**소프트웨어공학**

**최종 보고서**

**Smart CTS**

**Team CARM**

**2019.12.23**

**목차**

1. **팀 소개**

1-1. 팀 명

1-2. 팀 구성원 및 역할

1. **프로젝트 개요**

2-1. 프로젝트 목표

2-2. 프로젝트 개발 방향

1. **프로젝트 개발 계획**

3-1 개발 모델

3-2. 프로젝트 개발 환경

1) 성능 요구

2) 하드웨어 요구

1. **요구사항**

4-1. 목적

4-2. 시나리오

4-3. 요구사항 표

1. **분석 모델**

5-1. Data Model

5-2. Scenario-Based Model

1) Use-Case Diagram

2) Activity Diagram

5-3. Flow-Oriented Model

5-4. Object-Oriented Model

5-5. Class-Based Model

5-6. Behavioral Model

1) Sequence Diagram

2) State Diagram

5-7. CRC Model

5-8. Map Model

1. **설계 모델**

6-1. Layered Architecture

6-2. Architectural Context Diagram

6-3. Main / Sub Program Architecture

6-4. Component Architectural Structure

1. **CPS 제원 설명 및 회로도**

7-1. 제원 설명

7-2. 회로도

1. **코드**

8-1. Arduino Source Code

8-2. AdoScript

8-3. ADOxx Modeling

1. **유효성 테스트**

9-1. 시험 계획서

9-2. 테스트 결과

1. **회의록**
2. **팀 소개**

1-1. 팀 명

팀 명인 CARM은 Smart CTS를 나타내는 자동차의 CAR와 기계 팔의 ARM이 결합된 합성어로, Smart CTS의 주제를 직관적으로 나타낸다.

1-2. 팀 구성원 및 역할

1) 최성봉: 팀장, Adoxx Script 작성, Robot Arm 개발

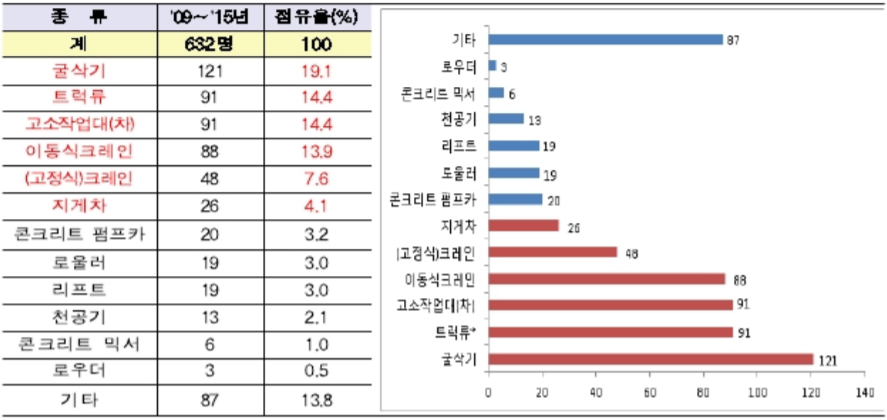
2) 이성주: 문서작성, Automobile 개발, 서기

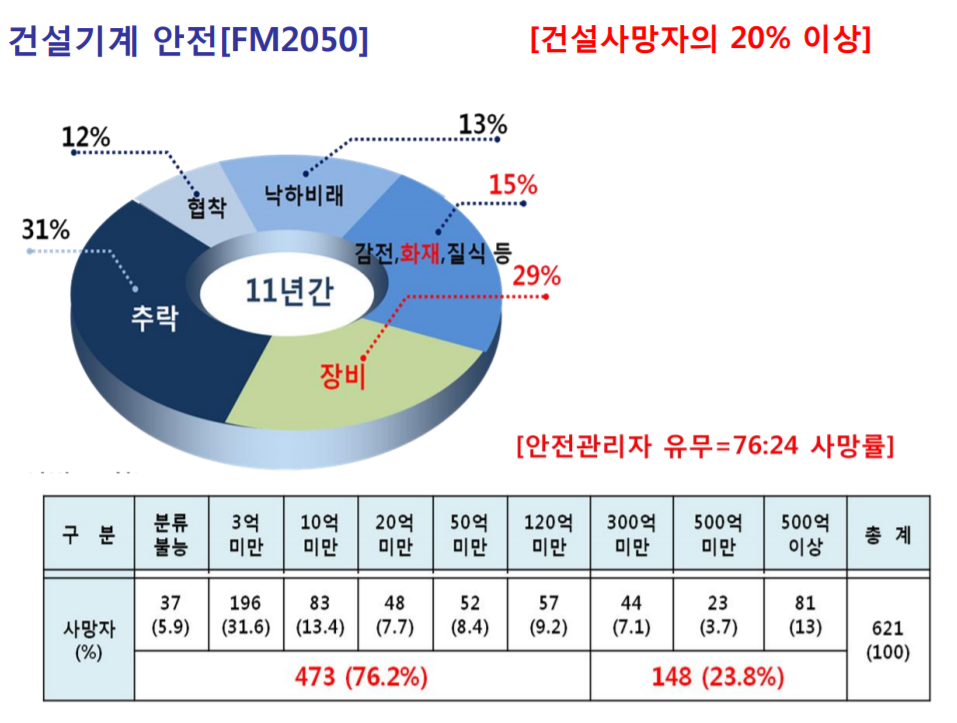
3) 이동규: 테스터, Automobile 개발

4) 가인온투: Server Parsing, Robot Arm 개발

1. **프로젝트 개요**

2-1. 프로젝트 목표





[출처 : 한국건설안전협회]

-동기

통계적인 지표로 확인 할 수 있듯이 많은 공사현장에서 운송장비로 인한 안전사고가 빈번하게 발생하고 있다. 안타깝게도 현재 이 부분에 대한 최선의 대안은 스스로 갖는 안전의식이 전부이다.

때문에 이런 사고들을 극복하고자 하는 의식에서 파생되 사람이 직접 운송장비들을 사용하는 것이 아닌 명령만 내리면 기계가 빠르고 정확하게 수행 하는 시스템을 개발.

-기대사항

운송장비로 인한 안전사고 발생률을 감축시킬 수 있으며 좀더 빠르고 정확한 물류 순환 속도를 개선할 수 있다.또한 안정적으로 정착이 된다면 사회적인 측면에서 새로운 패러다임을 제시 할 수 있게 된다.

* 1. 프로젝트 개발 계획 시 우선순위

Time

Satisfying requirements

Damage

1)정확한 위치

-장비가 요구사항에 맞는 목적지에 잘 찾아갈 수 있는가?

2)물품의 상태

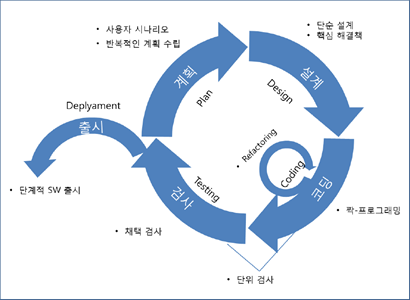
-물품이 파손되지 않고 안전하게 배송될 수 있는가?

3)운송 시간

-얼마나 빠른 시간 내에 운송이 될 수 있는가?

**3.프로젝트 개발 계획**

3-1. 개발 모델



* 1. eXtreme Programming (Agile Model)

- 짧은 시간 내에 높은 완성도의 프로그램 개발 가능

- 즉각적 피드백

- 품질 위주의 모델링

- 소규모 인원으로 개발 가능

3-2. 프로젝트 개발 환경

1) 성능 요구

- ADOxx 프로그램을 실행할 수 있는 Client, Server 성능

- ADOxx Client에서 ADOxx Server 통신을 할 수 있는 네트워크 속도

- ADOxx Server에서 Arduino 통신 할 수 있는 블루투스속도

- code 컴파일이 업로드 된 Arduino의 명령어 처리가 가능한 CPU속도

2) 하드웨어 요구

- ADOxx Client

- ADOxx Server

- Arduino Due

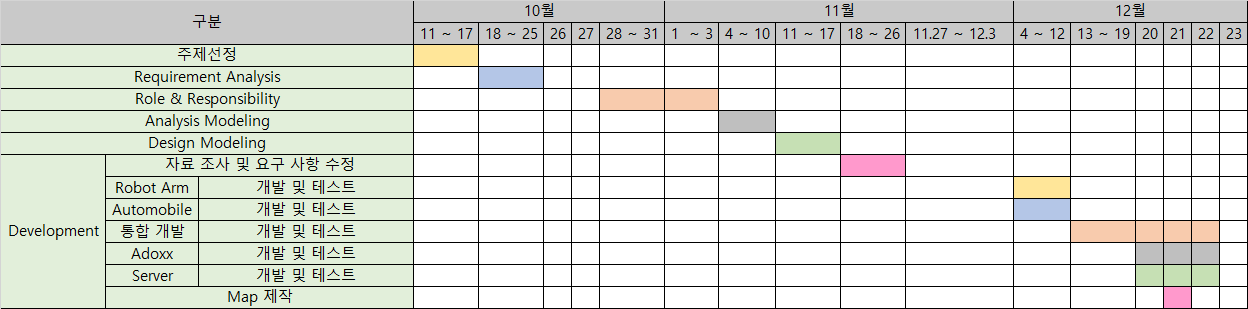
- Robot Arm

- Automobile

- Line tracer

- Bluetooth

3-3. 개발 일정



1. **요구사항**

4-1. 시나리오

1) 담당자는 원하는 장비를 ADoxx요청한다.

