LGE Internal Use Only

안내로봇 주행 매핑 설계

**SW Architectural Design**

**For**

**Guide robot Navigation mapping**

BS Jaejoon Shin

About This Document

Document Information

|  |  |
| --- | --- |
| Issuing authority | LGE |
| **Status of document** | In Progress / Approved / Released |

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Comment | Author | Approver |
| 0.3 | 2023-02-17 | 과제설계문서 초안작성 | 신재준 |  |
| 0.4 | 2023-02-24 | 멘토링 내용 반영 | 신재준 |  |
| 0.5 | 2023-02-27 | 심사 제출 버전 | 신재준 |  |
| 0.6 | 2023-03-21 | 1차 Feedback 반영 | 신재준 |  |
| 1.0 | 2023-04-11 | 2차 Feedback 반영 | 신재준 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 6](#_Toc132098580)

[1.1 Project Overview 6](#_Toc132098581)

[1.1.1 Background 6](#_Toc132098582)

[1.1.2 Project Purpose 7](#_Toc132098583)

[1.1.2.1 Business Purpose 7](#_Toc132098584)

[1.1.2.2 Technological Purpose 8](#_Toc132098585)

[1.1.3 Scope of Project 8](#_Toc132098586)

[1.1.3.1 Team Organization & Role 8](#_Toc132098587)

[1.1.3.2 Stakeholders 8](#_Toc132098588)

[1.1.3.3 Schedules 9](#_Toc132098589)

[2 Architectural Driver 10](#_Toc132098590)

[2.1 UX scenario Overview 10](#_Toc132098591)

[2.1.1 UX scenario – Vision SLAM Logging 10](#_Toc132098592)

[2.1.2 UX scenario - Grid Map 생성 11](#_Toc132098593)

[2.1.3 UX scenario - Vision Data Labeling 11](#_Toc132098594)

[2.1.4 UX scenario - Map Editing 12](#_Toc132098595)

[2.2 Constraints 12](#_Toc132098596)

[2.2.1 Business Constraints 12](#_Toc132098597)

[2.2.2 Technical Constraints 12](#_Toc132098598)

[2.3 Requirement Definition 13](#_Toc132098599)

[2.3.1.1 Functional Requirement 13](#_Toc132098600)

[2.3.2 Non Functional Requirement 13](#_Toc132098601)

[2.4 Quality Attribute definition 13](#_Toc132098602)

[2.4.1 Quality Attribute Priority Metrix 14](#_Toc132098603)

[2.4.1.1 Value Scale 14](#_Toc132098604)

[2.4.2 Quality Attribute List 14](#_Toc132098605)

[2.4.3 Quality Attribute Scenario 14](#_Toc132098606)

[2.4.3.1 QAS-01 배송로봇 Navigation 재사용 14](#_Toc132098607)

[2.4.3.2 QAS-02 Mapping 시간 감소 15](#_Toc132098608)

[2.4.3.3 QAS-03 뜬 장애물 가시화 기능 15](#_Toc132098609)

[2.4.3.4 QAS-04 동일 SLAM 성능수준 확보 16](#_Toc132098610)

[3 Architectural Design 17](#_Toc132098611)

[3.1 Architectural Design - QAS-01 17](#_Toc132098612)

[3.1.1 Design Discussion 17](#_Toc132098613)

[3.1.2 System Context 18](#_Toc132098614)

[3.1.3 Static Perspective 19](#_Toc132098615)

[3.1.4 Dynamic Perspective 21](#_Toc132098616)

[3.1.4.1 SLAM 21](#_Toc132098617)

[3.1.4.2 Navigation - Local Map 22](#_Toc132098618)

[3.2 Architectural Design - QAS-02 22](#_Toc132098619)

[3.2.1 Design Discussion 22](#_Toc132098620)

[3.2.2 System Context 23](#_Toc132098621)

[3.2.3 Static Perspective (Module View) 24](#_Toc132098622)

[3.2.4 Dynamic Perspective (C&C View) 25](#_Toc132098623)

[3.2.5 Interface Design - H/W 개발 26](#_Toc132098624)

[3.3 Architectural Design - QAS-03 27](#_Toc132098625)

[3.3.1 Design Discussion 27](#_Toc132098626)

[3.3.2 System Context 28](#_Toc132098627)

[3.3.3 Static Perspective (Module View) 29](#_Toc132098628)

[3.3.4 Dynamic Perspective (C&C View) 30](#_Toc132098629)

[3.3.5 Scenario Design – 사용 시나리오 정의 30](#_Toc132098630)

[3.4 Implementation - QAS-04 31](#_Toc132098631)

[3.4.1 Lidar SLAM 성능 향상 31](#_Toc132098632)

[3.4.2 S1 Lidar 대응 31](#_Toc132098633)

[3.4.3 고속주행 대응 31](#_Toc132098634)

[3.5 Architectural Design – Overall Design (Allocation View) 31](#_Toc132098635)

[3.6 Mapping Process 수립 32](#_Toc132098636)

[4 Validation & Verification 34](#_Toc132098637)

[4.1 Quality Attribute 결과 34](#_Toc132098638)

[4.1.1 배송로봇 Navigation 재사용 34](#_Toc132098639)

[4.1.2 Mapping 시간 감소 34](#_Toc132098640)

[4.1.3 뜬 장애물 가시화 기능 34](#_Toc132098641)

[4.1.4 동일 SLAM 성능수준 확보 35](#_Toc132098642)

[4.2 요구사항 검증방안 수립 36](#_Toc132098643)

[4.2.1 Goal #1: 저렴하고 빠른 Mapping 방법 개발 36](#_Toc132098644)

[4.2.1.1 Process 시간 비교 36](#_Toc132098645)

[4.2.1.2 Mapping 장비 가격 비교 37](#_Toc132098646)

[4.2.2 Goal #2: 동일 수준의 위치오차 검증 37](#_Toc132098647)

[4.2.2.1 Image Labeling Data 유효성 검증 37](#_Toc132098648)

[4.2.2.2 Mapping / Image Labeling 성능 확인 37](#_Toc132098649)

[4.3 검증결과 38](#_Toc132098650)

[4.3.1 Goal #1: 저렴하고 빠른 Mapping 방법 개발 결과 38](#_Toc132098651)

[4.3.1.1 Process 시간 비교 결과 38](#_Toc132098652)

[4.3.1.2 Mapping 장비 가격 비교 39](#_Toc132098653)

[4.3.2 Goal #2: 동일 수준의 위치오차 검증결과 40](#_Toc132098654)

[4.3.2.1 Image Labeling Data 유효성 검증 결과 40](#_Toc132098655)

[4.3.2.2 Mapping / Image Labeling 성능 확인 결과 41](#_Toc132098656)

[5 Lesson & Learned 42](#_Toc132098657)

# Introduction

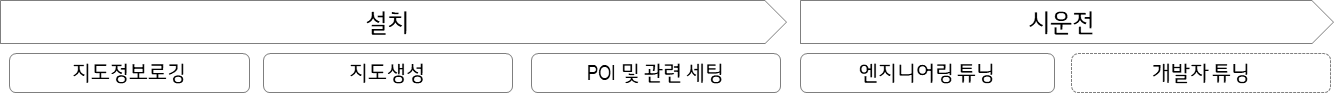
****Project Overview****

### ****Background****

설계문서를 보다 쉽게 이해하기 위해서, 주요단어(약어)와 그 설명을 기재한다.

|  |  |
| --- | --- |
| 주요단어 | 설명 |
| SLAM | Simultaneous localization and mapping, 주변 환경 지도를 작성하는 동시에 로봇의 위치를 작성된 지도 안에서 추정. 로봇에 부착된 센서에 따라 다양하게 구현할 수 있음. |
| Vision SLAM | 자사 중형배송로봇, 안내로봇의 SLAM 방식이며 Camera 입력을 통해서 받은 Image를 기반으로 현재 위치를 추. |
| Grid Map | SLAM의 결과물로 로봇이 주행할 수 있는 영역을 2D 지도형식으로 표시. 자사 Grid Map은 주행가능한 영역은 하얀색, 불가능한 영역은 검은색/회색으로 표시. |
| Image Labeling | Vision SLAM을 사용할 때 필요한 DB를 생성하기 위한 Raw data. 로봇의 카메라를 통해 받은 Image가 Grip Map상에 어느 지점에 위치했는지를 표시하는 작업. |
| Mapping Robot | SLAM은 위에서 작성한 것처럼, 지도 작성과 위치 추정을 동시에 하는 것을 말하지만, Vision SLAM은 지도 생성과 Image Labeling과 위치 추정을 동시에 하기 어려워 별도의 Mapping Robot을 통해서 지도생성과 Image Labeling을 진행 |
| Mapping Module | 위의 Mapping Robot이 고가이므로, 이를 저렴하게 하기 위해 배송로봇에 거치하는 형태의 모듈을 제작. |

로봇의 현장설치를 위해서는 하기 그림과 같은 프로세스로 진행되어야 한다.



<Figure 1>

위 단계 중 로봇의 주행을 위해서는 지도정보(Map)를 Logging(조이스틱으로 로봇 이동가능 영역으로 이동)하여 지도를 생성해야 한다. 자사 로봇의 기본 주행 방식은 Vision SLAM이므로, Grid Map과 Image Labeling (Camera로 촬영된 Image와 촬영된 좌표가 기록된 별도의 File을 생성)을 수행할 수 있는 방식이 필요하다.

지금까지는 Map을 생성하고 Image Labeling을 수행 할 수 있는 별도의 Mapping Robot을 사용하였으나, Mapping Robot의 비용이 고가여서 다수의 로봇을 운영하여 동시에 여러 사이트의 설치가 어려움이 있다.

설치가 용이하고 가격이 저렴하게 Mapping할 수 있는 방법이 필요하여 본 프로젝트를 시작하게 되었다.

### Project ****Purpose****

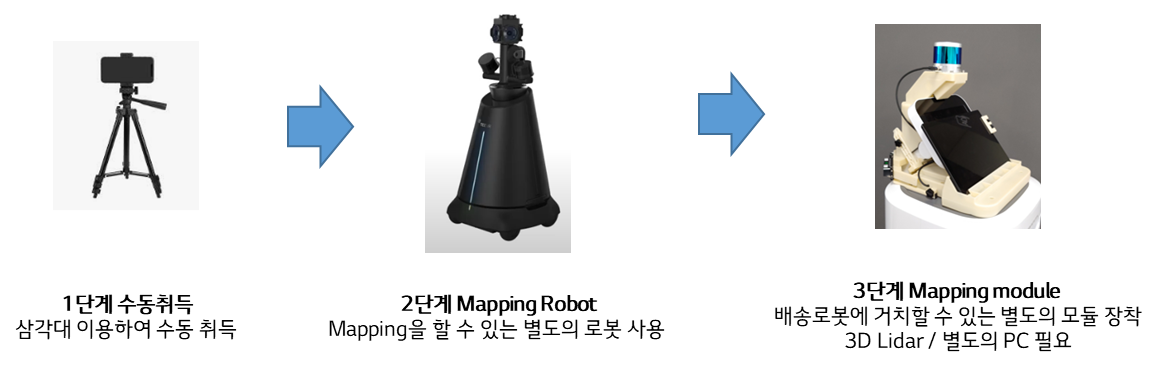
#### Business ****Purpose****

자사 로봇의 위치 추정방식은 VSLAM을 사용하고 있다. SLAM은 Mapping과 Localization으로 구분할 수 있고, VSLAM은 Vision Mapping이 추가적으로 필요하다. Vision Mapping을 하기 위해서는 현재까지는 별도의 장비가 필요한데, Mapping Robot을 사용하거나, Mapping Module을 사용해야 한다.

Mapper Robot은 단가가 1억가까이 되고, 대수가 1대밖에 없기 때문에 동시에 여러 사이트를 대응할 수 없다.

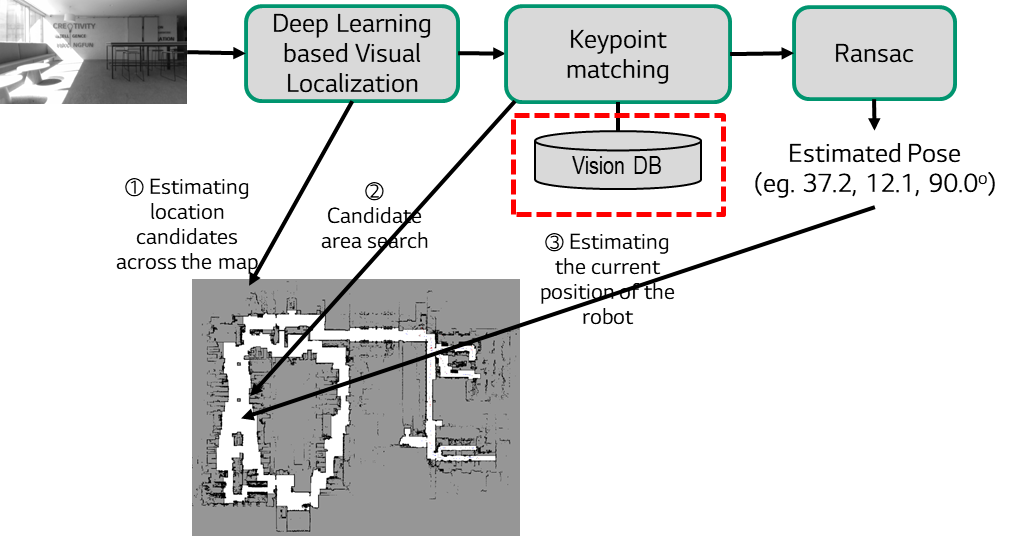
Mapping Module도 단가는 1000만원 수준이라 Mass production을 하기에는 어려움이 있다. 판매할 로봇의 대수가 늘어나고 설치해야 할 장소가 늘어나면 Mapping Robot과 Module로는 대응이 되지 않을 것으로 예상이 된다.

