**Software Design Specification  
for Simple ADAS System**

**Project Team**

**OOO Team**

Date

**0000-00-00**

**Team Information**

**Table of Contents**

[1 Introduction 4](#_Toc130153128)

[1.1 Purpose 4](#_Toc130153129)

[1.2 Scope 4](#_Toc130153130)

[1.3 Terms & Acronyms 4](#_Toc130153131)

[1.4 References 4](#_Toc130153132)

[2 Overall Description 4](#_Toc130153133)

[2.1 Context Diagram 4](#_Toc130153134)

[2.2 External Interface 5](#_Toc130153135)

[2.2.1 Input Data 5](#_Toc130153136)

[2.2.2 Output Data 5](#_Toc130153137)

[3 Structured Analysis & Design (Hierarchical Design) 5](#_Toc130153138)

[3.1 Software Component – Data Flow Level 1 6](#_Toc130153139)

[3.1.1 LKAS 6](#_Toc130153140)

[3.1.1.1 Description 6](#_Toc130153141)

[3.1.1.2 Inputs 6](#_Toc130153142)

[3.1.1.3 Outputs 6](#_Toc130153143)

[3.1.2 ACC 6](#_Toc130153144)

[3.1.3 FCA 6](#_Toc130153145)

[3.1.4 DMM 7](#_Toc130153146)

[3.1.5 VCU 7](#_Toc130153147)

[3.2 Software Component – Data Flow Level 2 7](#_Toc130153148)

[3.2.1 LKAS – Data Flow Level 2 7](#_Toc130153149)

[3.2.1.1 Function Name 7](#_Toc130153150)

[3.2.1.1.1 Description 7](#_Toc130153151)

[3.2.1.1.2 Inputs 7](#_Toc130153152)

[3.2.1.1.3 Outputs 7](#_Toc130153153)

[3.2.1.1.4 Detailed Design 8](#_Toc130153154)

[3.2.1.2 Function Name 9](#_Toc130153155)

[3.2.2 ACC 9](#_Toc130153156)

[3.2.3 FCA 10](#_Toc130153157)

[3.2.4 DMM 10](#_Toc130153158)

[3.2.5 VCU 10](#_Toc130153159)

1. Introduction
   1. Purpose

이 문서의 목적에 대해 서술하시면 됩니다. (요구사항 및 설계 문서 참조)

* 1. Scope

이 문서의 범위에 대해 서술하시면 됩니다.

* 1. Terms & Acronyms

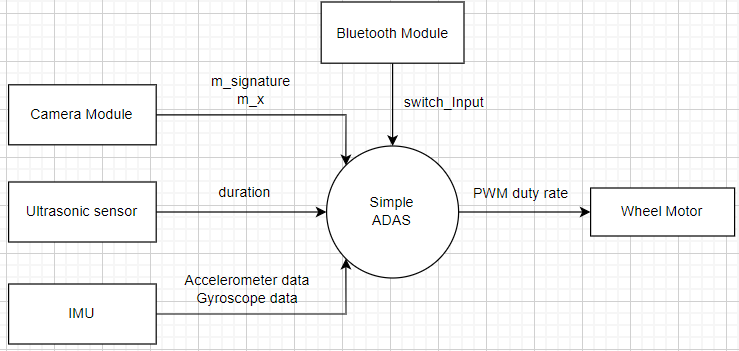
용어 정의

* 1. References

1. Overall Description
   1. Context Diagram - Data Flow Level 0

소프트웨어 설계 대상의 Context diagram을 기술합니다.

Context diagram은 개발하고자 하는 대상의 외부 요소와 외부 요소와의 인터페이스를 표현합니다.



ADAS는 첨단 운전자 보조 시스템으로, 운전자가 안전하고 편리하게 주행할 수 있도록 도와주는 모든 기능을 뜻한다. 실제 ADAS의 기능으로는 FCA, LKAS, BCA, SEA, ISLA, DAW, BVM, SCC, NSCC, LFA, HDA, HDA2, SVM, RCCA, PCA, RSPA 등이 있으나 본 프로젝트에서는 Simple ADAS라는 이름으로 LKAS, ACC, FCA 만을 구현하여 기능의 수를 제한하기로 한다. 이 때, FCA는 제동만을 사용해 동작하는 것으로 기능을 제한한다.

아두이노에 들어오는 입출력은 다음과 같다.

입력

- switch\_input

- m\_signature

- m\_x

- duration

- Accelerometer data (X축)

- Gyroscope data (Z축)

출력

-PWM 명령

소프트웨어는 판단SW, 구동SW로 구성되며 아두이노에 내장된다.

판단SW는 LKAS, ACC, FCA의 알고리즘과 DMM(Decision Making Model)으로 구성되어 있으며 주요 기능은 차선, 물체의 인식 및 추적, 주행 상황 판단 등이 있다. 추가로 판단 SW는 각 기능의 상태 정보를 PC로 전송한다.

구동SW는 VCU로 표현하기로 하며, VCU는 모터에 인가할 전압을 PWM 신호로 계산해 모터를 구동한다.