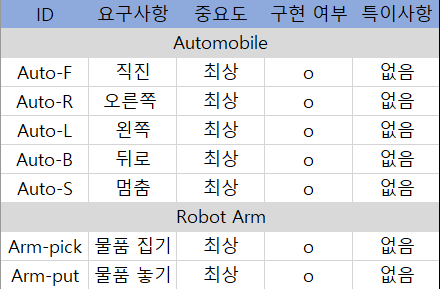
2) ADoxx는 전달받은 요청을 Web Server에게 전송한다.

3) Web Server는 전달받은 요청을 parsing 하여 다시 Arduino에게 전송한다.

4) Arduino는 전달받은 요청을 각 Module에 전달한다.

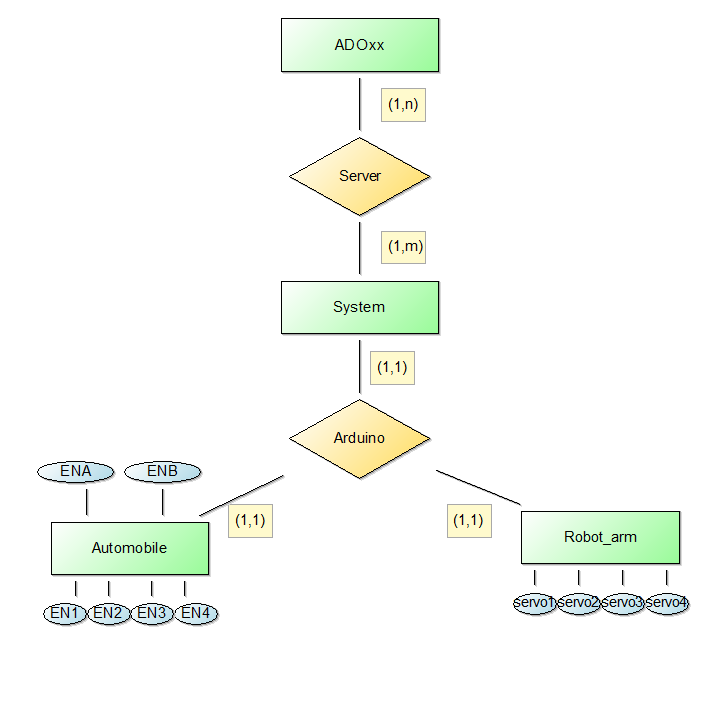
5) 각 Module은 요청을 수행한다.

4-2. 요구사항 표



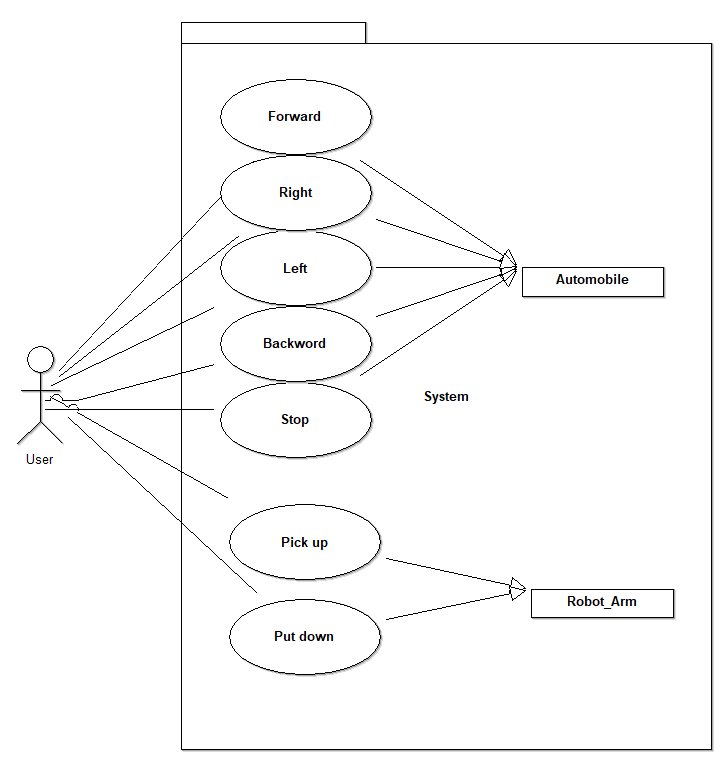
1. **분석 모델링**

5-1. ER - Diagram

ER Diagram은 시스템을 분석하는 단계에서 파악한 엔티티들을 작성하고 연결시킬 관계를 정의함으로써 엔티티들의 사이의 논리적인 관계와 상호 작용을 간단하게 보여주는 모델이다. 그리고 시스템이 어떤 방식으로 구성되어 있는지에 대한 전체적인 뷰를 알아볼 수 있는 역할을 하여 파악하기 쉽게 할 수 있다.

5-2. Scenario-Based Modeling

1) Use Case Diagram



Use-case Diagram은 시스템과 사용자의 상호작용을 다이어그램으로 표현한 것으로 사용자의 관점에서 시스템의 서비스 혹은 기능 및 그와 관련한 외부 요소를 보여주는 것이다.

사용자가 시스템 내부에 있는 기능 중에 어떤 기능을 사용할 수 있는지 나타내며 Use-case Diagram 사용함으로써 고객과 개발자가 요구사항에 대한 의견을 조율할 수 있다.

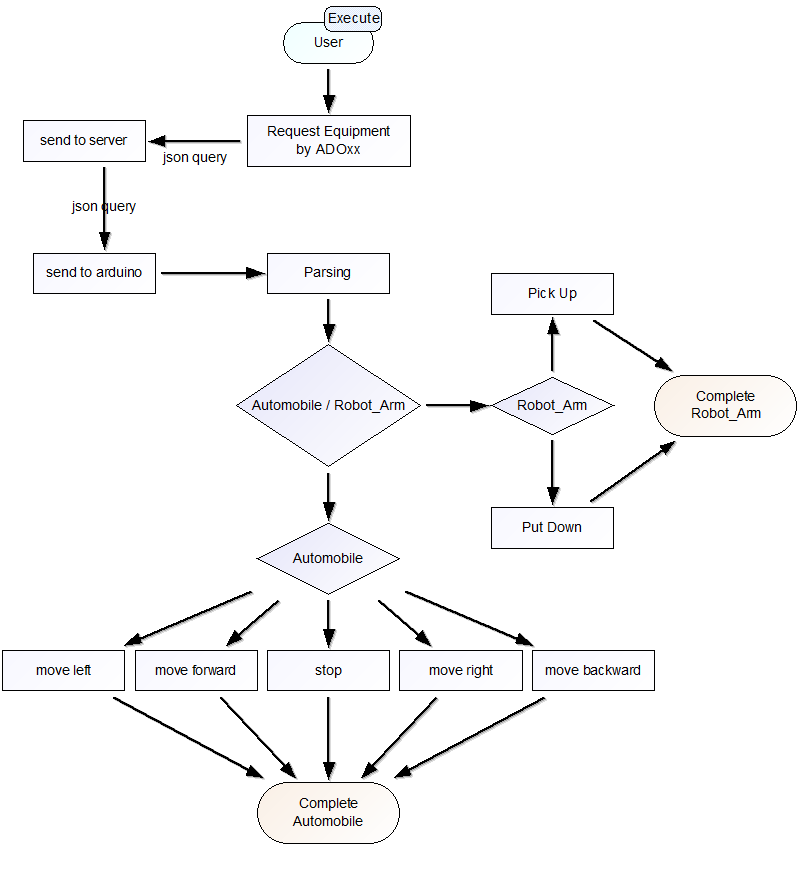
2) Activity Diagram



Activity Diagram은 use-case diagram에서 간단하게 설명한 내용들을 자세하게 보여줌으로써 내용을 보다 자세히 이해할 수 있다.

