 0
    SET first_y: 0
    SET x: first_x
    SET y: first_y
    SET visited[x][y]: 0
    SET queue[rear++]: x
    SET queue[rear++]: y
    WHILE (front < rear) {
        SET v: queue[front++]
        SET w: queue[front++]
        SET i: v
        FOR i from:v to: 3 {
            SET j: w
            FOR j from:w to:4
            {
                IF (V[i][j]= 1 && visited[i][j]= 1) {
                    visited[i][j]: 0
                    queue[rear++]: i
                    queue[rear++]: j
                }
            }
        }
    }
}
```



## 활동 사진



- 9차 회의	
회의 일시	2018년 12월 24일 월요일 11:00~21:00
회의 장소	공과대학 7호관 6층 디자인룸
활동 내용 및 문제 해결	<p>- 7차 회의에서 작성한 ADOxx의 너비우선탐색(BFS) 코드의 문제점을 확인하였다.</p> <ul style="list-style-type: none"> <li>■ 문제점 1. 로봇 팔에 사용되고 로봇 자동차에서는 지원되지 않는 코드가 포함되어 있었다. 전혀 사용되지 않는 코드가 있어서 가독성을 떨어뜨리고 개발 생산성을 저해하고 있었다.</li> <li>■ 해결 1. 해당 코드를 삭제하여 해결하였다.</li> <li>■ 문제점 2. 가로 세로의 크기가 4*5인 상황에서 정상적으로 동작하지 않는 것으로 보인다.</li> <li>■ 해결 2. 배열의 크기를 적당히 큰 201*201으로 수정하였다.</li> </ul>

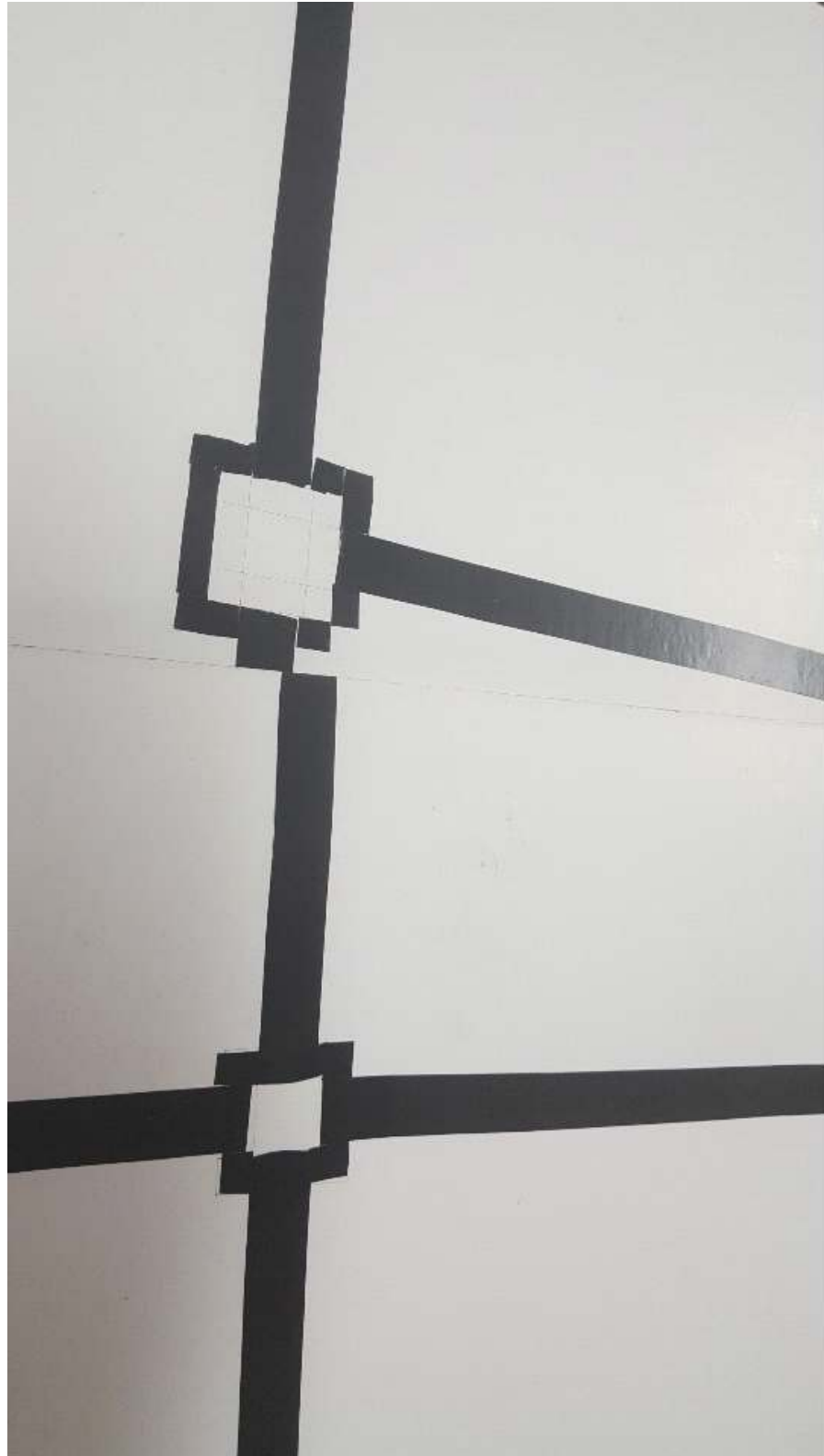
활동 내용  
및  
문제 해결

- 하드보드지 주행 도로판 위에 표시할 도로를 만들던 중, 절연 테이프가 부족하여 추가로 구매하였다.



활동 내용  
및  
문제 해결

## ■ 문제점 인식

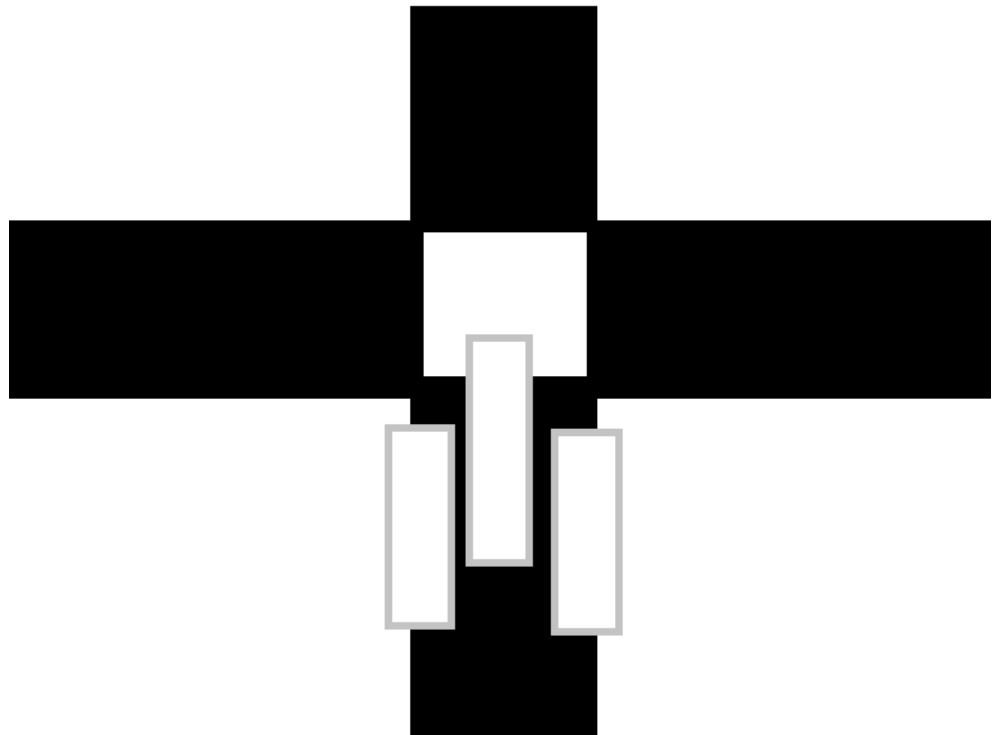


활동 내용  
및  
문제 해결

- 노드가 너무 작아서 라인 트레이서가 이를 잘 인식하지 못하고 있다고 판단하여, 위 사진의 아래 모습에서 위 모습과 같이 변경하였다.
- 노드의 흰색 박스의 크기를 더욱 크게 변경하였다.

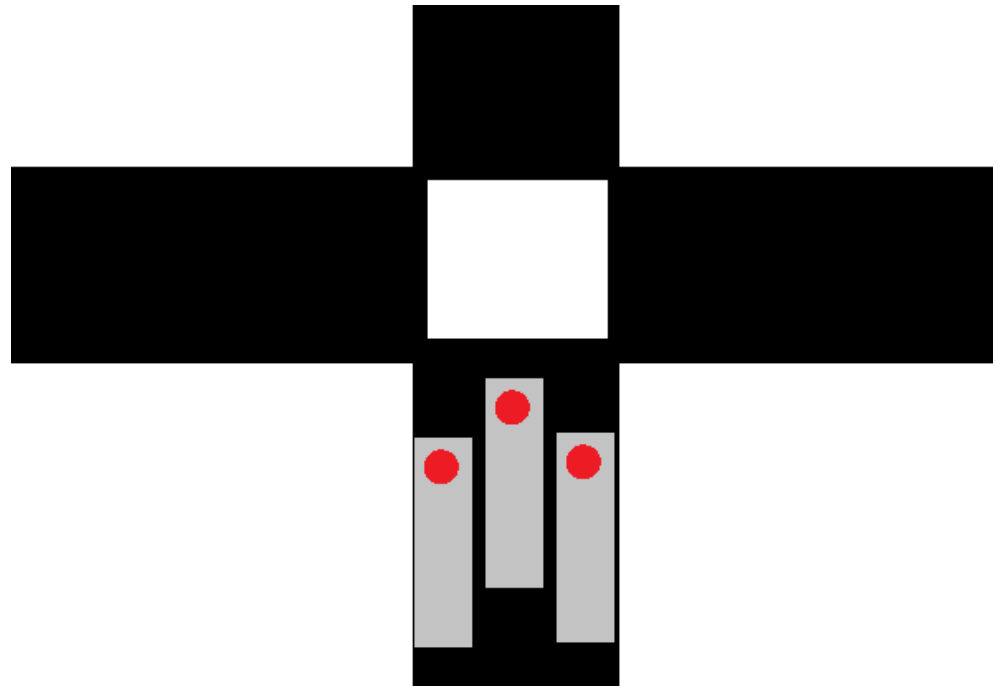
활동 내용  
및  
문제 해결

- 라인 트레이서를 위 사진과 같이 변경하였다.
