**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | **Depth map 을 활용한 후방카메라의 한계 개선** |
| 팀 명 | **천 리 안** |
| 문서 제목 | 계획서 |

|  |  |
| --- | --- |
| **Version** | 1.5 |
| **Date** | 12 |

|  |  |
| --- | --- |
| **팀원** | 최 승혁 |
| 신 태섭 |
| 박 성우 |
| 신 동호 |
| 최 성현 |
| 박 민욱 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “Deapth map을 활용한 후방카메라의 한계 개선”을 수행하는 팀 “천리안”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “천리안”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 계획서-Depth map을 활용한 후방카메라의 한계 개선.doc |
| **원안작성자** | **신태섭, 최승혁, 박성우** |
| **수정작업자** | **신태섭, 신동호, 최성현, 박민욱** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2015-03-06 | 신태섭 | 1.0 | 최초 작성 | 개요 |
| 2015-03-07 | 신태섭 | 1.1 | 내용 추가 | 개발 목표 및 내용 |
| 2015-03-07 | 박성우  최승혁 | 1.2 | 내용 수정 | 개발 목표 및 내용 |
| 2015-03-09 | 박민욱 | 1.3 | 내용 추가 | 배경기술 |
| 2015-03-09 | 최성현  신동호 | 1.4 | 내용 추가 | 프로젝트 팀 구성 및 역할 분담,  프로젝트 비용, 개발 일정 및 자원관리 |
| 2015-03-10 | 전원 | 1.5 | 최종 검토 | 오타 확인 및 검토 |
|  |  |  |  |  |

**목 차**

[**1** **개요** 4](#_Toc347412182)

[1.1 프로젝트 개요 4](#_Toc347412183)

[1.2 추진 배경 및 필요성 4](#_Toc347412184)

[**2** **개발 목표 및 내용** 5](#_Toc347412185)

[2.1 목표 5](#_Toc347412186)

[2.2 연구/개발 내용 6](#_Toc347412187)

[2.3 개발 결과 7](#_Toc347412188)

[2.3.1 결과물 목록 및 상세 사양 7](#_Toc347412189)

[2.3.2 시스템 기능 및 구조 7](#_Toc347412190)

[2.4 기대효과 및 활용방안 7](#_Toc347412191)

[**3** **배경 기술** 8](#_Toc347412192)

[3.1 기술적 요구사항 8](#_Toc347412193)

[3.2 현실적 제한 요소 및 그 해결 방안 9](#_Toc347412194)

[3.2.1 하드웨어 9](#_Toc347412195)

[3.2.2 소프트웨어 9](#_Toc347412196)

[3.2.3 기타 9](#_Toc347412197)

[**4** **프로젝트 팀 구성 및 역할 분담** 10](#_Toc347412198)

[**5** **프로젝트 비용** 10](#_Toc347412199)

[**6** **개발 일정 및 자원 관리** 11](#_Toc347412200)

[6.1 개발 일정 11](#_Toc347412201)

[6.2 일정별 주요 산출물 12](#_Toc347412202)

[6.3 인력자원 투입계획 13](#_Toc347412203)

[6.4 비 인적자원 투입계획 14](#_Toc347412204)

[**7** **참고 문헌** 15](#_Toc347412205)

# **개요**

## 프로젝트 개요

현재 모든 차량에는 운전자에게 후진 시 후방의 정보를 제공하기 위해 후방카메라 및

후방초음파센서(BWS)가 장착되어 있다. 이 분야에 있어서 차량 산업이 발전함에 따라 자율 주차, 무인 주행 등에 관련된 여러 부가기능이 활발히 개발되고 연구되고 있다.

하지만 현재 상용화된 후방 카메라 및 후방 초음파센서(BWS)는 측정 범위 및 물체에 많은 문제점을 가지고 있다.

본 프로젝트는 기존 후방카메라에서 Image 영상만을 제공하여 발생하는 문제점을 보완하기 위해 Kinect Device의 Depth map 영상을 활용하였다.

Kinect의 Depth map 영상과 Image 영상을 합성하여 운전자에게 장애물까지의 거리정보를 시각적으로 제공하는 영상소프트웨어를 개발한다.

이 영상소프트웨어를 통해 운전자가 보는 화면에 가까운 물체를 하이라이트 처리하여

운전자가 후방 정보에 대해 좀더 확실한 정보를 얻을 수 있도록 도와준다.

또한 본 프로젝트에서는 Kinect에서 얻어내는 정보를 모형차량에도 적용하여 후방의 장애물에 근접하면 차량의 구동이 제어되도록 하는 테스트 환경을 구축한다.