이를 해결하기 위해 Mapping 장비의 단가를 낮추는 방법에 대한 고민이 필요하다. **저렴하고 빠른 Mapping 방법 개발하고 이를 기존 Mapping Process에 적용하여 안내로봇에 적용하는 것**을 목표로 한다.



<Figure 2>

#### Technological ****Purpose****



<Figure 3>

자사 Robot의 위치추정방식인 VSLAM은 Figure3과 같은 방식으로 위치를 추정하는데, Deep Learning 모델과 더불어서 Vision DB의 성능이 위치 정확도에 미친다. 이는 Mapping 시점에 취득되는 Lidar Data와 Vision Data의 품질과 비례하며, 기존 중형배송로봇용 Mapping Robot이나 Mapping Module로 취득한 Data로는 위치오차 10cm ~ 15cm를 확보하였다. 본 프로젝트에서도 동일 SLAM알고리즘 기반에서 저렴한 Mapping 방법으로 취득된 Data의 **동일 수준의 위치오차를 목표**로 진행한다.

### ****Scope of Project****

별도의 Mapping Robot / Module을 사용하지 않고(3D Lidar를 미사용하고), Robot에 내장된 2D Lidar를 이용하여 Mapping& Image Labeling 하는 방법을 모색하고, 동등 수준의 성능을 확보하기 위해 Robot 주행 Engine 수정하는 방식으로 개발한다.

또한, 새로운 Mapping방식에 대한 Process를 설치 담당자와 같이 수립한다.

#### Team Organization & Role

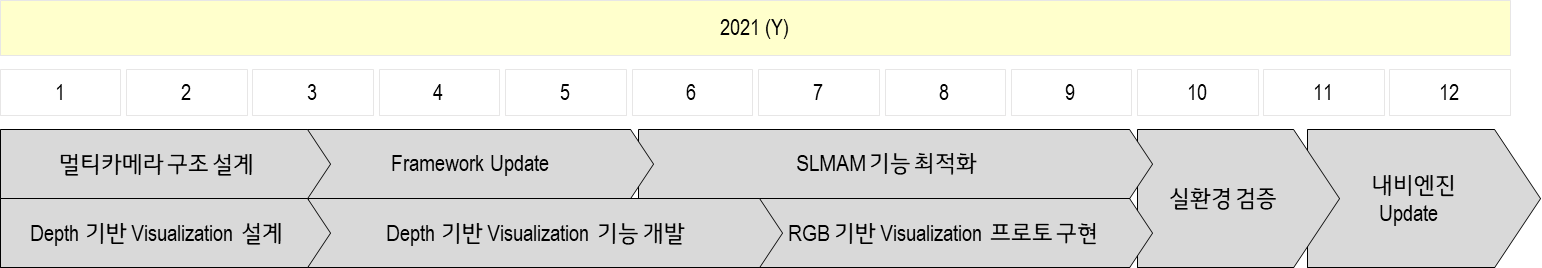
|  |  |
| --- | --- |
| 이름 | 업무 |
| 신재준 책임 | 개발 Leading, SW 구조설계, 성능평가항목 정리 |
| 김영한 선임 | Mapping 기술 개발 |
| 박찬수 선임 | SLAM 기능개발 |
| 이동학 선임 | Lidar 성능 향상 |

#### Stakeholders

|  |  |
| --- | --- |
| 이름 | 업무 |
| 설치담당자 | 현장에서 실제 설치를 하는 사람, Process 개선 효과가 있는지 판단 |
| QE담당자 | Mapping Robot으로 생성된 Map과 본 방식으로 생성된 Map의 성능을 비교 |
| 기구개발자 | Mapper를 기개발했던 인원으로 신규 Mapper의 개발 |
| System SW 담당자 | Lidar Firmware 및 성능에 대한 조언 |
| UI App 개발자 | Robot을 통해 Mapping을 해야하므로, 설치담당자가 사용할 UI app을 개발 |

#### Schedules

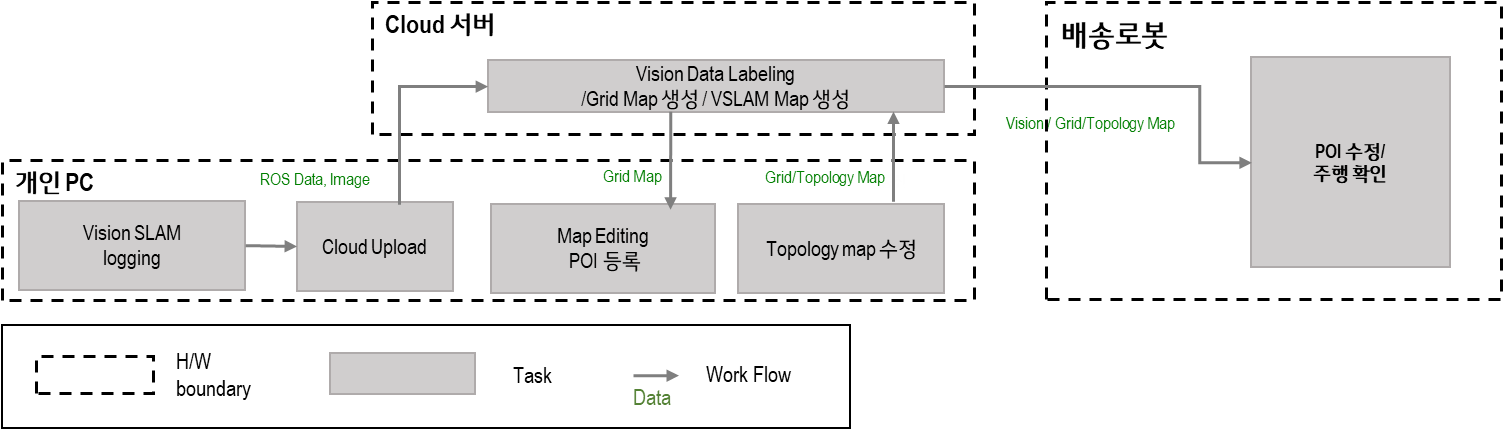
하기와 같은 일정으로 진행한다. (‘21년 상반기 개발 완료하여, 하반기 적용 목표)



<Figure 4>

# ****Architectural Driver****

****UX scenario Overview****



<Figure 5>

Mapping & Image Labeling의 최적화 방향을 정하기 위해 기존 Mapping 시나리오를 분석하였다. 기존 시나리오 기반으로는 <Figure 5>와 같은 단계로 설치는 진행되며, 최적화 필요한 Mapping은 VSLAM Logging, Vision Data Labeling, Grid Map 생성이 이에 해당되고, 이를 UX 시나리오로 정의하여 요구사항 정의 시에 사용하려 한다.

|  |  |  |
| --- | --- | --- |
| ID | Action | Description |
| UX-1 | Vision SLAM Logging | 조이스틱을 이용하여 로봇의 서비스영역을 이동시킴.  이동 시키면서 Lidar Data와 Image Data를 저장. |
| UX-2 | Grid Map 생성 | Lidar Data를 이용하여 Robot이 주행 가능한 지도를 생성. 주행 가능한 영역은 하얀색으로, 불가능한 영역은 회색/검은색으로 표시. |
| UX-3 | Vision Data Labeling | 저장된 Image data를 취득된 장소의 좌표 (x, y, 각도)로 정리하고 별도의 file(csv)형식으로 저장. |
| UX-4 | Map Editing | UX-3에서 생성된 Grid Map의 후처리를 진행하는 것으로 Lidar Sensor로 감지되지 않는 부분을 사람이 수기로 수정. |

### ****UX scenario –**** Vision SLAM Logging

|  |  |
| --- | --- |
|  | UX-1 |
| Use case title | Vision SLAM Logging |
| Actor | 현장 설치작업자 |
| Needs | Vision SLAM 위치추정에 필요한 Raw data를 수집(Logging)한다. |
| Precondition | 로봇에 카메라가 부착되어있는 모듈이 거치된 상태에서 진행한다. |
| Trigger | Logging UI를 동작하고, Joystick으로 로봇을 이동한다. |
| Post Condition | 이동영역에서 Vision Data를 수집한다. |
| Basic Flow | 현장설치작업자가 로봇에 모듈을 거치한다.  Logging UI를 동작하고 Joystick을 이용하여 로봇이 주행해야 하는 영역을 이동한다.  로봇이 주행해야 하는 영역을 모두 이동하면 Logging UI를 통해 작업을 종료한다. |
| Alternative | Mapping Robot을 사용하거나, 수동으로 Data를 수집해야 한다. |
| Exception | 중간에 Error가 발생하면 다시 처음부터 시작해야 한다. |

### ****UX scenario -**** Grid Map 생성

|  |  |
| --- | --- |
|  | UX-2 |
| Use case title | Grid Map 생성 |
| Actor | 현장 설치작업자 |
| Needs | Vision SLAM 위치추정에 필요한 Grid Map을 생성한다. |
| Precondition | VSLAM Logging이 되어있는 상태, Logging Data를 노트북으로 옮긴 상태 |
| Trigger | Cloud 서버로 Upload 한다. |
| Post Condition | Grid Map이 생성된다. |
| Basic Flow | Vision SLAM Logging종료되면, Raw Data를 노트북으로 이동한다.  노트북에서 Cloud서버 환경에 접속한다.  Cloud 서버로 Raw Data를 Upload한다.  Cloud 환경에서 Grid Map생성 UI를 통해 작업을 진행한다. |
| Alternative | Cloud 서버 대신 사용자 개인 노트북에서 해당 Script를 동작시켜야 한다. |
| Exception | Grid Map이 생성되지 않을 경우에는 Vision SLAM Logging을 다시 해야 한다. |

### ****UX scenario -**** Vision Data Labeling

|  |  |
| --- | --- |
|  | UX-3 |
| Use case title | Vision Data Labeling |
| Actor | 현장 설치작업자 |
| Needs | Vision SLAM 위치 추정에 필요한 Vision DB를 만드는데 필요한 Data를 생성한다. |
| Precondition | Grid Map이 있는 상태. |
| Trigger | Cloud 환경에서 UI 통해 작업을 진행한다. |
| Post Condition | Image file별로 취득된 위치가 별도의 file(csv)형식으로 저장된다. |
| Basic Flow | Cloud 환경에서 Grid Map생성 UI를 통해 작업을 진행한다. |
| Alternative | 사용자 Local PC에서 별도의 Script로 실행하여 진행한다. |
| Exception | Labeling Data가 생성되지 않을 경우에는 Vision SLAM Logging을 다시 해야 한다. |

### ****UX scenario -**** Map Editing

|  |  |
| --- | --- |
|  | UX-4 |
| Use case title | Map Editing |
| Actor | 현장 설치작업자 |
| Needs | 로봇이 고정 장애물을 파악하고 주행경로에 반영할 수 있도록 Grid Map을 수정 |
| Precondition | Grid Map이 생성되어야 하고, 고정장애물의 위치를 파악해야 한다. |
| Trigger | Map Editor를 실행시키면 |
| Post Condition | 고정장애물의 위치가 표시된 Grid Map |
| Basic Flow | Grid Map이 생성되면, 로봇 주행 구역 내 고정장애물의 위치를 파악한다.  고정장애물의 위치를 Map Editor를 통해 표시한다. |
| Alternative | Map Editor를 사용하지 않으면 그림판으로 수정 필요하다. |
| Exception | 장애물을 잘못 그리면 로봇 주행이 불가능할 수 있다. |

Constraints

주로 H/W 부품의 제약사항이 많으나, 이 문서에서 제약사항을 정의/기재 한다.

제약사항은 개발팀에서 선정한다. 비용은 요구사항이므로 제약사항으로 선정하지는 않는다.

### Business Constraints

|  |  |
| --- | --- |
| **ID** | Constraints |
| BC-01 | 안내로봇 구조 상 3D Lidar를 사용할 수 있는 Mapping Module을 신규 제작할 수 없다. |

### Technical Constraints

|  |  |
| --- | --- |
| **ID** | Constraints |
| TC-01 | **저렴한 Mapping장비의 비용**  Mapping Robot 1억, Mapping Module 1천만원 수준이나, 다양한 사이트를 동시에 설치하려면 이보다 가격이 낮아야 한다.  Robot에 부착된 2D Lidar를 사용해서 기존 Mapping Robot/Module과 동일한 성능을 확보해야 한다. |
| TC-02 | **Lidar 성능제약**  Mapping Robot/Module의 Lidar는 3D Lidar이기 때문에 수평 360도, 수직 약 120도의 영역의 Data를 수집할 수 있다. 하지만 BC-01의 제약처럼 안내로봇에는 3D Lidar를 사용할 수 없다. 별도의 장비나 Robot에 부착된 2D Lidar를 사용해야 하는데, 는 수평은 동일하나, 수직은 거의 0도에 가까워서 Lidar 부착 높이 (20cm)에 탐색되지 않는 장애물은 Map에 표시가 되지 않는다. Robot의 주행 가능한 영역이 실제와 다르게 표시될 수 있다. |

****Requirement Definition****

사전 정의된 UX 시나리오와 제약사항을 바탕으로 기능요구사항과 비기능 요구사항을 정의한다.