- 기존에는 라인 트레이서의 센서가 일렬로 있되 각도만 다르게 부착되어 있었다면, 지금은 山자와 같은 모양으로 가운데 센서만 앞쪽으로 돌출되어있는 형태를 띤다.

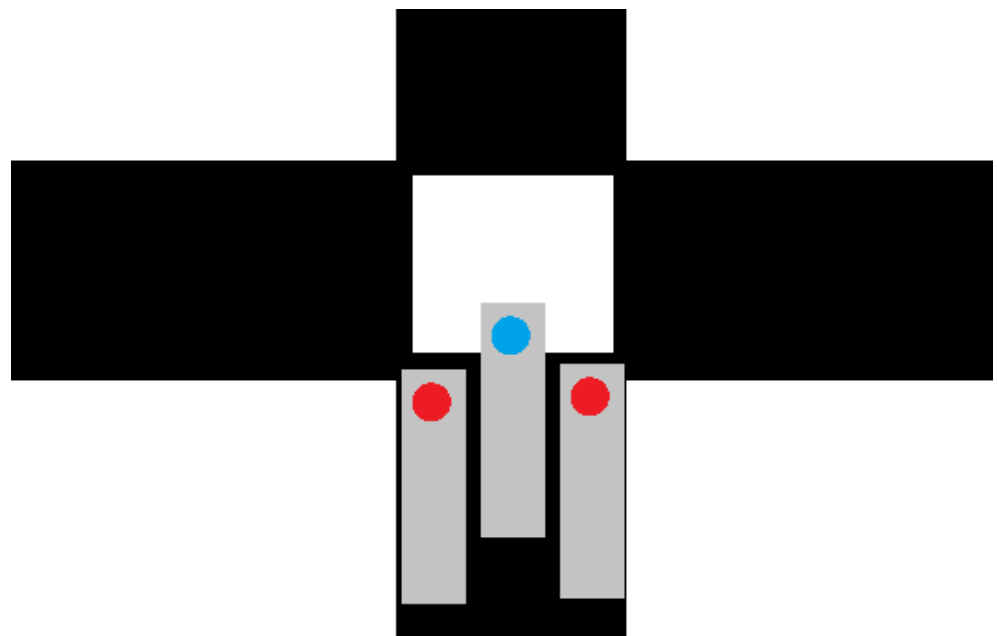


- 가운데 센서는 이제 길을 인식하는 용도보다는 노드를 발견하는 용도에 주로 사용한다.
- 위와 같은 라인 트레이서의 설치 구조로 일부 노드들을 테스트 이후 주행 보도판의 모든 노드에 똑같이 적용하였다.

활동 내용  
및  
문제 해결



- 3개의 라인 트레이서 센서가 노드에 닿기 전



- 중앙 센서가 노드에 닿았을 때 인식된다.

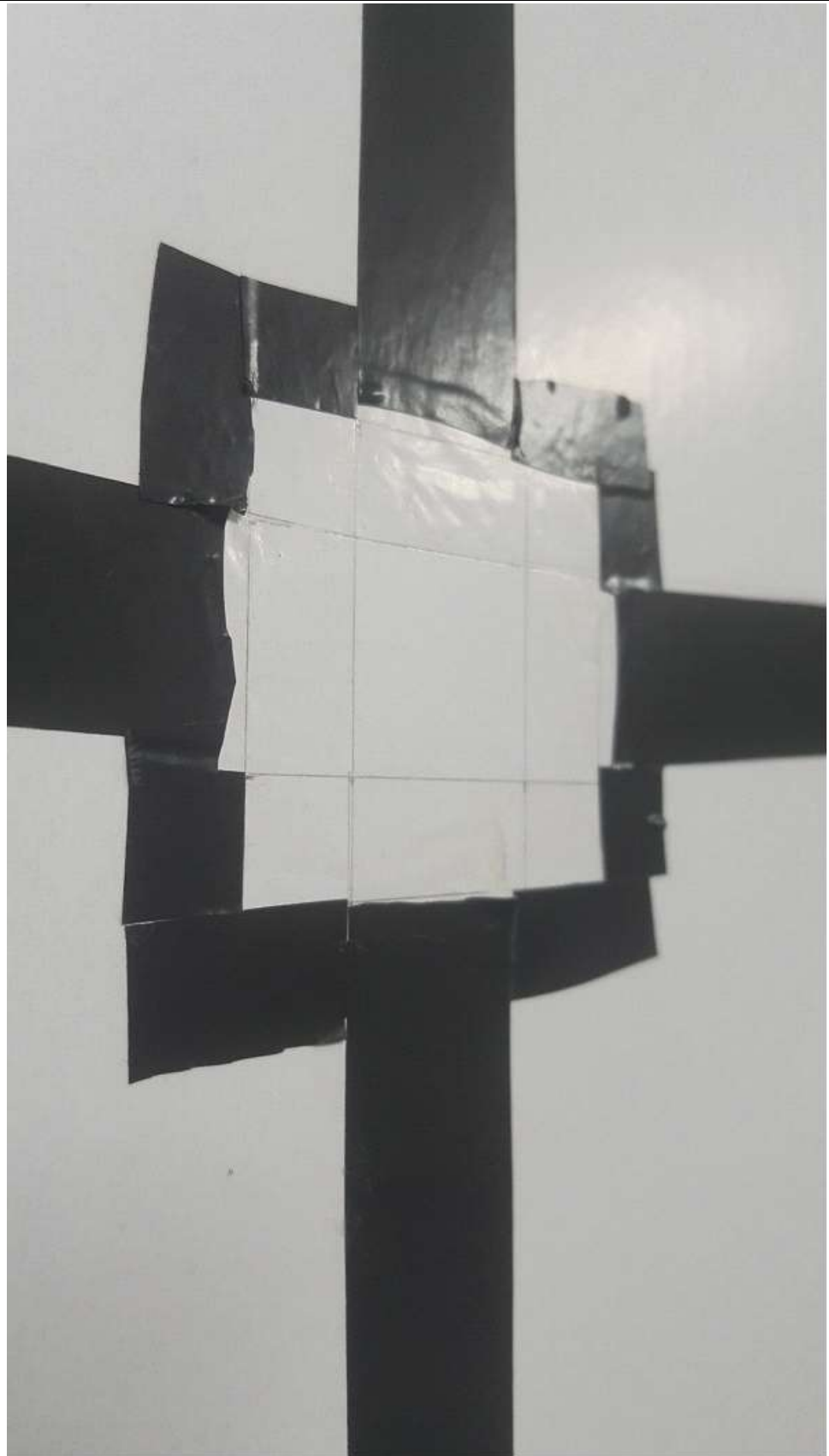


활동 내용  
및  
문제 해결





활동 내용  
및  
문제 해결



- 하나의 노드를 확대해서 보면 위와 같다.

활동 내용  
및  
문제 해결



활동 내용  
및  
문제 해결

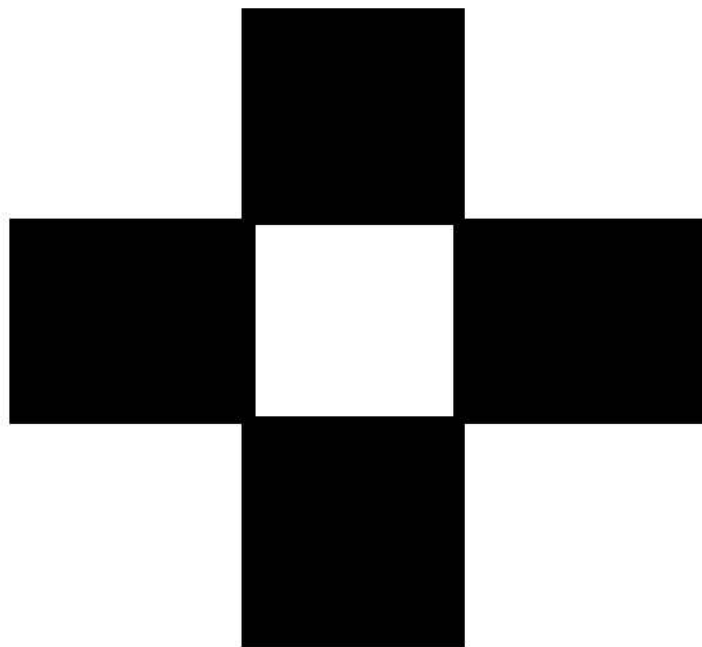
■ 문제 발생 및 원인 파악

- 자동차가 교차로(노드)에서 정확히 멈추지 못하였다.
- 원인을 파악한 결과, 라인 트레이서가 노드를 제대로 인식하지 못하여 노드에서 정확히 멈추지 못한 것으로 판단하였다.

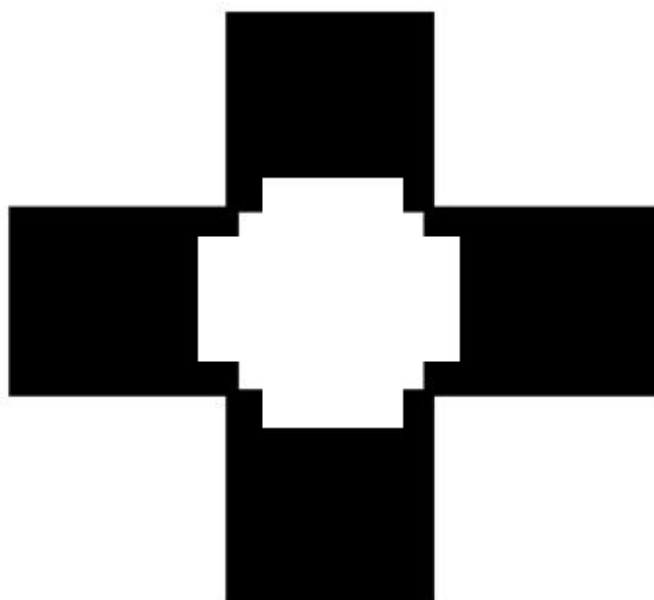
■ 문제 해결

- 노드의 인식률을 높이기 위해 도로의 끝 부분인 검은 변을 안쪽으로 조금 파내었고 그 형태는 다음과 같이 생겼다.
- 도로의 너비를 넓혀 검은선을 더 굵게 만들었다.

활동 내용  
및  
문제 해결



(변경 전)



(변경 후)

활동 내용  
및  
문제 해결

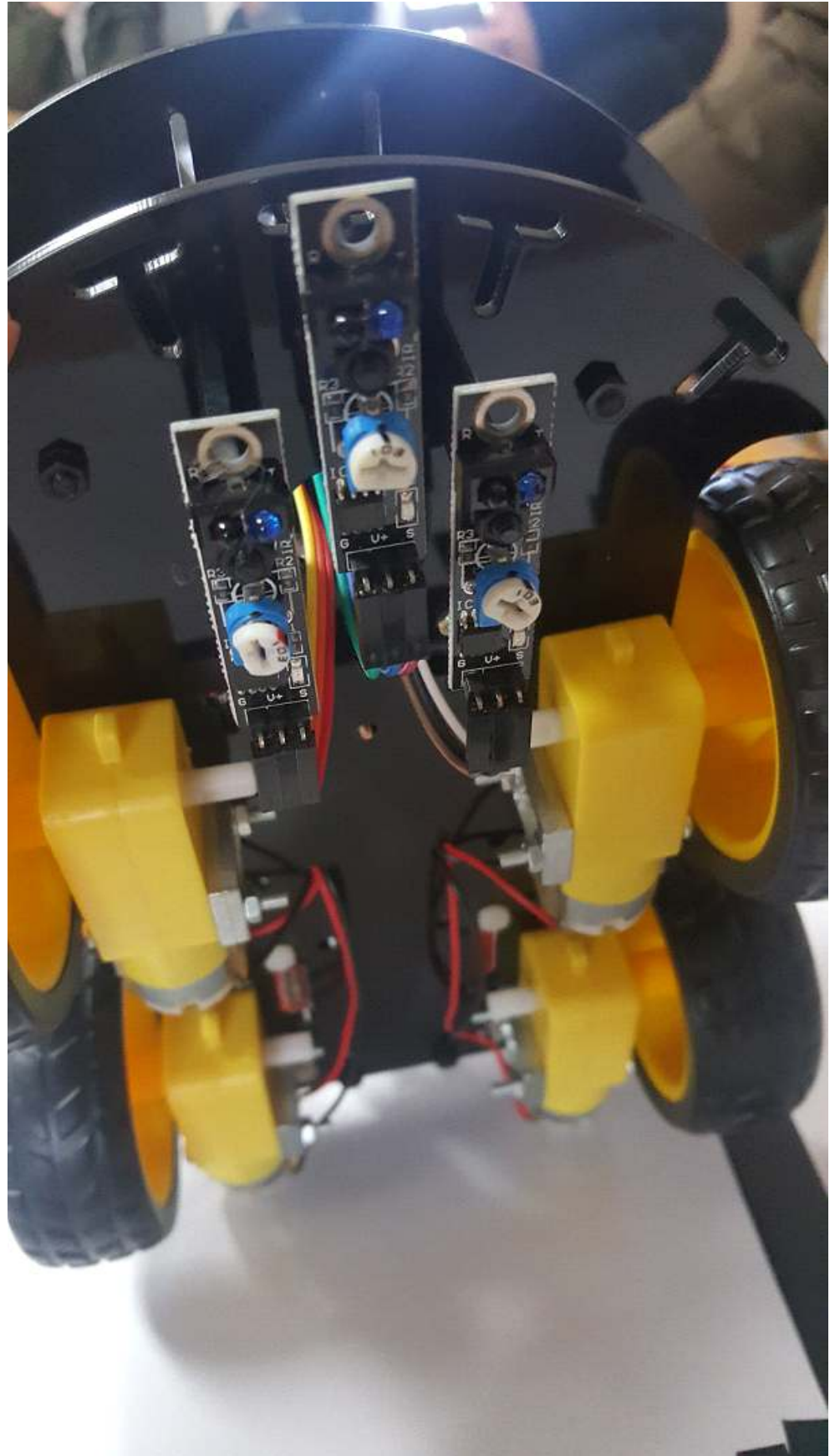


활동 내용  
및  
문제 해결





활동 내용  
및  
문제 해결



활동 내용  
및  
문제 해결

- 아두이노에서 json 형식의 데이터를 파싱하였다.
  - 수신된 데이터를 response라는 변수에 저장할 것이다.
  - 우선 테스트를 위해 데이터가 잘 수신되었다고 가정하고 진행하였다. 테스트 코드에서는 response에 아래와 같은 json 데이터를 직접 대입하여 개발을 진행하였다.
- ```
@{'state':3,'command':['AF','AR','AL']}
```
- json 파싱을 위해 ArduinoJson.h 헤더 파일의 StaticJsonBuffer 클래스를 사용하였다.
  - 파싱된 데이터를 JsonObject형 변수에 저장하여 사용하였다.
  - String을 분석하여 @를 구분자로 사용한다.
  - 명령이 몇 개인지 저장하는 변수 state
  - 각 명령어들을 반복문을 순회하면서 DIR이라는 배열에 하나씩 저장한 후, 저장이 잘 되었는지 디버깅용 출력으로 확인하였다.



## 아두이노 json 파싱 코드