5-3. Flow-Oriented Modeling

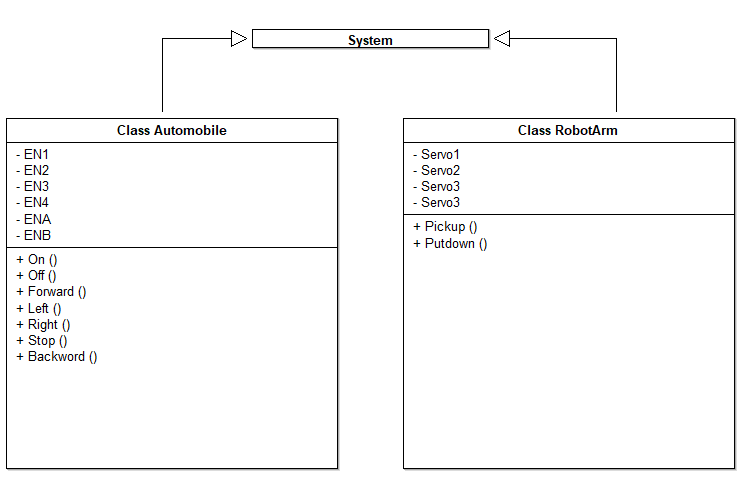
1) Flow Chart Diagram



Flow Chart Diagram은 시스템의 진행 과정을 나타내며 주어진 문제에 대한 솔루션 모델을 보여준다.

5-4. Class-Based Modeling

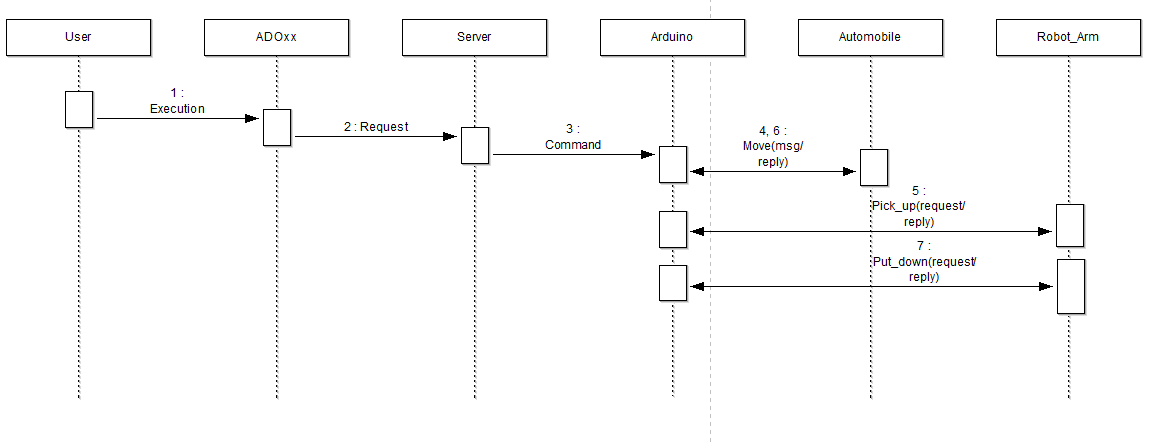
1) Class Diagram



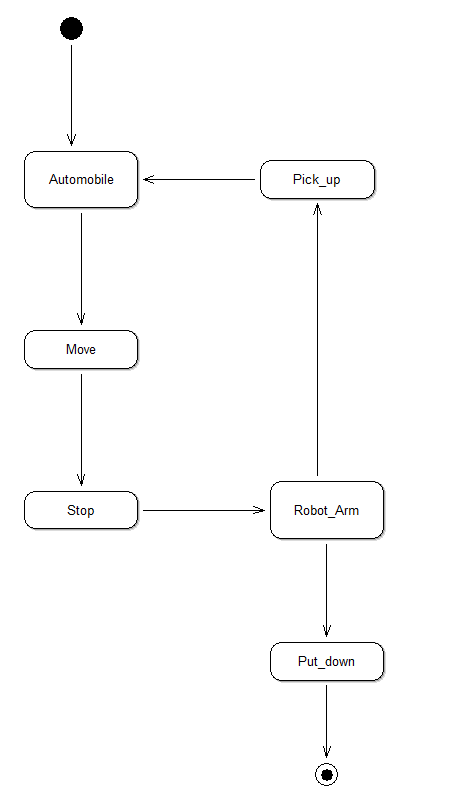
Class Diagram은 실제 소프트웨어의 설계 혹은 구현을 위한 용도로 사용된다. 앞으로 구현 할 혹은 구현된 실제 클래스를 의미하므로 소스코드와의 관계가 매우 깊으며 Class Diagram이 가지는 의미가 모호성을 띄지 않도록 주의해야한다.

5-5. Behavioral Modeling

1) Sequence Diagram

Sequence Diagram은 전체적인 시스템 흐름에 대해 각각의 요소들의 상호작용을 시간순으로 나타낸 Diagram이다. 시스템이 시간의 흐름에 따라 어떻게 동작하는지 쉽게 파악할 수 있으며 결함이 생길 시 빠른시간내에 발견 할 수 있다.

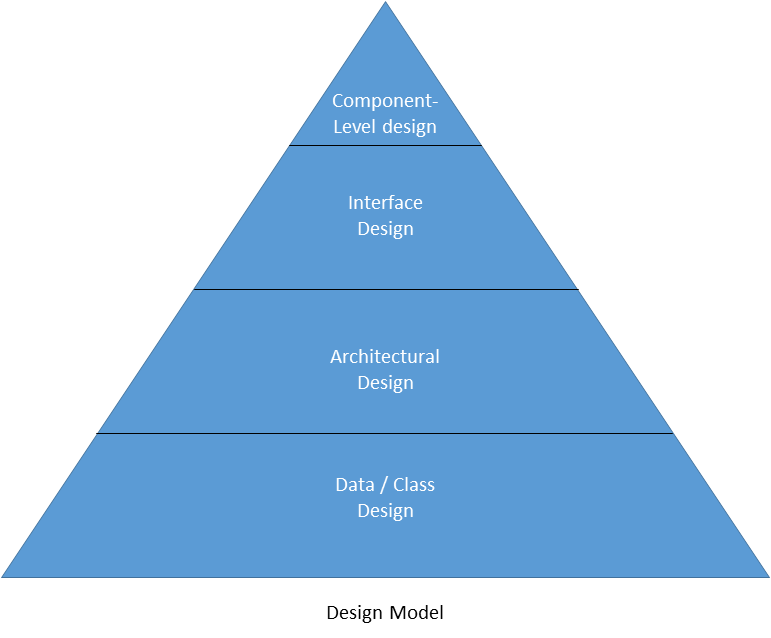
2) State Diagram



State Diagram은 상태와 상태천이를 표현하기 위해 도식화 해서 표현하는 그림을 말한다. 상태는 유한한 상태로 설정하고 추상화 해야 한다.

1. **디자인 모델링**

디자인 모델링은 시스템의 use-case 를 나타내는 그림이나, 시스템의 구현과 소스코드를 도식화 하는 것을 의미한다. 좀 더 사용자의 관점에 가까운 모델링이며 한눈에 각 계층에 해당하는 역할을 파악할 수 있어야 한다.