Mapping과 Image Labeling은 별도의 기획담당자가 없고, 설치/개발의 영역이기 때문에 설치담당자와 개발부서가 논의하여 요구사항을 정의하였다.

#### ****Functional Requirement****

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Description |
| FR-01 | **Grid Map과 Vision Data를 동시에 생성**  (UX-1, UX-2, UX-3단계를 한 번에 수행해야 한다) | 현재는 Mapper Robot이나 Mapping 모듈에서 VSLAM Logging 후 수집된 Data를 개발자 PC로 USB등을 통해 이동 한 후 Grid Map을 생성하고 Vision Data Labeling을 한다. Data 이동이나 별도의 PC를 사용하지 않고, Robot 자체적으로 Grid Map과 Vision Labeling Data를 생성해야 한다. |
| FR-02 | **로봇 주행공간 내 장애물이 표시**  (UX-4) | 별도의 Tool을 이용하여 장애물을 후처리 하지 않아야 설치시간이 줄어들기 때문에 Mapping Robot과 동일하게 주행공간 내 고정장애물은 별도의 처리 없이 표시되어야 한다. |

### ****Non Functional Requirement****

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Description |
| NFR-01 | 현장 설치담당자가 Mapping이 쉬워야 한다. | 타사 대비 Mapping 장비가 많고, Mapping단계가 많다.  장비 및 단계를 최소화해야 한다. |
| NFR-02 | 개발자 필요 M/M이 최소화 되어야 한다. | 개발일정 및 M/M이 여유롭지 않기 때문에 최소의 M/M으로 개발 진행해야 한다. |
| NFR-03 | 경쟁사 동일 수준의 설치 시간을 보장해야 한다. | 경쟁사는 3시간 이내 설치를 목표로 내세우고 있으나, 자사는 4.9시간정도가 소요되고 있다 |

## ****Quality Attribute definition****

도출된 요구사항과 제약사항을 기반으로 Quality Attribute를 정의한다.

Robot에 장착되어 있는 2D Lidar / Camera를 기반으로 Grid Map 생성과 Image 취득을 해야 하므로, 이와 연관이 있는 항목을 Quality Attribute로 선정하고, 그 업무 중요도에 따라 점수를 배점한다. 안내로봇의 개발일정은 선 정의되어 있으므로, 개발난이도는 평가의 지표로 삼지 않고, 필요성을 위주로 QA를 선정하고 그 내용부터 개발되도록 한다.

그 후의 내용은 MR때 반영한다.

### Quality Attribute Priority Metrix

#### Value Scale

|  |  |  |
| --- | --- | --- |
| Priority | Value | Score |
| High | 필수개선: 목표달성에 있어서 필히 반영(개선)되어야하는 항목 | 3 |
| Middle | 개선: 개선항목은 아니지만, 사용자에게 도움을 줄 수 있는 항목 | 2 |
| Low | 검토: 다른 방안으로 대체할 수 있는 항목 | 1 |

### Quality Attribute List

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Quality Attribute | Descriptions | Priority |
| QA-01 | 배송로봇 Navigation 재사용 | NFR-02에 해당한다. 최소의 M/M을 사용하기 위해 기존 배송로봇 기반의 코드를 재사용한다. | High |
| QA-02 | Mapping 시간 감소 | FR-01, NFR-01, NFR-03에 해당된다.  Lidar SLAM을 이용하여, Grid Map을 생성하고 50cm마다 Image를 Capture한다 | High |
| QA-03 | 뜬 장애물 가시화 기능 | FR-02, NFR-01에 해당된다.  2D Lidar는 20cm 높이의 장애물만 표시가 된다. 식탁과 같은 위와 밑이 다른 장애물도 표시가 되어야 한다 | Middle |
| QA-04 | 동일 SLAM 성능수준 확보 | TC-01, TC-02에 해당된다.  Lidar 변경에 무관하게 동일한 성능을 확보해야 한다. | High |

### Quality Attribute Scenario

#### QAS-01 배송로봇 Navigation 재사용

|  |  |
| --- | --- |
| **ID** | **QAS-01** |
| Narrative  Scenario | 개발기간의 단축을 위해 기존 Navigation 엔진의 구조를 동일하게 사용해야 한다.  기존 Code 동작에 영향을 주지 않는 범위에서 개발이 필요하다. |
| Quality  Attribute | Modifiability |
| Stimulus | Mapping후 고정장애물을 Grid Map에 표시할 때 |
| Source | 안내로봇 Navigation개발자 |
| Relevant  Environment  Conditions | 안내로봇 개발 시에 |
| System  Response | Navigation Engine내의 모듈 추가만 발생한다. |
| Response  Measure | 변경 모듈의 수 1 (SLAM 모듈만 변경) |

#### QAS-02 Mapping 시간 감소

|  |  |
| --- | --- |
| **ID** | **QAS-02** |
| Narrative  Scenario | Grid Map을 생성하고 50cm마다 Image를 Capture한다.  Capture된 Image에는 Grid Map상의 좌표가 기록된다.  이 작업은 동시에 진행되어 mapping 시간을 줄여야 한다. |
| Quality  Attribute | Usability |
| Stimulus | Mapping Application을 실행시키고, 조이스틱을 이용하여 Mapping을 진행한다. |
| Source | 현장설치인원 |
| Relevant  Environment  Conditions | 현장설치자가 초기설치를 계획할 때 |
| System  Response | Grid Map과 50cm마다 Capture된 Image, Image의 좌표가 생성 |
| Response  Measure | 기존 방식의 Mapping 시간 대비 10% 수준 감소되는지 확인 |

#### QAS-03 뜬 장애물 가시화 기능

|  |  |
| --- | --- |
| **ID** | **QAS-03** |
| Narrative  Scenario | 별도의 3D Lidar를 부착했을 경우에는 주위구조물들이 Grid Map에 표시가되는데, 2D Lidar는 20cm 높이의 장애물만 표시가 된다. 식탁과 같은 위와 밑이 다른 장애물도 표시가 되어야 한다. |
| Quality  Attribute | Usability |
| Stimulus | Mapping후 고정장애물을 Grid Map에 표시할 때 |
| Source | 현장설치인원 |
| Relevant  Environment  Conditions | 식탁과 같은 위와 밑이 다른 장애물이 있을 경우 |
| System  Response | Grid Map에 장애물의 가장 넓은 부분이 표시되어야 한다. |
| Response  Measure | 장애물의 크기가 실제 물리적 크기와 10% 수준의 오차를 보이는지 확인 |

#### QAS-04 동일 SLAM 성능수준 확보

|  |  |
| --- | --- |
| **ID** | **QAS-04** |
| Narrative  Scenario | Capture된 Image가 가지고 있는 좌표의 정확성을 증가 |
| Quality  Attribute | Performance |
| Stimulus | Mapping Application을 실행시키고, 조이스틱을 이용하여 Mapping을 진행 |
| Source | 현장설치인원 |
| Relevant  Environment  Conditions | 로봇만으로 현장설치자가 초기설치를 계획할 때 |
| System  Response | Grid Map과 50cm마다 Capture된 Image, Image의 좌표가 생성 |
| Response  Measure | 해당 Image의 좌표가 실제 물리적 좌표의 오차가 15cm 이하로 측정되는지 확인 (중형배송로봇 위치오차와 동일) |

# ****Architectural Design****

## ****Architectural Design - QAS-0****1

자사 중형배송로봇의 Navigation Engine을 분석하여 Mapping과 Image Labeling을 동시에 하기 위한 방안을 도출한다.

안내로봇도 기본적으로는 중형배송의 Navigation Engine 기반으로 제작이 될 예정이므로, 기존 코드를 분석하여 구조와 Data Flow를 확보한다.

### ****Design Discussion****

|  |  |  |
| --- | --- | --- |
| Title | Idea #1 | Idea #2 |
| Description | 중형배송 로봇 Navigation Engine의 SLAM 모듈 기반으로 Labeling을 할 수 있는 모듈을 추가하여 구현한다. | Mapping Module에서 동작하고 있는 ROS기반의 Data Gathering 모듈을 Navigation Engine에 Porting한다. |
| 장점 | 개발범위가 상대적으로 적고 개발자들이 익숙한 Platform이다. | 기존 코드에 영향이 가지 않고, 단독으로 테스트 및 검증이 가능하다. |
| 단점 | 기존코드의 분석이 필요하며, 기존 동작에 영향을 주지 않는지 추가 검증이 필요하다. | ROS기반의 개발이 익숙하지 않고, H/W Chipset의 성능에 영향을 받는다. |
| 고려사항 | 개발자의 편의성, 시스템의 안정성 | |
| 기타 |  | |

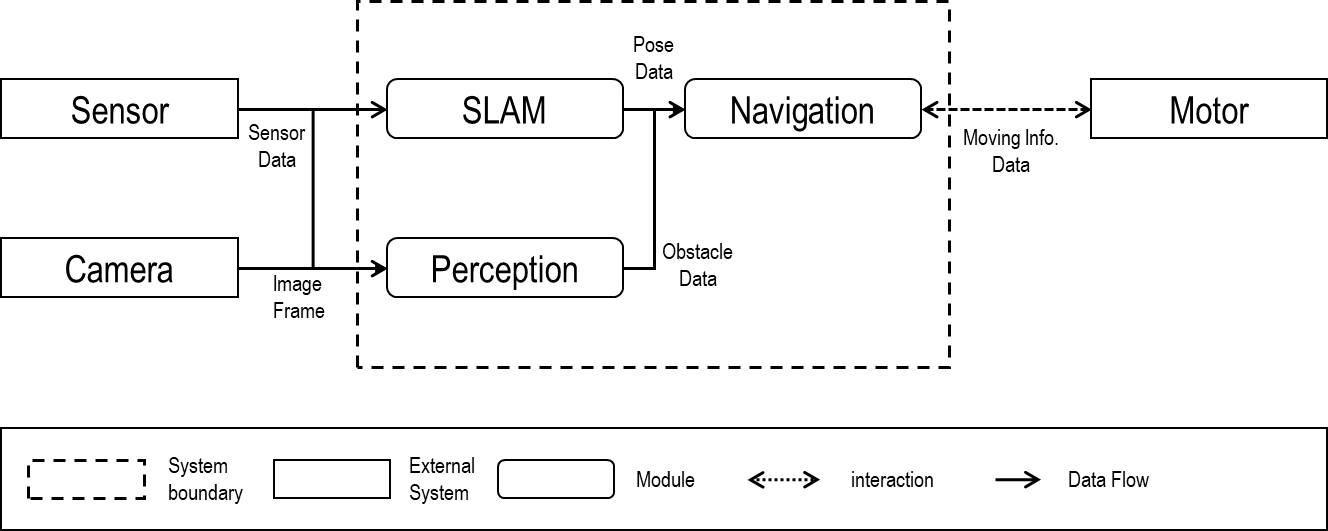
기존 Mapping Module에서는 ROS기반의 단독 Application을 사용하여 Mapping을 진행하고 있었다. 이를 중형배송 로봇 Navigation Engine에 Porting하려면 하기와 같은 문제점이 발생한다.

1) Navigation Engine의 Middleware는 자사에서 개발한 LGRP (LG Robot Platform)을 사용하고 있다. 여기에 ROS를 Porting하게 되면 2개의 Middleware가 동작하게 되어 시스템의 성능 저하 및 높은 복잡도가 예상된다.

2) 기존 소스의 Converting(ROS에서 LGRP로)을 하려면 3D Lidar Driver를 2D Lidar Driver로 Porting하는 등 기존의 BSP Level의 재사용이 불가하고 신규로 Data Path를 생성해야 한다.

이에 Idea #1을 채용하여 진행하기로 하였다.

### System Context

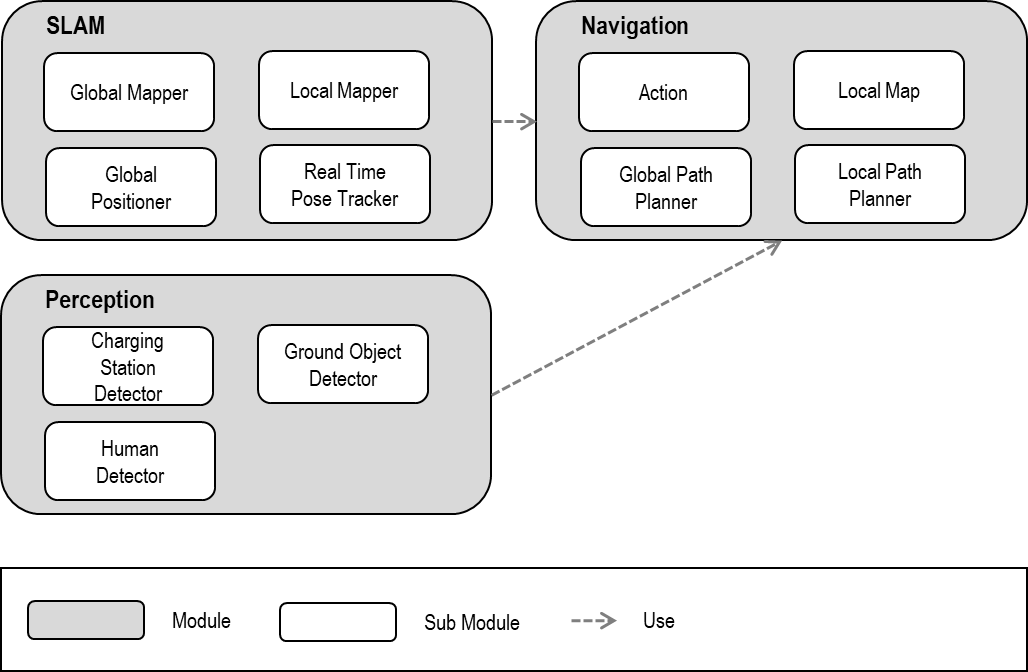


<Figure 6>

중형배송로봇 Navigation Engine의 System Context를 분석하였다. 로봇에 부착된 센서와 카메라에서 센서 Data (lidar, USS, 3D Depth Info.)와 Image data를 로봇의 위치를 추정하는 SLAM 모듈과 로봇의 장애물을 감지하는 Perception 모듈로 전달한다. SLAM 모듈에서는 현재 위치를 추정하고, Perception 모듈에서는 로봇 주위의 장애물을 감지하여 Navigation 모듈로 전달한다. Navigation 모듈에서는 전달된 위치와 장애물 정보를 이용하여 로봇이 이동할 경로를 생성하고, 이 경로에 맞게 Motor에 이동명령을 전달한다.