## 추진 배경 및 필요성.

### **1.2.1 후방카메라 및 후방초음파센서(BWS)의 시스템 현황**

1. 후방카메라는 현재 Image 영상만을 제공한다.
2. 후방초음파센서(BWS)는 시야 사각지대를 초음파센서를 사용, 감시하고

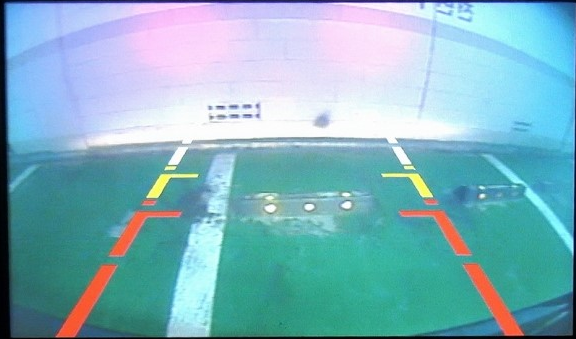
해당지역에 장애물이 있으면 운전자에게 경보를 통해 청각적 정보 제공한다.

1. 후방초음파센서(BWS)는 40kHz의 초음파를 발산, 초음파가 물체에 부딪쳐

되돌아 올 때까지의 시간을 측정하여 물체까지의 거리를 측정한다.

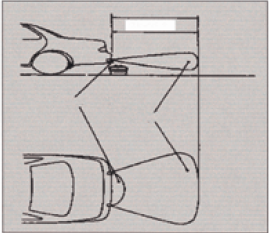
### **1.2.2 후방카메라 및 후방초음파센서(BWS)의 문제점**

1. 후방카메라는 Image 영상만을 제공하기 때문에 운전자가 후방 물체까지의 정확한 거리측정이 불가능하다.



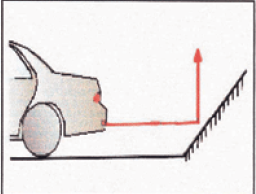
1. 후방초음파센서(BWS)는 자동차 범퍼와 가까운 지역은 감지 못하는 영역이

존재한다.



1. 후방초음파센서(BWS)는 초음파의 특성 때문에 철사, 체인, 자갈, 눈에 덮여 있는 소화전등 감지 못하는 물체가 존재한다.
2. 후방초음파센서(BWS)는 주변환경에 따라 초음파 반사로 인해 불규칙한

경보 음이 발생해 정확한 정보를 제공할 수 없다.



### **1.2.3 Depth map을 활용한 후방카메라(BWS)의 필요성**

현재 상용화 되어있는 후방카메라 및 후방초음파센서(BWS)는 일반 Image 영상과

초음파센서만을 이용하여 운전자에 후방 정보를 제공하기 때문에 인식되는 범위(사각지대, 거리) 및 장애물(모양, 재질)에 따라 한계가 있다. 이러한 한계 때문에 후방 진입 시 많은 사고가 발생하고 있다. 미국의 한 연구조사기관(IIHS)에 따르면 매년 후진 관련

사고로 평균 210명이 사망하고, 1만5천여 명이 부상당하는 것으로 집계된다.

이러한 조사 결과처럼 현재 상용화된 후방카메라 및 후방초음파센서(BWS)는 운전자에게 보다 정확하고 현실감 있는 정보를 제공하고 있지 못한다.

이러한 문제점을 해결하기 위해 Depth map 영상과 Image 영상을 합성하는 소프트웨어 개발을 통해 기존 시스템의 인식한계를 극복하고 더 현실감 있는 시각적 정보를 운전자에게 제공하여 후방 진입 및 주차 시 발생하는 사고를 절감한다.

또한 이렇게 개발한 소프트웨어를 모형차량에 테스트 환경을 구축하여 후방 장애물에

차량이 접근하였을 때 구동을 제어하는 테스트 환경을 구축하여 충돌사고를 막는다.

# **개발 목표 및 내용**

## 목표

1. Depth map 정보를 이용하여 자동차 운전자에게 자동차 후방과 장애물 사이의 직관적인 깊이 정보를 제공할 수 있는 영상처리 알고리즘을 구현한다.
2. 후방 영상카메라와 Depth map을 측정하는 Kinect의 Calibration을 통해 두 영상 사이의 이질감을 최소화한다.
3. 모형 자동차를 통하여 Depth map을 활용한 후방카메라의 기능을 테스트 가능하게 한다.
4. 기존의 후방카메라와 초음파센서의 한계점인 거리감의 부재와 사각지대의 존재를 Depth map을 이용하여 한계를 개선한다.
5. Depth map 정보를 활용하여 모형차량과 장애물 사이의 충돌 가능성이 높아진 경우 사용자에게 위험을 알리고, 모형차량에 제동을 하는 제어 알고리즘을 구현한다.

## 연구/개발 내용

**2.2.1 Depth map**

해상도 512 \* 424 정도의 비교적 고화질의 depth 영상을 출력하는 것을 목표로 Kinect부터 depth 영상을 제공받는다. Kinect를 pc와 연결하고, Kinect를 구동하기 위한 SDK를 설치한다. Kinect studio를 실행시키면, 기존의 카메라 영상 (color image)와 depth image를 함께 얻을 수 있다. 이 두 영상을 가지고 정확히 일치하도록 합성해야 한다.