* 1. External Interface
     1. Input Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **From** | **Description** |
| duration | Unsigned long | 0 | 116 | 23200 | Ultrasonic  Sensor |  |
| Accelerometer data | short int | 0 | -32768 | 32767 | IMU | MPU6050에 내장된 3축 가속도 센 |
| Gyroscope data | short int | 0 | -32768 | 32767 | IMU |  |
| m\_signature | Unsigned Int | - | 1 | 7 | Camera  Module | 감지된 물체의 signature number. 학습을 통해 특정 색상에 대한 signature number 부여 가능.  1: 라인  2: ACC 추종 대상 물체  3: 장애물 |
| m\_x | Unsigned int | - | 0 | 316 | Camera  Module | 감지된 물체 중심의 x 위치 |
| switch\_input | Unsigned Int | 0 | 0 | 4 | Cluster | SW ON/OFF, ACC ON/OFF 명령.  사용자가 블루투스 통신으로 활성화된 포트의 시리얼 모니터를 통해 입력.  1: SW ON  2: SW OFF  3: ACC ON  4: ACC OFF |

* + 1. Output Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| PWM duty rate | Float | 0 | 0 | 255 | Motor | 모터 구동을 위해 필요한 전압 명령 |

1. Structured Analysis & Design (Hierarchical Design)
   1. Software Component – Data Flow Level 1

텍스트, 도표, 평행, 평면도이(가) 표시된 사진

자동 생성된 설명

* + 1. LKAS
       1. Description

|  |  |
| --- | --- |
| **File Name** | Lane\_Keeping\_Assist\_System.ino |
| **Functional Description** | RC car가 주행 간 두 차선 사이를 벗어나지 않도록 필요한 조향 각속도를 계산해 자세각을 변경하도록 하는 알고리즘 |

* + - 1. Inputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **From** | **Description** |
| m\_signature | Unsigned Int | - | 1 | 7 | Camera  Module | 감지된 물체의 signature number. 학습을 통해 특정 색상에 대한 signature number 부여 가능.  1: 라인  2: ACC 추종 대상 물체  3: 장애물 |
| m\_x | Unsigned int | - | 0 | 315 | Camera  Module | SW ON/OFF, ACC ON/OFF 명령.  사용자가 블루투스 통신으로 활성화된 포트의 시리얼 모니터를 통해 입력.  1: SW ON  2: SW OFF  3: ACC ON  4: ACC OFF |
| Accelerometer data | short int | 0 | -32768 | 32767 | IMU |  |
| Gyroscope data | short int | 0 | -32768 | 32767 | IMU |  |

* + - 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| Vel\_x | Double | 0.0 | 0.0 |  | ACC, FCA | MPU6050의 가속도 센서에 의해 측정된 RC car 중심의 좌표계 기준 X축 가속도 정보를 m/s 단위로 변환한 값 |
| Ang\_z | double | 0.0 | -180.0 | 180.0 | - | MPU6050의 자이로센서에 의해 측정된 RC car 중심의 좌표계 기준 Z축 각도 정보를 deg 단위로 변환한 값 |
| Num\_Lanes | int | 0 | 0 | 2 | DMM | Pixy2 카메라 모듈에 의해 인식된 라인의 개수 |
| Des\_Angular\_Velocity | float | 0.0 | -30.0 | 30.0 | DMM | LKAS에 의해 계산된 RC car가 라인을 벗어나지 않도록 조향하기 위해 필요한 조향 각속도 |
| target\_exist | bool | False | false | True | ACC | Pixy2 카메라 모듈에 의해 인식된 추종 대상 물체의 유무 정보 |
| obstacle\_exist | Bool | false | False | True | FCA | Pixy2 카메라 모듈에 의해 인식된 장애물의 유무 정보 |
| LKA\_reset\_count | int | 0 | 0 | 11 | DMM |  |

* + 1. ACC
       1. Description

|  |  |
| --- | --- |
| **File Name** | Adaptive\_Cruise\_Control.ino |
| **Functional Description** | RC car가 전방의 추종 대상 물체와 거리를 유지하면서 주행하도록 RC car의 속도를 조절하는 알고리즘 |

* + - 1. Inputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **From** | **Description** |
| distance | Unsigned long | 0 | 2 | 400 | FCA | 초음파 센서의 전방에 존재하는 물체와의 거리 측정 값 |
| Vel\_x | double | 0.0 | 0.0 | - | LKAS | LKAS에서 구한 RC car의 주행 상태 가운데 RC car 중심 좌표계의 종방향 속도값 |
| target\_exist | bool | false | false | true | LKAS | LKAS에서 카메라 모듈을 통해 인식한 추종 대상 물체의 유무 정보 |

* + - 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| des\_vel\_ACC | Float | 0 | 0 |  | DMM | ACC 기능 수행을 위해 전방 추종 대상과의 거리를 유지하기 위한 RC car의 목표 속도 |

* + 1. FCA
       1. Description

|  |  |
| --- | --- |
| **File Name** | Forward\_Collision\_Avoidance\_Assist.ino |
| **Functional Description** | RC car가 주행 도중 전방 물체와의 충돌 위험이 감지되면 긴급 제동하는 알고리즘 |

* + - 1. Inputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **From** | **Description** |
| obstacle\_exist | bool | false | false | true | Camera  Module | Pixy2 카메라 모듈을 통해 인식된 장애물 유무 정보 |
| Vel\_x | double | 0.0 | 0.0 | - | LKAS |  |
| duration | Unsigned long | 0 | 116 | 23200 | Ultrasonic  Sensor | HC-SR04 초음파 센서에 의해 측정된 전방 물체와의 거리 정보를 담은 값 |