이동명령을 받은 Motor 구동부는 현재 이동상태와 이동결과를 다시 Navigation 모듈에 전달한다.

### Static Perspective



<Figure 7>

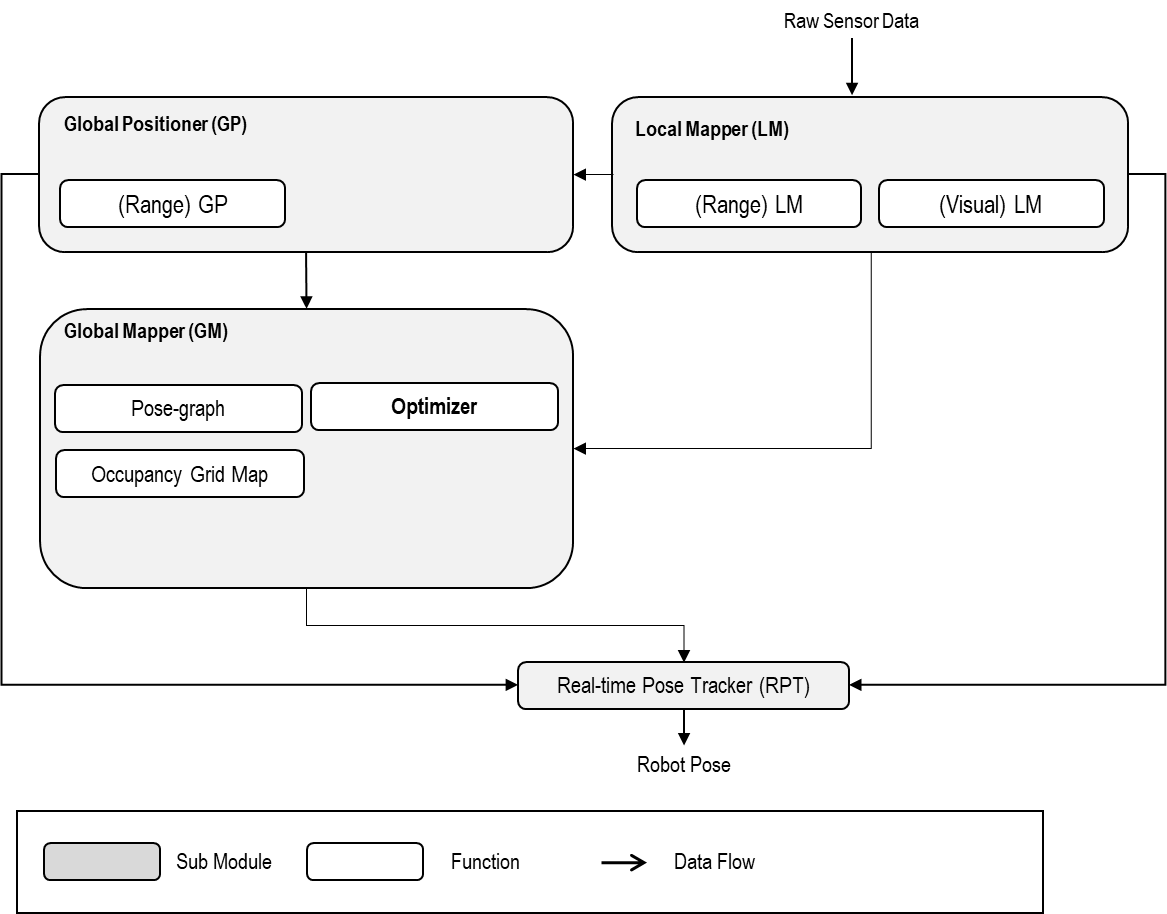
Navigation Engine각 Module별 Sub Module을 분석하여 하기와 같이 정리하였다.

|  |  |  |
| --- | --- | --- |
| Module | Sub Module | Description |
| SLAM | Global Mapper | Local Mapper로부터 Node 및 누적된 Node로부터의 상태 위치를 전달 받아 Pose Graph 및 Occupancy Grid Map을 생성하고 최적화를 수행한다. |
| Local Mapper | 연속된 Frame간의 상대적인 거리와 각도를 계산하는 모듈이다. 입력된 스캔 데이터에서 이전 스캔 데이터와 ICP를 통해 상대 좌표를 계산하게 된다 |
| Global Positioner | Global 좌표계 기준의 절대 위치를 추정하는 모듈이다. |
| Real-time Pose Tracker | 현재 로봇 위치를 추정하는 모듈이다. LM, GM, LD결과와 이전 Pose 등의 정보를 최적화하여 위치를 추정한다 |
| Navigation | Action | 로봇 주행 기능을 동작하기 위한 Action을 수행하는 서비스로  이 경로 기반으로 지정된 Action 기반으로 속도 명령(v, w)를 생성한다. |
| Local Map | 각 센서 데이터 및 주변 환경 인식 모듈로부터 로봇 주변 정보를 수집하여 로봇 일정 반경 이내 영역에 대한 격자 지도 작성한다.  데이터를 구분할 수 있도록 Layer Map (index map) 형식으로 publish하며, Local Map은 100ms 마다 update/publish 한다. |
| Global Path Planner | grid Map (static map) 내 두 지점 (start/goal)의 경로를 생성하는 모듈이다. 동적 장애물은 고려하지 않고, grid Map내의 주행가능영역으로 경로를 생성한다.  전역 경로 계획을 요청 시 경로 계획 1회 수행한다. |
| Local Path Planner | Global Path를 따라서 목표지점까지 이동하기 위해 현재 위치와 주변 장애물 정보를 고려하여 장애물을 회피하기 위한 경로를 생성한다. Global Path를 따라서 주행할 수 없을 경우 지속적으로 동작한다. |
| Perception | Ground Object Detector | 장애물과 낭떠러지 인식을 한다.  로봇 기울어짐을 인식하고 및 장애물 검출 반영하여 Local Map으로 전달한다.  센서값을 전달받으면, Point Cloud 변환하고 Ground Plane 인식하고 그 외의 부분은 Point Cloud Transform  하여 장애물 정보를 Update하고 publish 한다. |
| Station Detector | Point Cloud의 3D ICP 를 통한 충전대 자세 검출 |
| Human Detector Service | Darknet (Yolo V3) 를 이용한 사람 검출 |

본 프로젝트의 목적을 달성하기 위해서는 Navigation Engine내의 SLAM 모듈과 Navigation모듈 (Local Map)을 활용하거나 수정해야 할 것으로 예상되므로, 이 부분에 대한 내용을 상세 분석한다.

### Dynamic Perspective

#### SLAM

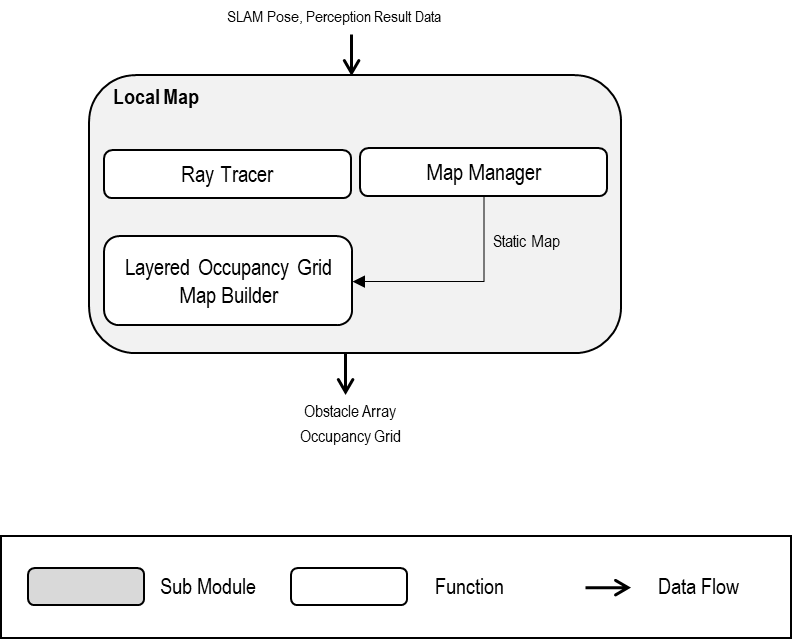


<Figure 8>

SLAM의 Data Flow는 위와 같은 구조로 되어있다. Camera와 Range, Odometry를 입력받아 Local Mapper가 주기적으로 동작하면서 Global Mapper에게 Node data를 전달하고 연결된 Node를 이용하여 Grid Map을 생성한다. 최종적으로 Optimizer 모듈이 동작하면서 Grid Map에 대한 최적화를 수행한다.

Image Labeling을 하려면 SLAM의 최종산출물인 Robot Pose를 사용해야 할 것으로 예상된다.

#### Navigation - Local Map



<Figure 9>

로봇을 중심으로 지역 장애물 정보를 가지고 있는 Map (Local Map)을 생성하는 모듈이다. 주기적으로 갱신되면 서 실시간으로 지역 장애물 정보를 갱신한다. (Occupancy Grid, Obstacle Array)

로봇에 부착되어있는 센서마다 하나의 Layer map을 만들고 이를 합쳐서 Local Map을 생성한다. 센서 구성별로 각 layer를 Enable/Disable 시킬수 있다. Local map은 16bit, 2D array로 각 bit가 각 장애물 들을 의미한다.

이 Local Map을 사용하면 실시간으로 장애물을 표시하는데 사용할 수 있다.

## ****Architectural Design - QAS-02****

### ****Design Discussion****

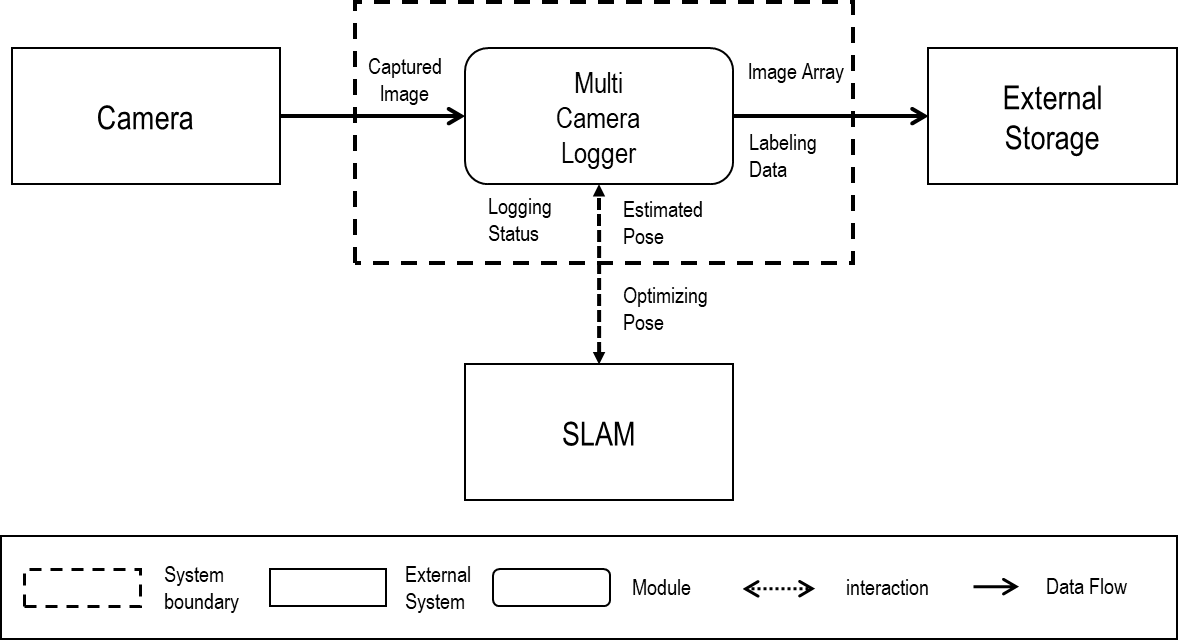
Mapping / Image Labeling 동시진행을 통한 Mapping 시간 감소에 대한 설계 아이디어를 도출하였다.

|  |  |  |
| --- | --- | --- |
| **Title** | **Idea #1** | **Idea #2** |
| Description | 기존 SLAM 코드에서 Graph 생성시마다 Image를 생성하도록 수정하게 한다.  멀티카메라에 대한 입력을 받게 하여 여러 각도의 Image를 수집하게 한다.  멀티카메라는 로봇 Mapping시에만 별도 Mapping Module을 부착하여 사용할 수 있게 한다. 별도의 Lidar는 부착하지 않는다. | 기존 SLAM 코드에서 Graph 생성시마다 Image를 생성하도록 수정하게 한다.  로봇이 회전하면서 이미지를 수집하여 여러 각도의 Image를 수집하게 한다. |
| 장점 | 현재 코드의 최소한의 수정 | 현재 코드의 최소한의 수정  별도의 모듈비용이 발생하지 않는다. |
| 단점 | 별도의 모듈이 들어가서 비용이 발생한다. | Mapping 시간이 과도하게 발생할 수 있다. |
| 고려사항 | 최소의 비용이 들어가야 한다. | |
| 기타 |  |  |

기존 SLAM Code에서 Graph Node 생성 시마다 저장하는 시점에 Image를 저장하고, 저장하는 시점의 좌표를 기록하는 방안으로 Idea를 도출하였다. 문제는 여러 각도의 이미지를 어떻게 수집할 것 인가였는데, 로봇이 회전하면서 수집하는 Idea #2 방안은 Mapping을 간소화하겠다는 본래의 취지에 어울리지 않기 때문에 별도의 Module을 생성하여 적당한 비용을 지불하면서 시간을 확보하는 Trade –off를 취할 수 밖에 없다.