****

**2.2.2 Calibration**

위에 제시된 그림이 Kinect 를 pc에 연결하고, Kinect studio를 실행하여 얻은 depth 영상과 color 영상을 캡쳐한 모습이다. 이제 이 두 개의 영상을 합성하는 작업을 해야 한다. 그런데 kinect의 카메라 중에서 color 영상을 촬영하는 카메라와 depth 영상을 촬영하는 카메라의 위치가 약간 떨어져 있기 때문에, 같은 곳을 촬영하더라도 두 영상의 위치가 완벽히 일치하지 않게 된다. 따라서 두 영상을 정확히 일치하도록 Calibration 알고리즘을 적용하여 깔끔하게 합성하는 작업이 필요하다. 그리고 depth 영상에서 나온 물체의 테두리가 깔끔하게 촬영되지 않는(데이터가 튀는) 현상이 종종 발생하게 된다. 이를 해결하기 위해서도 보정하기 위한 알고리즘을 정확하여 좀 더 정밀하고 정확한 영상을 획득할 수 있도록 해야 한다.

**2.2.3 영상처리**

합성한 영상에서 물체와 카메라 사이의 거리에 따라 하이라이트 표시를 해야 한다. 운전자에게 물체와 자동차 사이의 거리 정보를 확실하게 제공할 수 있고, 시각정보를 기존의 카메라보다 눈에 띄게 나타낼 수 있기 때문이다. Visual Studio 2012를 이용하여, 합성된 영상에서 일정 거리 이하의 물체에 도형이나 색깔로 표시를 하여 영상 처리를 한다. 그리고 필요하다면, 하이라이트 표시 이외에 경고음을 발생시켜서 운전자에게 확실히 각인시켜 줄 수 있는 방법도 생각해 볼 것이다.

**2.2.4 자동차 제어**

Kinect로부터 획득한 Depth정보를 이용하여 모형차량 제어를 해야 한다. 따라서, Kinect와 연결되어 있는 PC와 모형차량 간의 통신이 가능해야 하는데 그러기 위해선 먼저 모형차량의 전체적인 제어를 할 수 있는 Raspberry Pi를 설정해야 한다. Raspberry Pi설정을 단계별로 나타내면 다음과 같다.

1. SD카드에 OS 설치

Raspberry Pi에서 사용하는 운영체제는 일반적으로 Raspbian을 사용한다. Raspbian은 Debian linux가 Raspberry Pi용으로 포팅 된 운영체제이다.

1. Wireless 네트워크 환경 설정

운영체제가 설치되면 네트워크 설정을 해야 한다. Wireless환경을 설정하기 위해서 Wifi동글을 이용하거나 블루투스 모듈을 사용해야 한다. 데이터 전송속도를 고려하여 Wifi동글을 이용하여 설정한다.

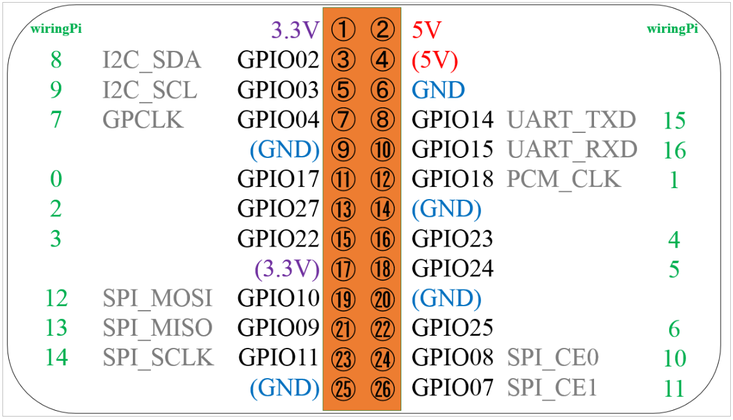
1. Client/Server 모델을 적용할 언어 선정

통신을 위하여 네트워크모델을 정해야 한다. PC를 Server로, Raspberry Pi를 Client로 하여 설정하고, 이 모델을 적용할 언어를 설정한다. 개발언어는 C언어, Java등을 사용 할 수 있다.

또한, Raspberry Pi와 모형차량의 각 모듈의 실질적인 제어를 하는 Micro controller와 UART를 통신을 위해 GPIO핀을 사용하기 위한 설정을 해야 한다. 이를 단계별로 나타내면 다음과 같다.

1. Wiring Pi 모듈 설치

Wiring Pi 모듈은 Raspberry Pi의 GPIO핀에 접근할 수 있게 하는 C언어로 작성된 라이브러리들을 사용할 수 있게 한다. Wiring Pi는 아두이노에서 사용하는 Wriring system과 유사하기 때문에, 쉽게 사용이 가능하다.