* + - 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| distance | Unsigned long | 0 | 2 | 400 | ACC | FCA에 의해 cm 단위로 변환한 초음파 센서의 거리 측정 값 |
| des\_vel\_FCA | Float | 1.0 | 0.0 | - | DMM | FCA 기능 수행을 위해 전방의 장애물과 충돌여부를 판단하여 계산한 RC car의 목표 종방향 속도 |

* + 1. DMM
       1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Functional Description** | 각 기능의 활성화 상태와 주행 상황을 통해 RC car의 주행 상태를 결정하기 위한 목표 종방향 속도와 목표 조향 각속도를 계산하는 결정 모델 |

* + - 1. Inputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **From** | **Description** |
| Num\_Lanes | int | 0 | 0 | 2 | LKAS | Pixy2 카메라 모듈에 의해 인식된 라인의 개수 |
| Des\_Angular\_Velocity | float | 0.0 | -30.0 | 30.0 | LKAS | LKAS에 의해 계산된 RC car가 라인을 벗어나지 않도록 조향하기 위해 필요한 조향 각속도 |
| des\_vel\_ACC | Float | 0.0 | 0.0 | - | ACC | ACC 기능 수행을 위해 전방 추종 대상과의 거리를 유지하기 위한 RC car의 목표 속도 |
| des\_vel\_FCA | Float | 1.0 | 0.0 | - | FCA | FCA 기능 수행을 위해 전방의 장애물과 충돌여부를 판단하여 계산한 RC car의 목표 종방향 속도 |
| switch\_input | int | 0 | 0 | 4 | Bluetooth\_Module | SW와 ACC의 활성화 여부를 결정하는 사용자의 시리얼 입력 |
| LKAS\_reset\_count | int | 0 | 0 | 11 | LKAS | LKAS 기능 동작 간 라인을 인식하지 못했을 때 동작모드에 의해 진행된 Reset의 횟수 |

* + - 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| DAD | Float | 0.0 | 0.0 | - | VCU | DMM에 의해 계산된 RC car의 목표 조향 각속도 |
| DVD | Float | 0.0 | 0.0 | - | VCU | DMM에 의해 계산된 RC car의 목표 종방향 속도 |

* 1. Software Component – Data Flow Level 2
     1. LKAS

텍스트, 도표, 평행, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - 1. Get\_state
         1. Description

|  |  |
| --- | --- |
| **File Name** | Lane\_Keeping\_Assist\_System.ino |
| **Prototype** | Vehicle\_Info get\_state(void) |
| **Functional Description** | IMU 센서의 데이터를 통해 RC카의 속도와 자세각을 계산하는 함수.  함수가 실행되면 아두이노는 MPU6050와 아날로그 핀인 A4, A5을 통해 I2C 통신으로 센서의 raw data를 받아온다. DMM의 sensor\_init 함수에서 센서의 offset을 계산했기 때문에 get\_state에서는 측정 센서값에 offset을 빼준 뒤 단위를 변환한다.  가속도 센서의 X축, Y축 raw data는 int16\_t형으로 -32768과 32767 사이의 값으로 표현이 된다. 해당 범위를 ±2g로 표현하는 것으로 하면 Scale Factor는 16384가 되고 이는 중력가속도 단위인 1g에 해당하는 값이다. 이를 m/s^2 단위의 가속도 값으로 변환하기 위해서는 먼저 중력 단위를 구하기 위해 16384로 나누어 준 뒤, 중력가속도인 9.8을 곱한다.  자이로센서의 Z축 raw data는 rad/s로 나타나는데 이를 deg/s 단위로 변환하기 위해 131.0으로 나누어준다.  RC카의 속도와 자세각을 구하기 위해 각 값들을 아두이노의 기본 라이브러리 함수인 micros()를 통해 계산한 시간 간격에 따라 적분한다. 자세각의 경우 X축을 기준인 0도로 위에서 내려다봤을 때 반시계 방향을 양(+)의 각, 시계 방향을 음(-)의 각으로 표현하여 범위를 ±180도로 정의한다. 적분에 의해 각도를 계산하기 때문에 범위 내의 값으로 표현하기 위해 적분값이 180도 보다 클 경우에는 360도를 빼주고, -180도보다 작을 경우에는 360도를 더해주는 것으로 한다. |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| Vel\_x | double | 0.0 | 0.0 | - | FCA, ACC |  |