Idea #1을 선정하기로 하고, 멀티카메라를 사용할 수 있는 외부 H/W도 설계하도록 한다. 외부 H/W는 저렴하게 구성할 수 있도록 최소한의 H/W만 반영하도록 한다. Idea #1 도출시에는 안내로봇과 동일한 Camera를 부착하는 것으로 논의되었고, 그 제품은 C3D Depth Camera이므로 그 비용이 160,000원 정도 예상되었다. 멀티이미지에 대한 입력을 받기 위해서는 최대 5개 정도의 카메라가 필요할 것으로 예상된다.

### ****System Context****



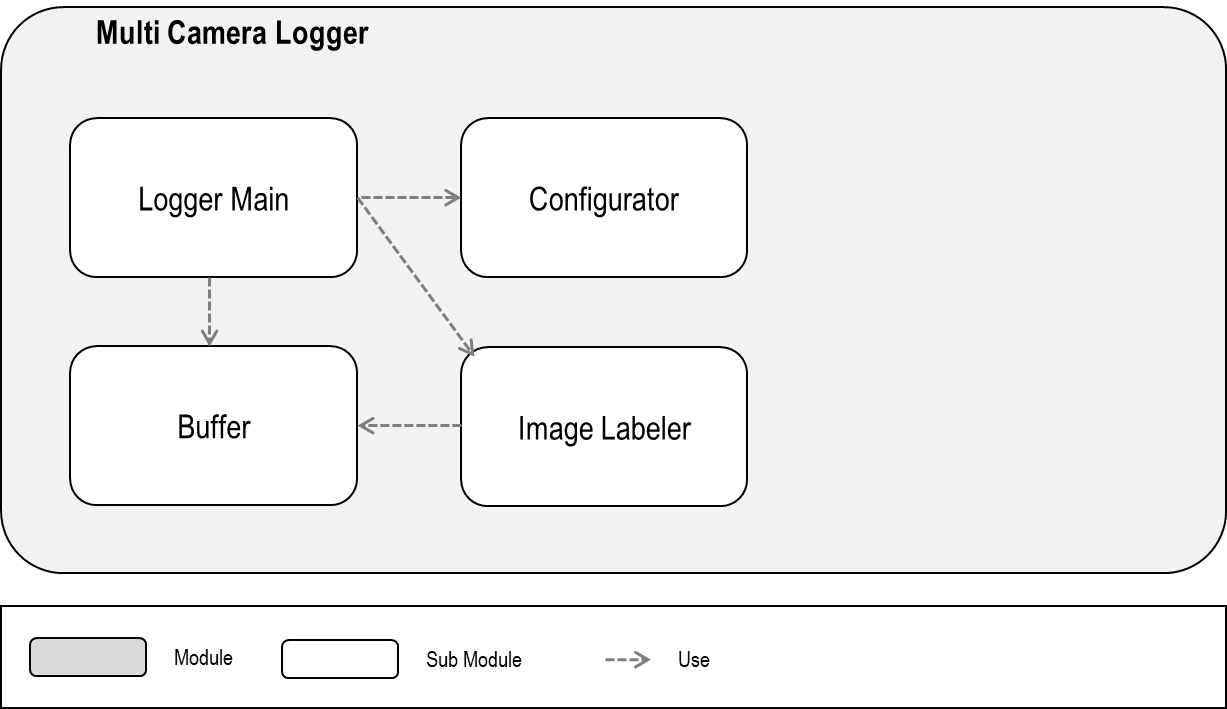
<Figure 10>

멀티카메라의 이미지와 SLAM 모듈에서의 Pose data를 이용하여 Image를 저장하는 방식으로 설계 진행이 필요하다.

하기와 같은 명세를 하였다.

|  |  |  |
| --- | --- | --- |
| Name | Description | Data |
| Multi camera Logger | MCL은 주기적으로 Camera 이미지와 Grid Map을 받아서 이미지를 저장하고, 각 이미지가 촬영된 위치를 기록하는 역할을 한다. | [Input]  Camera Image  SLAM Pose  [Output]  Camera Image Array  Camera Image Position Info. | |

### Static Perspective (Module View)



<Figure 11>

Camera에서 주기적으로 수집된 Image를 저장하는 모듈을 추가로 제작하고, GM에서 추출된 Grid Map과 Pose Data를 이용하여 Labeling하는 기능을 개발한다.

|  |  |
| --- | --- |
| Name | Description |
| Logger Main | 서비스 형태로 설계되어 SLAM과 상태 Message를 주고받고, Camera 이미지를 받아 Buffer에게 전달하는 역할을 한다. | |
| Configurator | 멀티카메라의 정보와 Calibration값 등을 저장하는 역할을 한다. | |
| Buffer | 주기적으로 Camera 이미지와 Pose Data를 저장한다. | |
| Image Labeler | Buffer에 저장된 Camera Image와 Pose Data를 이용하여 특정 Format으로 파일이 압축되도록 한다.  SLAM에서 최적화된 Data를 받으면 다시 한 번 Labeling을 하여 그 정확도를 높인다. | |

|  |  |  |
| --- | --- | --- |
| AA\_BB\_F1\_vision.tar | colmap | \*.jpg |
| pose | optimized\_pose.txt |
| cal | camera.yml |
| **verification** | \*.jpg |

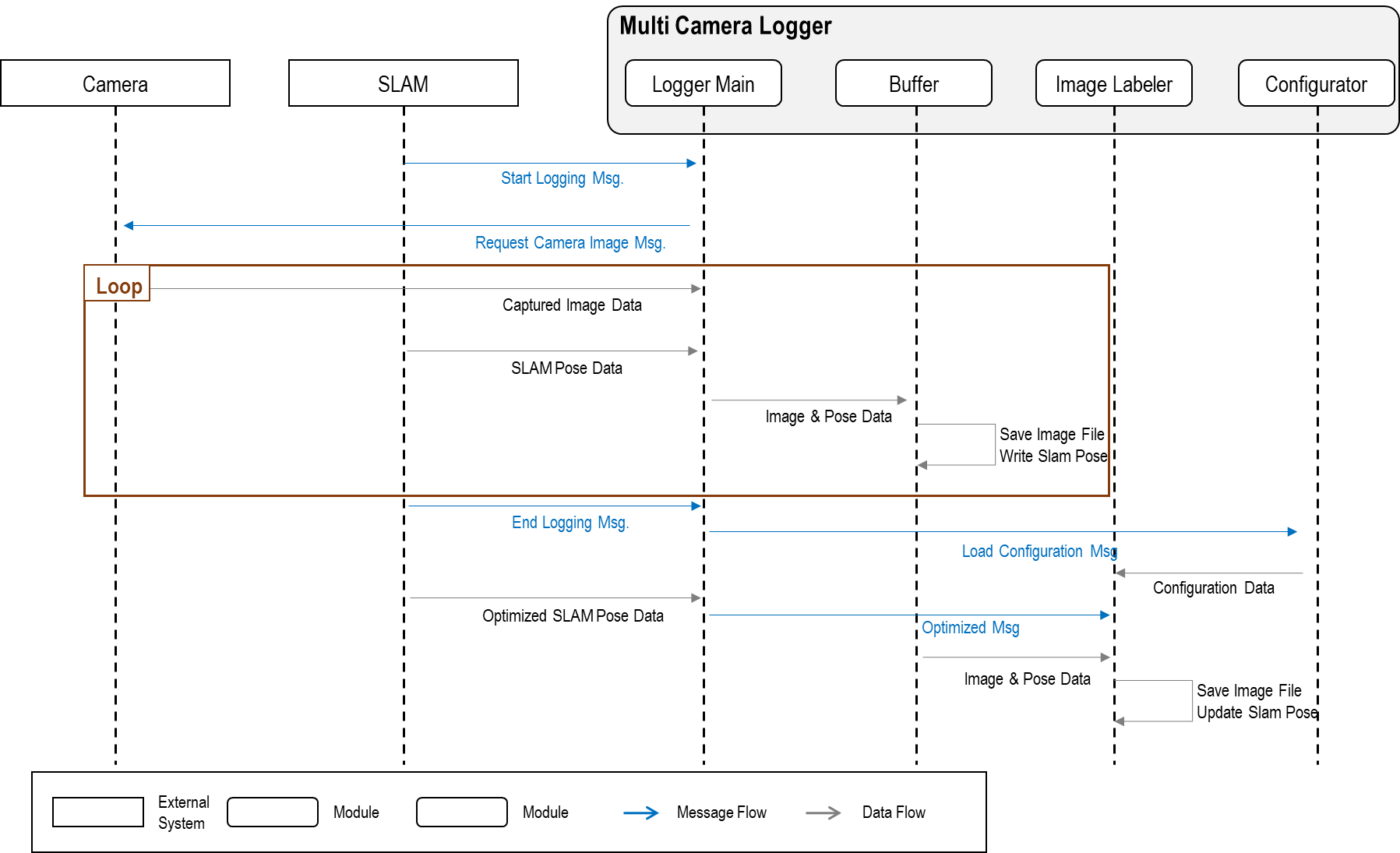
파일 Format은 위와 같은 형식으로 진행한다.

### Dynamic Perspective (C&C View)

Logger main은 SLAM 에서 지도 제작을 시작하면서 Logging 시작을 요청하면 Camera에 복수의 카메라 영상(Multi-camera image)을 요청하면서 시작되며 주기적으로 Camera에서 Multi-camera image를 입력 받는다.

로봇이 지도를 제작하면서 SLAM 에서는 일정한 거리마다 Node를 생성하며 이를 pose-graph에 등록한다. 새로운 Node가 등록되면 SLAM는 Logger main에 로깅을 요청하면서 새롭게 등록된 Node의 Pose Data를 전달한다. Logger main은 입력 받은 Node의 timestamp를 기준으로 Buffer에 전달하고 Buffer는 복수의 카메라 영상을 찾아 indexing하고 Pose 정보를 기록한다.

지도 제작이 종료되면 SLAM은 로깅 종료를 Logger main에 요청하고 현재 최적화 된 pose-graph를 Logger main에 전달한다. Logger Main은 최적화 된 Node 및 pose-graph와 Image와 Pose 정보를 Image Labeler에 전달하고 Image Labeler는 최적화 정보를 활용하여 Image와 Pose 정보를 재구성 하여 File 형태로 저장할 수 있게 한다.



<Figure 12>

### Interface Design - H/W 개발



<Figure 13>

설계 아이디어 도출 시에는 안내로봇과 동일한 Camera를 부착하는 것으로 논의되었고, 그 제품은 C3D Depth Camera이므로 그 비용이 160,000원 정도, 최대 5개 정도의 카메라가 필요할 것으로 예상되었으나, 비용 문제로 RGB Depth기능이 없는 RGB Camera를 부착하고, 그 개수도 2개로만 제한하였다.