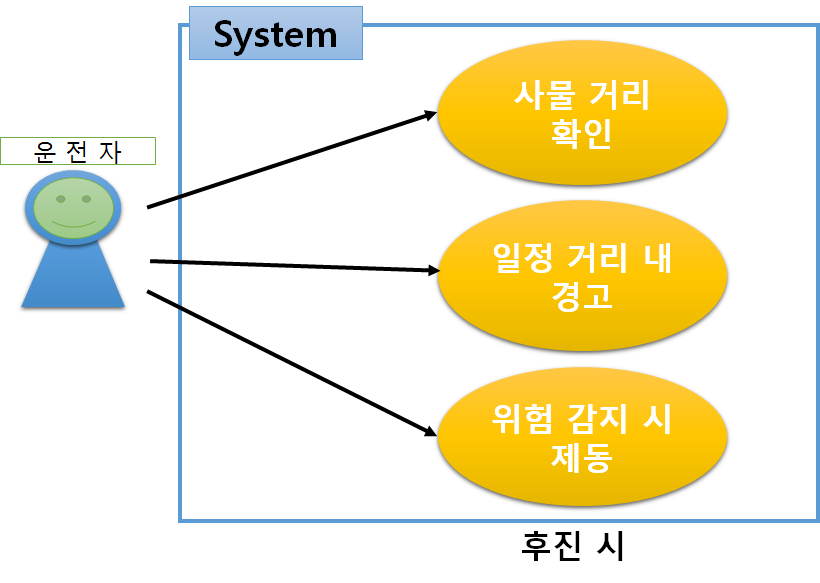
위의 GPIO그림과 같이 GPIO14, 15번 핀에 Wiring Pi 라이브러리를 이용하여Controller와 값을 주고 받을 수 있다.

1. Wiring Pi 라이브러리를 이용하여 차량 제어 프로그래밍

앞서 선택한 개발 언어를 이용하여 알고리즘을 구현한다. 이 프로그램에는 Wiring Pi 라이브러리와 서버/클라이언트 라이브러리를 사용한다. 이를 통해 PC에서 Depth map 정보를 이용하여 해당하는 제어를 Raspberry Pi로 전송할 수 있고, 해당하는 제어에 맞도록 차량을 구동할 수 있게 된다.

## 개발 결과

### **2.3.1 시스템 기능 요구사항**



### **2.3.2 시스템 비기능(품질) 요구사항**

성능 : 0.5초 이내에 운전자가 물체를 인식 할 수 있을 정도로 영상 합성을 실시간으로 처리한다.

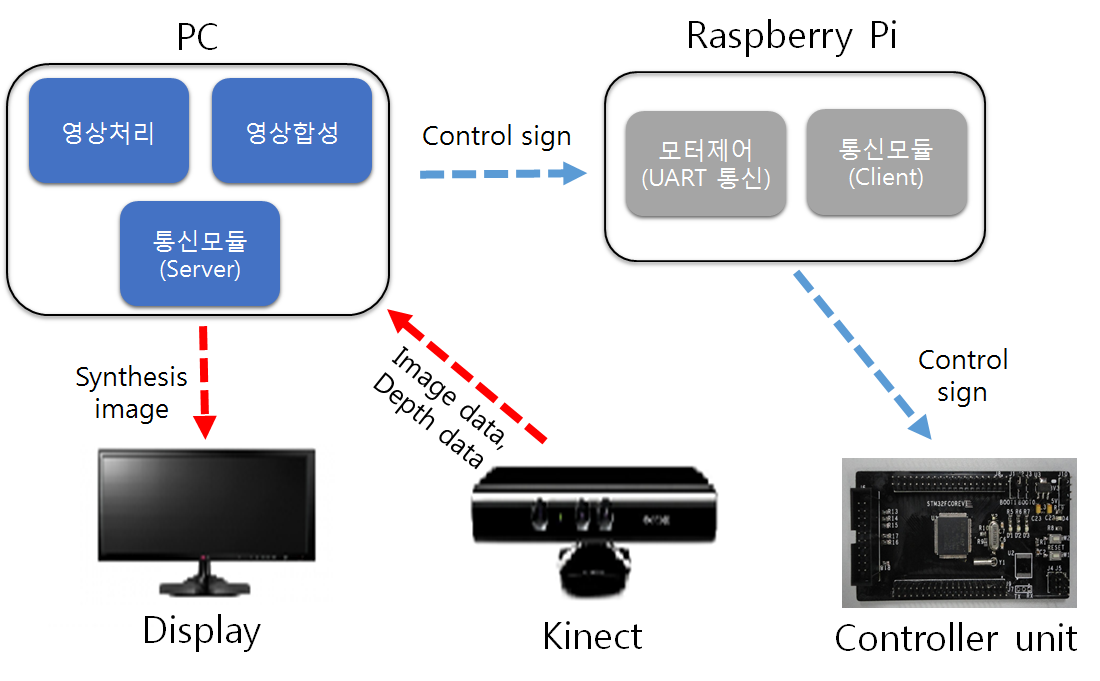
카메라가 1초당 30 frame 정도 촬영할 수 있을만한 해상도를 지닌다.

정확성 : 둘레 3CM 이하 정도의 가느다란 물체나 면적이 25CM^2 정도의 작은 크기의 물체도 인식이 가능하게 해야 한다.

안정성 : 키넥트와 컴퓨터, 모형자동차 간의 통신이 언제든지 끊기지 않을 수 있도록 안정성 있는 통신구조를 갖추어야 한다.