* + - 1. Detect\_obj
         1. Description

|  |  |
| --- | --- |
| **File Name** | Lane\_Keeping\_Assist\_System.ino |
| **Prototype** | Detection\_result detect\_obj(void) |
| **Functional Description** | 카메라 모듈을 통해 라인 및 물체를 인식 및 구분하고 필요한 조향 방향을 결정하는 함수.  SRS에서 가정한 라인, 장애물, 추종 대상 물체의 색은 각각 초록색, 빨간색, 파란색으로 각각 pixy2 카메라 모듈에 학습되어 있다. 카메라가 물체 구분없이 인식 가능한 총 개체 수는 4개로, 각 case 별 최대 인식 가능 개체 수는 2개로 제한한다.  pixy.ccc.getBlocks()를 통해 한 프레임 내에서 인식된 개체 정보를 받는다.  만약 객체를 인식하게 되면 pixy.ccc.numBlocks()는 라인이 인식될 수 있는 최소 개수인 1 이상이어야 한다. 즉, pixy.ccc.numBlocks()가 0이면 어떠한 객체도 인식하지 못한 경우이다. 이 때는 LKAS\_reset\_count를 증가시키고 동작모드에 의해 pixy2\_init와 all\_sensors\_init를 false로 변경하며, sensor\_init()함수를 실행시켜 pixy2 카메라 모듈의 initialize를 다시 시도하도록 한다.  객체를 인식했다면 객체를 분류하기 위해 for문을 실행한다.  만약 i번째 개체에 대해 라인이 인식될 경우, 추종 대상 물체가 인식될 경우, 장애물이 인식될 경우 pixy.ccc.blocks[i].m\_signature 값은 각각 1, 2, 3이다. 라인의 개수에 따라 lanes라는 변수가 1씩 증가하며, 추종 대상 물체를 인식할 경우 target\_exist가 true로, 장애물을 인식할 경우 obstacle\_exist가 true로 변경된다.  이때, 라인의 경우 pixy를 통해 얻은 데이터 가운데, 블록의 중심의 x좌표를 mid라는 변수에 더한다.  객체 분류를 완료하면, lanes의 값을 확인해 라인을 인식했는지 확인한다. 라인을 인식하지 못했다면 LKAS\_reset\_count를 증가시키고 라인을 인식했다면 mid를 lanes로 나눠 라인의 중심의 평균값을 구해 mid로 다시 저장한다. mid와 lanes는 바로 다음 절에 언급할 calc\_SA()함수의 매개변수로 전달하여 목표 조향 각속도를 결정하도록 한다. |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| DR | Detection\_result | - | - | - | DMM(Num\_Lanes, Des\_Angular\_Velocity),  FCA(obstacle\_exist),  ACC(target\_exist) | Pixy2 카메라 모듈에 의해 얻어진 영상의 한 프레임에서 라인과 물체의 인식 결과를 나타낸 구조체이다. 라인의 개수를 나타내는 Num\_Lanes, 장애물 유무를 나타내는 obstacle\_exist, 추종 대상 물체 유무를 나타내는 target\_exist, 목표 조향 각속도인 Des\_Angular\_Velocity를 원소로 한다.  Num\_Lanes는 2이하의 정수로 detect\_obj에서 한 프레임에 대해 지역 변수 lanes를 선언해 라인의 지정 색인 초록색을 인식할 경우 lanes가 증가한다. 프레임 내 인식된 모든 개체를 확인하고나면 Num\_Lanes에 lanes를 할당된다.  obstacle\_exist는 장애물의 지정 색인 빨간색을 인식할 경우 true가 된다.  target\_exist는 추종 대상 물체의 지정 색인 파란색을 인식할 경우 true가 된다.  Des\_Angular\_Velocity는 float형 변수로 LKAS에서 RC카가 두 라인 사이를 주행하도록 계산한 각속도이다. 라인 중심의 평균 x 좌표인 mid와 라인의 개수인 lanes을 매개변수로 calc\_SA에 전달했을 때 calc\_SA의 return 값으로 한다. |

* + - 1. Calc\_SA
         1. Description

|  |  |
| --- | --- |
| **File Name** | Lane\_Keeping\_Assist\_System.ino |
| **Prototype** | float calc\_SA(float mid, unsigned int lanes) |
| **Functional Description** | LKAS에 의해 RC카가 두 라인 사이를 벗어나지 않도록 선회 주행하는 데 필요한 각속도를 계산하는 함수.  인자로 detect\_obj에서 계산한 mid와 lanes를 받는다.  Pixy2 카메라 모듈의 영상 프레임에서 라인 2개를 인식했을 경우에는 라인 1개를 인식한 경우에 비해 차량이 중앙선(라인 2개의 중심선을 중앙선이라고 하자.)에서 크게 벗어나지 않은 상황이므로 중앙에 대한 영역을 지정해준다.  Frame\_width\_mid는 프레인의 중심선의 x좌표를 의미하며, tol은 중심영역을 나타내도록 하는 중심선으로부터의 간격을 나타내며 중심영역의 끝부분의 x좌표는 각각 mid\_left, mid\_right로 선언한다.  두 라인의 중심의 x좌표인 mid가 중심영역 왼쪽에 있을 경우에는 좌회전이므로 15deg/s, 중심영역 오른쪽에 있을 경우에는 우회전이므로 -15deg/s, 중심영역에 포함되어 있는 경우에는 선회하지 않고 직진하도록 0deg/s를 Des\_Angular\_Velocity로 한다.  라인 1개를 인식한 경우에는 차량이 중앙선에서 크게 벗어난 경우이므로 라인 2개의 경우와 똑같이 좌회전 우회전 직진 규칙을 적용하되 선회 운동이 필요한 case이므로 중심영역이 아니라 중심선을 기준으로 좌우를 구분한다. 따라서 mid가 중심선 왼쪽에 있을 경우에는 좌회전으로 30deg/s, 오른쪽에 있을 경우에는 우회전으로 -30deg/s를 Des\_Angular\_Velocity로 한다. |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| Des\_Angular\_Velocity | float | 0.0 | -30.0 | 30.0 | DMM |  |