6-1. Data / Class Design

1) Data Dictionary

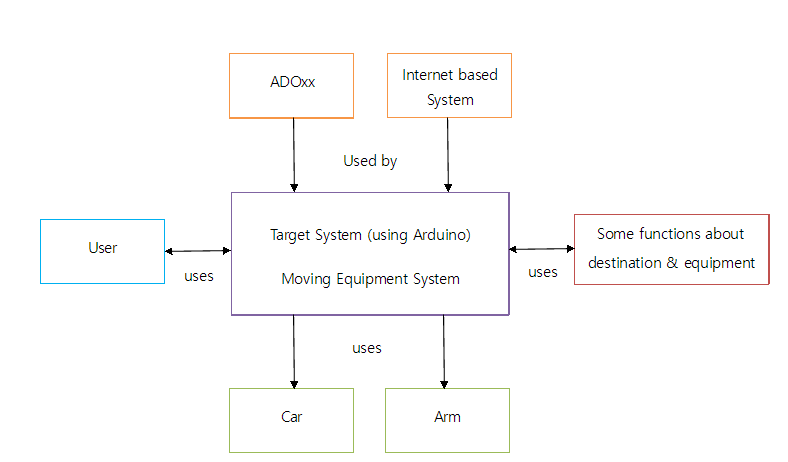


6-2. Architectural Design

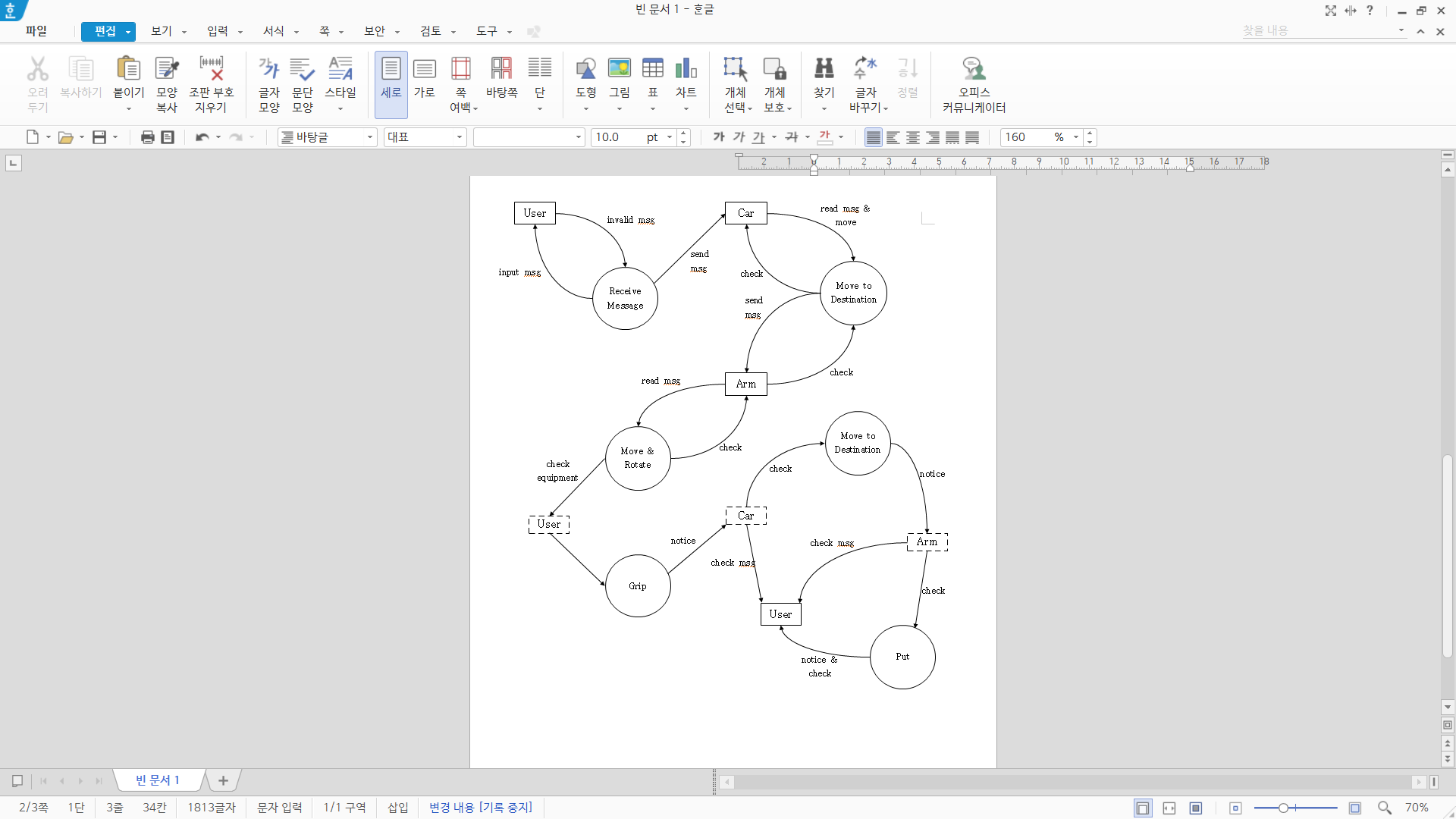
Architectural Design은 컴퓨터 기반 시스템을 구축하는데 필요한 데이터와 프로그램 구성요소의 구조를 나타낸다. Architectural Design은 운영체제가 아닌 표현을 의미하며, 소프트웨어 공학이 설계의 효과를 분석하여 구조적 대안을 고려하고 위험을 줄일 수 있는 구조임을 의미한다.

Architectural Design에서는 소프트웨어 엔지니어가 명시된 요구사항을 충족시키는 데 있어 설계의 효율성을 분석할 수 있다. 그리고 설계 변경을 수행하는 단계에서 아키텍처 대안을 고려하는 것이 상대적으로 수월 하기 때문에 엔지니어에게 도움이 될 수 있다.

특히 Architectural Design이 중요한 이유는 시스템이 어떻게 구조화되어 있고 그 구성요소들이 어떻게 작동하는지에 대해 간결하고 이해하기 쉬운 모델로 표현하기 때문에 개발자들과 이해 관계자 간의 커뮤니케이션이 잘 이루어지도록 도와준다.

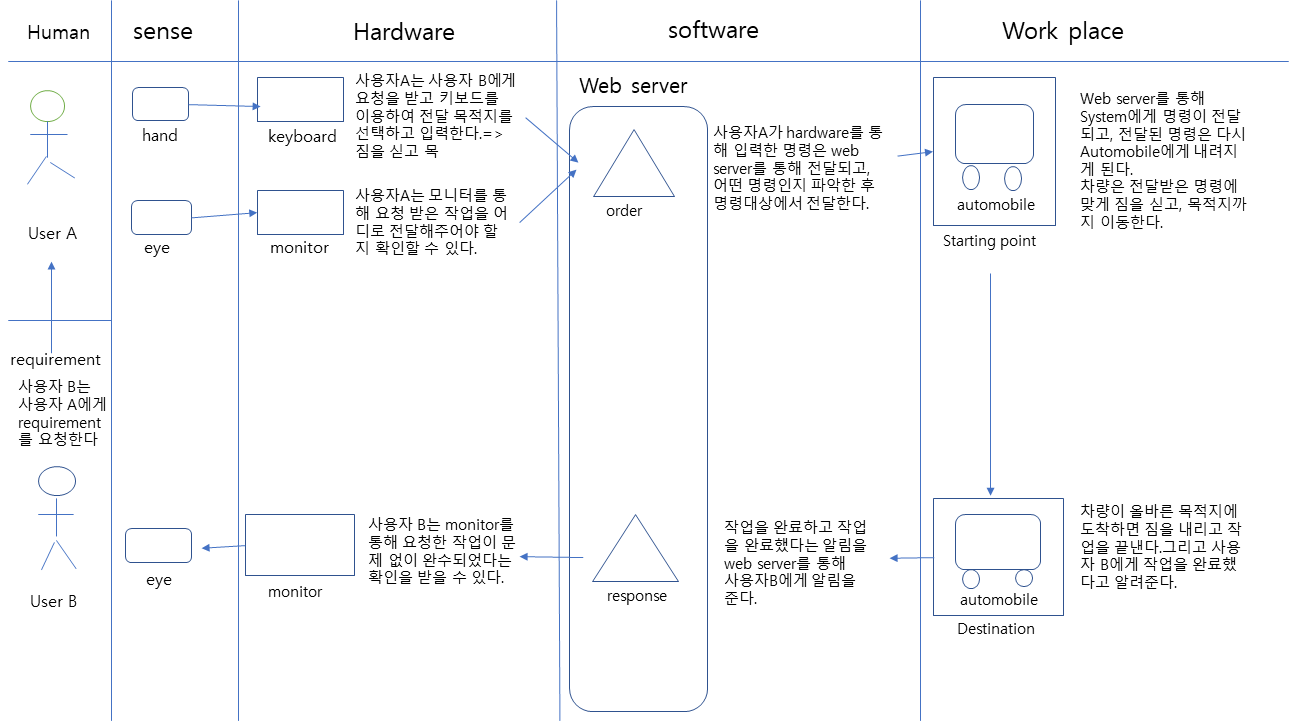
****

* Target System은 공사현장 장비 운송 시스템이다.
* ADOxx를 통해 msg를 송, 수신 한다

****

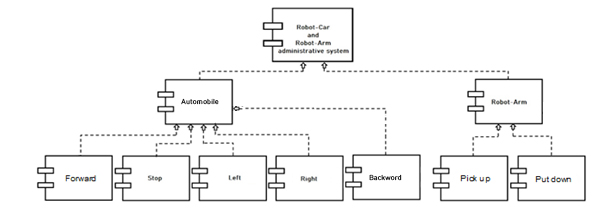
6-3. Interface Design

Interface Design은 일반 사용자들이 소프트웨어를 사용할 때 data 입력이나 동작을 제어하기 위해 사용하는 명령어 또는 기법 절차를 설계하는 것을 말한다. Interface Design에서는 사용자가 소프트웨어와 의사소통을 원활하게 할 수 있도록 하는 것이 주된 목적이다. 때문에 최대한 사용자의 관점에서 접근해야 하며 전문적인 용어의 사용을 줄이고 한눈에 알아보기 쉬운 표나 그림을 이용하여 작성하는 것이 좋다.