```
#include <ArduinoJson.h>

void setup() {
    Serial.begin(9600);

    // put your setup code here, to run once:
    String response = "";

    response="@{'state':3,'command':['AF','AR','AL']}";
    //테스트 json파일, 이런 형식으로

    /////JSON 파싱 시작/////
    String seperator="@";
    int startindex = response.indexOf(seperator);
    if (startindex != -1)
    {
        String json_raw_data =
response.substring(startindex + 1); //@바로 앞 인덱스
        StaticJsonBuffer<2500> jsonBuffer;
        JsonObject& json_data =
jsonBuffer.parseObject(json_raw_data);

        int state = atoi(json_data["state"]); //명령 몇개인지
        String command=json_data["command"]; //명령
배열 문자열로 가져옴
        String DIR[state]; //명령어 하나씩 들어갈 배열

        int a = 0;

        for(int i=0; i<command.length(); i++){
            if(command[i] == 'A'){
                DIR[a] = command[i+1];
                a++;
            }
        }

        Serial.println(command[3]=='F');

        Serial.println("sb");
        Serial.println(state);
        Serial.println(command);
    }
    /////JSON 파싱 끝/////
}

void loop() {
    // put your main code here, to run repeatedly:

}
```

활동 내용  
및  
문제 해결

■ 문제 발생 및 원인 파악

- 좌회전 시 라인 트레이서의 센서가 노드를 계속 인식한 채로 방향을 꺾으면서 차체가 경로를 이탈하는 현상이 발생했다.
- 노드를 감지하지 않도록 하는 방법을 생각하는 것이 필요하였다.

■ 문제 해결

- 기존의 알고리즘: 노드를 감지하자마자 정지한다.
- 변경 후 알고리즘: 노드를 감지하면 약간 더 앞으로 이동(직진)한 후 정지한다.
- 노드 감지와 관련한 주행 알고리즘을 위와 같이 변경하면서 노드가 계속 감지되어 경로를 이탈하던 현상을 해결하였다.

활동 내용  
및  
문제 해결



- 검은색 사각형을 반복하도록 하여 경로 이탈이 발생하는 지 확인하였다.
- 검증 결과 문제가 해결된 것을 확인하였다.

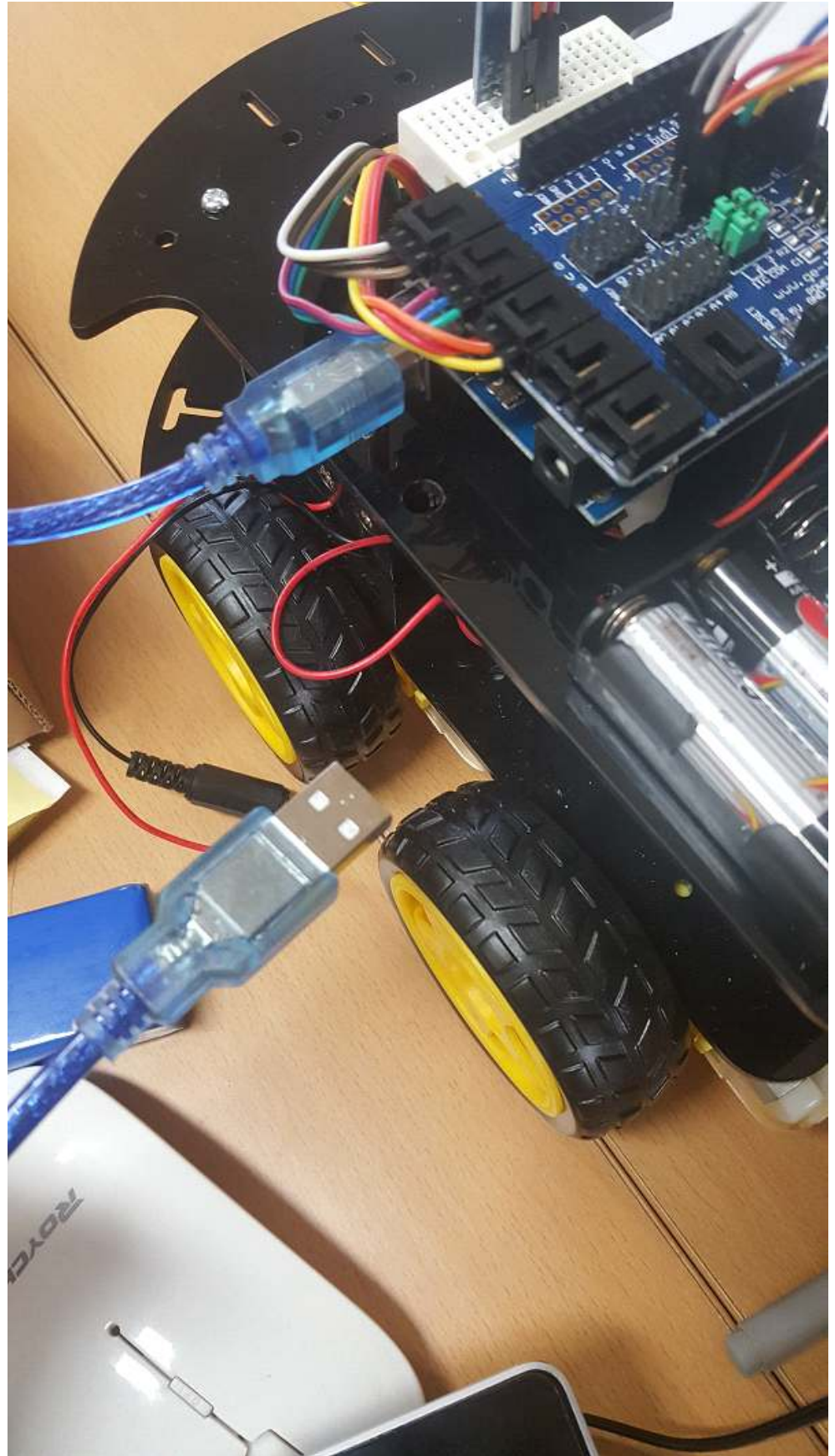
활동 내용  
및  
문제 해결

■ 문제 발생

- 배터리에 연결된 양극과 음극의 전선 중 하나(빨간 선)가 끊어졌다.
- 일단 USB 포트를 사용하여 노트북과 연결하는 방식으로 전원을 대체하여 프로젝트의 개발을 계속 진행하였다.

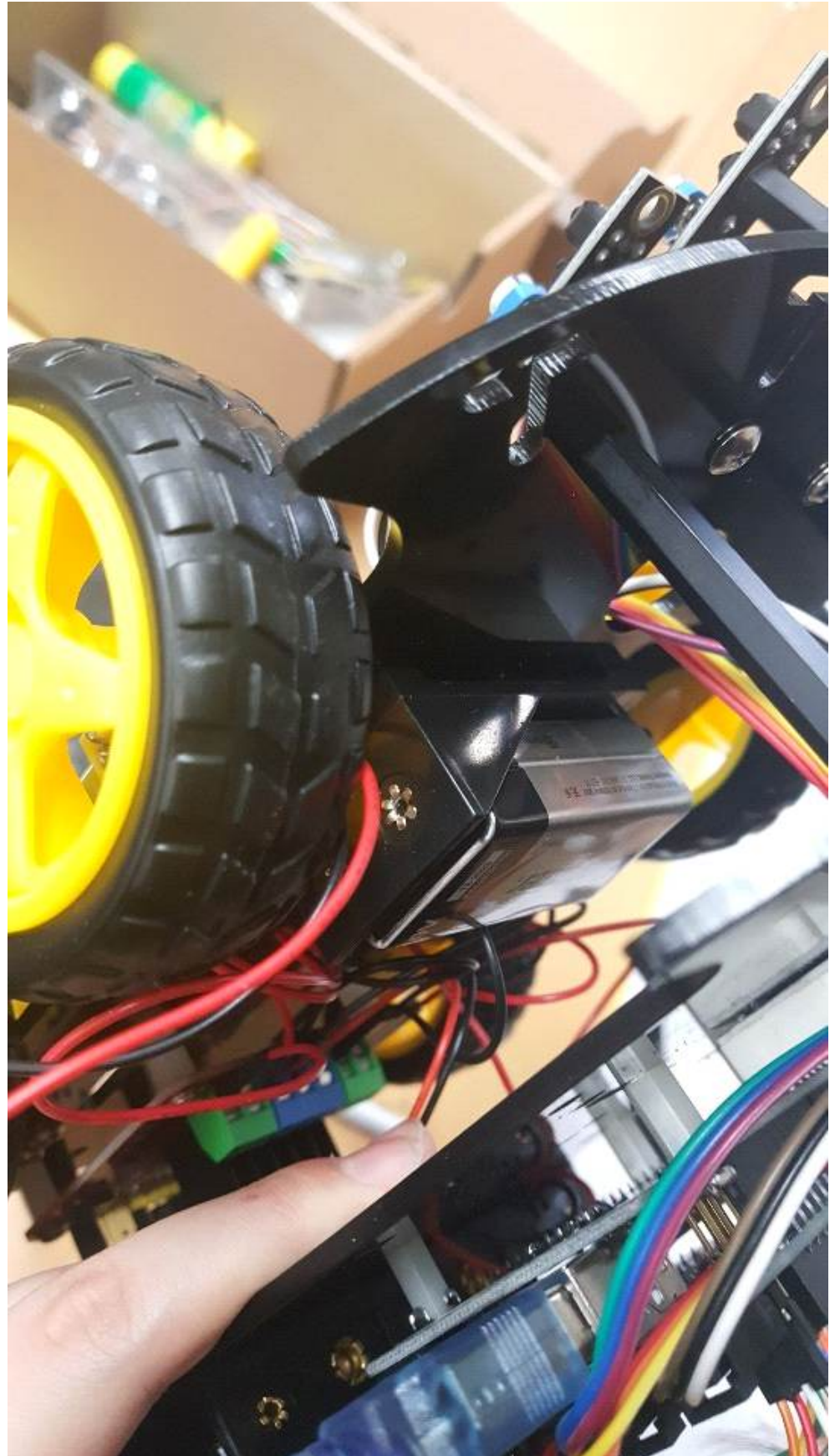


활동 내용  
및  
문제 해결





활동 내용  
및  
문제 해결

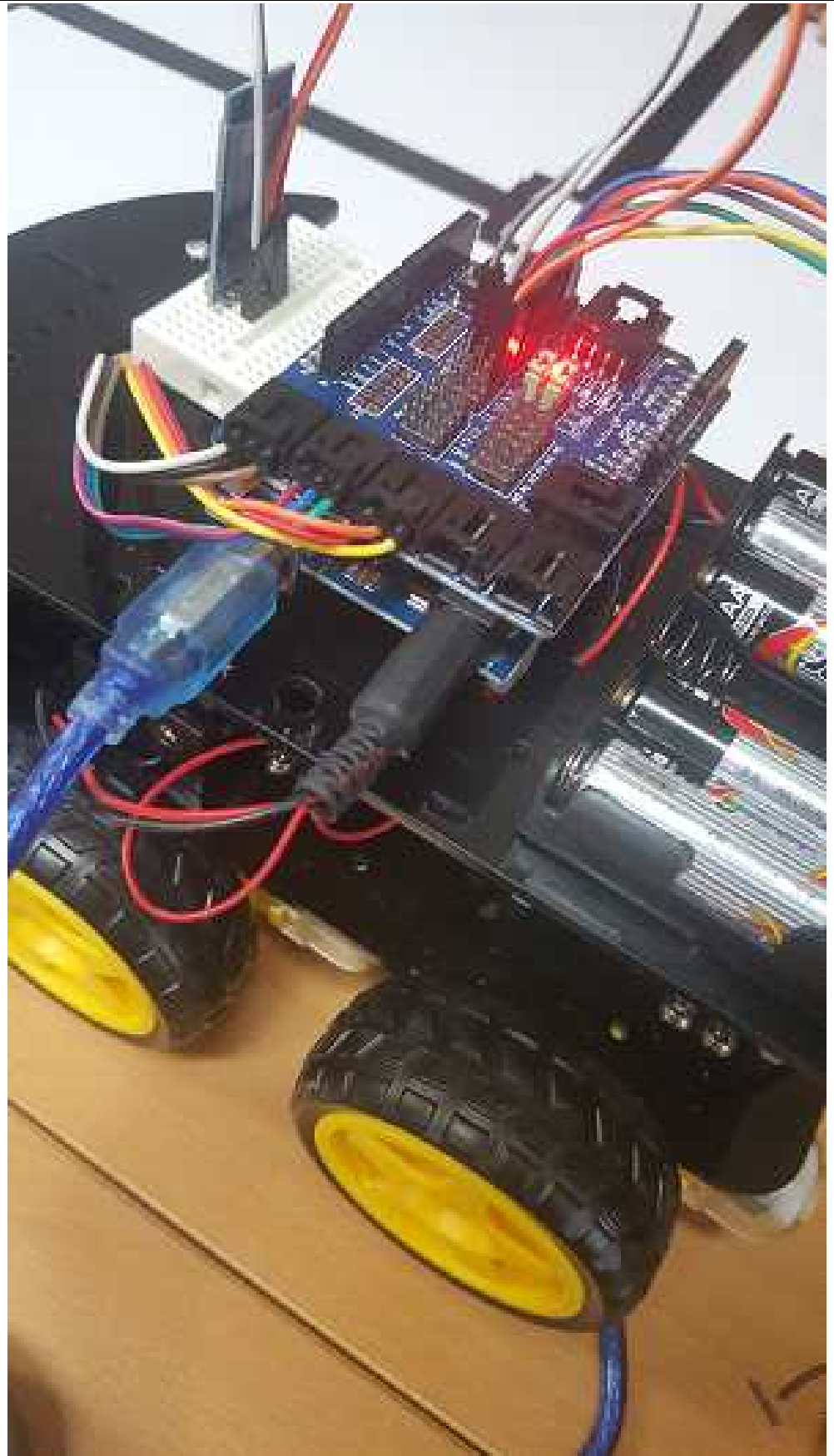


활동 내용  
및  
문제 해결

■ 문제 해결

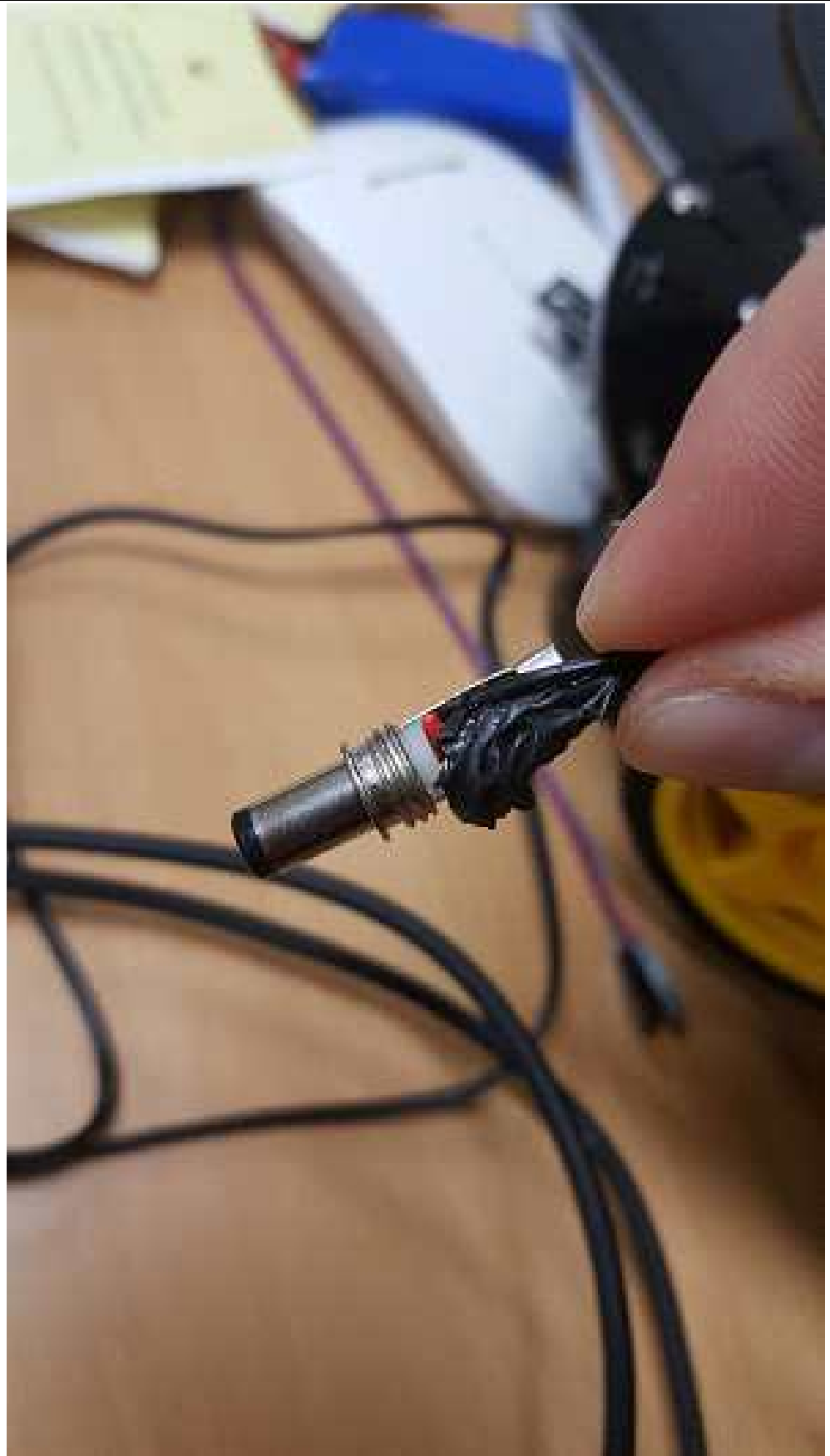