* + 1. ACC

도표, 라인, 텍스트, 평행이(가) 표시된 사진

자동 생성된 설명

* + - 1. Description

|  |  |
| --- | --- |
| **File Name** | Adaptive\_Cruise\_Control.ino |
| **Prototype** | float ACC(unsigned long distance, bool target\_exist, double Vel\_x) |
| **Functional Description** |  |

* + - 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| des\_vel\_ACC | float | 0.0 | 0.0 | - | DMM |  |

* + 1. FCA

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - 1. Get\_distance
         1. Description

|  |  |
| --- | --- |
| **File Name** | Forward\_Collision\_Avoidance\_Assist.ino |
| **Prototype** | unsigned long get\_distance() |
| **Functional Description** | HC-SR04 센서의 데이터를 거리 값으로 변환하는 함수.  duration이라는 변수를 선언해 센서의 raw data를 저장한다.  Trigger핀을 활성화해 초음파를 송신하며 Echo핀을 활성화해 물체와 반사해 돌아오는 초음파를 수신한다. 이때 pulseIn을 통해 초음파 신호의 왕복 시간을 계산한다. 거리 계산을 위해 왕복 거리를 2로 나눠 편도 거리를 구하고, pulseIn의 출력값이 마이크로초이므로 초 단위로 계산하기 위해 1000000으로 나누고, 초음파의 속도는 340m/s이므로 cm단위로 거리를 계산하기 위해 34000을 곱하여 distance에 저장한다.  Distance는 HC-SR04의 측정 범위인 4m를 넘을 경우 튀는 값으로 취급하여 전역변수인 prev\_distance를 갱신하지 않으며 distance에 prev\_distance를 저장하도록 한다. 첫 측정이 아닌 경우에는 추가로 차량의 기본 속력이 1m/s인 점을 감안해 측정값이 이전 값과의 차이가 1m를 넘을 경우도 튀는 값으로 취급한다. 그 외의 경우는 정상적인 측정값으로 인정하고 prev\_distance 값을 갱신한다. |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| distance | Unsigned long | 0 | 2 | 400 | ACC | HC-SR04에서 방출한 초음파 신호가 물체에 반사되어 돌아온 시간으로부터 구한 cm 단위의 거리 측정값. |

* + - 1. FCA
         1. Description

|  |  |
| --- | --- |
| **File Name** | Forward\_Collision\_Avoidance\_Assist.ino |
| **Prototype** | FCA\_result FCA(bool obstacle\_exist, double Vel\_x) |
| **Functional Description** | 전방 장애물과의 거리와 RC카의 주행 속도를 통해 장애물과 충돌하지 않도록 목표 속도를 계산하는 함수.  Get\_distance()함수로 전방 물체와의 거리를 구해 현재 속도에 대한 충돌 예상 시간(TTC)을 구한다. 이때, RC카의 속도가 0이하 즉, 정지나 후진일 경우에는 물체가 가만히 있다는 전제하에 충돌이 불가능하므로 0.01로 나눈다.  LKAS에 의해 장애물이 인식되어 obstacle\_exist가 true이고 TTC가 2.2초 이하라면 충돌 위험이 있는 상황으로 간주하고 목표 속도를 0으로 한다. 그렇지 않으면 기본 주행 속도인 1m/s로 한다. |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| des\_vel\_FCA | float | 1.0 | 0.0 | - | DMM | 전방 물체와의 충돌 위험을 고려한 RC카의 목표 주행 속도. LKAS에 의해 장애물이 인식되어 obstacle\_exist가 true이고 TTC가 2.2초 이하라면 충돌 위험이 있는 상황으로 간주하고 목표 속도를 0으로 한다. 그렇지 않으면 기본 주행 속도인 1m/s로 한다. |

* + - * 1. Detailed Design
    1. DMM

텍스트, 도표, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

* + - 1. System\_exit
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | void system\_exit(void) |
| **Functional Description** |  |

* + - 1. Valid\_input\_message
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | void valid\_input\_message(int switch\_input) |
| **Functional Description** |  |

* + - * 1. Detailed Design
      1. Invalid\_input\_message
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | void invalid\_input\_message(void) |
| **Functional Description** |  |

* + - * 1. Detailed Design
      1. Check\_bluetooth
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | bool check\_bluetooth(void) |
| **Functional Description** |  |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| bluetooth\_Connected | Bool | False | False | True | check\_switch | HC-06 모듈과 PC의 블루투스 통신 상태.  Check\_bluetooth 함수가 실행될 때마다 해당 값은 false로 초기화되며 통신 중임을 확인하면 true가 된다. |