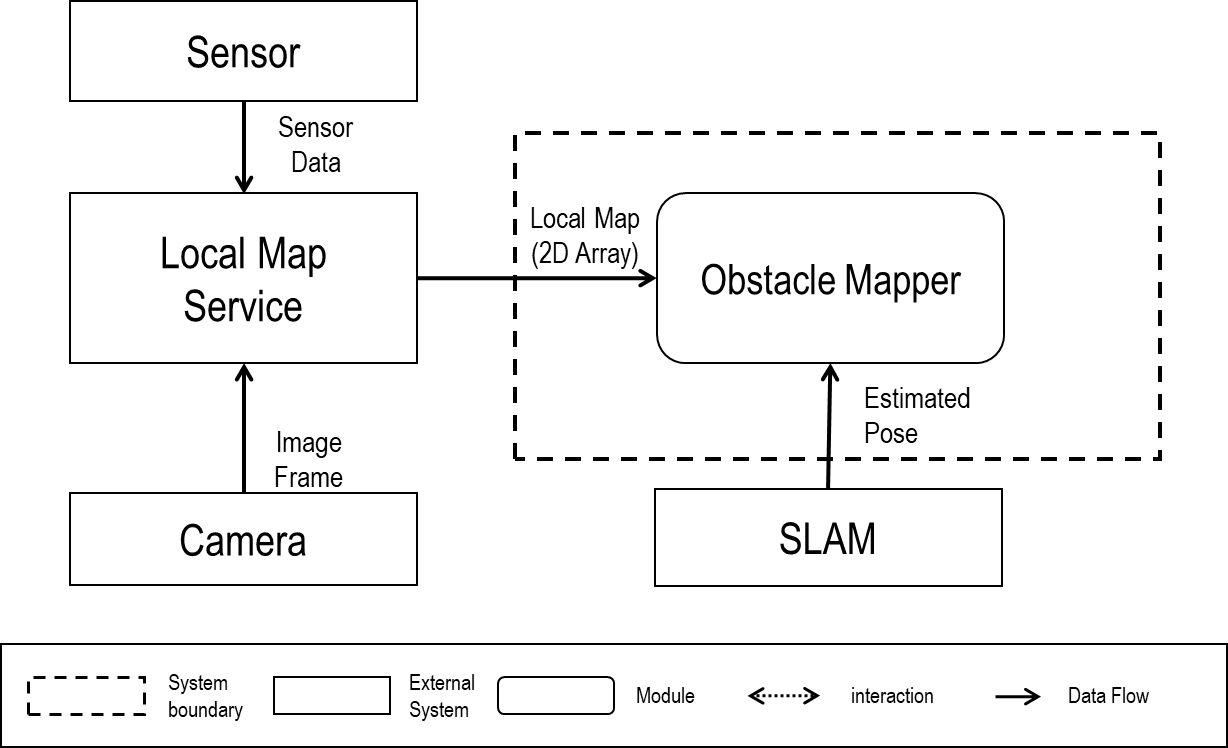
## ****Architectural Design - QAS-0****3

### ****Design Discussion****

|  |  |  |
| --- | --- | --- |
| Title | Idea #1 | Idea #2 |
| Description | Mapping 진행하면서 Grid Map과 Local Map을 사용하여 별도의 Obstacle Map을 생성한다. | Mapping후 Grid Map을 검토하고, 장애물이 누락된 부분에는 Local Map 중 3D Camera를 이용한 부분만 발췌하여 별도의 Obstacle Map을 생성하고, 이를 Overlay 시켜 Grid Map을 Update하게 한다. |
| 장점 | Map 생성시간을 최소화 할 수 있다. | 성능이 상대적으로 안정적이다. |
| 단점 | 지나가는 사람과 책상다리 등도 표시될 수 있어 후처리가 필요할 수 있다.  로봇의 성능상 실제와 다른 Data가 표시될 수 있다. | 상대적으로 장애물의 정확도가 떨어진다. 뜬 장애물만 표시할 수 있다. |
| 고려사항 | 설치담당자의 편의성 | |
| 기타 |  | |

뜬 장애물 대응을 위해 로봇의 장애물 감지방식을 차용하는 방식으로 진행을 하는 것에는 동의하였으나, 실시간으로 생성을 할 것인지, 필요부분만 후처리 할 것인지가 주요 의사결정 항목이었다. 고려해야 할 항목은 로봇의 형태와 운영공간이었다. 이 프로젝트의 목적은 안내로봇이므로 안내로봇은 뜬 장애물이 기존 배송로봇 보다는 적을 것이고, 건물의 로비 등을 돌아다니기 때문에 그 이동공간도 비교적 넓은 편으로 확보 될 것 이기 때문에 Idea #2를 사용하는 것으로 개발 조직 / 설치담당자가 합의 하였다.

### ****System Context****

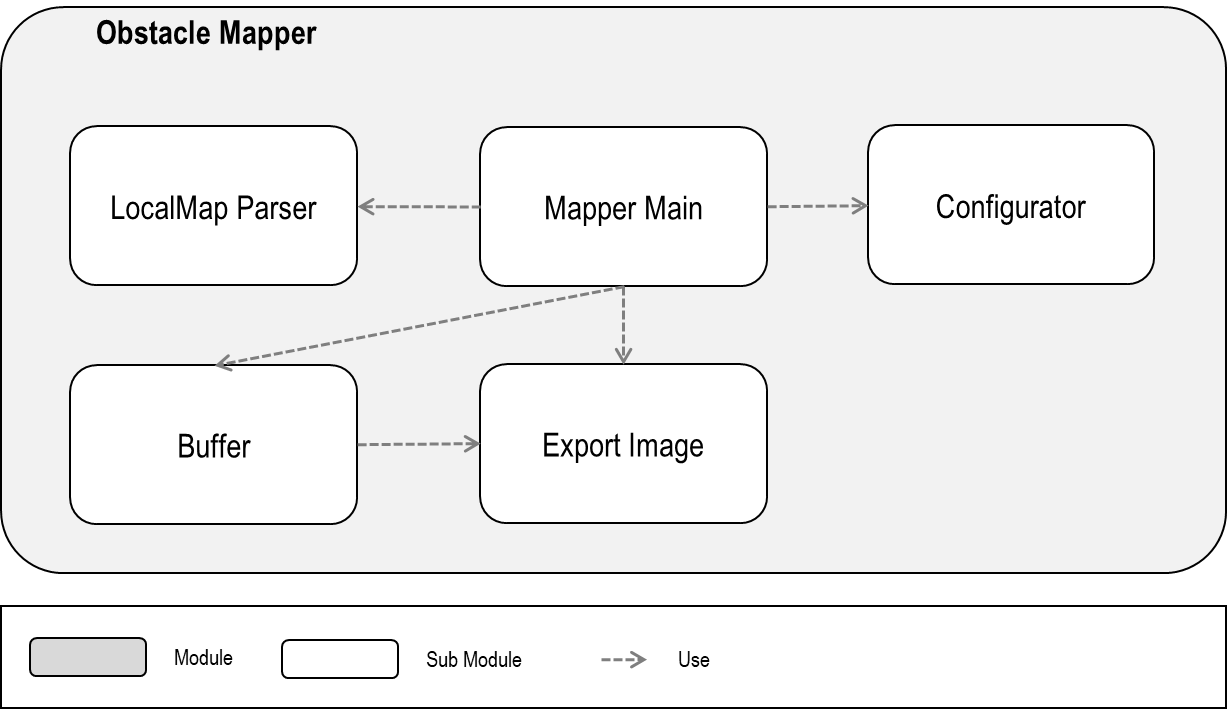


<Figure 14>

Mapping이 끝난 후 조이스틱으로 주행하면서 필요한 부분에만 장애물 Map을 생성할 수 있도록 설계가 필요하며, 하기와 같이 명세하였다.

|  |  |  |
| --- | --- | --- |
| Name | Description | Data |
| Obstacle Mapper | 3D 센서의 Ground object 기반의 장애물Map을 추가로 생성하여 Grid Map과 Overlay 할 수 있도록 이미지로 구현 | [Input]  장애물정보  SLAM Pose  [Output]  장애물 이미지 파일 |

### Static Perspective (Module View)



<Figure 15>

추가적으로 Obstacle Mapper를 생성하여 별도의 3D점유 Grid Map을 생성할 것이고, 그 명세는 다음과 같다.

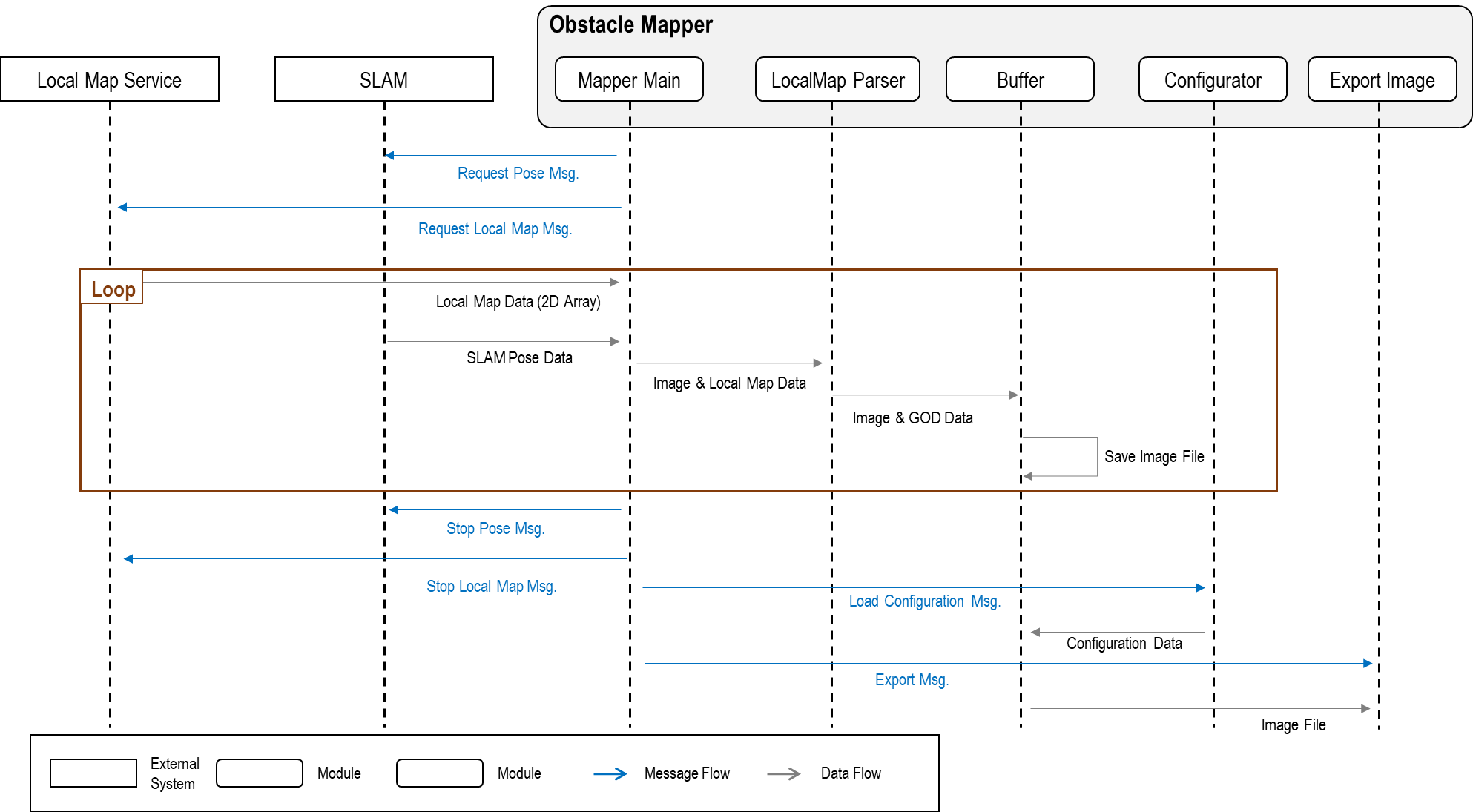
|  |  |
| --- | --- |
| Name | Description  Data |
| Mapper Main | 서비스 형태로 설계되어 Local Map 정보와 SLAM Pose를 받아 LocalMap Parser로 전달한다. | |
| Configurator | 센서 구성과 운영형태에 따라 사용할 LocalMap의 정보를 설정한다. | |
| Buffer | Parsing된 LocalMap정보와 위치를 저장한다. | |
| LocalMap Parser | LocalMap정보를 수신하여 그 중에서 설정값에 따라 Data 분리한다. | |
| Export Image | Buffer에 저장된 LocalMap정보와 위치를 이용해 실제 설치담당자가 사용할 수 있도록 .jpg 형태로 저장한다. | |

### Dynamic Perspective (C&C View)

Mapper main은 Obstacle Map 제작을 시작하면서 SLAM과 Local Map Service에 현재 Robot의 위치와 Local Map 정보를 요청한다. 이는 주기적으로 입력 받는다.

LocalMap Parser에서는 로봇이 이동하면서 수신되는 Local Map 정보에서 기 설정된 Sensor (3D Camera GOD)만을 분리하고, 그 정보와 위치를 Buffer에 저장한다.

Obstacle Map 제작이 Mapper main은 더 이상 정보를 수신하지 않고, 현재 Buffer에 기록된 Data를 JPG 형태로 제작하여 Export한다.



<Figure 16>

### Scenario Design – 사용 시나리오 정의

1) Mapping 후, 총 2개의 Map 이미지가 생성된다. (Grid Map과 Obstacle Map이 생성된다.)

2) 현장설치 담당자는 Obstacle Map을 참고하여 Grid Map에 테이블, 의자 등 장애물 영역을 설정한다.

3) 장애물 영역 설정 후 Grid Map을 저장하여 사용한다

Implementation - QAS-04

QAS-04는 기존 Legacy구조에서의 S/W Architecture적인 변경 없이, 알고리즘 및 Parameter 변경으로 대응가능하기에 개선사항만 기재한다. 본 문서는 설계문서이므로, 구현사항을 상세히 기재하지는 않는다.

### Lidar SLAM 성능 향상

로봇에 부착되어있는 Lidar는 2D Lidar 이다. Mapping Robot이나 Module에 부착되어 있는 3D Lidar에 비해서는 감지할 수 있는 영역과 그 정확도가 상대적으로 떨어진다. FR-02를 만족하려면 Lidar SLAM의 성능이 향상되어야 하기에 하기와 같은 항목을 선정하여 개선 진행하였다.