우선순위 : 성능 > 안정성 > 정확성

### **2.3.3 시스템 구조**



### **2.3.4 결과물 목록 및 상세 사양**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 대분류 | 소분류 | 기능 | 형식 | 비고 |
| 파일 | *Kinect image 영상 저장,열기* | Kinect image 영상을  저장, 열기한다.. | C++ 함수 |  |
| *Kinect Depth*  *영상 저장,열기* | Kinect Depth영상을  저장, 열기한다 | C++ 함수 |  |
| 입력 | *Kinect image, depth*  *영상 information* | Kinect로부터 image, depth 정보를 얻는다. | Module |  |
| *모형자동차*  *Controller Unit* | 차량제어 | Module |  |
| 출력 | *Display* | Image와 depth 합성영상 출력 | Scene |  |
| *모형자동차* | 차량제어 | Module |  |
| 알고리즘 | *Depth interface* | Kinect에서 얻은 depth 정보를  분석하여 사용자가 의도한 interface를 알아낸다. | C++ |  |
|  | *모형차량 interface* | 차량 제어 |  |  |
| 모양 | *글꼴* | 정렬 |  |  |
|  |  | 폰트 바꾸기 |  |  |
|  |  |  |  |  |

## 기대효과 및 활용방안

**2.4.1 깊이 정보를 포함한 후방 영상**

현재 후방 주차를 위해 후정 정보를 얻기 위하여 운전자들이 보편적으로 사용하고 있는 방법으로는 초음파 센서를 통한 후방 장애물과의 거리 정보를 이용한 소리 경고음, 후방 카메라를 이용한 후방 영상이다. 이 두 가지 정보를 통해서는 후방 장애물 과의 거리 정보를 정확히 알 수 없을 뿐만 아니라, 초음파 센서를 통한 거리정보는 사각지대가 많고 오류도 잦다. 여기에 깊이 정보를 알 수 있는 깊이 카메라를 설치하여 후방 정보에 장애물 과의 거리정보를 추가한다면 운전자는 더 정확한 후방의 시야를 확보할 수 있다.

**2.4.2 기존 사용하던 후방카메라에 확장 가능**

현재 장착되어 있는 후방 카메라에 Depth 카메라를 더 설치해 준다면 두 영상의 Calibration을 통하여 후방 장애물에 대한 깊이 정보를 획득할 수 있다.

**2.4.3 사각지대 제거**

Depth map을 활용하여 장애물과의 거리분석을 통해 초음파 센서로 잡아내지 못한 장애물도 잡아낼 수 있다. 향후 Depth 카메라의 개선에 따라 더 정밀한 측정이 가능해 진다면 장애물 판별의 기능이 크게 향상될 것이다.

# **배경 기술**

## 기술적 요구사항

**3.1.1 Kinect 카메라 기술적 요구사항**

우선 Kinect v1을 가지고 개발 할 예정이지만, 차후에 Kinect v2 버전으로 바꾸게 될 것을 대비하여 v1과 v2 각각에 대한 기술적 요구사항을 나열해보도록 하겠다.

첫 번째로, Kinect v1과 v2의 개발 환경을 살펴보면 다음과 같다.

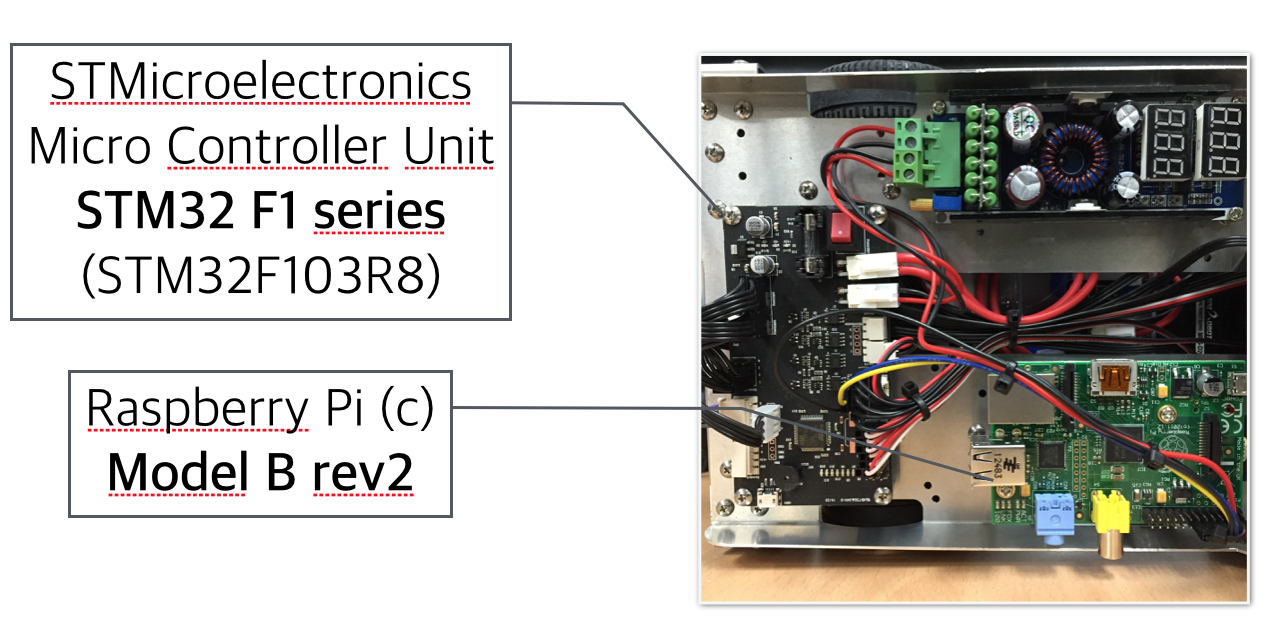
|  |  |
| --- | --- |
| **Kinect v1** | **Kinect v2** |
| **Windows 7 또는 Microsoft Windows Embedded Standard 7** | **Embedded 8 Standard, Windows 8, Windows 8.1** |
| **32비트(x86) 또는 64비트(x64) 프로세서** | **64 비트(x64) 프로세서** |
| **Dual-core 2.66GHz 이상의 프로세서** | **Dual-core 3.1 GHz 이상의 프로세서** |
| **전용 USB 2.0 버스** | **전용 USB 3.0 버스** |
| **2GB RAM** | **4GB RAM** |
| **Visual Studio 2010 이상** | **Visual Studio 2012 또는 2013** |