* + - * 1. Detailed Design
      1. Check\_switch
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | void check\_switch(bool Bluetooth\_connected) |
| **Functional Description** | 사용자 입력에 따라 SW와 ACC의 활성화 여부를 결정하는 함수.  사용자 입력은 시리얼 입력으로 받으며 블루투스 통신에 의해 PC에서 아두이노로 입력된다. 따라서 블루투스 통신 여부를 확인하기 위해 함수는 인자로 Bluetooth\_connected를 받으며 해당 변수가 true일 경우 사용자 입력을 받을 수 있도록 한다.  Switch\_input은 사용자의 입력값을 나타내는 지역변수로, 함수 실행 시 선언되며 0, 1, 2, 3, 4를 입력으로 받게 될 경우 조건문이 동작하도록 하며 그 외의 값이 입력될 경우 switch\_false\_count가 1 증가한다. 올바른 입력이 들어올 경우 switch\_false\_count는 0으로 초기화 된다. Sw\_status와 acc\_status는 전역변수로 각각 SW와 ACC의 활성화 상태를 나타내며 초기값은 fasle이다.  각 숫자를 스위치 관점에서 설명하면 다음과 같다.  1: sw switch on  2: sw switch off  3: acc switch on  4: acc switch off  조건은 10개와 모두 해당하지 않는 fault 상황이 있으며, sw\_status, acc\_status를  조건에 포함해 각각 SW, ACC의 활성화 상태를 확인한다. Fault 상황의 경우는 이전 문단에 설명되어 있다.  1. SW 비활성화, 입력: 0  SW 비활성화의 입력 대기상태이다.  2. SW 비활성화, 입력: 1  SW를 활성화시키는 상황으로 sensor\_init()함수를 실행해 센서의 initialize를 실시한  다.  3. SW 활성화, ACC 비활성화, 입력: 1  SW 활성화를 유지하는 상황  4. SW 활성화, ACC 활성화, 입력: 1  3과 동일.  5. 입력: 2  SW를 종료하는 상황이다. System\_exit()함수를 실행시켜 sw를 종료한다.  6. SW 비활성화, 입력: 3 or 4  SW 비활성화 상태에서 ACC의 활성화 switch를 조작한 상황으로 잘못된 입력이다.  Switch\_falut\_count를 1 증가시킨다.  7. SW 활성화, ACC 비활성화, 입력: 3  ACC를 활성화하는 상황이다. Acc\_status를 true로 한다.  8. SW 활성화, ACC 활성화, 입력: 3  ACC 활성화를 유지하는 상황.  9. SW 활성황, ACC 비활성화, 입력: 4  ACC를 비활성화하는 상황이다. Acc\_status를 false로 한다.  10. SW 활성화, ACC 비활성화, 입력: 4  ACC 비활성화 상황임에도 비활성화를 시도하는 상황이다. Prev\_input을 1로 하여  3번 상황으로 전환한다.  6번 상황을 제외하고는 나머지 9개의 상황은 유효한 입력이 들어오는 상황이다.  Prev\_input은 초기값이 0인 전역변수로, 사용자로부터 유효한 입력이 들어올 경우 갱신이 되며 사용자로부터 시리얼 입력이 들어오지 않을 경우 switch\_input의 값으로 쓰인다. Save\_input은 1, 4, 7, 9 상황과 같이 사용자 입력에 의한 스위치 명령이 유효할 경우 사용자 입력인 prev\_input 값을 저장한다. 사용자 입력은 유효한데 유효하지 않는 스위치 명령이 들어왔을 때, 해당 입력 이전에 마지막 스위치 상태를 유지하도록 prev\_input 값에 save\_input 값을 사용한다.  Switch\_fault\_count가 4가 되면 system\_exit()함수를 실행해 SW를 종료한다. |

* + - * 1. Detailed Design
      1. Sensor\_init
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | void sensor\_init(void) |
| **Functional Description** | 초음파 센서, IMU 센서, 카메라 모듈, 모터의 initialize를 실시하는 함수  해당 함수는 check\_switch()함수에서 sw\_status가 false인 상황에서 시리얼 입력으로 1이 입력된 상황에서 initialize 목적으로 실행되므로 단 1회만 실행된다.  함수를 실행 시 init\_fault\_count라는 변수를 선언하여 initialize의 reset 횟수를 확인하도록 한다. Init\_fault\_count가 4가 아닌 경우 while문을 통해 모든 센서와 모터의 initialize를 시도한다.  초음파 센서 HC-SR04는 trigger 핀으로 초음파 신호를 송신하며 echo 핀으로 수신하는 것으로 initialize를 실시한다.  IMU 센서 MPU6050은 LKAS의 get\_state()와 마찬가지로 데이터를 구하되 해당 함수에서는 get\_state()에서 구하고자하는 X, Y축 가속도값과 Z축 각속도의 정확도를 높이기 위해 오프셋인 AcXoff, AcYoff, GyZoff를 구한다. 오프셋은 cnt\_sample 값인 1000개의 샘플 데이터를 받아 모두 더한 뒤, 이들의 평균값을 구해 각각 AcXoff, AcYoff, GyZoff에 저장한다. (초기값은 모두 0이어야 하므로 평균값을 get\_state()에서 측정값에 빼주기 위함이다.)  카메라 모듈 Pixy2는 부품 자체 라이브러리인 pixy.init()을 통해 initialize를 시도한다.  모터는 ㅁㄴㅇㄻㄴㄹㄴㅁㅇㄻㄴㅇㄹ  각 센서의 initialize 여부를 나타내는 변수가 모두 true일 경우 전역 변수 all\_sensors\_init을 true로 바꾼다. 하나의 센서라도 initialize 여부를 나타내는 변수가 false일 경우 init\_fault\_count는 1씩 증가한다.  init\_fault\_count가 4가 될 경우 while문을 빠져나오며 system\_exit()를 실행해 시스템을 종료한다. |