* Global SLAM 성능 개선 및 Grid map 작성 방법 개선으로 이중 벽 현상을 최소화
* Local map 작성 시 확률 기반 outlier 제거: 동적 장애물 및 반사체의 잔상 감소
* Local SLAM 개선에 따른 Local map 누적시 정확도 향상
* 대면적 mapping 을 위한 Map 구조 개선: 사용 메모리 감소 및 LiDAR Loop detection 연산 속도 향상
* Voxel filter 적용: Local map 경량화

### S1 Lidar 대응

FR-02를 만족하기 위해 Lidar 특성에 대한 수정 개발을 진행하였다. 중형배송로봇에는 S1 Lidar가 부착되어 있는데, 안내로봇에도 동일한 Lidar로 개발 논의되고 있다. 이를 대응하기 위해서 S1 Lidar의 특성 (분해능과 회전속도의 반비례)을 고려하여 SLAM정확도를 향상 시킬 수 있는 보정 알고리즘을 개발하였다.

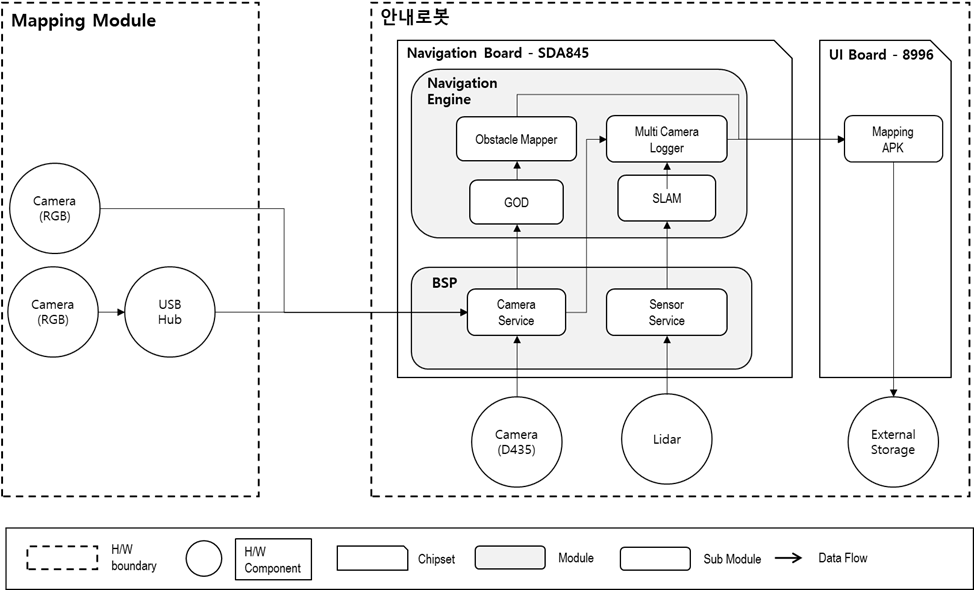
* S1 LiDAR distortion correction
  + -135° ~ 135° scan data 측정까지 첫-끝 data간 93.75ms 차이 발생 (8hz)
  + 로봇의 이동속도가 빨라짐에 따라 회전하며 취득하는 LiDAR data의 distortion 발생 -> 위치 오차 영향
  + LiDAR data frequency 사이의 Odometry 이동량 및 LiDAR resolution에 따른 비율을 고려하여 distortion 보상

### 고속주행 대응

NFR-02를 대응하기 위해 Odometry / 카메라 / LiDAR 처리 프로세스 Thread blocking을 해제하였다.

## ****Architectural Design – Overall Design (Allocation View)****

신규 H/W(Mapping Module)가 제작되었으므로, 각 H/W간 Data flow와 Module 구성을 정의하였다.



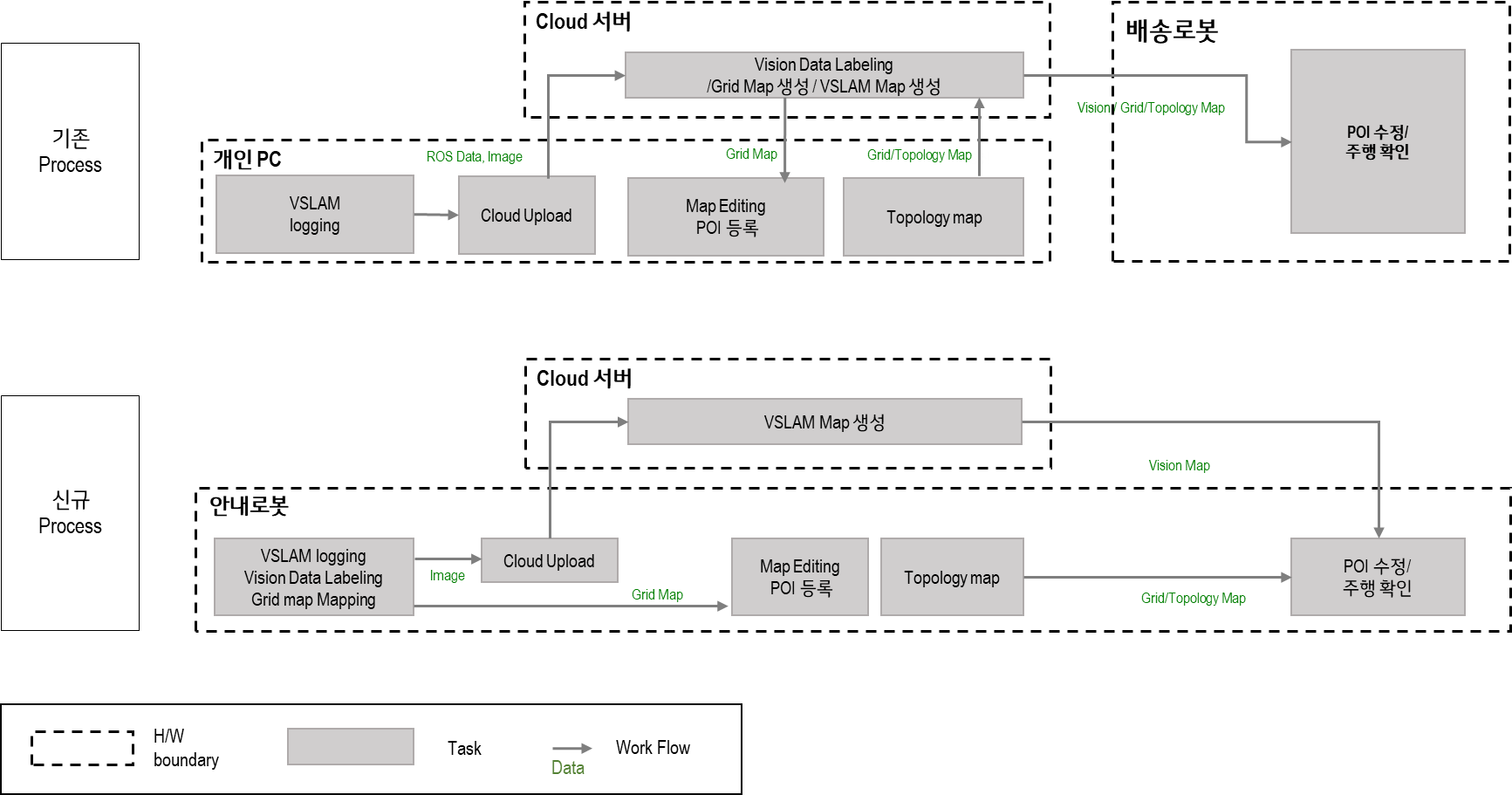
<Figure 17>

## Mapping Process 수립

Mapping Process는 중형배송로봇 초기에 ‘로봇 설치/셋업 Task’ 진행을 통해서 수립되었다. 하기 figure 18에서의 기존 Process가 그 부분이다. (Task 완료보고 문서 추가)

H/W 변경 및 Process의 변경이 발생하여 안내로봇 Mapping Process를 수립하였고, 이를 설치담당자에게 교육하였다.

기본적으로 설치담당자가 해야 하는 Task는 크게 다르지 않지만, 여러가지 H/W를 사용하면서 Data를 이동하는 번거로움은 줄어들도록 Process 수립하였다.



<Figure 18>

# ****Validation & Verification****

Quality Attribute 결과

본 프로젝트 초반에 정의한 QA의 진행결과를 하기와 같이 정리하였다.

자체평가는 개발팀 자체로 내부논의 하였고, 5점을 만점으로 하였다,

### 배송로봇 Navigation 재사용

QAS – 01에 대한 결과를 하기와 같이 정리하였다.

|  |  |
| --- | --- |
| 항목 | 설명 |
| 진행결과 | 중형배송로봇의 Navigation Engine기반으로 작성하여 코드의 변화를 최소화 하였다. 기존 SLAM모듈에만 수정을 하여 다른 Navigation Engine내의 모듈에 영향을 주지 않았다. SLAM 모듈 안에서도 기존 모듈의 기능을 수정하지 않고, 별도의 모듈로 제작하여 변화량을 최소화 하였다. |
| 자체평가 | **4점**  안내로봇에 실제 적용완료 하였다. |
| 기타 |  |

### Mapping 시간 감소

QAS – 02에 대한 결과를 하기와 같이 정리하였다.

|  |  |
| --- | --- |
| 항목 | 설명 |
| 진행결과 | SLAM에서의 Grid Map 생성 시, Image Labeling 기능 적용하여, Mapping과 Image Labeling을 동시에 진행 가능하도록 적용완료 하였다. 다양한 각도에서의 Image가 필요하기 때문에 신규로 Mapping Module을 제작하였고, 이에 따른 설치 Process를 최적화하였다.  기존 Mapping Module은 외부 노트북에서 Data를 생성하여 Mapping과 Image Labeling이 동시에 되지 않기 때문에 금번 구현사항 대비 시간이 더 소요된다. |
| 자체평가 | **4점**  안내로봇에 실제 적용완료 하였고, Mapping 시간 감소 확인하였다. |
| 기타 |  |

### 뜬 장애물 가시화 기능

QAS – 03에 대한 결과를 하기와 같이 정리하였다.

|  |  |
| --- | --- |
| 항목 | 설명 |
| 진행결과 | 3D Depth 기반 장애물 인식과 SLAM 노드 연결하여 뜬 장애물 가시화 기능 개발하였다. |
| 자체평가 | **2점**  GOD 성능의 제약으로 장애물 표현의 정확도 저하와 Mapping 작업 동시에 생성되는 것이 아닌, Mapping후 추가작업이 들어가야 한다는 사유로 크게 사용되지 않았다. |
| 기타 |  |

### 동일 SLAM 성능수준 확보

QAS – 04에 대한 결과를 하기와 같이 정리하였다.

|  |  |
| --- | --- |
| 항목 | 설명 |
| 진행결과 | 로봇 회전 시 LiDAR 데이터 Distortion 영향 최소화 위한 알고리즘 적용하였다. |
| 자체평가 | 2점  어느정도 개선사항은 확인되었으나, 다양한 Mapping을 진행하면서 성능의 한계점이 확인되어 H/W를 교체하였다. |
| 기타 |  |

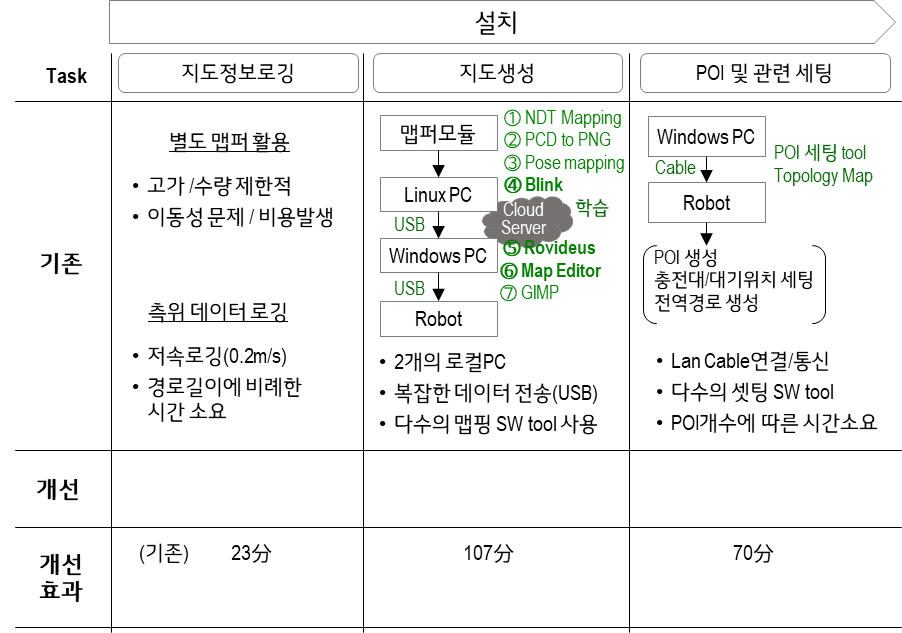
요구사항 검증방안 수립

### Goal #1: 저렴하고 빠른 Mapping 방법 개발

#### ****Process 시간 비교****

설치담당자가 별도로 관리하는 설치 Activity Tool을 이용하여 검증한다.

Process와 각각의 시간이 기재되어 있으며, 개선 후 효과를 기재한다.