6-4. Component Level Design

Component Design은 시스템의 부분별 동작 특성을 고려하고 각 부분 간의 관계를 단계별로 규정해준다.



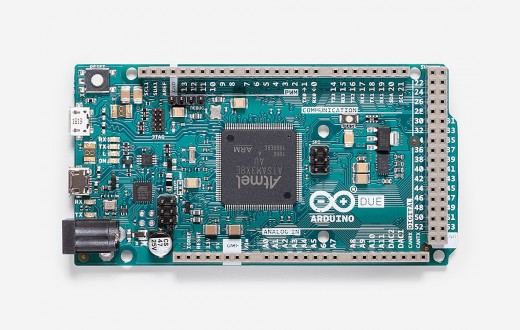
|  |  |  |
| --- | --- | --- |
| 메인 모듈 | 서브 모듈 | 모듈 설명 |
| Smart CTS | Automobile | 자동차 |
| Robot Arm | 기계 팔 |

|  |  |  |
| --- | --- | --- |
| 메인 모듈 | 서브 모듈 | 모듈 설명 |
| Automobile | Forward | 차량이 앞으로 이동함. |
| Left | 차량이 좌회전함. |
| Right | 차량이 우회전함. |
| Stop | 차량이 정지함. |
| Robot Arm | Backward | 차량이 뒤로 이동함. |
| Pick up | Robot Arm이 물건을 집는다. |
| Put down | Robot Arm이 물체를 놓는다. |

1. **CPS 제원 설명 및 회로도**

7-1. 제원 설명

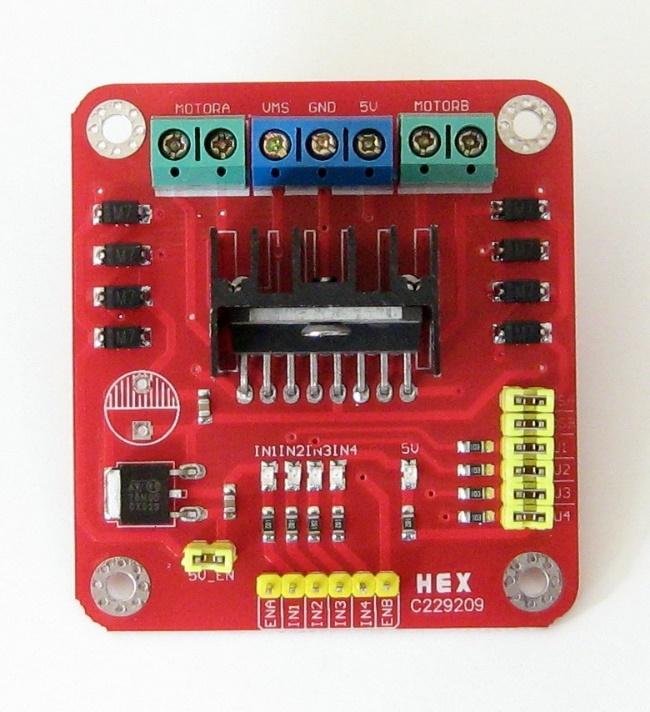
1) Arduino Due



제품 사양

* 마이크로 컨트롤: AT91SAM3X8E
* 동작전압: 3.3V
* 정격전압: 7-12V
* 허용전압: 6-16V
* 디지털 입출력핀: 54개(이 중 12개는 PWM 사용 가능)
* 아날로그 입력 핀: 12개
* 아날로그 출력 핀: 2개
* DC 입출력 핀의 출력 전류: 130 mA
* DC 3.3V 일 때 출력 전류: 800 mA
* DC 5V 일 때 출력 전류: 800 mA
* 플래시 메모리: 512KB (256KB 2개)
* SRAM: 96KB
* 클럭속도: 84MHZ
* 무게: 36g

2) L298N Dual H-Bridge 모터 드라이버



제품 사양

- 드라이버 칩: L298N 듀얼 H-브리지 드라이버 칩

– 드라이브 부분의 공급전압 VMS: +5 V ~ 35 V-

– 드라이브 부분의 최대전류: 2A/bridge

– 로직 부분의 공급전압 VSS: 4.5-5 0.5 V

– 로직 부분의 작동 전류 범위 :0 ~ 36mA

– 입력제어신호 전압 범위: H: 4.5~ 5.5V / L: 0V

– 최대 소비 전력 : 20W

– 보관 온도 : -25 ℃까지 130 ℃

– 드라이버 보드 크기 : 55mm \* 60mm \* 30mm

– 드라이버 보드 무게 : 33g

– 기타 기능 : 방향 제어 표시LED, 전원 표시 LED

3) MG90S 9g Servo



제품사양

           -사이즈            23mm x 12,2nn x 29mm

           -무게                       9g

           -작동 속도            0.12초/60도 (4.8V)

           -토크                       1.8kg/초 (4.8V)

           -작동 온도            –30 ~ +60도

           -동작 밴드 폭 5us

           -동작전압            DC 3.5V~6V

4) DC Motor

제품 사양

- 작동전압: DC 3-6V

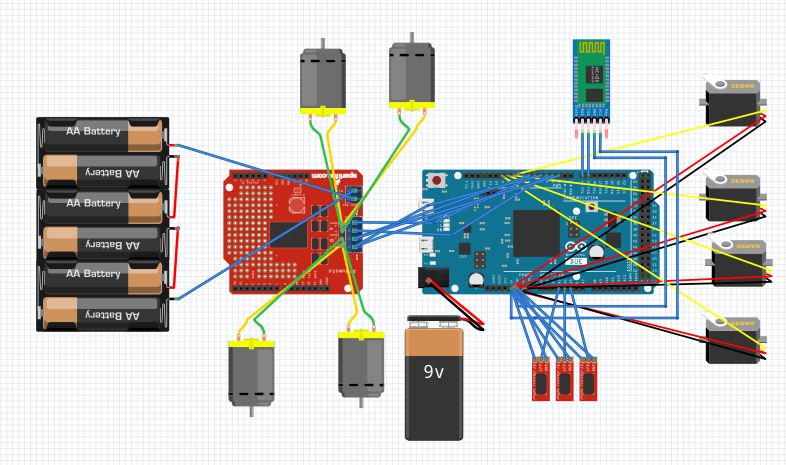
- 감속비: 1:48

5) HC06



제품 사양

* 동작 전압: 3.6~6V
* 통신 방식: Serial(UART) Interface (AT Command)
* 프로토콜: Bluetooth 2.0+ EDR standard
* 운영주파수 : 2.4Ghz ISM frequency band
* 전송속도 : 2.1Mbps(Max)/160 kbps(Asynchronous)； 1Mbps/1Mbps(Synchronous)

7-2. 회로도

1. **코드**

8-1. Arduino Source Code

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Robot Car-Arm System \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* \*/

/\* Final Project \*/

/\* Smart Construction Transportation System(Smart CTS) \*/

/\* Due on 23 December 2019 (Mon) \*/

/\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Programming Copright & Environment : \*/

/\* Team Name : CARM \*/

/\* Author : 최성봉, 이성주, 이동규, 가인온투 \*/

/\* Date : 2019.12.22 \*/

/\* PL : C / C++ \*/

/\*---------------------------------------------------------------------------\*/

/\* Objective : Script for a smart construction transportation system \*/

/\* run on arduino \*/

/\*---------------------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Included Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <Servo.h>

#include <ArduinoJson.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* DEFINE's \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Automobile \*/

#define ENA 6

#define EN1 7

#define EN2 3

#define EN3 4

#define EN4 2

#define ENB 5

/\* Bluetooth \*/

#define HC06 Serial3

/\*Line Tracer \*/

#define LT\_L A3

#define LT\_C A4

#define LT\_R A5

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Global Variables \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Robot\_Arm Servo Motor \*/

Servo servo1, servo2, servo3, servo4;

char automobile\_Direction = 'S';