* + - * 1. Detailed Design
      1. Decision\_making
         1. Description

|  |  |
| --- | --- |
| **File Name** | Decision\_Making\_Model.ino |
| **Prototype** | DMM\_result decision\_making(int Num\_Lanes, float Des\_Angular\_Velocity, float des\_vel\_acc, float des\_vel\_fca, bool acc\_status) |
| **Functional Description** | 시스템 동작을 제어하고 시스템 상태 및 LKAS, ACC, FCA 기능의 동작 여부에 따라 주행 상황을 판단하여 RC car의 목표 각속도와 목표 종방향 속도를 계산하는 함수.  인자로 LKAS()의 return인 LKAS\_result와 ACC()의 return인 des\_vel\_acc, FCA()의 return인 des\_vel\_fca, 전역 변수인 acc\_status를 받는다.  LKAS(), FCA(), ACC()를 모두 실행한 이후에 모든 상황을 종합하기 위해 마지막에 실행되는 함수이다.  LKAS에 의해 라인의 개수가 존재한다면 동작모드에 의해 driving mode로 전환한다는 출력문을 보내고, LKAS 활성화 상태를 나타내는 LAS를 true로 한다. ACC의 활성화 상태를 나타내는 AAS는 acc\_status와 동일하게 적용이 되며, FCA의 활성화 상태를 나타내는 FAS는 des\_vel\_fca가 0일 때는 FCA()함수에 의해 충돌 위험을 판단한 상황이므로 true로하고 0이 아닐 때는 false로 한다.  LKAS에서 결정된 목표 각속도는 DMM에서 그대로 적용되어 Desired Angular Velocity of DMM인 DAD에 저장된다.  Desired Velocity of DMM인 DVD는 FCA 기능 활성화 여부를 우선으로 하여 FAS가 true일 경우 des\_vel\_fca로 선정한다. FCA가 활성화 상태가 아니라면 ACC 기능의 활성화 상태일 때 des\_vel\_acc를 선정하며, 그렇지 않을 경우 init\_vel을 선정하도록 한다. |

* + - * 1. Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Min** | **Max** | **To** | **Description** |
| DMM\_result | struct | - | - | - | VCC(DAD, DVD) | DAD, DVD, function\_state로 이루어진 구조체.  DAD는 Desired Angular Velocity of DMM으로 DMM에 의해 구해진 목표 각속도로 초기값이 0.0인 float형 변수이다. LKAS의 calc\_SA()함수의 return값이 그대로 사용되며 별다른 계산과정을 거치지 않는다.  DVD는 Desired Velocity of DMM으로 DMM에 의해 구해진 목표 종방향 속도로 초기값이 0.0인 float형 변수이다. Decision\_making()의 입력 인자인 des\_vel\_acc, des\_vel\_fca와 전역 변수인 init\_vel 중 하나의 값이 되며 FAS가 true면 des\_vel\_fca, FAS는 false, AAS는 true일 때 des\_vel\_acc, FAS와 AAS가 모두 false일 때 init\_vel이 된다. |
| Function\_state | struct | - | - | - | - | 주행 기능들의 활성화 상태인 LAS, AAS, FAS 로 이루어진 구조체로 모두 초기값이 false인 bool형 변수이다. |

* + - * 1. Detailed Design

1. Data Design
   1. Struct, Enum, Union

4.1.1 Struct

[Detection\_result]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| target\_exist | bool | False | Pixy2 카메라 모듈에 의해 인식된 추종 대상 물체의 유무 정보 |
| obstacle\_exist | bool | False | Pixy2 카메라 모듈에 의해 인식된 장애물의 유무 정보 |
| Num\_Lanes | int | 0 | Pixy2 카메라 모듈에 의해 인식된 라인의 개수 |
| Des\_Angular\_Velocity | float | 0.0 | LKAS 알고리즘에 의해 라인을 벗어나지 않도록 조향하기 위해 인가하는 각속도 |

[LKAS\_Result]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| Detection\_result | struct | - | Pixy2 카메라 모듈을 통한 인식 정보 |
| Vehicle\_Info | struct | - | IMU 센서에 의해 측정된 RC카의 상태 정보 |

[Vehicle\_info]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| Vel\_x | double | 0.0 | IMU의 가속도 센서로부터 계산된 차량의 X 방향 속력 |
| Vel\_y | double | 0.0 | IMU의 가속도 센서로부터 계산된 차량의 Y 방향 속력 |
| Ang\_z | double | 0.0 | IMU의 자이로센서로부터 계산된 차량의 Z축 자세각 |

[FCA\_Result]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| distance | Unsigned long | 0 | 초음파 센서로부터 측정된 전방 물체와의 거리 |
| des\_vel\_fca | float | 0.0 | FCA 알고리즘에 의해 계산된 목표 속도 |

[DMM\_Result]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| DAD | float | 0.0 | DAD는 Desired Angular Velocity of DMM으로 DMM에 의해 구해진 RC car의 목표 각속도로 초기값이 0.0인 float형 변수이다. LKAS의 calc\_SA()함수의 return값이 그대로 사용되며 별다른 계산과정을 거치지 않는다. |
| DVD | float | 0.0 | DVD는 Desired Velocity of DMM으로 DMM에 의해 구해진 RC car의 목표 종방향 속도로 초기값이 0.0인 float형 변수이다. Decision\_making()의 입력 인자인 des\_vel\_acc, des\_vel\_fca와 전역 변수인 init\_vel 중 하나의 값이 되며 FAS가 true면 des\_vel\_fca, FAS는 false, AAS는 true일 때 des\_vel\_acc, FAS와 AAS가 모두 false일 때 init\_vel이 된다. |
| Function\_state | struct | - | 주행 기능들의 활성화 상태인 LAS, AAS, FAS 로 이루어진 구조체로 모두 초기값이 false인 bool형 변수이다. |