그 다음으로 프로젝트 결과물 확인 환경은 개발 환경과 동일한 환경이 모두 갖추어져 있어야 하며, 직접 pc를 통하여 카메라에서 출력되는 영상을 확인해야 한다. 위에 나열되어 있는 시스템적인 요구사항을 모두 갖추었다면, 기본적으로 Kinect SDK 프로그램을 설치하여 영상 출력 결과를 확인할 수 있다.

**3.1.2 모형차량 기술적 요구사항**

모형차량의 기술적 요구 사항을 파악하기 위해서는 우선 모형차량의 하드웨어구성을 알아야 한다. 아래 표는 모형차량의 구성요소와 개발에 필요한 기술적인 요구사항들을 나타낸 것이다.

|  |  |  |
| --- | --- | --- |
| **항목** | **규격 및 내용** | **요구사항** |
| 싱글 보드 컴퓨터 | Raspberry pi  Model B타입, rev2  (700 MHz ARM1176) | - linux 계열의 OS환경  - PC와 pi간 통신환경 설정  - Wiring Pi모듈 설정  - Micro controller와 UART통신 |
| 마이크로 컨트롤러 | STM32F103R8  (72 MHZ ARM Cortex-M3) |  |
| 거리 센서 | 라인감지용 7조 | - 제어CODE Sheet필요 |
| 적외선 센서 | 적외선 거리센서 – 6SET | - 제어CODE Sheet필요 |
| 구동모터 | DC서보 모터 엔코더 | - 제어CODE Sheet필요 |
| 서보모터-3EA  (조향 및 카메라) | 6V. 9Kg (180도) | - 제어CODE Sheet필요 |

****

모형차량의 Micro controller에는 이미 제조사에서 작성된 프로그램이 업로드 되어 있고, Controller가 실행될 때 이 프로그램이 실행된다. 이 프로그램은 Pi로부터 UART통신을 통해 읽어 들이는 제어 Data에 따라 차량의 각 구성 요소를 제어하게 된다. 제어 Code sheet는 다음과 같은 형식으로 제공된다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **Data1** | **Data2** | **Read/Write** | **설명** |
| 0x80 | 0-255 |  | R | 로봇 ID |
| 0xA0 | High(L-Speed) | Low(L-Speed) | W | L모터 속도설정 |
| 0xC8 |  |  | W | 전진(mm) |
| 0xC9 |  |  | W | 후진(mm) |

모형차량의 테스트나 결과물은 모형차량의 구동을 통해 직접 눈으로 확인할 수 있기 때문에, 결과물을 확인할 수 환경을 위한 기술적 요구사항은 따로 기술하지 않았다.

## 3.2현실적 제한 요소 및 그 해결 방안

### **3.2.1** **하드웨어**

**1) Kinect**

Depth map을 얻기 위한 카메라의 경우 외부에서 사용 가능한 depth camera는 그 비용이 예산을 초과하므로 차선책인 Kinect를 사용하여 실내를 기반으로 프로그래밍 하며 프로그램이 검증 될 경우 좋은 성능의 카메라와 호환 가능하게 구성한다.

현재까지 알려진 Kinect v1의 사양 중 depth 영상의 해상도는 320 \* 240으로, 실제로 실험을 위한 영상을 촬영해 보았더니, 뚜렷하게 인식되지 않는 부분들이 많이 보였다. 특히 테두리 부분에서 그림자가 심하게 진다거나, 테두리가 명확히 드러나지 않는 부분들이 많이 보여서 확실한 depth 정보가 확보되지 않았다. 따라서 Kinect v2를 이용하여 실험을 해 보았는데, 해상도가 512 \* 424으로 많이 향상되어서 좀 더 세밀한 부분까지 출력이 가능한 것을 확인할 수 있었다.