/\* Automobile Speed Variable \*/

int speed = 110;

int Automobile\_Right\_speed = 122;

int Automobile\_Left\_speed = 125;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* parsing function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: Response \*/

/\* Logic : When Bluetooth available, read bluetooth, add to another variable \*/

/\* and return the variable \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

String parsing()

{

String response = "";

if (HC06.available())

{

long int time = millis(); //현재 시간 설정

int timeout = 3000; //ms

while ((time + timeout) > millis())

{

while (HC06.available())

{

char c = HC06.read();

response += c;

}

}

}

return response;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* line function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Configures all 3 line tracing sensors as Input \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void line()

{

pinMode(LT\_L, INPUT);

pinMode(LT\_C, INPUT);

pinMode(LT\_R, INPUT);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* lt\_isLeft function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: true/false \*/

/\* Logic : If left tracer detects light return true, otherwise false \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool lt\_isLeft()

{

int ret = digitalRead(LT\_L);

return ret == 1 ? true : false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* lt\_isRight fucntion \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: true/false \*/

/\* Logic : If right tracer detects light return true, otherwise false \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool lt\_isRight()

{

int ret = digitalRead(LT\_R);

return ret == 1 ? true : false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* lt\_isCenter function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: true/false \*/

/\* Logic : If center tracer detects light return true, otherwise false \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool lt\_isCenter()

{

int ret = digitalRead(LT\_C);

return ret == 1 ? true : false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Automobile\_Forward function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Makes the robot-car go forward \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Automobile\_Forward()

{

digitalWrite(EN1, HIGH);

digitalWrite(EN2, LOW);

analogWrite(ENA, speed);

digitalWrite(EN3, LOW);

digitalWrite(EN4, HIGH);

analogWrite(ENB, speed);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Automobile\_Left function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Makes the robot-car turn left \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Automobile\_Left()

{

digitalWrite(EN1, LOW);

digitalWrite(EN2, HIGH);

analogWrite(ENA, Automobile\_Left\_speed);

digitalWrite(EN3, LOW);

digitalWrite(EN4, HIGH);

analogWrite(ENB, Automobile\_Left\_speed);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Automobile\_Right function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Makes the robot-car turn right \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Automobile\_Right()

{

digitalWrite(EN1, HIGH);

digitalWrite(EN2, LOW);

analogWrite(ENA, Automobile\_Right\_speed);

digitalWrite(EN3, HIGH);

digitalWrite(EN4, LOW);

analogWrite(ENB, Automobile\_Right\_speed);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Automobile\_Backward function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Makes the robot-car go reverse \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Automobile\_Backward()

{

digitalWrite(EN1, LOW);

digitalWrite(EN2, HIGH);

analogWrite(ENA, speed);

digitalWrite(EN3, HIGH);

digitalWrite(EN4, LOW);

analogWrite(ENB, speed);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Automobile\_Stop function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Makes the robot-car motionless \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Automobile\_Stop()

{

digitalWrite(EN1, LOW);

digitalWrite(EN2, LOW);

analogWrite(ENA, 0);

digitalWrite(EN3, LOW);

digitalWrite(EN4, LOW);

analogWrite(ENB, 0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* lt\_mode\_update function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : update car status according to line tracing sensor update \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lt\_mode\_update()

{

while (true)

{

if (automobile\_Direction == 'U' || automobile\_Direction == 'D')

break;

int ll = lt\_isLeft();

int cc = lt\_isCenter();

int rr = lt\_isRight();

if (cc && ll && rr)

{

if (automobile\_Direction == 'L' || automobile\_Direction == 'R')

{

Automobile\_Stop();

break;

}

else

Automobile\_Forward();

}

else if ((cc && ll && !rr) || (!cc && ll && !rr)) // 왼쪽만 인식되면 좌회전

Automobile\_Left();

else if ((cc && !ll && rr) || (!cc && !ll && rr)) // 오른쪽 인식인식되면 우회전

Automobile\_Right();

else if (!cc && !ll && !rr) // 정지

{

if (automobile\_Direction == 'F')

{

Automobile\_Forward();

delay(300);

Automobile\_Stop();

break;

}

else if (automobile\_Direction == 'L')

Automobile\_Left();

else if (automobile\_Direction == 'R')

Automobile\_Right();

else

;

}

else

;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Arm\_On function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Attach servo 1,2,3,4 variable accordingly to 8,9,10 and 11 pin \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Arm\_On()

{

servo1.attach(8);

servo2.attach(9);

servo3.attach(10);

servo4.attach(11);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Arm\_Off function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Detach servo 1,2,3,4 variable accordingly from its pin \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Arm\_Off()

{

servo1.detach();

servo2.detach();

servo3.detach();

servo4.detach();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Servo\_Set function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : servo, pos \*/

/\* Output: N/A \*/

/\* Logic : Servo motor is engaged untill it reaches our desired position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Servo\_Set(Servo servo, int pos)

{

int cur = servo.read();

if (cur == pos)

;

else if (cur < pos)

while (cur <= pos)

{

servo.write(++cur);

delay(7);

}

else

while (cur >= pos)

{

servo.write(--cur);

delay(7);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Servo\_Set function (Overloaded) \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : servo\_num, pos \*/

/\* Output: N/A \*/

/\* Logic : Passes information to the other Servo\_set function according to \*/

/\* the inputted servo motor number \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Servo\_Set(int servo\_num, int pos)

{

if (servo\_num == 1)

Servo\_Set(servo1, pos);

else if (servo\_num == 2)

Servo\_Set(servo2, pos);

else if (servo\_num == 3)

Servo\_Set(servo3, pos);

else if (servo\_num == 4)

Servo\_Set(servo4, pos);

else

;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* pick\_up function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : With a define movement, servo motors are engaged to pick up \*/

/\* object \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void pick\_up()

{

Arm\_On();

delay(20);

Servo\_Set(servo2, 90);

delay(10);

Servo\_Set(servo4, 0);

delay(10);

Servo\_Set(servo1, 90);

delay(10);

Servo\_Set(servo3, 170);

delay(10);

Servo\_Set(servo1, 0);

delay(10);

Servo\_Set(servo3, 90);

delay(10);

Servo\_Set(servo4, 90);

delay(10);

Servo\_Set(servo2, 60);

delay(10);

Arm\_Off();

delay(10);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* put\_down function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : With a define movement, servo motors are engaged to put down \*/

/\* the object \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void put\_down()

{

Arm\_On();

delay(10);

Servo\_Set(2, 90);

delay(10);

Servo\_Set(4, 180);

delay(10);

Servo\_Set(3, 170);

delay(10);

Servo\_Set(1, 90);

delay(10);

Servo\_Set(3, 90);

delay(10);

Servo\_Set(1, 0);

delay(10);

Servo\_Set(4, 90);

delay(10);

Servo\_Set(2, 60);

delay(10);

Arm\_Off();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Arm\_Init function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : With fixed values, arm servo motors are set to a initial position\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Arm\_Init()

{

Arm\_On();

delay(10);

Servo\_Set(1, 0);

delay(10);

Servo\_Set(4, 90);

delay(10);

Servo\_Set(3, 90);

delay(10);

Servo\_Set(2, 60);

delay(10);

Arm\_Off();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* setup function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Initialize pin modes, serial and bluetoth begins listening for \*/

/\* incoming connections \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup()

{

Serial.begin(9600);

HC06.begin(9600);

pinMode(ENA, OUTPUT);

pinMode(EN1, OUTPUT);

pinMode(EN2, OUTPUT);

pinMode(EN3, OUTPUT);

pinMode(EN4, OUTPUT);

pinMode(ENB, OUTPUT);

Arm\_Init();

line();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Control function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Main control section which controls the arms and car part \*/

/\* according to automobile\_Direction value \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Control()

{

if (automobile\_Direction == 'U')

{

pick\_up();

delay(500);

}

else if (automobile\_Direction == 'D')

{

put\_down();

delay(500);

}

else if (automobile\_Direction == 'L')

{

delay(500);

Automobile\_Left(); //Automobile\_Left

}

else if (automobile\_Direction == 'R') //Automobile\_Right

{

delay(500);

Automobile\_Right();

}

else if (automobile\_Direction == 'F') //Automobile\_Forward

{

Arm\_Init();

Automobile\_Forward();

}

else if (automobile\_Direction == 'B')

Automobile\_Backward(); //Automobile\_Backward

else if (automobile\_Direction == 'S')

Automobile\_Stop(); //stop

else

;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* loop function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Input : N/A \*/