[Function\_state]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| LAS | bool | False | LAS는 LKAS Active State로 LKAS 기능의 활성화 상태이다. LKAS에서 라인의 개수가 인식될 경우 true가 된다. |
| AAS | bool | False | AAS는 ACC Active State로 ACC 기능의 활성화 상태이다. Decision\_making()의 입력 인자인 acc\_status의 값으로 저장된다. |
| FAS | bool | False | FAS는 FCA Active State로 FCA 기능의 활성화 상태이다. FCA()의 return인 des\_vel\_fca가 0이면 true이며 0이 아니면 false이다. |

* 1. Global Variable

[check\_switch]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| sw\_status | bool | False | 사용자의 입력을 통한 전체 SW의 활성화 상태.  DMM의 check\_switch()에서 sw\_status가 false인 경우 사용자 입력으로 1이 들어올 때 true가 된다. System\_exit()가 실행되거나, Check\_switch()에서 사용자 입력이 2가 들어올 때 false가 된다. |
| acc\_status | bool | False | 사용자의 입력을 통한 ACC 기능의 활성화 상태  DMM의 Check\_switch()에서 7번 상황에 true가 되며 9번 상황에 false가 된다. Decision\_making()에서는 del\_vel\_acc가 적용되기 위해서 acc\_status가 true여야 한다. |
| switch\_fault\_count | int | 0 | 조건에 맞지 않은 사용자 입력의 연속적인 오류 횟수.  4 미만일 경우 유효한 입력이 들어오면 0으로 초기화된다.  4회 연속 유효하지 않은 입력이 들어오면 시스템 종료 조건이 된다. |
| prev\_input | int | 0 | 사용자에 의한 스위치 입력 바로 직전의 입력 상태 |
| valid\_input | int | - | 새로운 스위치 입력이 들어오기 전까지, 기존의 입력 상태를 유지하기 위해 저장되고 불러오는 변수 |

[sensor\_init]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| All\_sensors\_init | Bool | false | 전체 센서들과 모터의 initialize 여부  DMM의 sensor\_init()에서 HC-SR04, MPU6050, Pixy2, 모터의 initialize 변수가 모두 true일 경우 true가 된다. |
| Hc\_sr04\_init | Bool | False | 초음파 센서 HC-SR04의 initialize 여부.  DMM의 sensor\_init()에서 Trigger 핀과 Echo 핀의 설정 및 활성화가 진행되면 true가 된다. |
| Mpu6050\_init | Bool | False | IMU 센서 MPU6050의 initialize 여부.  DMM의 sensor\_init()에서 센서 데이터의 오프셋을 구하면 true가 된다. |
| Pixy2\_init | Bool | False | Pixy2 카메라 모듈의 initialize 여부.  DMM의 sensor\_init()에서 Pixy 라이브러리 내 init 함수가 정상적으로 실행되면 true가 된다. |
| Motor\_init | bool | False | 모터의 initialize 여부. DMM의 sensor\_init()에서 |
| AcXsum | Int32\_t | 0 | MPU6050의 가속도센서로부터 구한 X축 가속도 데이터의 합 |
| AcXoff | Int16\_t | 0 | MPU6050의 가속도센서로부터 구한 X축 가속도 데이터의 평균 오프셋 |
| AcYsum | Int32\_t | 0 | MPU6050의 가속도센서로부터 구한 Y 축 가속도 데이터의 합 |
| AcYoff | Int16\_t | 0 | MPU6050의 가속도센서로부터 구한 Y축 가속도 데이터의 평균 오프셋 |
| GyZsum | Int32\_t | 0 | MPU6050의 자이로센서로부터 구한 Z축 각속도 데이터의 합 |
| GyZoff | Int16\_t | 0 | MPU6050의 자이로센서로부터 구한 Z축 각속도 데이터의 평균 오프셋 |
| Cnt\_sample | Static int | 1000 | MPU6050의 가속도센서, 자이로센서의 오프셋을 구하기 위한 샘플링 데이터 수 |
| T\_prev | Unsigned long | 0 |  |

[calc\_SA]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| Frame\_width\_mid | int | 157 | Pixy2 카메라 모듈에서 얻어지는 영상 및 이미지 프레임에 대해 중심선을 나타내기 위한 프레임 중심의 x 좌표 값 |
| Tol | int | 20 | 중심으로부터 좌, 우측 영역을 정의하기 위한 간격 |

[get\_distance]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| Prev\_distance | Unsigned long | 0 |  |

[steer\_control]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| Axle\_length | float | 0.2 |  |
| Wheel\_diameter | Float | 0.1 |  |
| enA | Int | 9 |  |
| In1 | Int | 5 |  |
| In2 | Int | 6 |  |
| enB | Int | 10 |  |
| In3 | Int | 7 |  |
| In4 | Int | 8 |  |
| motorSpeed | Int | 0 |  |

[etc.]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Initial Value** | **Description** |
| Init\_vel | float | 1.0 | RC car의 기본 주행 시 종방향 속도 |