<Figure 19>

#### Mapping 장비 가격 비교

각 Mapping 장비 별 가격을 비교하여 저렴한 Mapping 방식의 만족도를 평가한다.

|  |  |  |
| --- | --- | --- |
| 방식 | Description | 가격 |
| Mapping Robot | 적용 로봇: 중형배송로봇  Mapping을 할 수 있는 별도의 로봇 | 약 100,000,000 원 |
| Mapping Module  1세대 | 적용 로봇: 중형배송로봇  Mapping Robot이 고가라서 유사한 기능을 하는 Module | 약 10,000,000 원 |
| Mapping Module  2세대 | 적용 로봇: 안내로봇  RGB Camera만 부착하여 1세대 Module과 유사한 기능을 하는 Module | ? |

### Goal #2: 동일 수준의 위치오차 검증

#### ****Image Labeling Data 유효성 검증****

**Image Labeling Data는 누락되지 않고, 주기적으로 생성이 되어야며 취득된 멀티카메라의 위치가 동일해야 한다. 이렇게 검증된 Data를 이용하여야만 VSLAM 성능이 확보된다.**

**Image Labeling Data가 누락되지 않고, 멀티카메라의 위치가 동일한지를 판단하여 검증한다.**

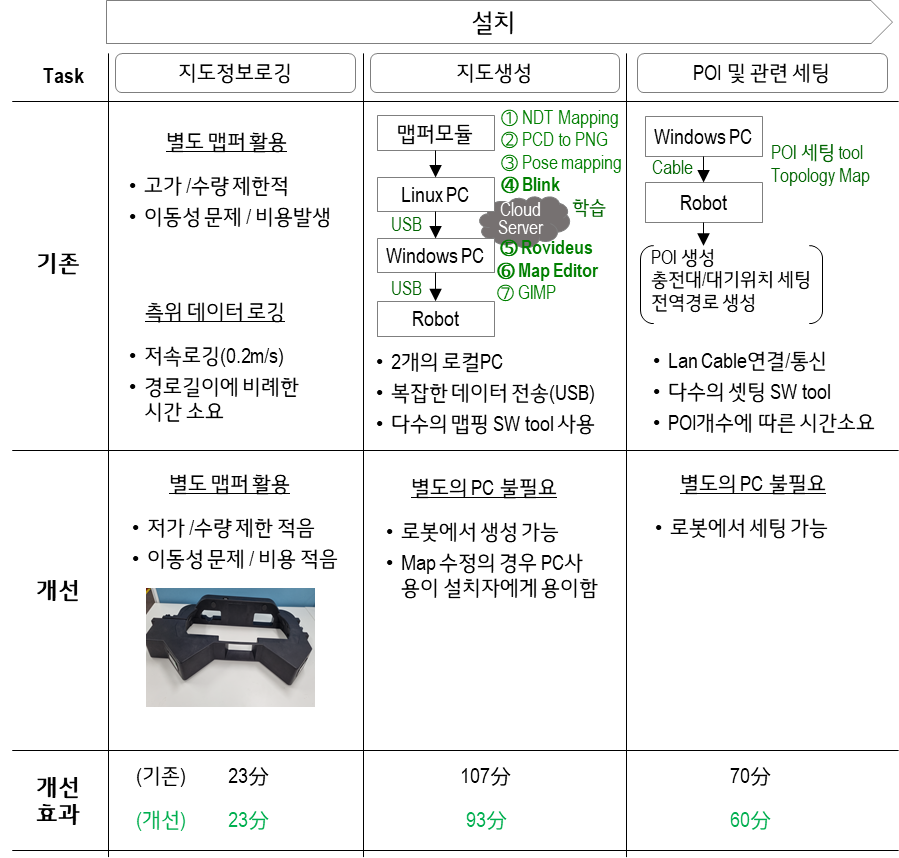
#### ****Mapping / Image Labeling 성능 확인****

구조물이 있는 지점을 기준으로 레이저 거리 측정계 실측치와 동일 구간을 Mapping 한 Pixel간 거리를 비교한다.

검증결과

### Goal #1: 저렴하고 빠른 Mapping 방법 개발 ****결과****

#### ****Process 시간 비교 결과****



<Figure 20>

개선효과로 총 24분의 설치시간 절감효과 발생하였다. 시간만으로는 큰 절감이 없는 것으로 판단되지만, 측정한 사람은 비교적 능숙한 설치담당자였고, 다수의 사이트가 발생하여 설치담당자들이 다수가 생기고 그 능숙도가 다를 수 있으므로, 시간절감 및 별도의 PC가 불필요함은 의미 있는 항목으로 판단된다.

#### Mapping 장비 가격 비교

|  |  |  |
| --- | --- | --- |
| 방식 | Description | 가격 |
| Mapping Robot | 적용 로봇: 중형배송로봇  Mapping을 할 수 있는 별도의 로봇 | 약 100,000,000 원 |
| Mapping Module  1세대 | 적용 로봇: 중형배송로봇  Mapping Robot이 고가라서 유사한 기능을 하는 Module | 약 10,000,000 원 |
| Mapping Module  2세대 | 적용 로봇: 안내로봇  RGB Camera만 부착하여 1세대 Module과 유사한 기능을 하는 Module | 1,238,168원 |

간소화된 모듈로 Mapping을 할 경우 1세대 Module대비 약 12.3%의 재료비만 소요되었고, Mapping Robot 대비로는 1.2%의 재료비가 소요되었다.

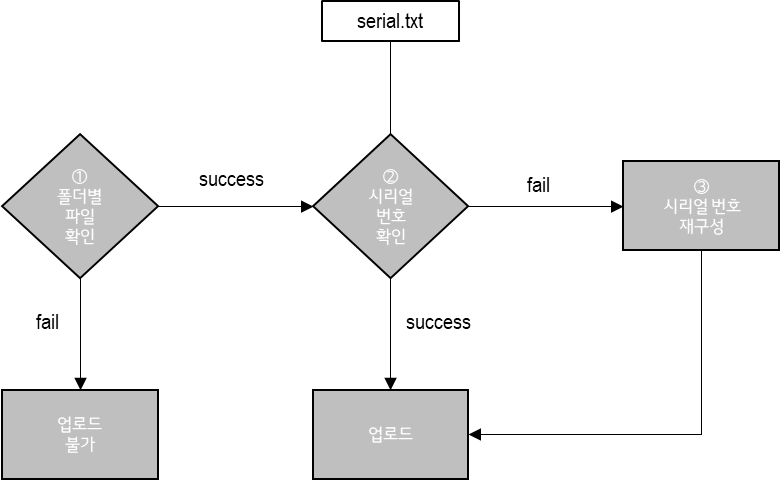
목표로 했던 저렴한 Mapping 방법 성과는 매우 만족스러운 수준이다.

### Goal #2: 동일 수준의 위치오차 검증결과

#### **Image Labeling Data 유효성 검증 결과**

Image Labeling Data의 유효성을 검증한 결과, 누락된 Data가 발견되었다. 사유는 Module에 부착된 Camera의 Serial No.를 이용하여 카메라의 위치를 판단하는데 불특정한 상황에 Serial No.가 사라지는 경우가 발생하였다.

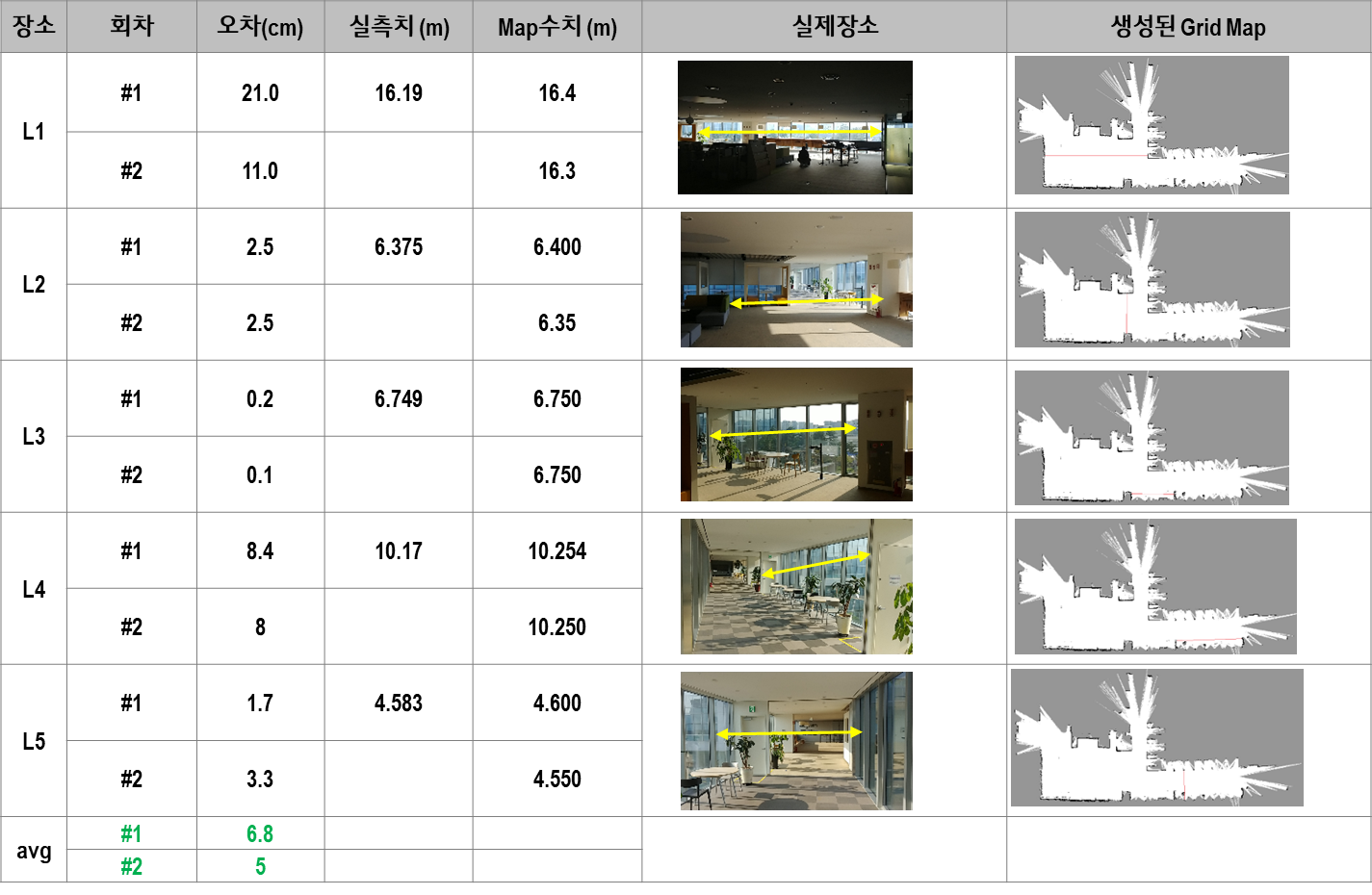
이를 개선하기 위해 하기와 같은 검증Process를 만들고 별도의 Tool을 제작하였다.

****

<Figure 21>

**검증 Process와 Too제작 이후에는 자체 검증결과 이슈가 재현되지는 않았으나, 지속 확인 중에 있다.**

#### ****Mapping / Image Labeling 성능 확인 결과****



측정장소는 마곡 사이언스 파크 3층을 기준으로 2회측정하였으며, 실측대비 오차는 6.8cm, 5cm 수준으로 측정되었다.

제품 측면에서는 최종적으로는 안내로봇의 위치오차를 측정해야 하지만, 위치오차는 Mapping 오차 외에 로봇의 위치추정오차, Motor 오차 등도 반영될 수 있기 때문에 본 프로젝트의 범위에서는 Mapping 오차만을 측정하는 것으로 기재하였다.

안내로봇 위치오차는 제품 Spec인 15cm를 만족하여 출시하였다.

# ****Lesson & Learned****

본 프로젝트의 결과를 하기와 같이 정리하였다.

**1) 저비용 Mapping 방식 개발 및 실적용**

1천만원에서 1억의 개발비가 필요한 Mapping 방식에서 약 120만원의 개발비로 절감하였고, 유사성능을 확보하였다.

다만, 현장에서는 아직도 별도의 Mapper가 있는 것이 경쟁사 대비 취약하다고 Voice를 내고있고, 이 부분에 대한 지속적인 개선이 필요하다.

**2) 설치 Process의 재정립**

로봇 현장 설치 시 별도의 PC가 필요했는데, 별도의 PC 없이 Mapping과 Image Labeling을 할 수 있게 되었다. 이로 인해 설치 Process 중에서 Data를 이동하는 단계를 간소화/생략 할 수 있었다.

설치 중, Image Labeling에 대한 오류가 지속 발생하여 현장에서 기존과 동일하게 PC를 사용하는 경우가 있었다. Image Labeling을 검증할 수 있는 Tool을 제공하였으나, 근본적인 해결방안을 모색해야 한다.

**3) SLAM 성능 확보**

Mapping 성능을 확보하기 위해 성능개선을 한 결과로 SLAM 성능이 개선되었다.

하지만, 최종적으로는 H/W의 한계(S1 Lidar)로 최종에는 Sick Lidar로 부품을 변경하였다.

개인적으로는 다양한 요구사항과 많은 이해당사자 있는 프로젝트였다. S/W적인 측면뿐만 아니라 신규 H/W와도 연동이 되고, 설치 Process와도 연관이 있었다. 프로젝트의 전체적인 규모는 크지 않지만, 다양한 업무 범위를 경험할 수 있었고, 여러 개발자 / 이해당사자들간 조율에 대한 Skill도 익힐 수 있었다.