- 조원 중 한 명이 대체할 부품을 찾으러 떠났다.
- 대체할 부품을 찾으러 다녔으나 알맞은 부품을 찾지 못했고, 주변 물건을 활용하여 전원이 연결되도록 수리하는 방법을 궁리하였다.
- 프로젝트의 진행에 사용되었던 재료 중 PVC 전열 테이프를 사용하여 전선이 포트의 끊어진 부분과 접촉될 수 있도록 전선을 고정하였다.

활동 내용  
및  
문제 해결



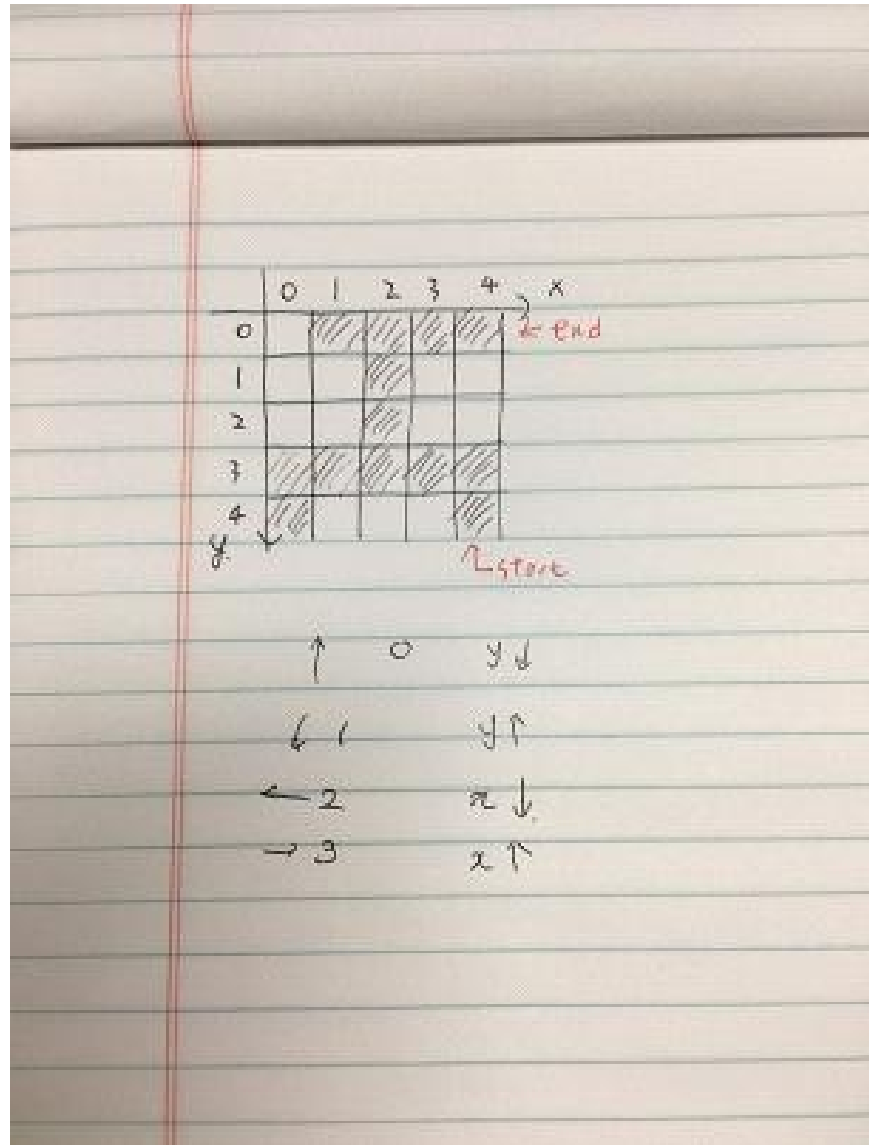


활동 내용  
및  
문제 해결



활동 내용  
및  
문제 해결

■ 경로 설정



- 주행 도로판의 각 교차로를 하나의 칸으로 보고, 길은 검정색으로 색칠된 칸이다.
- 0은 직진, 1은 후진, 2는 좌회전, 3은 우회전을 의미한다.
- 위 사진과 같은 상황에서 움직이는 과정은 다음과 같다.

활동 내용  
및  
문제 해결

- 먼저 (4, 4)에서 출발하며 차체는 북쪽을 바라보고 있다고 가정한다.
  - 목적지인 (0, 4) 위치에 도달하기까지 아래의 순서대로 명령이 수행되었다고 시나리오를 작성하였다.
1. (4,4)에서 직진한다.
  2. (3,4)에서 좌회전한다.
  3. (3,3)에서 직진한다.
  4. (3,2)에서 직진한다.
  5. (3,1)에서 직진한다.
  6. (3,0)에서 좌회전한다.
  7. (4,0)에서 좌회전을 두 번 수행한 후 직진한다.
  8. (3,0)에서 우회전한다.
  9. (3,1)에서 직진한다.
  10. (3,2)에서 우회전한다.
  11. (2,2)에서 직진한다.
  12. (1,2)에서 직진한다.
  13. (0,2)에서 좌회전한다.
  14. (0,1)에서 좌회전을 두 번 수행한 후 직진한다.
  15. (0,2)에서 직진한다.

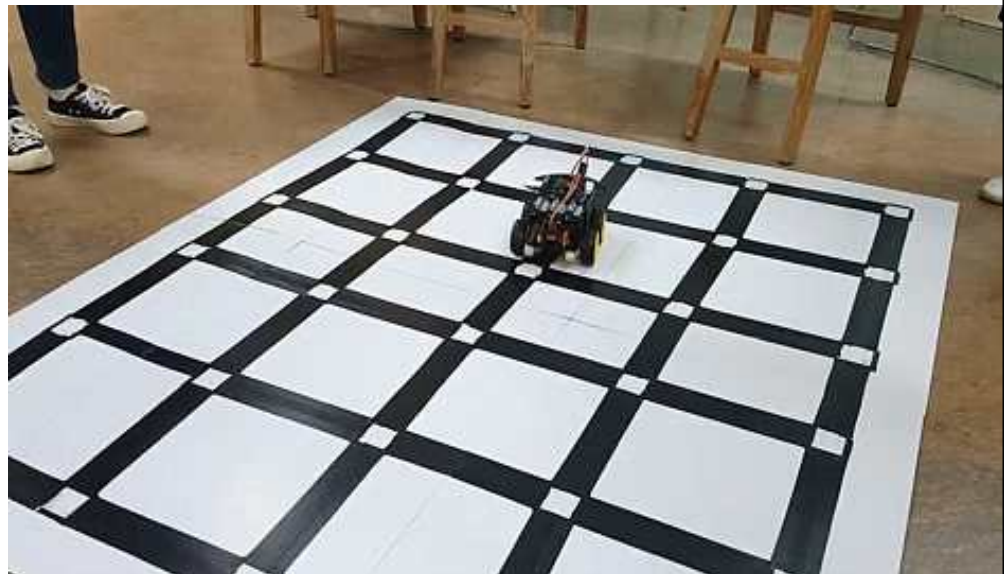
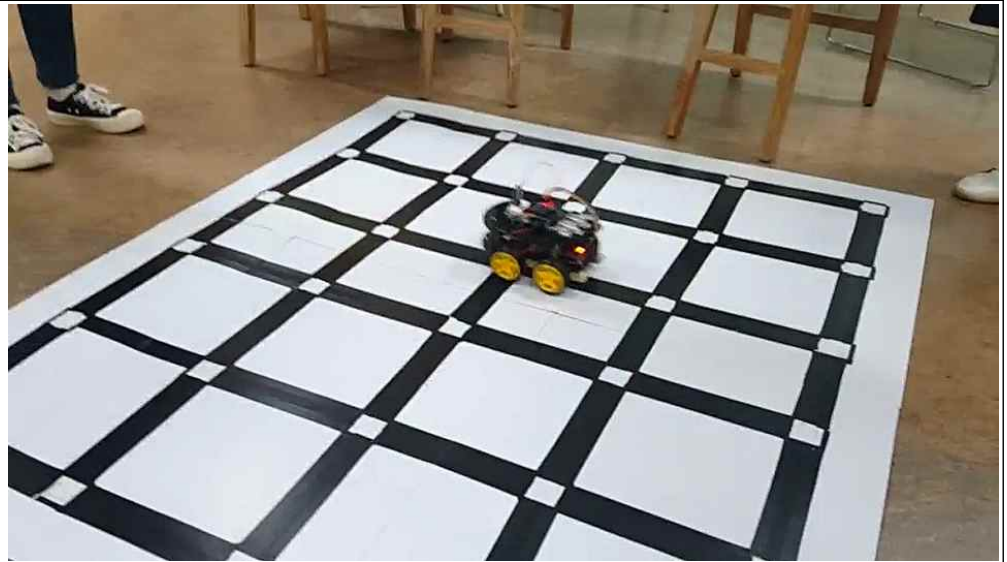
활동 내용  
및  
문제 해결

16. (0,3)에서 직진한다.

17. (0,4)에서 직진한다.

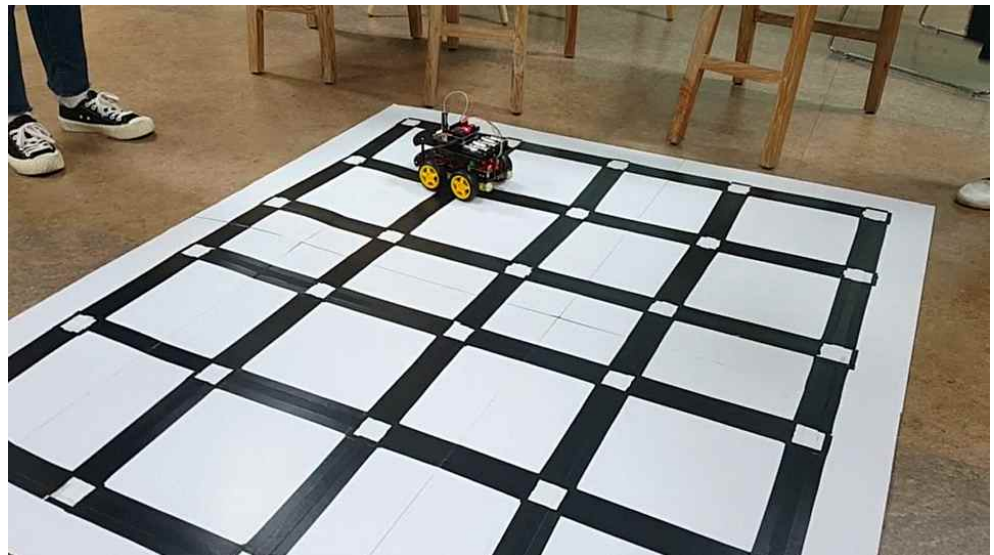
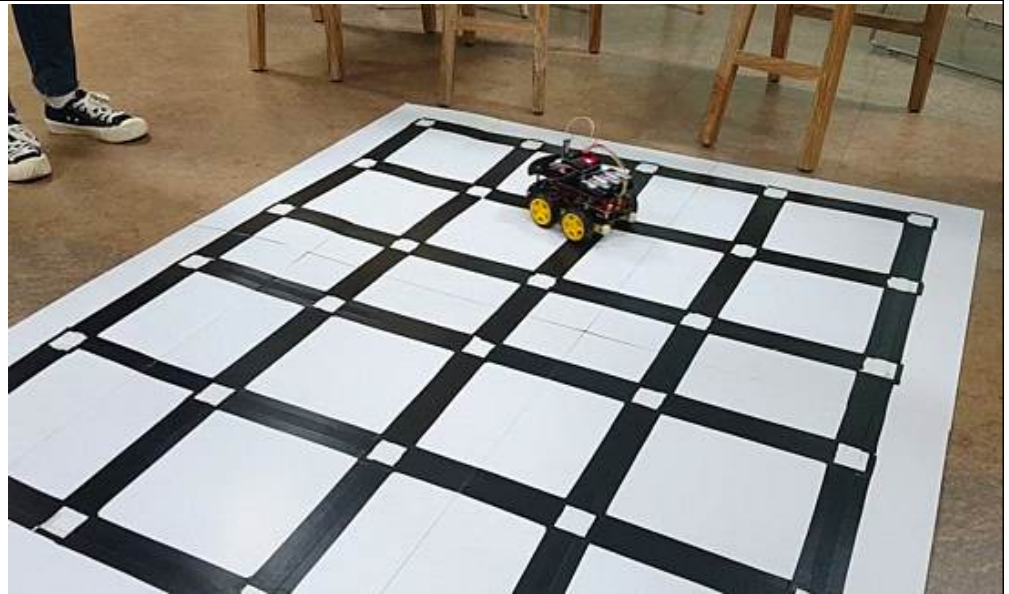
- 각 명령어를 숫자로 표현하면 0, 2, 0, 0, 0, 2, 2, 2, 0, 3, 0, 3, 0, 0, 2, 2, 2, 0, 0, 0, 0의 순서대로 수행되어야 한다.
- 명령어를 위와 같이 설정하였을 때, 해당 명령대로 움직이는지를 확인하는 검증 단계가 필요하다.

활동 내용  
및  
문제 해결



[해당 동영상은 시연 시 재생할 예정]

활동 내용  
및  
문제 해결

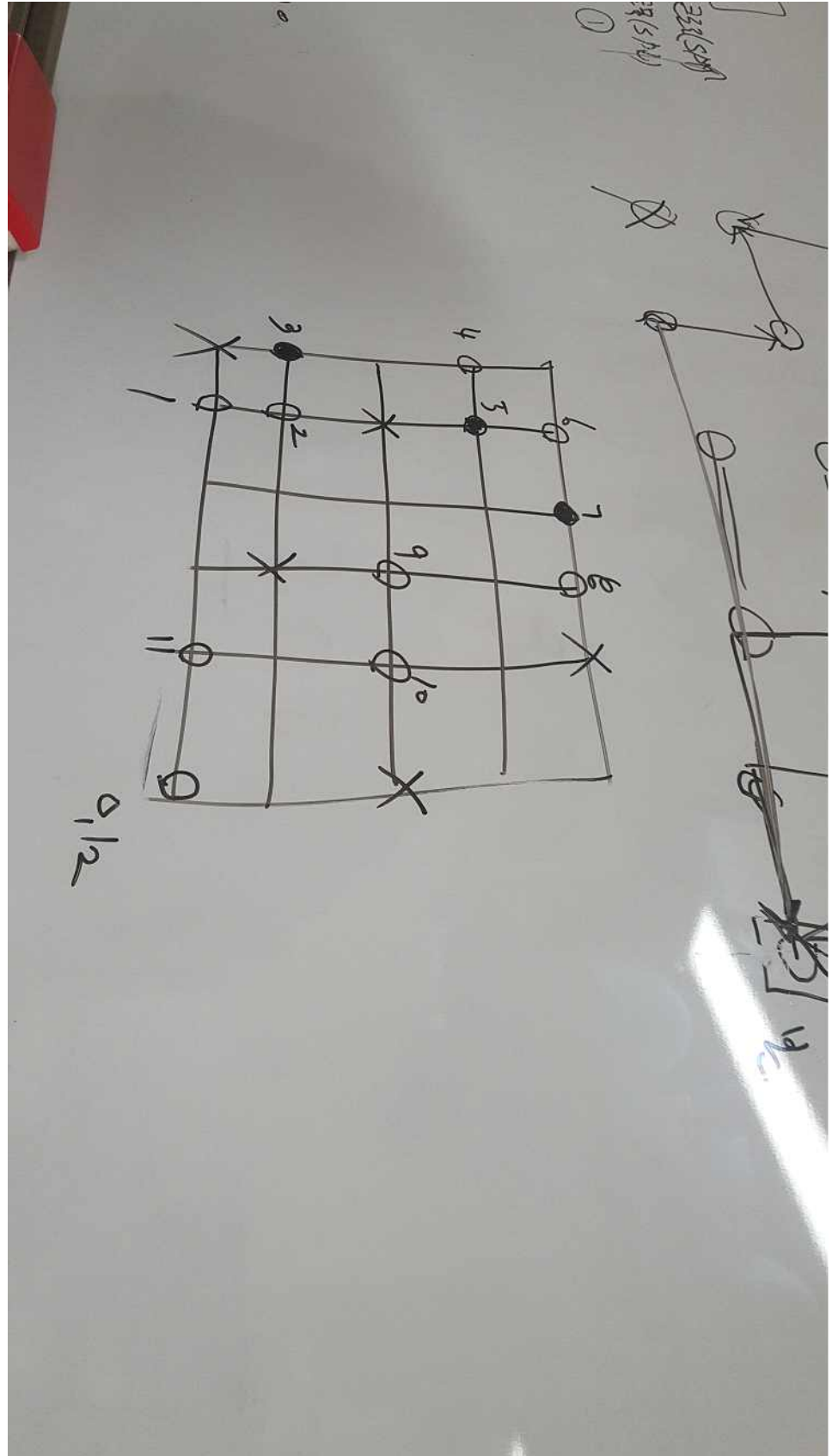


- 정해진 경로를 따라 아두이노 스마트 로봇 자동차 키트가 주행하는 것을 확인하였다.

| - 10차 회의            |                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 회의 일시               | 2018년 12월 25일 화요일 09:00~18:00                                                                                                                                                                                                                                                                                                                                          |
| 회의 장소               | 공과대학 7호관 6층 디자인룸                                                                                                                                                                                                                                                                                                                                                       |
| 활동 내용<br>및<br>문제 해결 | <p>■ 목표 설정</p> <ul style="list-style-type: none"> <li>- 블루투스 페어링을 통한 서버와의 통신</li> <li>- 명령어들을 미리 설정하여 서버에서 아두이노로 데이터를 전송한다.</li> <li>- @로 구분된 json의 형태를 아두이노가 수신하면 파싱 및 그에 알맞은 동작을 수행하는 것을 이번 회의의 목표로 한다.</li> </ul> <p>■ 세부 계획</p> <ul style="list-style-type: none"> <li>- 각 노드에 도착하는 순서를 정하였다.</li> <li>- X는 화재 지점, ○는 가야할 곳, ●는 환자의 위치로 설정하고 지도로 표시하였다.</li> </ul> |

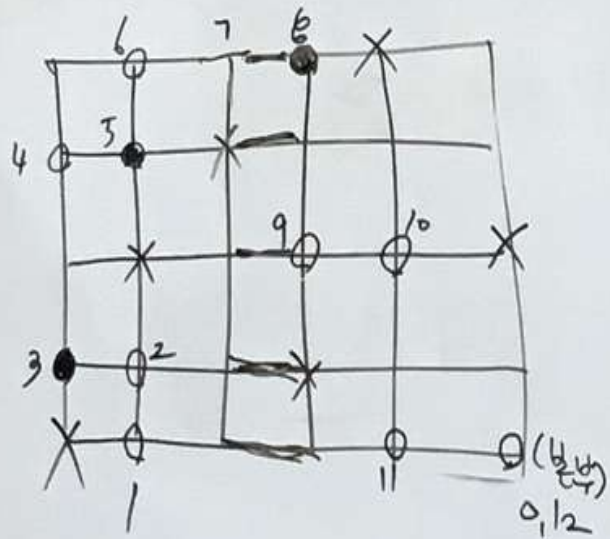


활동 내용  
및  
문제 해결





$X$  = 현상  
 $O$  = 지나가던 곳  
 $\bullet$  = 현상 위치



활동 내용  
및  
문제 해결

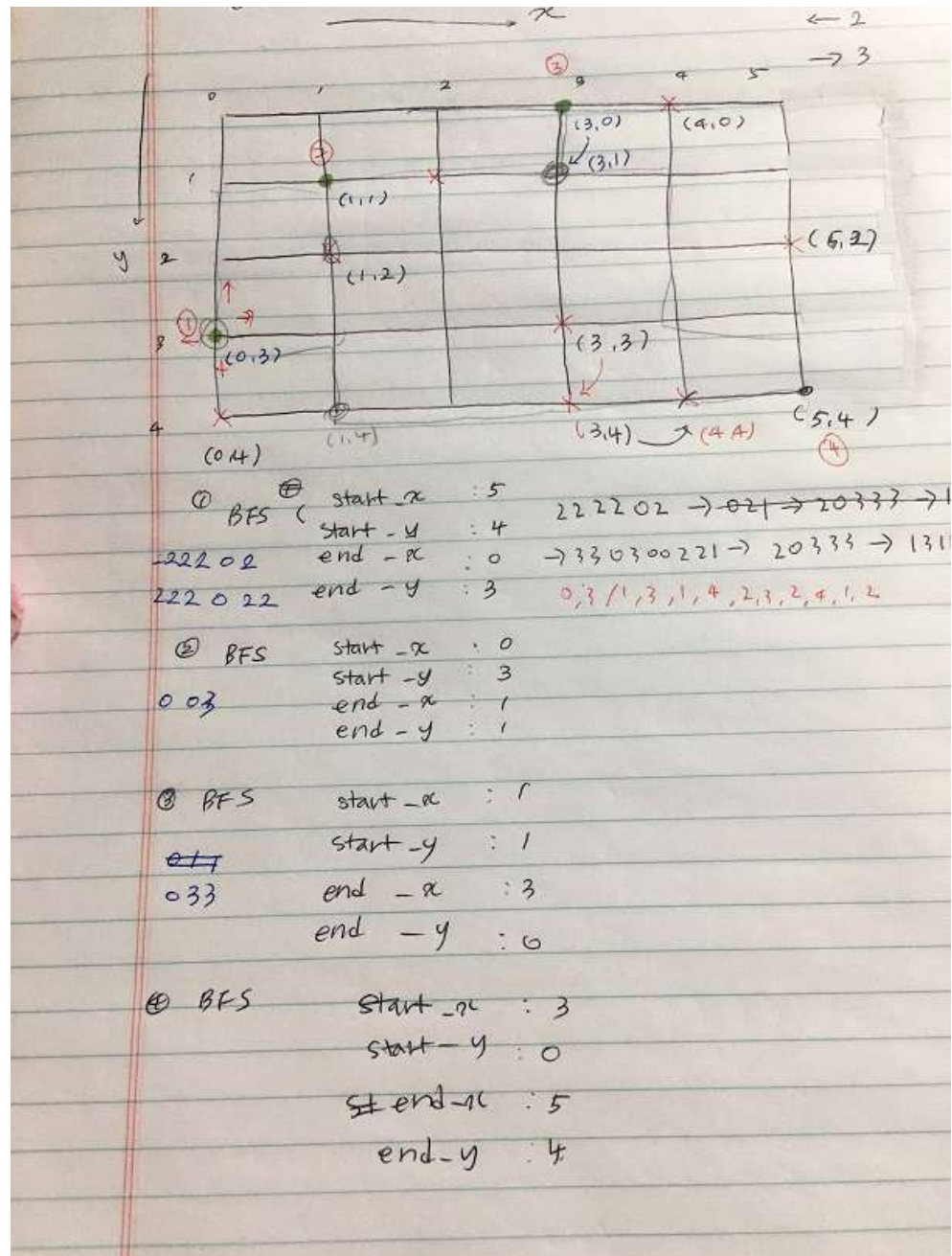
■ 문제 인식 후 주제 변경

- 재난 상황에서 재난이 발생한 지점에 도로가 평탄하게 있다는 것이 이상하다는 의견이 나옴
- 회의를 거친 후, 이대로 주제를 진행하기엔 현실적으로 납득이 되지 않는 부분이 많다고 판단하여 다른 상황 또는 배경을 생각하기로 하였다.
- 그 결과, “카풀 서비스”로 주제를 변경하였다.

■ 새로운 주제에 대한 변경 사항

- 기존의 “재난 센터”는 “카풀 본부”가 된다.
- 기존의 “재난 상황”, “재난 발생지”는 “교통 체증”으로 변경하였다.
- 기존의 “환자” 또는 “긴급 구조해야 할 대상”은 “카풀 서비스 이용자”로 변경하였다.
- 출발 지점으로부터 1번 사용자, 2번 사용자, 3번 사용자, ..... 다시 목적지로 돌아오는 구성을 취한다.
- 각 사용자에게 가기 위해 현재 위치로부터 너비 우선 탐색(BFS)을 사용하여 경로를 찾아가는 카풀 서비스이다.

## 활동 내용 및 문제 해결



활동 내용  
및  
문제 해결

■ 클래스 분석 및 작성

- ADOxx에서 사용될 클래스의 종류와 목적을 정하고 코드로 작성하였다. 다음과 같다.

1. end:

- 프로그램의 종료 지점이다.

2. Move:

- 아두이노 작동을 위한 명령어로 변환하는 부분이다. BFS\_Start의 결과는 동서남북의 방향만 나타내기 때문에, 기계가 직접 동작하기 위하여 앞쪽, 왼쪽, 오른쪽 회전의 명령어들로 전환하고 json의 형태로 가공하여 웹 서버로 전송한다.

3. BFS\_Start:

- BFS로 경로를 탐색하기 위한 부분이다.

4. Field\_Data\_Setting:

- 맵 데이터를 초기화하는 부분이다.

5. BFS:

- BFS 탐색을 위한 프로시저이다.

- 실제 연산을 이 프로시저에서 수행하고 있다.

6. Automobile\_motor:

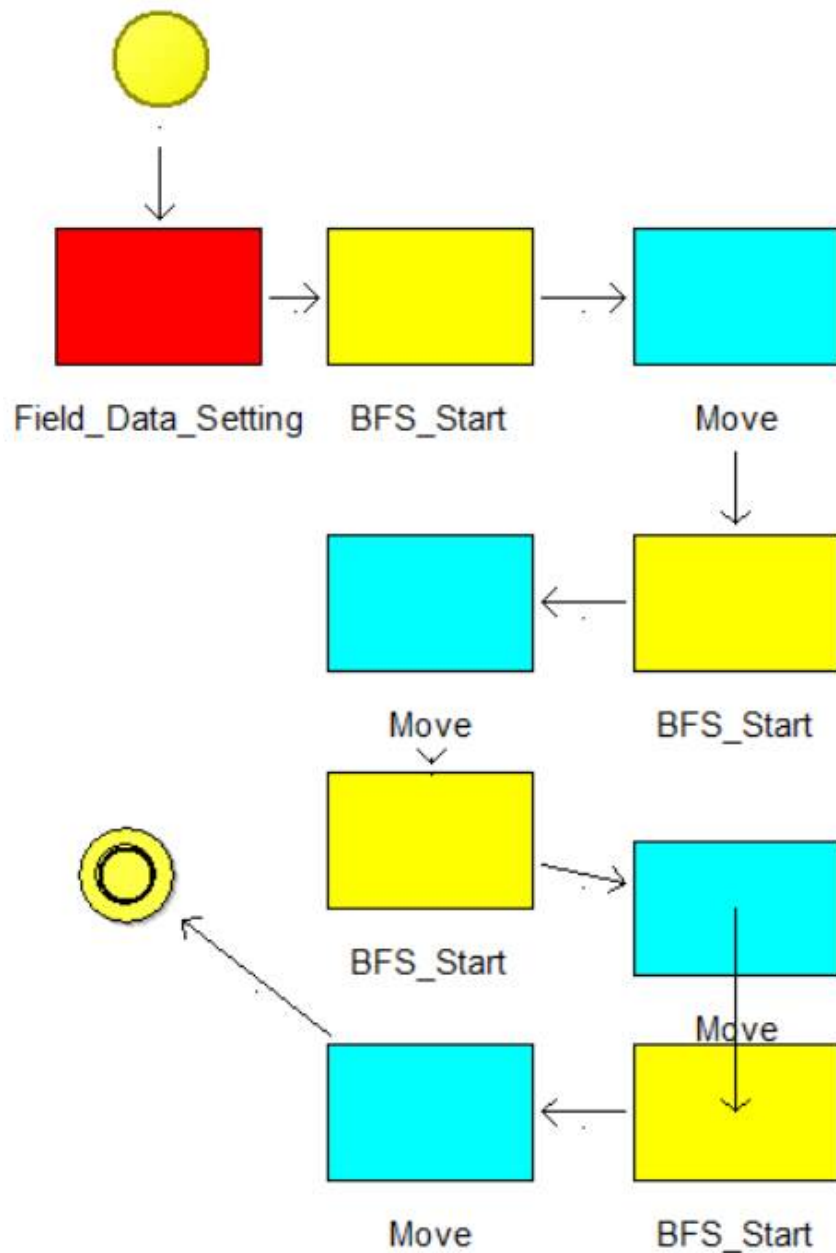
- 예제로 있었던 코드이나 수정하지 않았다.

7. path:

- 테스트를 위해 사용되었다.

활동 내용  
및  
문제 해결

■ 클래스에 의한 프로젝트 진행



- 프로그램이 시작되면 먼저 초기화 단계를 거친 후, 탐색(BFS)과 이동(Move)을 모든 목적지를 방문할 때 까지 반복하다 모두 마치면 프로그램을 종료한다.

활동 내용  
및  
문제 해결

■ 기존 BFS 코드를 변경

- 기존의 BFS 코드는 시작점(x, y)가 주어지면 배열을 참고하여 특정 값을 찾기만 하였다.
- 여기에는 목적지가 없었기 때문에 아무 일도 일어나지 않았었다.
- 이번 회의에서 BFS 프로시저의 매개변수를 first\_x, first\_y 2개가 아닌, start\_x, start\_y, end\_x, end\_y인 4개로 변경하였다.
- 따라서 매번 다음 목적지를 가기 위해 현재 위치로부터 BFS를 수행하며, 가장 가까운 목적지를 선택하여 경로를 저장하게 된다.

■ 장애물의 종류 설정

- 장애물은 “교통혼잡” 1가지로만 선정하였다.
- 장애물의 종류는 여러 가지가 있을 수 있으나 이름과 그림 표시만 다를 뿐 그 의미와 역할은 동일하다고 생각하여 하나의 이름만 사용하였다.
- 또한, 현대 도심에서 교통 혼잡은 길 찾기 알고리즘에서 가장 중요한 요소로 작용한다.

활동 내용  
및  
문제 해결

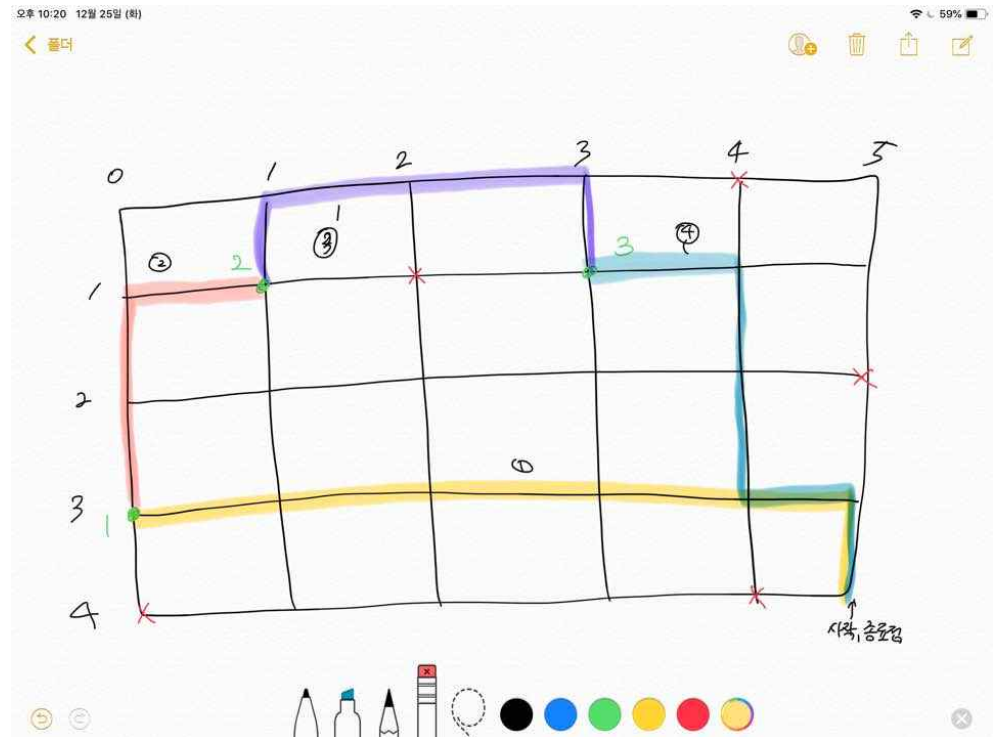
■ 경로 구성

- 자동차는 특정 목적지에 도달하기 위해 카풀 이용자(필수적으로 방문해야 하는 노드)와 교통 혼잡(갈 수 없는 노드) 2가지를 신경써야 한다.
- 맵 데이터에 교통 혼잡과 카풀 이용자의 각 위치를 등록하게 되면 프로그램은 너비 우선 탐색(BFS; Breadth First Search)를 따라 자동으로 다음 목적지를 결정하며, 그 과정에서 큐를 사용하여 올바른 경로를 저장한다.

■ 시나리오

- 모든 카풀 이용자를 탑승시키기 위해 도로를 따라 각 이용자의 위치로 주행한다.
- 주행하는 경로는 교통 혼잡을 최대한 피하는 것만 선택된다.
- 모든 카풀 이용자를 탑승시킨 이후에는 처음 위치로 돌아온다.

## 활동 내용 및 문제 해결



- 위는 각 사용자를 찾아가는 경로를 색깔별로 구분하여 표시한 그림이다.

### ■ 힘들었던 점

- AdoScript가 가지고 있는 문법과 특징을 파악하기 어려웠다.
- AdoScript를 사용하여 개발하는 도중 공백 문자 등의 단순 오타나, 모든 변수에 괄호가 필요한 등 다소 생소한 문제를 많이 마주하였다.
- 또한 사용자가 많은 언어가 아니라서 관련 커뮤니티가 없고, 관련 자료가 무척 적어 오류를 해결하는 데에 많은 어려움을 겪었다.



활동 내용  
및  
문제 해결

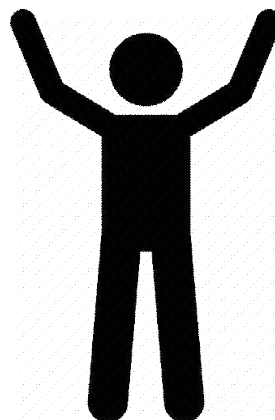


활동 내용  
및  
문제 해결

- 주행 도로판 위에 “교통 체증”과 “카풀 사용자”를 의미하는 그림을 부착하여 자동차의 움직임이 시연 시 외부인에게 잘 이해될 수 있도록 하였다.
- 교통 체증은 차가 막혀있는 다음과 같은 사진을 사용하며,

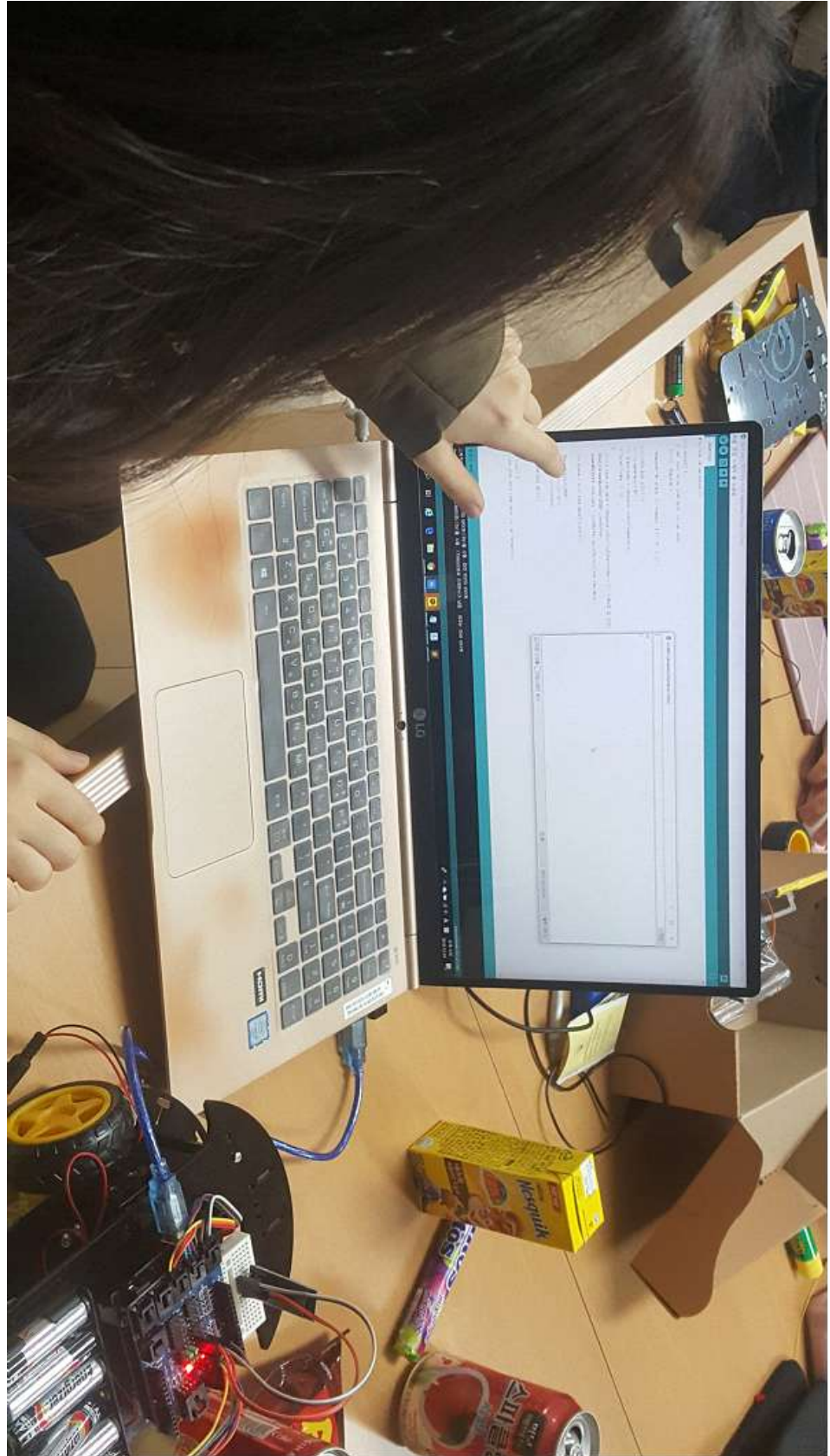


- 카풀 사용자는 사람이 두 팔을 벌리고 택시를 부르는 듯한 자세를 취하는 그림을 사용하고 있다.



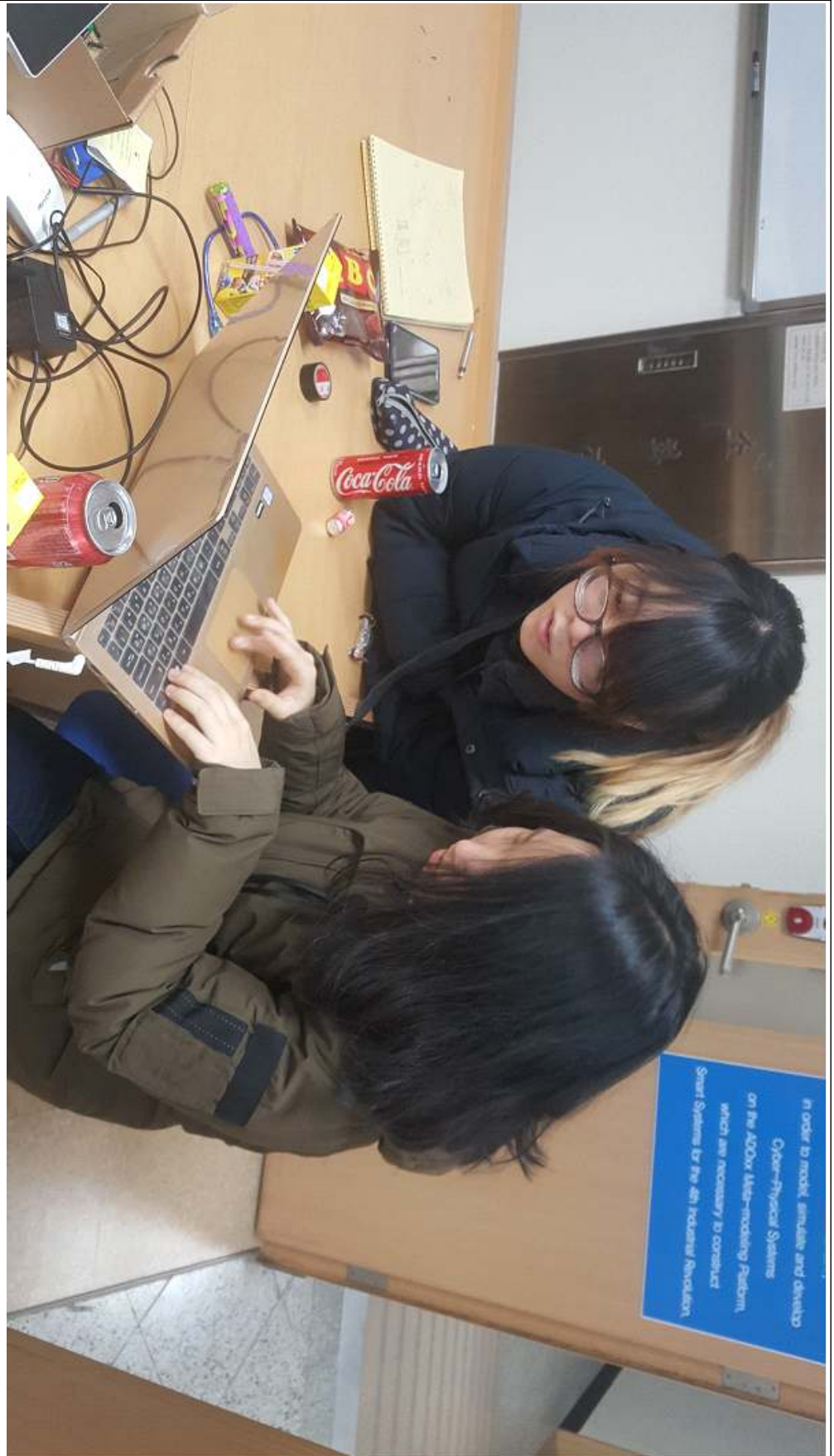
## b. 활동 사진

활동 사진





## 활동 사진



## 활동 사진



## 활동 사진





## 활동 사진



활동 사진





활동 사진



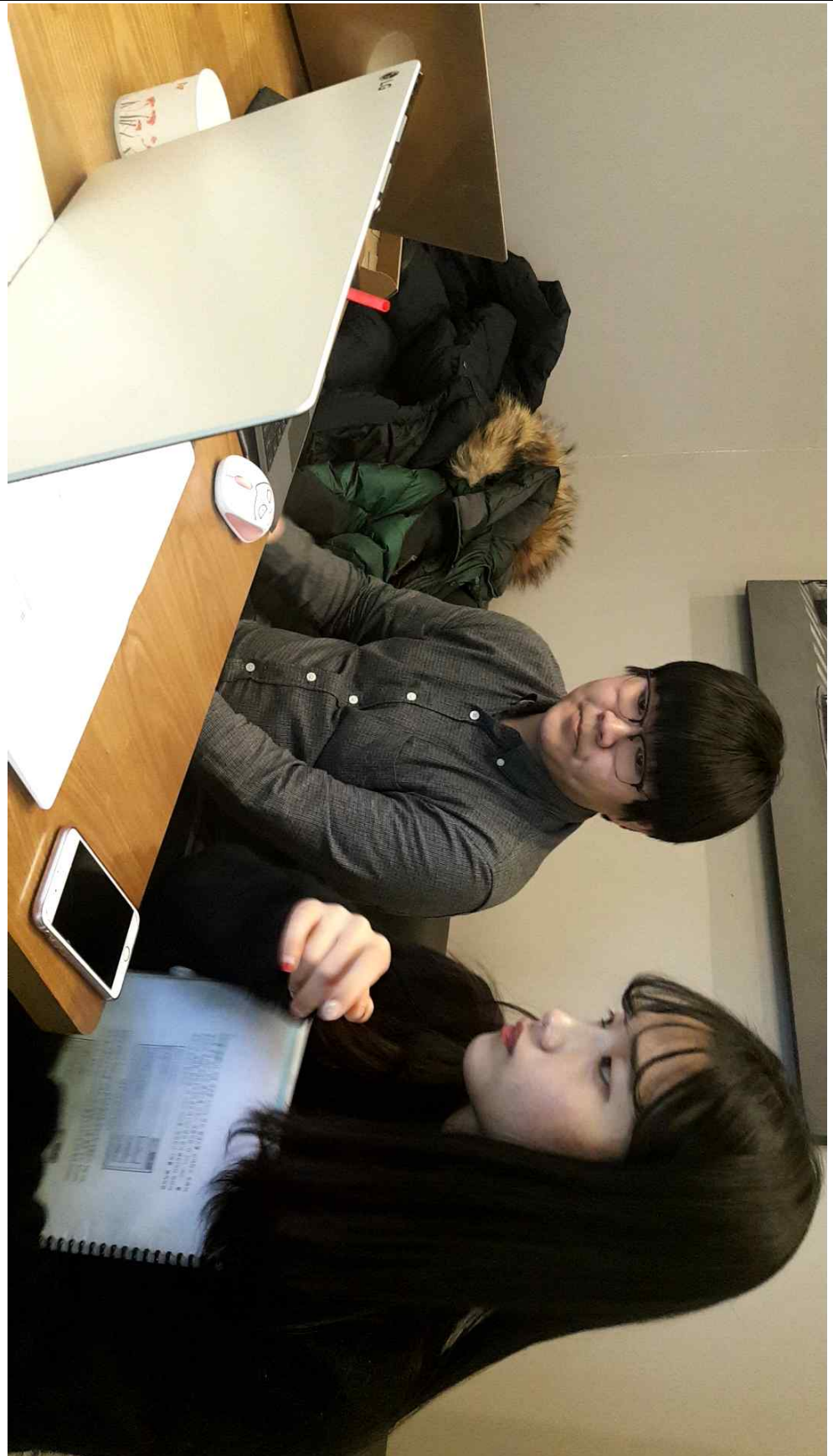
## 활동 사진



활동 사진



## 활동 사진

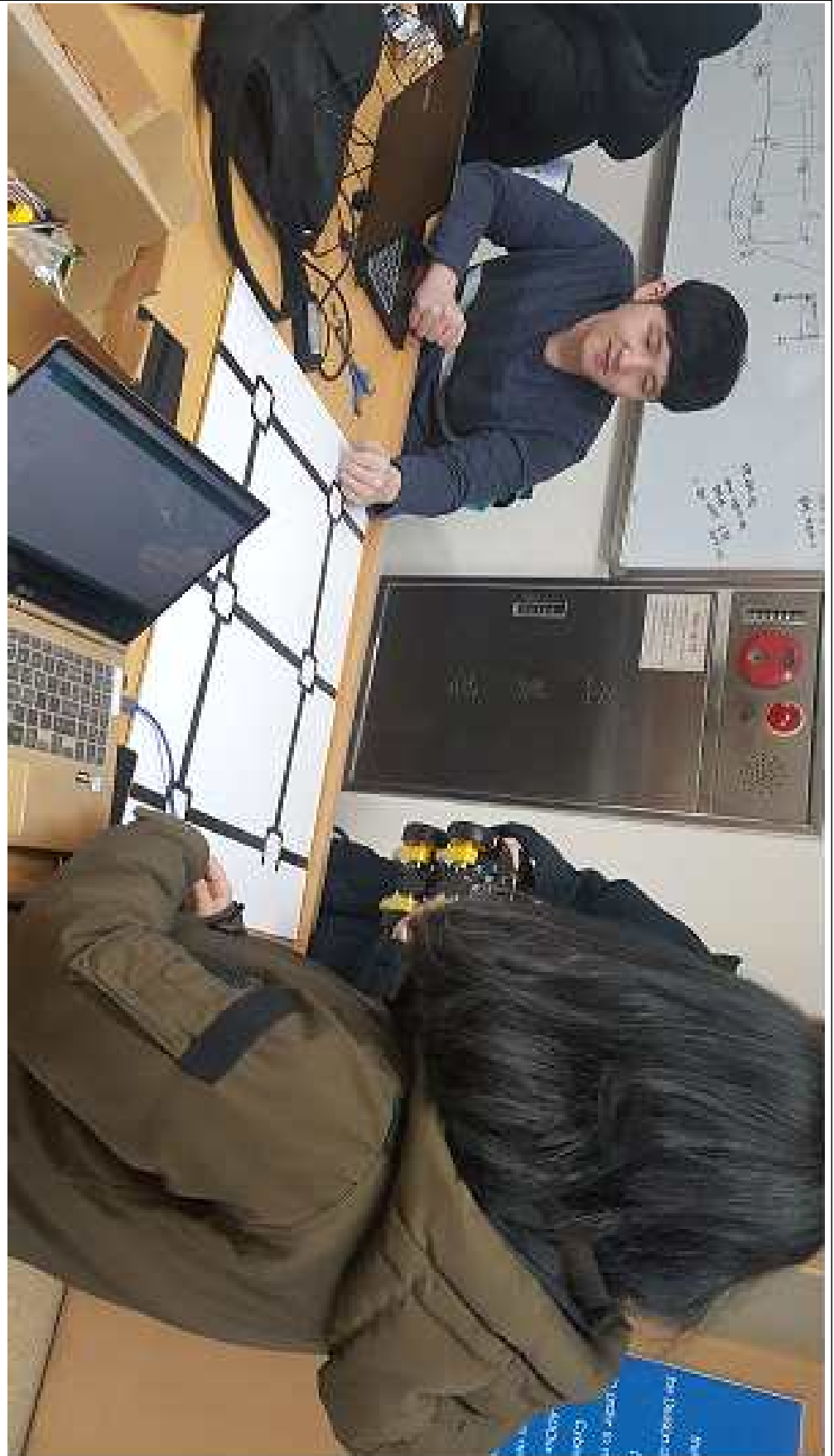


## 활동 사진





## 활동 사진



## 3. 결과

### a. 아두이노 최종 코드

#### 아두이노 최종 코드

```
// Line Tracer basic code
#include <SoftwareSerial.h>
#include <ArduinoJson.h>

#define CAR_DIR_FW 0    //전진
#define CAR_DIR_BK 1    //후진
#define CAR_DIR_LF 2    //좌회전
#define CAR_DIR_RF 3    //우회전
#define CAR_DIR_ST 4    //정지
#define CAR_DIR_FIN 5    //운행 안 함

#define F 0
#define L 1
#define R 2
#define FIN 3

#define BT_RXD 8
#define BT_TXD 9
SoftwareSerial bluetooth(BT_TXD, BT_RXD);

int state;
int num=1;
int END = 0;    //마지막 JSON 파일 읽기 전까지는 0

int g_carDirection = CAR_DIR_FIN; //운행방향 변수. 초기는
정지상태
int g_carSpeed = 2;    //최대속도의 50~60%

int rotation=0;

#define Size 10 //json 파일 하나당 들어갈 명령어 개수
int DIR[Size] = {FIN, FIN, FIN, FIN, FIN, FIN, FIN, FIN, FIN,
FIN}; //명령어 하나씩 들어갈 배열

#define ENA 6
#define EN1 7
#define EN2 3
#define EN3 4
#define EN4 2
#define ENB 5
```

## 아두이노 최종 코드

```
#define Front 0
#define Left 1
#define Right 2
int a [] = {Front,Left,Right};

void init_car_controller_board() {
    pinMode(ENA , OUTPUT);
    pinMode(EN1 , OUTPUT);
    pinMode(EN2 , OUTPUT);
    pinMode(ENB , OUTPUT);
    pinMode(EN3 , OUTPUT);
    pinMode(EN4 , OUTPUT);
}

void car_forward() {
    digitalWrite(EN1 , LOW);
    digitalWrite(EN2 , HIGH);
    digitalWrite(ENA , g_carSpeed);
    digitalWrite(EN3 , LOW);
    digitalWrite(EN4 , HIGH);
    digitalWrite(ENB , g_carSpeed);
}

void car_left() {
    digitalWrite(EN1 , HIGH);
    digitalWrite(EN2 , LOW);
    digitalWrite(ENA , g_carSpeed);
    digitalWrite(EN3 , LOW);
    digitalWrite(EN4 , HIGH);
    digitalWrite(ENB , g_carSpeed);
}

void car_right() {
    digitalWrite(EN1 , LOW);
    digitalWrite(EN2 , HIGH);
    digitalWrite(ENA , g_carSpeed);
    digitalWrite(EN3 , HIGH);
    digitalWrite(EN4 , LOW);
    digitalWrite(ENB , g_carSpeed);
}
```



## 아두이노 최종 코드

```
void car_stop() {
    delay(400);

    analogWrite(ENA, 0);
    analogWrite(ENB, 0);

    rotation = 1;
}

void car_finish() {
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
}

void car_update() {
    //매번 loop에서 호출되자만 커맨드가 있을 경우에만 호출될 수
    있도록 변경해야함
    if(g_carDirection == CAR_DIR_FW) {car_forward();}
    else if(g_carDirection == CAR_DIR_LF) car_left();
    else if(g_carDirection == CAR_DIR_RF) car_right();
    else if(g_carDirection == CAR_DIR_ST) car_stop();
    else if(rotation == 0) car_stop();
    else if(g_carDirection == CAR_DIR_FIN) car_finish();
}

#define LT_MODULE_L A0
#define LT_MODULE_F A1
#define LT_MODULE_R A2

void init_line_tracer_modules() {
    pinMode(LT_MODULE_L, INPUT);
    pinMode(LT_MODULE_F, INPUT);
    pinMode(LT_MODULE_R, INPUT);
}

bool It_isLeft() {
    int ret = digitalRead(LT_MODULE_L);
    return ret == 1? true:false;
}

bool It_isForward() {
    int ret = digitalRead(LT_MODULE_F);
    return ret == 1? true:false;
}

bool It_isRight() {
    int ret = digitalRead(LT_MODULE_R);
    return ret == 1? true:false;
}
```

## 아두이노 최종 코드

```
void It_mode_update() {
    int ll = It_isLeft();
    int ff = It_isForward();
    int rr = It_isRight();

    if(rotation == 2){
        if(state == L){
            if(ff && rr){
                rotation = 0;
                if(num==Size){
                    state = FIN;
                }
            }
            else{
                state = DIR[num];
                num++;
            }
            g_carDirection = CAR_DIR_FW;
        }
        else g_carDirection = CAR_DIR_LF;
    }
    else if(state == R){
        if(ff && ll){
            rotation = 0;
            if(num==Size){
                state = FIN;
            }
        }
        else{
            state = DIR[num];
            num++;
        }
        g_carDirection = CAR_DIR_FW;
    }
    else g_carDirection = CAR_DIR_RF;
}

else if(rotation == 3){
    rotation = 0;
    if(num==Size){
        state = FIN;
    }
    else{
        state = DIR[num];
        num++;
    }
    g_carDirection = CAR_DIR_FW;
}

else{
    if(!ff) g_carDirection = CAR_DIR_ST;
    else if(ff && rr && ll) g_carDirection = CAR_DIR_FW;
    else if(!ll) g_carDirection = CAR_DIR_RF;
    else if(!rr) g_carDirection = CAR_DIR_LF;
    else g_carDirection = CAR_DIR_FW;
}
}
```

## 아두이노 최종 코드

```
void Parse(){
    String response = "";

    if (bluetooth.available()) { //서버에서 오는게 없으면 안들어가짐
        long int time=millis(); //현재 시간 설정
        int timeout = 3000; //ms
        while((time+timeout)>millis()) {
            while(bluetooth.available()) {
                char c = bluetooth.read();
                response+=c;
            }
        }
        rotation = 1;
    }

    if (Serial.available()) {
        bluetooth.write(Serial.read());
    }

    num = 1;

    /////JSON 파싱 시작/////
    String seperator="@";
    int startindex = response.indexOf(seperator);

    if (startindex != -1)
    {
        String json_raw_data = response.substring(startindex + 1);
        //@바로 앞 인덱스
        StaticJsonBuffer<2500> jsonBuffer;
        JsonObject& json_data =
        jsonBuffer.parseObject(json_raw_data);

        String command=json_data["Command"]; //명령 배열
        문자열로 가져옴

        int a = 0;

        for(int i=0; i<command.length(); i++){
            if(command[i] == 'A'){
                char str = command[i+1];
                if(str=='F') DIR[a] = F;
                else if(str=='L') DIR[a] = L;
                else if(str == 'R') DIR[a] = R;
                else{
                    //위에서 안 걸러지면 마지막 JSON파일의
                    마지막 문자열
                    DIR[a] = FIN;
                    END = 1;
                    break;
                }
                Serial.println(DIR[a]);
                a++;
            }
        }
    }
}
```

## 아두이노 최종 코드

```
//첫번째 상태대로 방향 잡음
switch(DIR[0]){
  case FIN:
    if(END == 1) rotation = 4;
    break;
  case F:
    rotation = 0;
    //첫상태 방향 잡았으니 다음 노드에선 두번째 방향으로 가야함
    state = DIR[num];
    num ++;
    break;
  case L:
    car_left();
    delay(375);
    state = DIR[0];
    rotation = 2;
    break;
  case R:
    car_right();
    delay(375);
    state = DIR[0];
    rotation = 2;
    break;
}
/////JSON 파싱 끝/////
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  bluetooth.begin(9600);
  delay(500); //그냥 차 놓을시간 번거
  init_car_controller_board();
}
```

## 아두이노 최종 코드

```
void loop() {
    // put your main code here, to run repeatedly:
    if((END ==0 && DIR[0] == FIN) || (END == 0 && (num
    == Size))) Parse();

    else if(rotation==1){
        switch(state){
            case FIN:
                if(END == 0){ //마지막 json 파일은 아니었는데 파싱은
                끝났음
                    Parse(); //다시 파싱
                    rotation = 0;
                }
                else{ //마지막 json파일이었음
                    car_finish();
                    rotation = 4; //주행종료
                }
                break;
            case F:
                rotation = 3; //회전 필요없고 상태 업데이트 필요
                break;
            case L:
                car_left();
                delay(375);
                rotation = 2;
                break;
            case R:
                car_right();
                delay(375);
                rotation = 2;
                break;
        }
    }

    else if(rotation != 4){
        lt_mode_update();
        car_update();
    }

    else if(rotation == 4){
        if(END == 1) {} //마지막 파스파일 끝까지 주행함. 사실상
        종료
        else if(END == 0) Parse();
    }
}
```

## b. ADOxx 최종 코드

### AdoScript Source

```
CC "CoreUI" MODEL_SELECT_BOX #읽어들일 모델들을 선택
oktext:"Ok" boxtext:"Please select the execution model:"
multi-sel title:"Execution model select box" # without 2nd "_",
hehe...
modeltype:"Automobile"
extra:{
#      CHECKBOX "Automatically arranged processes"
checked:1 result-var:bAutoArrangProcesses
#      CHECKBOX "Use default zoom factor (default:
x=2,y=1)" checked:1 result-var:bZoomFactorOption
}

IF (endbutton = "ok") #ok 버튼을 누르면 모델을 로드함
{
  FOR modelid in:(modelids)
  {
    CC "Core" LOAD_MODEL modelid:(VAL modelid)
  }
}

#json 데이터 포맷
# state: state의 개수
# state_type
# 그리고 타입에 맞는 attr을 state_attr 나열...

SET ip:("http://192.168.1.21:10000") #서버의 ip주소와 포트를 넣은
변수
SET max_couter_limit:(5) #보내는 명령어의 최대 길이
SET _m:(array(201,201)) #필드 전체 데이터
FOR i from:0 to:200 {
  SET j:0
  FOR j from:0 to:200
  {
    SET _m[i,j]:0
  }
}
}
```

## AdoScript Source

```

FOR modelid in:(modelids) #모든 모델을 읽어들이기 때까지 반복함
{

    SETL nModelID:(VAL modelid) #Model id를 nModelID 변수에
    저장
    CC "Core" GET_MODEL_BASENAME modelid:(nModelID)
    #모델의 이름을 불러옴
    SET state:(1)
    CC "Core" GET_ALL_OBJS_OF_CLASSNAME
    modelid:(nModelID) classname:("Start") # 클래스 이름에 해당하는
    objid를 불러옴
    SET nObjId:(VAL objids)
    CC "Core" GET_ATTR_VAL objid:(nObjId) attrname:("IP")
    #start node의 "IP"란 속성에 동작시킬 아두이노에 대한 정보가
    저장되어 있음
    #"IP"속성의 값을 불러들여 중계용 서버에 전송하기 위해 함수를
    호출한 것임
    SETL basenane:(val) #모델의 이름을 저장

    SET t:(0)
    SETL json:("@{ W"commandsW" : [") #json 형식으로 만들기
    위해 변수 생성, 앞의 @는 아두이노 상에서 파싱처리를 위해 임의로
    넣은 토큰임
    WHILE (1) #모델 안에 있는 모든 객체의 정보를 읽어들이기 후
    종료됨
    {
        SETL sState:(STR state) #상태 번호 저장
        # CC "AdoScript" INFOBOX ()
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Next") #다음 객체 번호가 저장된 속성 값 불러오기
        SET nObjId:(VAL val) # 다음 객체 번호 저장
        CC "Core" GET_CLASS_ID objid:(nObjId) #객체 번호의
        classid 불러오기
        CC "Core" GET_CLASS_NAME classid:(classid) #classid를
        기반으로 class이름 불러오기
        SETL sClassName:(classname) #class 이름 저장, class
        이름에 따라 아래의 분기가 실행됨

        IF (sClassName = "End") #종료 클래스일 경우
        {
            SET state:(state-1)
            SETL sState:(STR state)
            # CC "AdoScript" INFOBOX ("The end of Class")
            BREAK
        }
    }
}

```

## AdoScript Source

```

        ELSIF (sClassName = "Automobile_Motor") #자동차 모터일
경우
    {
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Quick_Start") #속성값을 불러옴
        SETL qs:(val) #값 저장
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Execution_Time(ms)") #속성값을 불러옴
        SETL time:(STR val) #값 저장
        SETL type:("car") #유형 저장
        #
        #
        # SETL json:(json+ "W"+sState + "_" + "tW"+":W" +
type + "W",
        # + "W"+sState + "_" + "qsW"+":W" + qs + "W",)
#json 형식으로 text 저장

        SETL json:(json+"{WcomW" : "+qs+",Wtime:W" :
"+time+"},")
    }

    ELSIF (sClassName = "path")
    {
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Str")
        SETL path:(val)

        CC "AdoScript" INFOBOX (path)
        FOR word in: (path)
        {
            CC "AdoScript" INFOBOX (word)
        }
    }
    ELSIF (sClassName = "BFS")
    {
        CC "Core" GET_ATTR_VAL objid:(nObjId)
        attrname:("Str")
        SETL path:(val)
    }

```



## AdoScript Source

```

ELSEIF (sClassName = "Move")
{
    CC "Core" GET_ATTR_VAL objid:(nObjId)
    attrname:("Path")
    SETL path:(val)
    CC "Core" GET_ATTR_VAL objid:(nObjId)
    attrname:("Compass")
    SETL compass:(val)

    CC "AdoScript" INFOBOX (path)
    SETL counter:(0)
    SETL json_start:("@{W"CommandW":[")
    #SETL json_start_2:(",")
    SETL json_end:("]}")
    SETL json:("")
    SETL tempmap:({"Content-Type": "text/plain"})
    SETL compass:(0)

    SETL path:(path+"-1")

    FOR comm in: (path)
    {
        SETL str_comm:("")

        IF (comm != "")
        {
            SETL str_comm:("")

            IF (compass=0) #North
            {
                IF ((VAL comm)=0)
                {
                    SETL str_comm:("AF")
                    SETL compass:(0)
                }
                ELSEIF ((VAL comm)=1)
                {
                    SETL str_comm:("AB")
                    SETL compass:(2)
                }
                ELSEIF ((VAL comm)=2)
                {
                    SETL str_comm:("AL")
                    SETL compass:(1)
                }
                ELSEIF ((VAL comm)=3)
                {
                    SETL str_comm:("AR")
                    SETL compass:(3)
                }
            }
        }
    }
}

```

## AdoScript Source

```

ELSIF (compass=1) #West
{
    IF ((VAL comm)=0)
    {
        SETL str_comm:("AR")
        SETL compass:(0)
    }
    ELSIF ((VAL comm)=1)
    {
        SETL str_comm:("AL")
        SETL compass:(2)
    }
    ELSIF ((VAL comm)=2)
    {
        SETL str_comm:("AF")
        SETL compass:(1)
    }
    ELSIF ((VAL comm)=3)
    {
        SETL str_comm:("AB")
        SETL compass:(3)
    }
}
ELSIF (compass=2) #South
{
    IF ((VAL comm)=0)
    {
        SETL str_comm:("AB")
        SETL compass:(0)
    }
    ELSIF ((VAL comm)=1)
    {
        SETL str_comm:("AF")
        SETL compass:(2)
    }
    ELSIF ((VAL comm)=2)
    {
        SETL str_comm:("AR")
        SETL compass:(1)
    }
    ELSIF ((VAL comm)=3)
    {
        SETL str_comm:("AL")
        SETL compass:(3)
    }
}
    
```

## AdoScript Source

```

ELSIF (compass=3) #East
{
    IF ((VAL comm)=0)
    {
        SETL str_comm:("AL")
        SETL compass:(0)
    }
    ELSIF ((VAL comm)=1)
    {
        SETL str_comm:("AR")
        SETL compass:(2)
    }
    ELSIF ((VAL comm)=2)
    {
        SETL str_comm:("AB")
        SETL compass:(1)
    }
    ELSIF ((VAL comm)=3)
    {
        SETL str_comm:("AF")
        SETL compass:(3)
    }
}
IF ((VAL comm)=-1)
{
    SETL str_comm:("AZ")
}
IF (counter = max_couter_limit)
{

    SETL json:(json_start+json+json_end)
    SETL counter:(0)
    #HTTP_SEND_REQUEST
    (ip+"? id=adoxx&name="+basename) str_method:("GET")
    map_reqheaders:(tempmap) str_reqbody:(json)
    val_respcode:respstat map_respheaders:respheaders
    str_respbody:respbody
    #=SLEEP ms:(500)
    #CC "AdoScript" INFOBOX (json)
    SETL json:("")
}
SETL json:(json + "W"+str_comm+"W",)
SETL counter:(counter+1)
}
}

```

## AdoScript Source

```
#SENDTOSERVER linetrace:commands

SETL json:(json_start+json+json_end)
#HTTP_SEND_REQUEST
(ip+"? id=adoxx&name="+basename) str_method:("GET")
map_reqheaders:(tempmap) str_reqbody:(json)
val_respcode:respstat map_respheaders:respheaders
str_resbody:resbody
#SLEEP ms:(500)
#CC "AdoScript" INFOBOX (json)
CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("Next")
SET next_node:(val)
CC "Core" SET_ATTR_VAL objid:(VAL next_node)
attrname:("Compass") val:(compass)
}
ELSIF (sClassName = "BFS_test_case")
{
SETL get:0
BFS start_x:4 start_y:4 end_x:4 end_y:0 result:get
CC "AdoScript" INFOBOX (get)
}
ELSIF (sClassName = "BFS_Start")
{
#CC "AdoScript" INFOBOX ("!")
CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("Start_Pos_X")
SET start_pos_x:(val)
CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("Start_Pos_Y")
SET start_pos_y:(val)
CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("End_Pos_X")
SET end_pos_x:(val)
CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("End_Pos_Y")
SET end_pos_y:(val)

SETL res:""
BFS start_x:(start_pos_x) start_y:(start_pos_y)
end_x:(end_pos_x) end_y:(end_pos_y) result:res
#CC "AdoScript" INFOBOX (res)
#CC "AdoScript" INFOBOX (nObjId)

CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("Next")
SET next_node:(val)
CC "Core" SET_ATTR_VAL objid:(VAL next_node)
attrname:("Path") val:(res)

CC "Core" GET_ATTR_VAL objid:(nObjId)
attrname:("Compass")
SET compass:(val)
CC "Core" SET_ATTR_VAL objid:(VAL next_node)
attrname:("Compass") val:(compass)
#CC "Core" SET_ATTR_VAL objid:(nObjId)
attrname:("Next") val:("path-16600")
}
}
```

## AdoScript Source

```
ELSIF (sClassName = "Field_Data_Setting")
{
    CC "AdoScript" INFOBOX ("Map Set")
    SET _m[0,0]:1
    SET _m[1,0]:1
    SET _m[2,0]:1
    SET _m[3,0]:1
    SET _m[4,0]:0
    SET _m[5,0]:1

    SET _m[0,1]:1
    SET _m[1,1]:1
    SET _m[2,1]:0
    SET _m[3,1]:1
    SET _m[4,1]:1
    SET _m[5,1]:1

    SET _m[0,2]:1
    SET _m[1,2]:0
    SET _m[2,2]:1
    SET _m[3,2]:1
    SET _m[4,2]:1
    SET _m[5,2]:0

    SET _m[0,3]:1
    SET _m[1,3]:1
    SET _m[2,3]:1
    SET _m[3,3]:0
    SET _m[4,3]:1
    SET _m[5,3]:1

    SET _m[0,4]:0
    SET _m[1,4]:1
    SET _m[2,4]:1
    SET _m[3,4]:1
    SET _m[4,4]:1
    SET _m[5,4]:1

    #SET _m[1,0]:1
    #SET _m[2,0]:1
    #SET _m[3,0]:1
    #SET _m[4,0]:1

    #SET _m[2,1]:1

    #SET _m[2,2]:1
```

## AdoScript Source

```
#SET _m[0,3]:1
#SET _m[1,3]:1
#SET _m[2,3]:1
#SET _m[3,3]:1
#SET _m[4,3]:1

#SET _m[0,4]:1
#SET _m[4,4]:1

}

SET state:(state+1)

IF (t=10)
{
    CC "AdoScript" INFOBOX ("t is "+STR t)
    BREAK
}
SET t:(t+1)
}

#SETL tempmap:({"Content-Type": "text/plain"}) #http
request를 보내기 위한 초기 설정
#SETL json:(json+"}") #json 변수 저장
#HTTP_SEND_REQUEST (ip+"? id=adoxx&name="+basename)
str_method:("GET") map_reqheaders:(tempmap)
str_reqbody:(json) val_respcode:respstat
map_respheaders:respheaders str_respbody:respbody
#HTTP_SEND_REQUEST을 실행하여 중계용 서버로 보냄
#SLEEP ms:(500) # 서버에서 데이터를 완전히 받을 때까지 대기
#CC "AdoScript" INFOBOX
(ip+"? id=adoxx&name="+basename)

}
CC "AdoScript" INFOBOX ("The end of AdoScript")
# CC "Modeling" GET_ACT_MODEL #현재 활성화된 모델의 id를
불러옴
# SETL nModelID:(modelid)
# SET ecode:0

PROCEDURE SLEEP ms:integer
{
    SETL time:(800*ms)
    FOR i from:1 to:(time) {}
}
```

## AdoScript Source

```

PROCEDURE BFS start_x:integer start_y:integer end_x:integer
end_y:integer result:reference
{
    SET rear:0
    SET front:0

    #SET _m:(array(4,5))
    SET visited:(array(201,201))
    SET queue:(array(2001))

    SET i:0
    FOR i from:0 to:200 {
        SET j:0
        FOR j from:0 to:200
        {
            SET visited[i,j]:1
        }
    }

    #CC "AdoScript" INFOBOX (_m)
    SET i:0
    FOR i from:0 to:2000
    {
        SET queue[i]:-1
    }

    SET x: (start_x)
    SET y: (start_y)
    SET visited[x,y]: 0
    SET queue[rear]: (x)
    SET rear:(rear+1)
    SET queue[rear]: (y)
    SET rear:(rear+1)
    SET queue[rear]: (-1)
    SET rear:(rear+1)

    #초기화
    WHILE (front < rear) {
        SET i: (queue[front])
        SET front:(front+1)
        SET j: (queue[front])
        SET front: (front+2)

        IF ((i=end_x) AND (j=end_y)) # 도착지에 도착하면 종료
        {
            CC "AdoScript" INFOBOX ((STR queue))
            BREAK
        }
    }
}
    
```

## AdoScript Source

```

SET _j: (i+1)
SET _j:(j+1)

SET i_: (i-1)
SET j_: (j-1)
#CC "AdoScript" INFOBOX ("i : "+(STR i)+" j : "+(STR j)+"
_i : "+(STR _i)+" j : "+(STR j)+" i_ : "+(STR i_)+" j_ : "+(STR
j_))

IF (((_i)>=0) AND ((j)>=0))
{
    IF ((_m[_i,j] = 1) AND (visited[_i,j] = 1))
    {
        SET i:(_i)
        SET visited[i,j]: 0
        #CC "AdoScript" INFOBOX ("i"+ (STR i)+" j" +(STR
j))

        SET queue[rear]: (i)
        SET rear: (rear+1)
        SET queue[rear]: (j)
        SET rear:(rear+1)
        SET queue[rear]: 3
        SET rear:(rear+1)
    }
}
IF (((i)>=0) AND ((_j)>=0))
{
    IF((_m[i,_j] = 1) AND (visited[i,_j] = 1))
    {
        SET j:(_j)
        #CC "AdoScript" INFOBOX ("i"+ (STR i)+" j" +(STR
j))

        SET visited[i,j]: 0
        SET queue[rear]: (i)
        SET rear: (rear+1)
        SET queue[rear]: (j)
        SET rear:(rear+1)
        SET queue[rear]: 1
        SET rear:(rear+1)
    }
}
}

```



## AdoScript Source

```

IF (((i_)>=0) AND ((j)>=0))
{
  IF((_m[i_,j]= 1) AND (visited[i_,j]= 1))
  {
    SET i:(i_)
    #CC "AdoScript" INFOBOX ("i"+ (STR i)+" j" +(STR
j))

    SET visited[i,j]: 0
    SET queue[rear]: (i)
    SET rear: (rear+1)
    SET queue[rear]: (j)
    SET rear:(rear+1)
    SET queue[rear]: 2
    SET rear:(rear+1)
  }
}

IF (((j_)>=0) AND ((i)>=0))
{
  IF((_m[i,j_]= 1) AND (visited[i,j_]= 1))
  {
    SET j:(j_)
    #CC "AdoScript" INFOBOX ("i"+ (STR i)+" j" +(STR
j))

    SET visited[i,j]: 0
    SET queue[rear]: (i)
    SET rear: (rear+1)
    SET queue[rear]: (j)
    SET rear:(rear+1)
    SET queue[rear]: 0
    SET rear:(rear+1)
  }
}
CC "AdoScript" INFOBOX ((STR queue))
#ELSE
#{

#경로 없음!
#SET result: (-1)

#}
IF ((i=(end_x)) AND (j=(end_y)))
{
  SET result:(queue)
}

}

```

## AdoScript Source

```

SET route:(array(2001))
FOR i from:0 to:2000
{
    SET route[i]:-2
}
SET route_count:0

SET rear:(rear-1)
#루트를 찾아가는 함수
SET par_x:(end_x)
SET par_y:(end_y)

WHILE (rear >= 3)
{
    SET poi:(queue[rear])
    SET rear:(rear-1)
    SET pos_y:(queue[rear])
    SET rear:(rear-1)
    SET pos_x:(queue[rear])
    SET rear:(rear-1)

    IF ((par_x = pos_x) AND (par_y = pos_y))
    {
        SET route[route_count]:(pos_x)
        SET route_count:(route_count+1)
        SET route[route_count]:(pos_y)
        SET route_count:(route_count+1)
        SET route[route_count]:(poi)
        SET route_count:(route_count+1)

        IF (poi = 0)
        {
            SET par_x:(pos_x)
            SET par_y:(pos_y + 1)
        }
        ELSIF (poi = 1)
        {
            SET par_x:(pos_x)
            SET par_y:(pos_y - 1)
        }
    }
}
    
```

## AdoScript Source

```

        ELSIF (poi = 2)
        {
            SET par_x:(pos_x + 1 )
            SET par_y:(pos_y)
        }
        ELSIF (poi = 3)
        {
            SET par_x:(pos_x - 1)
            SET par_y:(pos_y)

        }
        ELSIF (poi = -1)
        {
            BREAK
        }

    }
}

#CC "AdoScript" INFOBOX ("route_count : "+ (STR
route_count))

#CC "AdoScript" INFOBOX ("route : "+ (STR route))
SET path_count: 0
SET route_path:""
#route_count: 16
SET route_count: (route_count-1)
WHILE (route_count>0)
{
    SET route_path:(route_path+(STR (route[route_count]))+"
")
    SET route_count: (route_count-3)

    ## 방향 -> x-> y

}

#CC "AdoScript" INFOBOX ("    path    :" +(STR
path_count))

#CC "AdoScript" INFOBOX ("    path    :" + route_path)

SET result:(route_path)
}
    
```

### 3. 종합

#### a. 결론 및 평가

- 학부 수업에서 소프트웨어를 개발하는 프로젝트는 많지만, 하드웨어를 직접 제어하는 경우는 드물었다. 그런 면에서 이번 프로젝트는 쉽게 경험할 수 없는 개발 프로젝트였다는 점이 가치가 있다고 생각한다. 그리고 소프트웨어 개발 공정 기법을 사용하여 프로젝트팀 내에서 역할을 분리하고 팀마다 그 결합도를 최대한 낮추고 응집도를 높이기에 좋았다. 아두이노의 관리 및 조작과 ADOXX를 통한 서버의 연산을 각자 개발 및 검증한 후, 마지막에 합치는 상황에서 큰 무리없이 하나의 구현체가 완성될 수 있었다는 점이 소프트웨어 공정 기법을 조금이나마 사용하여 개발을 마칠 수 있었던 것 같아서 뿌듯하다.
- 처음에는 화재나 수재 등의 재난 상황에서 응급 환자나 재해 복구 등을 위한 무인 자동차 운영 시스템을 계획하였으나, 프로젝트 진행 중 완전한 도로의 상태나 올곧은 경로 등 여러 면에서 현실적인 문제를 다루지 못한다고 생각하였다. 그래서 현재 제작된 도로 주행판이나 자동차의 안정적인 주행 등이 우리 도시의 모습을 담을 수 있을 것이라 생각하였고, 카풀 서비스로 프로젝트 주제를 변경했고 만족스러운 결과를 낼 수 있었다.
- 하드웨어를 사용한 프로젝트라 직접 눈으로 확인할 수 있어서 좋았고 그것이 현실적인 문제를 직접 다루고 있다는 느낌이 들어서 프로젝트에 대한 자긍심이 더 고취되었다.

## b. 아쉬운 점

- 서버와 양방향으로 통신하는 실시간성을 띄지 못한 점이 아쉽다고 생각한다. 이는 웹 서버와 아두이노 사이에서 발생하는 네트워크의 속도 등의 이유로 해결하지 못하였다. 양방향 통신이 가능했다면 더 다양한 주제를 고려하거나 추가적인 기능을 구상할 수 있었을 것이라 생각한다. 하지만 시뮬레이션의 형태로 진행되는 이 프로젝트에서는 필드(도로와 교통체증 등)의 상황과 그 외 환경들을 모두 알고 있는 상태에서 시연되므로, 사실 서버와의 통신이 단방향인 것과 양방향인 것의 차이는 없을 것이라는 점에서 아쉬움을 덜 수 있었다.
- 아두이노 키트의 H/W가 생각보다 연산 속도가 빠르지 않아서 아쉬웠다. Delay를 위한 Sleep 함수가 존재했으나 일부 경우, 반복문 등의 많은 연산이 빠른 시간 안에 처리되어야 하는 상황 등에서 속도를 내지 못해 딜레이가 조금씩 발생하기도 하였다. 이는 아두이노가 레지스터를 직접 제어하는 속도를 나타내는 주파수에 따라 다르다는 것을 알아내었고, digitalWrite 등의 함수에서 다른 함수보다 많은 클럭 수가 걸리는 것이 아쉬웠다.