Kinect v1의 단점을 보안하기 위해 Kinect v2를 사용하면 되지만, 해외 구매대행을 해야 하기 때문에 기기를 확보하는 데까지 시간이 많이 소요된다. 따라서 우선 v1을 통하여 개발을 진행하고, v2를 구매하게 된다면 기기를 바꾸어 카메라의 성능을 향상시킬 수 있도록 할 것이다.

**2) 모형자동차**

Depth map 을 구하기 위한 Kinect 의 요구전력이 모형 차량에 내장된 파워로는 부족하기 때문에 외부에서 전력을 공급해야 된다.  
무선으로 차량을 구성할 경우 Raspberry Pi의 무선 네트워크를 통하여 영상을 pc로 전송해야 되는데 무선 네트워크의 경우 거리나 외부 변수에 따라 그 속도가 불안정하고 평균적으로 최대 지원 속도가 54mbps이며, 연결방식이 일반적인 n방식으로 이루어질 경우 속도는 20mbps 에 머문다.

Depth map 의 영상정보와 일반 이미지 영상의 경우 초당 23Mbyte의 용량을 가지고 있으며 이를 실시간으로 pc에 전송함과 동시에 pc에서 차량제어 신호를 보내야 한다. 무선 네트워크로 구성 하였을 경우 좀 더 유동적인 움직임이 가능하지만 정보의 상호교환과 실시간 처리에 있어서 전송속도문제가 생길 여지가 있으므로 유선 랜 구성을 차선책으로 둔다. Kinect의 시야각은 수평 57도 수직 43도로 좁은 편이나 시야를 위해 두 개의 Kinect를 모형 차량에 설치하기엔 어댑터의 수나 공간의 협소함 같은 물리적 한계가 있다.

### **3.2.2 소프트웨어**

**1) Kinect**

미흡할 수 있는 Kinect의 영상처리에 부분에 대해서 인터넷 검색 및 전문 서적을 활용하여 개선 방안에 관련된 알고리즘을 찾아봐야 한다. Image 영상과 depth 영상을 합성하였을 때, 오차가 많이 발생할 수 있으므로 이를 대비하여 자연스럽게 영상을 합성할 수 있는 프로그래밍 방안을 찾아보고 적용시켜야 한다. 또한 좀 더 정밀한 depth 영상을 출력하기 위해 무조건 하드웨어를 교체하기보다는 소프트웨어 부분에서 최대한 개선시킬 수 있을만한 방법을 알아보는 것이 중요하다.

**2) 모형자동차**

Raspberry Pi의 경우 os가 linux 기반인 반면 Kinect는 windows 기반에서 사용할 수 있게 설계되어 있기 때문에 Raspberry Pi를 통한 영상 처리에 문제가 있으므로 Kinect는 pc에 직접 연결하여 영상을 처리한다.  
Micro controller unit을 제어 하기 위한 프로그램을 작성하려면, 해당 Controller의 매뉴얼을 통해 각 칩의 입출력 값 등의 하드웨어 중점적인 내용을 심도 있게 파악해야 한다. 이 프로젝트에서는 Controller에서 실행될 프로그램을 새로 프로그래밍 하는 것이 의미가 없으며, 많은 시간을 소요하므로 제조사에서 제공하는 제어 프로그램을 활용, 응용하여 사용한다

### **3.2.3 기타**

키넥트나 모형 자동차에 대해서 오랜 시간동안 전문적으로 다루어본 팀원들이 없기 때문에, 프로젝트를 수행하는 데 필요한 정보들이 많이 부족하다. 그래서 팀원들 간의 스터디 및 자료 조사를 철저히 하여야 한다. 계획한 대로 프로젝트가 진행되지 않을 수도 있는 가능성이 있기 때문에 개발을 할 때 신중하게 알아보고 해야 한다.

# **프로젝트 팀 구성 및 역할 분담**

| 이름 | 역할 |
| --- | --- |
| 최승혁 | * Software Project Leader * Kinect 영상처리 알고리즘 개발 |
| 신태섭 | * 모형자동차Interface 담당 * Kinect 자료수집, 모형자동차 자료수집 분석 |
| 박성우 | * 모형자동차 자료수집 및 분석 * 모형자동차의 STM32F10과 컴퓨터 통신 |
| 신동호 | * Kinect 자료수집 및 분석 * Depth map 의 정밀성 향상 알고리즘 |
| 최성현 | * Kinect calibration 알고리즘 * Kinect Interface 담당 |
| 박민욱 | * 모형자동차 자료분석 * 모형자동차 회로설계, 제어 알고리즘 개발 |