/\* Output: N/A \*/

/\* Logic : Arduino board is actively controlled using a collaboration \*/

/\* of control and lt\_mode\_update function \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop()

{

String response = parsing();

for (int i = 0; i < response.length(); i++)

{

automobile\_Direction = response[i];

Control();

if (automobile\_Direction != 'F')

delay(500);

if (automobile\_Direction != 'S')

lt\_mode\_update();

}

}

8-2. AdoScript

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"

name:"Automobile01"

extra:{}

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://210.117.182.28:10000/") #서버의 ip주소와 포트를 넣은 변수

FOR modelid in:(modelids) #모든 모델을 읽어들일 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:("{") #json 형식으로 만들기 위해 변수 생성, 앞의 @는 아두이노 상에서 파싱처리를 위해 임의로 넣은 토큰임

WHILE (1) #모델 안에 있는 모든 객체의 정보를 읽어들인 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+ "\""+sState + "\_" + "t\""+":\"" + "\_" + type + "\","

# + "\""+sState + "\_" + "qs\""+":\"" + qs + "\",") #json 형식으로 text 저장

SETL json:(json+ "\""+sState + "\_" + "t\""+":\"" + "\_" + type + "\","

+ "\""+sState + "\_" + "qs\""+":\"" + qs + "\","

+ "\""+sState + "\_" + "time\""+":\"" + 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+ "\""+sState + "\_" + "t\""+":\"" + "\_" + type + "\",")

}

SET state:(state+1)

IF (t=50)

{

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+""+ "\"state\":\"" + "\_" + sState + "\"}") #json 변수 저장

HTTP\_SEND\_REQUEST (ip+"?id=adoxx&name=Automobile"+basename) str\_method:("GET") map\_reqheaders:(tempmap) str\_reqbody:(json) val\_respcode:respstat map\_respheaders:respheaders str\_respbody:respbody

SLEEP ms:(500) # 서버에서 데이터를 완전히 받을 때까지 대기

}

CC "AdoScript" INFOBOX ("The end of AdoScript")

PROCEDURE SLEEP ms:integer

{

SETL time:(800\*ms)

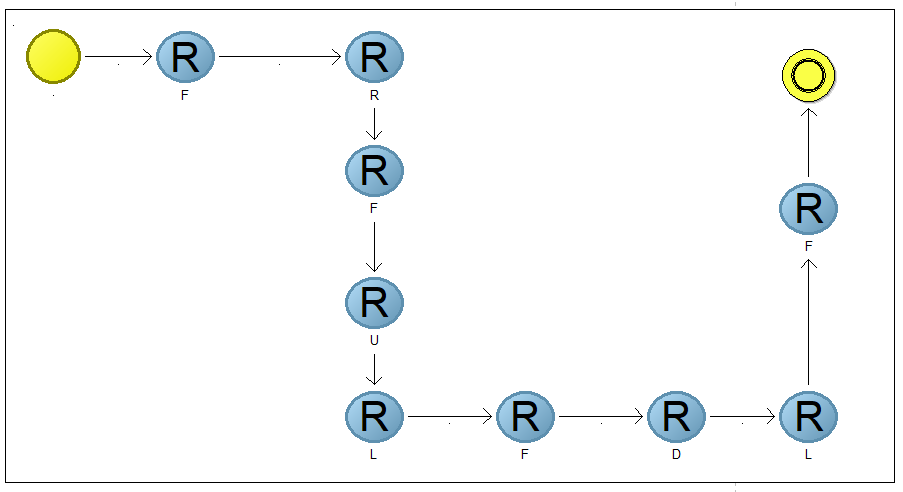
FOR i from:1 to:(time) {}

}

8-3. ADOxx Modeling

시나리오 1

시작 - 직진 – 우회전 – 직진 – 물건 들기 – 좌회전 – 직진 – 물건 내려놓기 – 좌회전 – 직진 – 종료

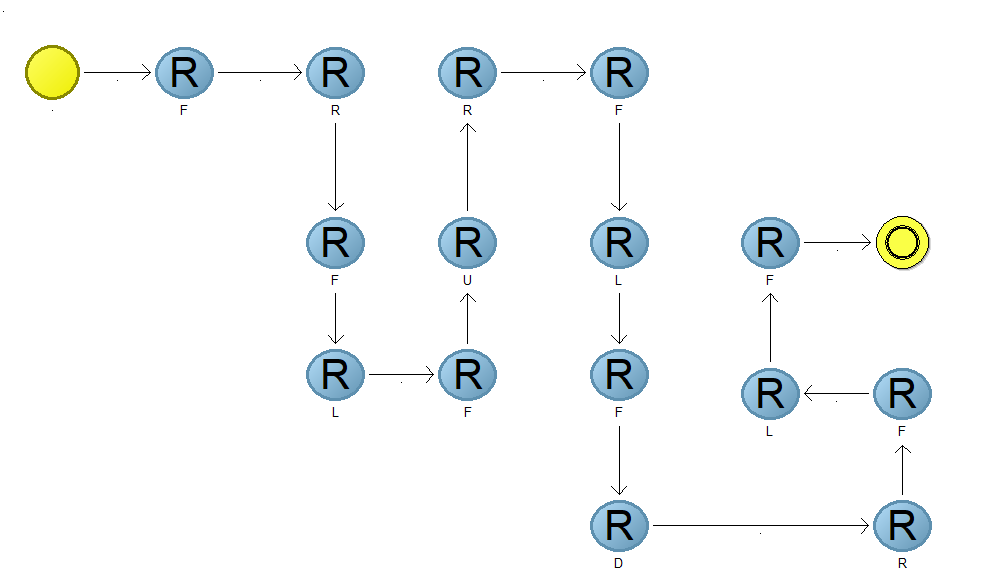


시나리오 2

시작 - 직진 – 우회전 – 직진 – 좌회전 – 직진 – 종료



시나리오 3



1. **유효성 테스트**

9-1. 시험 계획서

1)Automobile

-차량이 명령에 맞게 작동하는가?

-직진 명령을 잘 수행할 수 있는가?

-오른쪽 회전 명령을 잘 수행할 수 있는가?

-왼쪽 회전 명령을 잘 수행할 수 있는가?

-후진 명령을 잘 수행할 수 있는가?

-제동 명령을 잘 수행할 수 있는가?

-차량이 라인 트레이서를 정상적으로 인식할 수 있는가?

-라인을 잘 인식하는가?

-왼쪽으로 벗어났을 때 정상적으로 복귀할 수 있는가?

-오른쪽으로 벗어났을 때 정상적으로 복귀할 수 있는가?

-라인이 아닌 경우 정상적으로 멈출 수 있는가?

2)Robot-ARM

- robot\_arm 이 명령에 맞게 잘 수행하는가?

- pick\_up 명령을 정상적으로 수행하는가?

- put\_down 명령을 정상적으로 수행하는가?

9-2. 테스트 결과

1)Automobile

-차량이 명령에 맞게 작동하는가?

-직진 명령을 잘 수행할 수 있는가?

: Yes

-오른쪽 회전 명령을 잘 수행할 수 있는가?

:Yes

-왼쪽 회전 명령을 잘 수행할 수 있는가?

:Yes

-후진 명령을 잘 수행할 수 있는가?

:Yes

-제동 명령을 잘 수행할 수 있는가?

:Yes

-차량이 라인트레이서를 정상적으로 인식할 수 있는가?

-라인을 잘 인식하는가?

:Yes

-왼쪽으로 벗어났을 때 정상적으로 복귀할 수 있는가?

:Yes

-오른쪽으로 벗어났을 때 정상적으로 복귀할 수 있는가?

:Yes

-라인이 아닌 경우 정상적으로 멈출 수 있는가?

:Yes

2)Robot-ARM

-robot\_arm 이 명령에 맞게 잘 수행하는가?

-pick\_up 명령을 정상적으로 수행 하는가?

:Yes

-put\_down 명령을 정상적으로 수행 하는가?

:Yes

1. **회의록**

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 10.16 (수) |
| 회의 주제 | 프로젝트 주제 선정 및 모델링 선정 |
| 참가인원 | 가인온투, 이성주, 최성봉 |
| 회의 내용 | * 1. 프로젝트 주제 선정 * 선정 주제: 공사현장 장비 운송을 위한 스마트 시스템 * 선정 이유: 공사현장에서 사람이 직접 중장비를 운반함으로 발생하는 인명 피해 및 물적 피해를 줄이고자 주제로 선정함 * 활동 사진        * 1. 프로젝트 모델링 선정 * 선정 모델 : eXtreme Programming(Agile Model) * 선정 이유   짧은 시간내에 완성도 높은 프로그램 개발 가능  즉각 적인 피드백  퀄리티 중심의 모델링  소규모 인원으로 개발 가능   * 1. 프로젝트 역할   - Project Leader(PL): 최성봉  - Project Engineer(PE): 이동규, 이성주, 가인온투  - Documents Writer: 이성주  - Tester: 이동규  - Debugger: 가인온투  4) 프로젝트 세부 사항  Work product: 공사현장 장비 운송을 위한 스마트 시스템  Activity: 장비 적재 및 운송  Action 1) 장비 적재 (Equipment loading)  Task 1) 물건을 든다. (pick up)  Task 2) 차량에 싣는다. (loading on vehicle)  Action 2) 운송(Transport)  Task 1) 현 위치 확인(check for current location)  Task 2) 목적지 확인(check for destination location)  Task 3) 차량 이동 (vehicle movement)  Action 3) 장비 하역(Equipment unloading)  Task 1) 올바른 목적지인지 확인(correct destination verify)  Task 2) 물건을 든다(pick up)  Task 3) 물건을 목적지에 내려놓는다(Put down at the destination) |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 10. 23(수) |
| 회의 주제 | 요구 사항 분석 |
| 참가인원 | 가인온투, 이성주, 최성봉, 이동규 |
| 회의 내용 | 요구사항 분석을 위한 요구 사항 파악   1. 공사현장에서 담당자가 원하는 장비 및 배달 받을 장소를 시스템에 요청한다. 2. 시스템은 담당자의 요청을 확인하여 장비의 위치를 확인한다. 3. 장비의 위치가 확인되면 차고지에 있는 자율화 기기를 보낸다. 4. 위치에 도착한 자율화 기기는 물건을 싣는다. 5. 목적지의 위치를 확인한 뒤 이동한다. 6. 목적지에 도착한 후 현 위치와 목적지 위치가 일치하는지 확인한다.    1. 위치가 일치하는 경우 물건을 내려놓고, 차고지로 돌아간다.    2. 위치가 일치하지 않는 경우 5로 돌아간다. |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 11. 02(토) |
| 회의 주제 | 사전 회의 |
| 참가인원 | 이성주, 최성봉 |
| 회의 내용 | 1. Task 분할,   서버관리  아두이노 통합 관리  문서작업 및 관리  모델링 관리  Planning   1. 공사현장을 어떻게 시각화 하여 보여줄 것 인가? 2. 차량은 어떻게 제어할 것인가 3. 차량은 어떻게 개발할 것인가 4. 차량과 어떻게 통신할 것인가   Design   1. 공사현장 map은 ~로 만든다. 2. 차량은 Adoxx로 제어한다. 3. 차량에 대한 개발은 아두이노 코딩을 한다. 4. Web sever를 통해 통신.   Coding  매주 개발내용 확인 및 평가 후 피드백  책임감, 책임을 부여하여 수행을 안 했을 시 ~한다.  지속적인 통합, 어느정도 완성이 되었을 경우 통합테스트를 진행한다.  Pair programming – 한 task를 2명이상이 함께 맡아 진행한다.  Testing  Coding 하면서 중간 중간하는 테스팅과  완성이 되었을 때 구동이 잘 되는지에 대한 테스트 진행  문제 발생시 feedback 진행  채택검사 각 요구사항을 얼마만큼 완성도 있게 수행할 수 있는가를 평가  조건 미흡 시 다시 plan으로  Deployment – 완성 후 제품 출시 |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 11. 04(월) |
| 회의 주제 | 역할 선정 및 임무 분배. |
| 참가인원 | 가인온투, 이성주, 최성봉 |
| 회의 내용 | EMB0000503c9593  (Project Modeling – eXtreme Programming)  **1. Plan**  1) 공사현장을 어떻게 시각화할 것인가?  2) 차량을 어떻게 제어할 것인가?.  3) 차량을 어떻게 개발할 것인가?  4) 차량과 어떻게 통신할 것인가?  위의 4개 주제에 대해서 각자 사전 조사 진행.  **2. Design**  1) 공사현장 시각화  EMB0000503c9594  EMB0000503c9595- start, end point , EMB0000503c9596- load, unload point  EMB0000503c9597- Equipment point, EMB0000503c9598- Obstacle  위와 같은 형식으로 구성하기로 결정.  2) 차량 제어  - Adoxx를 이용하여 차량을 제어하기로 함. 본격적인 차량 개발이 진행되기 이전에는 Adoxx에 대한 활용법을 분석 및 연구하기로 함.  3) 차량 개발  - 아두이노를 통해 차량에 대한 개발을 진행하기로 결정. 본격적인 차량에 대한 개발이 진행되기 이전에는 아두이노 상에서 프로그래밍 하는 법에 대해서 Study 하기로 결정.  4) 차량과의 통신  - web server를 구축하여 통신, python, Django, json 등을 활용하여 web server를 직접 구축하는 것을 고려하였으나 많은 시간이 소요될 것으로 판단하여 aws를 이용하여 구축하기로 결정.  **3. 역할 및 임무 분배**  1) 공사현장 MAP 제작에 대한 세부사항 구상  – 이동규(정), 이성주(부)  2) 차량 제어를 위한 Adoxx 분석 및 연구  – 최성봉(정), 이동규(부)  3) 차량 개발을 위한 아두이노 Study  – 가인온투(정), 최성봉(부)  4) 차량과의 통신을 위한 Web Server 구축  – 이성주(정), 가인온투(부)  EMB0000503c9599EMB0000503c959a |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 11. 9(월) |
| 회의 주제 | 분석 모델링 |
| 참가인원 | 화상회의(전원 참여) |
| 회의 내용 | 분석 모델링에 대한 간단한 토론  자료 조사 역할 분배  가인온투 – 클래스 다이어그램 외 1개 선택  최성봉 – ER Diagram 외 1개 선택  이동규 - Sequence Diagram 외 1개 선택  이성주 – UseCase 다이어그램 외 1개 선택 |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 11. 13(수) |
| 회의 주제 | 디자인 모델링 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | EMB0000503c95aa  Design modeling에 대한 기본적인 내용을 의논하였고, 어떻게 해야 할지 고민함.  각 Design 단계에 대해서 어떻게 구축해야 할지 찾아보았으며, 과정속에서 난관에 봉착함.  자세한 예시가 나와있지 않으며 강의자료의 설명도 부족하였다. 그러므로 각자 좀더 시간을 투자하여 자료조사를 해야 하는 필요성이 느껴졌다. 또한 강의자료의 이 부분을 기반으로 방향을 잡아 나가기로 하였으며 팀원들에게 역할을 나누기로 정하였다.  Interface – 이성주  Component – 가인온투  Architecture – 이동규  Data / class - 최성봉  위와 같이 역할을 분배하였으며, 목요일에 각자 자료조사한 내용으로 회의를 진행하기로 함. |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 4(수) |
| 회의 주제 | 아두이노 개발 역할 분배 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | 아두이노 개발에 대한 역할 분배를 함  자동차 – 이성주, 이동규  로봇 팔 – 최성봉, 가인온투  12월 13일까지 기본 개발 완료 후 합치기로 함 |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 13(금) |
| 회의 주제 | 자동차와 로봇 팔 결합 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | 자동차와 로봇 팔 결합 및 테스트를 실시함 |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 16(월) |
| 회의 주제 | 기기 블루투스 테스트 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | 기기에 블루투스를 연결하여 작동 테스트를 시행함. |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 18(수) |
| 회의 주제 | ADOxx 라이브러리 확인 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | ADOxx에 대한 활용방안에 대해서 논의 및 회의 |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 19(목) |
| 회의 주제 | ADoScript 적용 시도 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | ADoxx를 통한 Arduino 제어 방법을 알아봄  지속적인 실패로 인해 문제점을 파악하였는데, 기본 제공 ADoScript의 문제였음,  8시간동안 삽질. |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 21(토) |
| 회의 주제 | 맵 제작 및 Arduino 테스트 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | 우드락과 절연 테이프를 이용하여 맵을 제작함.    초기 테이프만을 이용하여 맵을 제작하였지만, 라인 트레이서의 인식 문제로 인하여 하얀색 네모를 추가함. |

|  |  |
| --- | --- |
| 회 의 록 | |
| 날짜 | 19. 12. 22(일) |
| 회의 주제 | 최종 테스트 및 보고서 작성 |
| 참가인원 | 이동규 이성주 최성봉 가인온투 |
| 회의 내용 | 12.23(월) 실시할 발표를 준비하여 최종 점검 및 보고서를 작성함. |