# **프로젝트 비용**

|  |  |
| --- | --- |
| **항목** | **예상치 (MD)** |
| Depth map과 영상처리 관련 Study | 80 |
| 초음파 센서와 후방카메라 한계 측정 | 15 |
| Kinect v1과 v2개발환경 구축 | 35 |
| Kinect 영상과 depth map calibration | 40 |
| 모형자동차 STM32F10과 컴퓨터 통신 | 25 |
| 모형자동차 제어 알고리즘 | 50 |
| 깊이 정보에 따른 영상처리알고리즘 | 60 |
| 모형자동차와 Kinect 연동 및 보정 | 30 |
| 실제 주차환경 구축을 통한 test | 25 |
| 합 | 360 |

# **개발 일정 및 자원 관리**

## 개발 일정

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 항목 | 세부내용 | 1월 | 2월 | 3월 | 4월 | 5월 | 6월 | 비고 |
| 요구사항  분석 | 아이디어 구상회의 |  |  |  |  |  |  |  |
| 아이디어 선정 및  정보수집 |  |  |  |  |  |  |  |
| 관련분야  연구 | Depth map 과 영상관련 연구 |  |  |  |  |  |  |  |
| Kinect interface 연구 |  |  |  |  |  |  |  |
| 모형자동차interface연구 |  |  |  |  |  |  |  |
| 설계 | 하드웨어 설계 |  |  |  |  |  |  |  |
| 프로그램 설계 |  |  |  |  |  |  |  |
| 구현 | 영상과 depth map Calibration |  |  |  |  |  |  |  |
| 모형차량과 컴퓨터 통신 |  |  |  |  |  |  |  |
| 모형차량 제어 알고리즘 |  |  |  |  |  |  |  |
| 깊이 정보 영상처리 |  |  |  |  |  |  |  |
| 모형차량과 Kinect 연동 |  |  |  |  |  |  |  |
| 테스트 | 주차환경 구축 및 테스트 |  |  |  |  |  |  |  |

## 일정 별 주요 산출물

|  |  |  |  |
| --- | --- | --- | --- |
| 마일스톤 | 개요 | 시작일 | 종료일 |
| 계획서 발표 | - Depth map 과 영상관련 study  - 프로젝트 아이디어 수립 및 설계  - 프로젝트 관련 자료 수집  **산출물 :**   1. 프로젝트 수행 계획서 2. 프로젝트 소개 발표 자료 | 2015-01-05 | 2015-03-12 |
| 설계 완료 | * 하드웨어 설계 완료 * 프로그램 설계 완료   **산출물 :**   1. 시스템 설계 사양서 | 2015-03-10 | 2015-03-20 |
| 1차 중간 보고 | * Kinect 깊이 인식 구현 완료 * 깊이 정보 영상처리 완료 * 영상과 Depth map calibration 완료   **산출물 :**   1. 프로젝트 1차 중간 보고서 2. 프로젝트 진도 점검표 3. 1차분 구현 소스 코드 및 하드웨어 | 2015-03-16 | 2015-04-08 |
| 2차 중간 보고 | * 모형차 와 컴퓨터 통신 완료 * 모형차와 Kinect 연동 완료   **산출물 :**   1. 프로젝트 2차 중간 보고서 2. 프로젝트 진도 점검표 3. 2차분 구현 소스 코드 및 하드웨어 | 2015-04-09 | 2015-05-13 |
| 구현 완료 | 시스템 구현 완료  **산출물:**   1. 구현 완성 프로젝트 결과물 | 2015-03-16 | 2015-05-21 |
| 테스트 | 시스템 통합 테스트  **산출물:**   1. 오류 수정 목록 2. 프로젝트 수정 결과물 | 2015-03-16 | 2015-05-26 |
| 최종 보고서 | 최종 보고 | 2015-05-13 | 2015-05-28 |

## 인력자원 투입계획

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 이름 | 개발항목 | 시작일 | 종료일 | 총개발일(MD) |
| **최승혁** | *Kinect Algorithm* | 2015-03-10 | 2015-05-28 | 40 |
| **신태섭** | *모형자동차 interface* | 2015-03-10 | 2015-05-28 | 40 |
| **박성우** | *모형자동차 STM2F10, 컴퓨터통신* | 2015-03-10 | 2015-05-28 | 40 |
| **신동호** | *Depth map Algorithm* | 2015-03-10 | 2015-05-28 | 40 |
| **최성현** | *Kinect calibration Algorithm* | 2015-03-10 | 2015-05-28 | 40 |
| **박민욱** | *모형자동차 제어 Algorithm* | 2015-03-10 | 2015-05-28 | 40 |

## 비 인적자원 투입계획

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 항목 | Provider | 시작일 | 종료일 | Required Options |
| **개발용 PC 6대** | Samsung 4대  DELL 2대 | 2015-01-10 | 2015-05-28 |  |
| **Kinect** | Microsoft Windows | 2015-01-10 | 2015-05-28 |  |
| **모형자동차** | 국민대학교 | 2015-01-10 | 2015-05-28 |  |

# **참고 문헌**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| **1** | 웹사이트 | 마이크로 컨트롤러 제원, 매뉴얼 | Microsoft |  |  |  |
| **2** | 기사 |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |