제17회 임베디드SW경진대회

개발완료보고서 [지능형 휴머노이드]

0. 작성 시 주의사항

※아래의 작성 양식(제출분량, 폰트, 크기, 줄 간격 등)을 미준수 시 서류 평가의 감점요인됨

※ 제출 분량 : A4 용지 상세내용 포함 30 page 이내

※ 작성 양식 (폰트 : 맑은 고딕 / 폰트 크기 : 10pt / 자간 : 0% / 장평 : 100% / 줄 간격 : 130%)

※ 제출 포맷 : pdf

1. 팀 정보

팀명	드래곤볼	팀장	김재현
팀원	김종연	팀원	박범규
팀원	서지연	팀원	안주영

2. 개발완료보고서

1. 개요

1.1. 작품 개요

2족 보행을 하는 미니 지능형 휴머노이드 2대를 이용하여 긴급한 재난현장에서 임무를 수행하는 시스템을 개발한다. 미니 로봇 2대는 재난 상황에서 구조 임무를 수행하는 '구조 로봇'과 구조를 받는 '시민 로봇'으로 이루어진다.

1.2. 개발 목표

최근 발생한 강원도 속초에서의 산불, 후쿠시마 원전 사고와 같이 규모가 매우 큰 사고의 경우 인간이 직접 사고 현장에 투입되어 임무를 수행하는 데 큰 어려움이 있다. 산불의 경우, 그 사고 현장이 매우 험난하고 산불로 인한 엄청난 열기로 인해 인간이 구조 작전을 펼치기에는 무리가 있다. 또한, 원전 사고의 경우 인간에게 엄청난 해를 가하는 방사선 피폭으로 인해 사고 현장 내부로 진입하기에는 많은 어려움이 있다. 이처럼 직접 임무를 수행하기 어려운 재난 상황에서 인간을 대신해 로봇이 투입되어 구조자에 대한 인명피해를 줄일 수 있다.

이번 미션은 여러 가지 재난 상황에서 앞서가는 '구조 로봇'이 구조 임무를 수행하고, '구조

로봇'의 도움을 받아 뒤따라가는 '시민 로봇'이 안전하게 대피한다. '구조 로봇'이 '시민 로봇'을 안전하게 대피시키는 것이 이 대회의 목표이고 더 나아가 실제로 발생한 긴급하고 복잡한 재난 현장에 로봇이 투입되어 민간인, 시민 등 인간을 구출하는 것을 최종 목표로 한다.

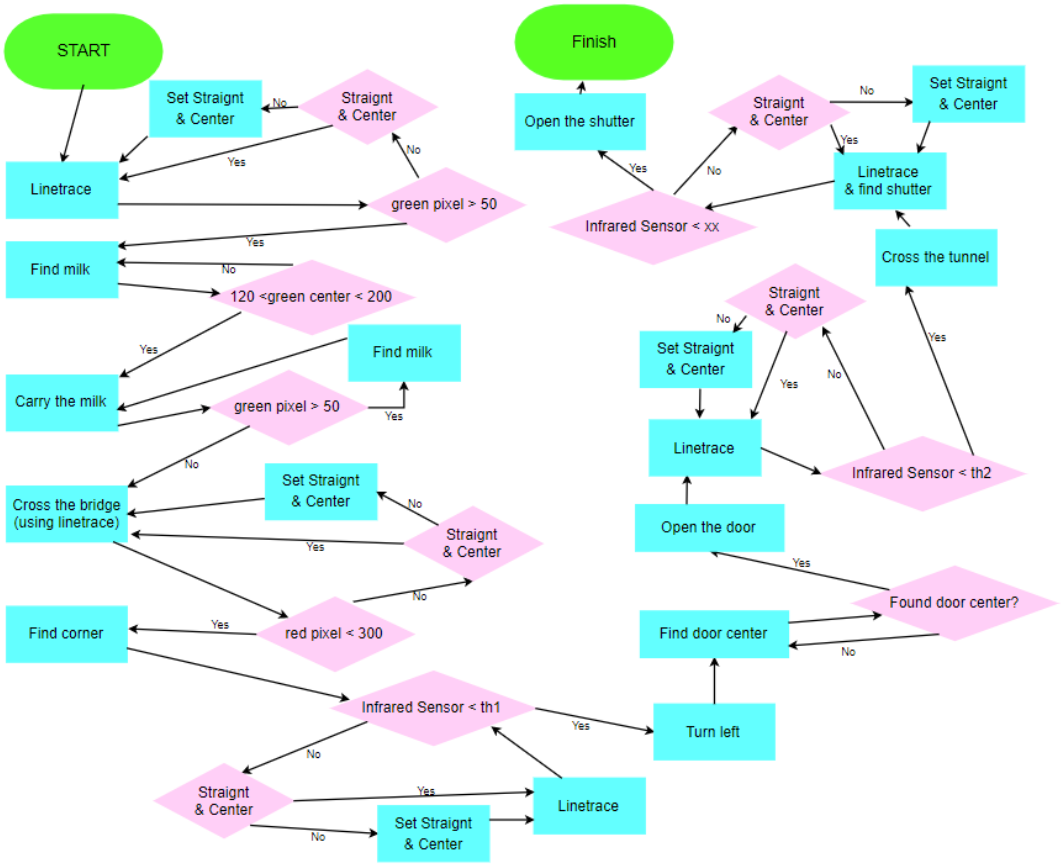
2. 개발 환경 설명 (최대한 자세하게 기술)

2.1. Software 구성

이 작품은 두뇌보드 소프트웨어와 제어보드 소프트웨어 2가지가 존재한다. 두뇌보드 소프트웨어는 라즈베리 파이를 이용하는 것으로 파이썬과 openCV를 이용해 카메라로 받아온 영상을 실시간으로 처리하고, 이를 통해 인식한 상황에 맞는 결과 값을 시리얼 통신을 이용하여 제어보드로 송신한다. 제어보드는 로보베이직을 이용하는 것으로 시리얼 통신을 이용해 받은 값으로 그에 맞는 동작을 취하도록 각각의 모터를 제어한다. 둘 사이에 통신할 때 한 동작이 끝나기 전에 다른 값을 받지 않게 하기 위해 한 동작이 끝나면 제어보드는 그에 맞는 값을 두뇌보드로 송신한다.

2.2. Software 설계도(흐름도 및 클래스 다이어그램 등 (개발언어에 따라 선택))

<흐름도>



2.3. Software 기능 (알고리즘 설명 포함)

카메라로 실시간 영상을 받아 그에 맞는 영상처리를 파이썬을 이용해 한다. 각 구간마다 조건을 주었고 조건이 성립될 시 그 구간의 알고리즘을 적용시킨다. 한 구간의 동작을 모두 수행했을 때 다음 구간으로 넘어가게 조건을 주어 이를 통해 모든 미션수행을 한 후 경기장을 완주하도록 했다.

2.4. 프로그램 사용법 (Interface)

1. 로봇의 전원을 켠다.
2. 라즈베리파이가 켜지면 와이파이 방식을 이용해 VNC로 원격연결 한다.
3. 터미널을 켠 뒤 리눅스 명령어를 이용하여 파이썬 코드를 실행시킨다.
4. 파이썬 코드가 실행되면 로봇이 구동되기 시작한다.

2.5. 개발환경 (언어, Tool, 사용시스템 등)

OS	WINDOWS 7 / Ubuntu / Raspbian Linux
개발언어	Python / Robobasic
개발도구	Pycharm / Robobasic
개발환경	Pycharm / Robobasic
기타	OpenCV library

3. 개발 프로그램 설명 (최대한 자세하게 기술)

3.1. 파일 구성

'구조 로봇'과 '시민 로봇'이 각자 다른 동작을 수행하므로 파이썬 파일을 따로 구동한다. 로봇을 구동하는데 필요한 모든 구간의 코드는 각각 하나의 파일로 구성했다.

① '구출 로봇' 파이썬 파일

② '시민 로봇' 파이썬 파일

③ '구출 로봇' 로보베이직 파일

④ '시민 로봇' 로보베이직 파일

3.2. 함수별 기능

3.2.1 line trace

'구조 로봇'은 라인을 주행하기 시작한다. 라인 주행을 위해 바닥의 노란 선에 맞게 HSV 범위를 지정하고 영상을 이진화 한다. 이진화 된 영상을 hough transform을 이용하여 선을 그린다. 그려진 선의 기울기 및 위치 정보를 바탕으로 line trace를 한다. 휴머노이드가 라인을 잘 따라왔을 경우 기울기(slope)가 무한대이고 line의 중심점이 화면의 가운데에서 멀어지거나, 기울기가 일정 각도 이상 틀어진 경우 로봇이 라인에서 벗어났다고 판단한다. 이에 각 상황에 조건을 주어서 몸을 틀거나 좌우로 이동하여 라인에 맞추어 똑바로 걷는다.

'시민 로봇'은 일정 횟수의 line trace를 하고, 고개를 들어 앞에 로봇이 있는지 확인을 한다. 만약 로봇이 없다면 다시 일정 횟수의 line trace를 하고, 로봇이 있다면 잠시 멈추어 기다린다. 이때 고개를 들고 판단을 제대로 하기 위해 2초 동안 고개를 들고 있는다.

3.2.2 우유팩 정리

'구조 로봇'은 line trace를 하다 일정 픽셀 이상의 녹색을 발견하면 우유팩을 찾기 시작한다. 일정 사이즈 이상의 녹색 픽셀이 뭉쳐 있는 곳을 우유팩이라 판단하고 Rectangle 함수를 이용해서 우유팩에 사각형을 그린다. 그 후 사각형의 꼭지점 좌표를 이용하여 x축의 센터 값, y축의 센터 값을 추출한다. 센터 값을 정해준 위치 값 사이에 존재하도록 로봇을 이동시킨다. 이동을 통해 x축, y축 센터 값이 정해진 위치 값 사이에 존재하게 된다면 로봇이 우유팩을 집는다. 우유팩을 집으면 좌, 우 정해진 위치에 우유팩을 갖다 놓아 우유팩을 중심에서 멀리 떨어트려 우유팩을 정리 하여 구간의 목표를 달성한다.

'시민 로봇'은 고개를 들어 로봇의 뒷편에 달린 녹색 픽셀의 개수를 판단하여 아직 로봇이 우유를 정리하고 있다면 기다린다. 이를 반복 수행하여 로봇이 앞으로 갔다 판단하면 line trace를 하여 따라간다.

3.2.3 외나무다리 구간

'구조 로봇'은 우유팩을 정리하고 line trace를 하면서 일정 픽셀의 빨간색 값이 검출되면 외나무다리 구간을 분석하기 시작한다. 외나무다리 구간은 기본적으로 line trace를 하면서 화면의 HSV를 분석하고 빨간색 픽셀 수를 이용하여 외나무다리 구간의 시작과 끝을 판단한다. 이 구간에서 로봇은 기존의 line trace 방법을 사용하여도 실격영역으로 벗어나는 일이 없음을 실험을 통해 알아내었다. 구간의 끝에 도달한 로봇은 고개를 들고 다음 구간에 필요한 행동을 한다.

'시민 로봇'은 고개를 위 아래로 움직이며 앞의 '구조 로봇'을 확인하며, 거리를 유지한 채 같은 알고리즘으로 외나무다리 구간을 통과한다.

3.2.4 문 통과 구간

1. 코너 인식

'구조 로봇'은 외나무다리를 건넌 후 line trace를 하면서 거리센서를 이용하여 거리를 판단한다. 이때, 두 가지 threshold를 설정해 corner를 돈다. 거리센서가 첫 번째 threshold 이상으로 잡히면 문으로 가기 전 코너에 도달했다고 생각한다. 이후 천천히 앞으로 전진하다 두 번째 거리센서 threshold에 만족할 때에 100도로 회전하고 문을 정면으로 바라본다. 이후 문 쪽을 향해 천천히 돌면서 문에 있는 EXIT를 인식한다.

2. 문 센터 잡는 함수

문에 표시되어 있는 EXIT의 녹색 픽셀을 보며 Rectangle 함수를 이용하여 사각형을 친다. 사각형의 꼭지점을 이용하여 EXIT 사각형의 왼쪽 가장자리가 중심이 되도록 로봇을 움직인다. 문과의 거리를 가깝게 하기 위해 일정 값 이상의 녹색 픽셀 값이 화면에 나올 때까지 전진한다.

3. 문 통과 함수

문과 로봇 사이에 장애물이 끼어 동작 수행을 방해하는 것을 방지하기 위해 아래 장애물을 치우는 동작을 수행한다. 장애물을 치운 뒤 문을 여는 동안 중심이 잘 무너지지 않게 하기 위해 옆으로 문을 통과한다. 이를 위해 로봇은 몸을 옆으로 90도 회전하고 일정 시간 동안 문을 통과하는 동작을 수행한다. 문을 통과하고 난 후 로봇은 다시 라인을 인식하여 line trace를 수행한다.

'시민 로봇'도 '구조 로봇'과 같은 알고리즘을 사용하여 구간을 통과한다. 이때 앞에 로봇이 있는지 고개를 들어 확인해준다.

3.2.5 터널 구간

'구조 로봇'은 문을 통과한 후 거리센서를 이용해 터널을 찾기 시작한다. 일정 거리 내 물체가 인식되기 전까지 line trace를 하면서 앞으로 나아간다. 일정거리 내 물체가 인식되면 터널로 판단하여 조금씩 앞으로 전진하다가, 적절한 거리가 되면 터널 구간을 통과하기 위한 동작을 시작한다. 로봇은 '림보 자세'로 구간을 통과한다. 로봇이 일어날 때 머리가 터널에 부딪치지 않도록 하기 위해 충분히 동작을 수행한 후 일어난다.

'시민 로봇'도 '구조 로봇'과 같은 알고리즘을 사용하여 구간을 통과한다.

3.2.6 셔터 구간

'구조 로봇'은 터널을 통과한 후 셔터가 나올 때까지 line trace를 통하여 전진한다. 거리센서를 이용하여 셔터를 발견하면 셔터 앞의 장애물을 치운다. 이후 셔터 쪽으로 한 걸음 이동한 로봇은 셔터를 올리는 동작을 수행한다. 이 때 로봇이 최대한 작은 틈 사이에 손을 넣을 수 있도록 옆으로 몸을 튼 후 동작을 수행한다. 셔터를 안정적으로 올리기 위해 로봇의 팔을 머리 위까지 올려준다. 이후 셔터를 통과할 때 머리가 부딪치지 않도록 머리를 숙인 채 셔터를 통과한다.

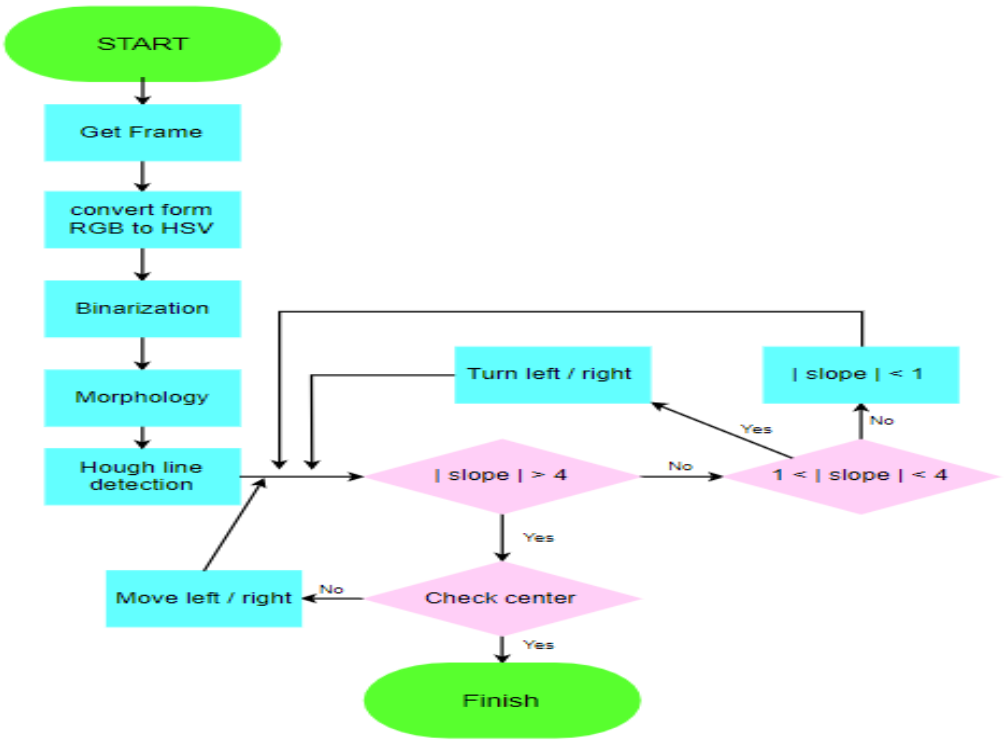
'시민 로봇'은 터널을 통과한 후 구조로봇이 작업할 시간을 위해 잠시 대기한다. 일정시간 대기 후 line trace를 하며 라인이 끝날 때까지 전진하다가 라인이 보이지 않으면 코스가 끝났다고 판단하여 앞으로 전진하여 셔터를 통과한다.

3.2.7 시민로봇의 앞의 로봇 판단 함수

고개를 드는 동작을 수행한 후 2초 동안 영상을 판단한다. 이때 앞의 로봇의 특징이 되는 녹색 픽셀이 일정픽셀 이상이면 로봇이 있고, 일정픽셀 이하면 로봇이 없다고 판단한다. 로봇이 있으면 고개를 든 채 로봇이 멀리 떨어질 때까지 가만히 있는다. 로봇이 없다면 고개를 숙이고 일정 횟수의 line trace를 한다. 이를 반복하며 주기적으로 앞에 로봇이 없는지 확인한다.

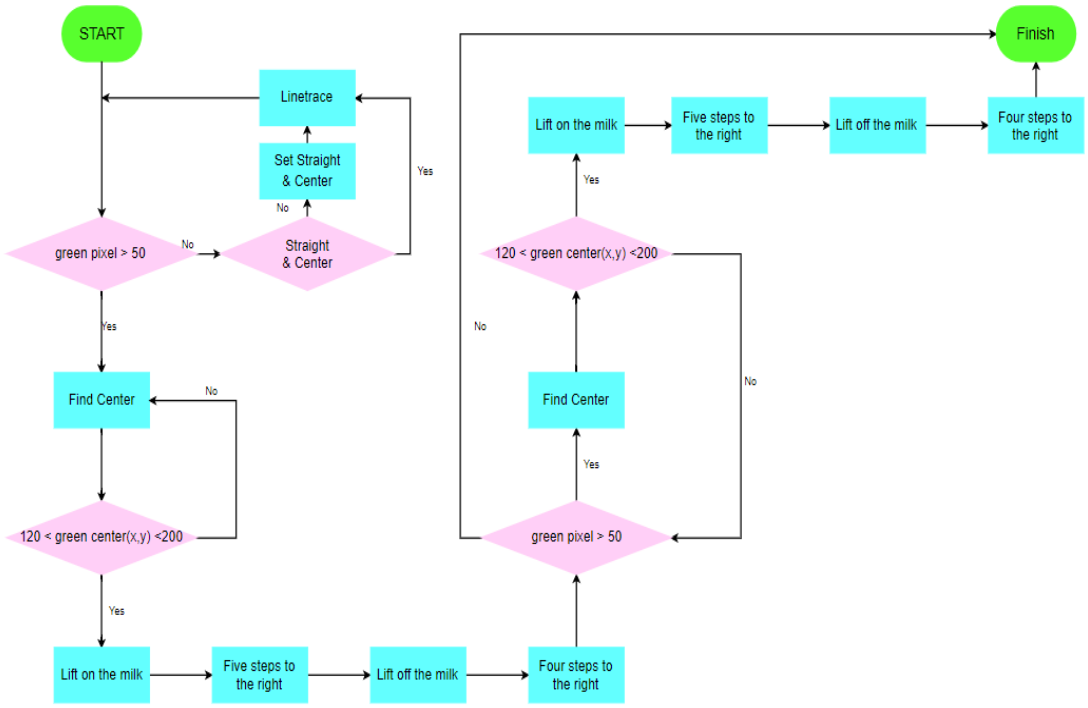
3.3. 주요 함수의 흐름도

<Line trace>



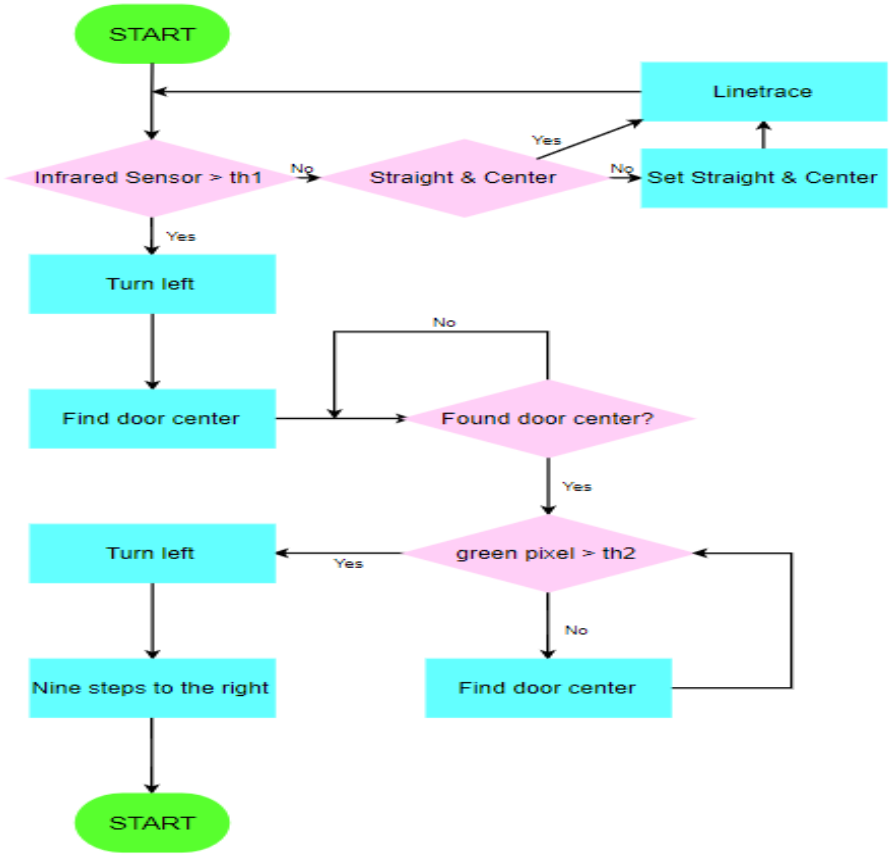
Line trace 알고리즘은 크게 직선의 기울기를 측정하는 변수와 직선의 중앙값 위치를 측정하는 변수를 이용한다. 로봇의 기울기를 총 네 구간으로 나눠 로봇의 회전 정도를 정하고 이때 중앙값의 위치에 따라 로봇의 좌우 이동 여부를 결정한다.

<우유팩 옮기기>



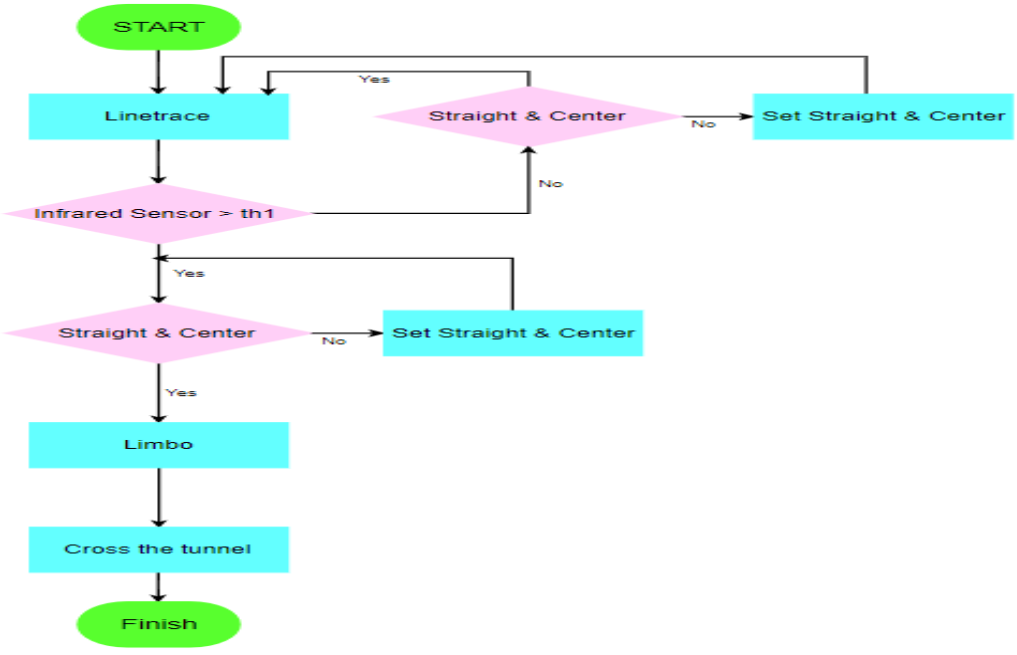
우유팩 구간은 초록색 pixel 수를 정하고, 이때 중앙점을 계산하여 우유팩의 위치를 판별 후 로봇을 이동시켜 우유팩을 옮긴다. Robobasic을 이용하여 우유팩을 놓은 후 로봇이 원위치로 이동한 뒤, 초록색 pixel 수를 센다. 이때 일정 pixel 이상이면 우유팩이라고 판단하여 미션을 수행하고 아니면 우유팩이 아니라고 판단해 다음 미션을 수행한다.

<코너돌기, 문 통과>



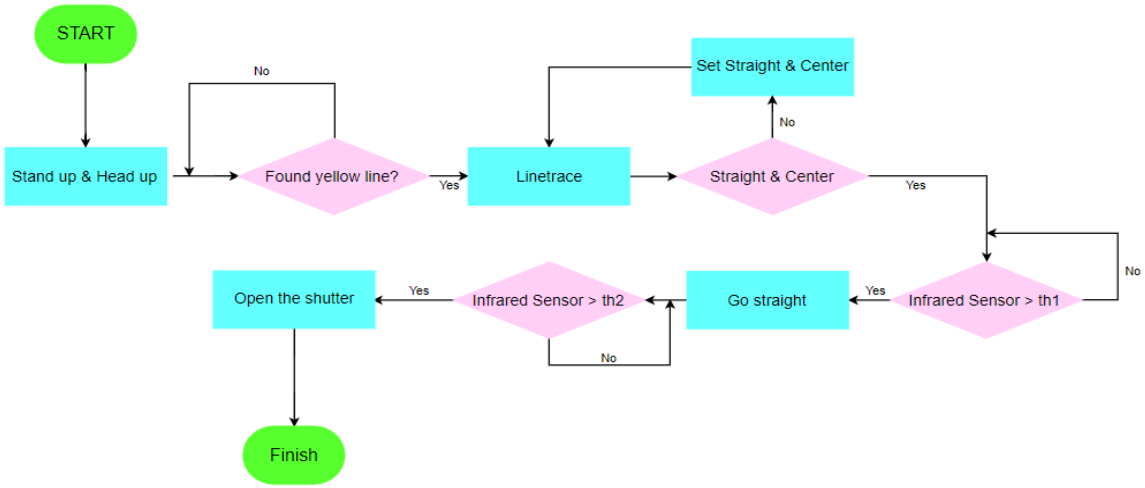
적외선 센서를 이용해 코너를 검출한다. Th1이상의 센서 값이 검출되면 왼쪽으로 회전 후 이동한다. 이때 문의 EXIT 표시에 중심을 두면서 로봇을 위치 시킨 뒤 왼쪽으로 90도 회전이동하고 9보 옆으로 전진하면서 문을 통과한다.

<터널 통과>



적외선 센서를 이용하여 터널을 인식한다. Line trace를 끝낸 후 로봇은 터널과의 거리 센서 값을 받는다. 이때 임계 값 이상을 센서가 인식하면 림보 자세를 취하고 터널을 지나가도록 하였다.

<셔터 올리고 탈출>



셔터를 통과하고 일어난 로봇이 라인 검출을 하기 위해 노란 선을 찾아 line trace를 한다. 이후 셔터도 거리센서를 이용해 검출하는데, threshold 이상의 값이 검출되면 로봇이 셔터 앞으로 이동하여 셔터를 열도록 동작한다.

3.4. 기술적 차별성

기존의 line trace 영상처리 방식은 좌/우 이동과 직진으로만 구성되있어 line trace 도중 불필요한 동작이 추가되어 비효율적인 구현방식을 선택했다. 그런데 이번 line trace는 기울기를 이용하는 방식으로 라인의 기울기를 확인하여 회전이동이라는 동작을 추가시켰다. 이에 따라 더 빠르고 정확한 동작을 수행할 수 있다고 예상한다. 더 나아가 시간이 촉박한 재난구조사항에서 보다 더 정확한 동작을 하여 미션을 수행 할 수 있을 것이라고 예상한다.

또한 기존의 Robobasic 동작 함수는 통신을 한 후 로봇을 제어하는 방식을 선택했다. 이와 같은 방식을 사용하면 라즈베리파이에서 보낸 신호를 즉각적으로 처리하는 장점을 가지고 있으나, 우유팩 집기 같은 동작 수행시간이 긴 함수를 실행했을 때 수행을 완료 하지 못하고 다른 동작을 실행하라는 신호를 수행하는 문제점이 생긴다. 이를 해결하기 위해 통신 후 동작하는 방식으로 로봇을 제어하는 방식을 선택했다. 이에 따라 동작 수행시간이 긴 함수를 완료한 뒤 다른 동작을 수행할 수 있다. 또한 신호를 안 받은 경우 계속적으로 신호를 보내는 방식으로 기존의 시스템이

가지는 단점을 해결하였다.

4. 개발 중 장애요인과 해결방안

4.1 통신 문제

처음에 제어보드와 두뇌보드 사이의 통신 방법을 알지 못해 파이썬 코드를 다 짜놨음에도 불구하고 로봇을 구동해 보지 못하는 문제가 발생했었다. 이를 해결하기 위해 기존에 제공해 준 파이썬 코드와 로보베이직 코드를 계속 연구하고 동시에 돌려봄으로써 통신 방법을 찾았다.

4.2 코너 문제

처음에 line trace 코드로 코너를 도는 방법을 선택했다. 하지만 수평라인이 보이는 순간 수평라인의 중심 값을 찾느라 앞으로 못 가고 제자리에서 좌/우로만 계속 움직이는 문제가 발생했다. Harris코너도 적용해 보았지만 문제는 똑같았다. 그래서 거리센서를 이용해 벽과의 거리를 잰 후 일정 거리가 되면 코너라고 인식하고 왼쪽으로 돌게 만들었다.

4.3 빛 반사 문제

장소와 시간이 다름에 따라 빛이 달라지면서 로봇이 동일한 물체임에도 불구하고 다른 색으로 인식하는 문제가 생겼다. 이를 해결하기 위해 실시간 영상에서 원하는 곳을 클릭하면 그와 동일한 색들만 보여주고 그에 맞는 HSV값을 알려주는 코드를 짜서 매번 실험 시작할 때마다 적용시켰다. 또한 처음에 로봇을 실행시키면 화면이 매우 밝아졌다가 어두워져서 처음과 시간이 흐른 후에 색을 인식하는데 차이가 발생했다. 그래서 로봇을 실행시킨 뒤 2초 뒤에 미션수행을 시작하도록 했다.

4.4 우유팩 2개 인식 문제

우유팩이 2개 있을 때 이미 갖다 놓은 것과 갖다 놓지 않은 것을 구분하는데 문제가 있었다. 이를 해결하기 위해 우유팩 stage부분에서는 ROI를 씌워서 라인 근처만 볼 수 있게 화각을 줄였고, 우유팩을 벽에 가까이 갖다 놓음으로써 이미 갖다 놓은 건 라인 근처에서 보이지 않게 했다.

4.5 우유팩 stage 탈출 문제

우유팩이 1개 있을 때와 2개 있을 때 구분 없이 우유팩을 다 갖다 놓은 후 stage를 탈출하게 하는 알고리즘을 짜는데 어려움이 있었다. 이를 위해 우유팩을 하나 갖다 놓은 후 뒤로 조금 걸어간 뒤 앞에 빨간 픽셀이 어느 정도 이상 보이기 전까지 우유팩 stage로 설정해놨다. 그때까지 초록색 픽셀이 발견 되지 않으면 우유팩이 없다고 판단하고 외나무다리 stage로

넘어가는 방법을 선택했다.

4.6 다음 stage로 넘어가는 알고리즘 구성 문제

한 stage가 끝난 후 다음 stage로 넘어가는 지점을 찾는데 어려움이 있었다. 그래서 stage를 탈출하는데 문제가 발생했었다. 이를 해결하기 위해 각 stage마다 특징점을 찾았고, 그 특징점들마다 변수를 설정해주는 방법을 사용해서 문제를 해결했다.

4.7 영상처리 속도 문제

처음 로봇을 받은 상태 그대로 영상처리를 하려 했을 때 해상도가 높아서 영상처리 속도가 느린 문제가 발생했다. 이 때문에 실시간이 되지 않고 로봇이 한 박자 느리게 행동하는 문제가 발생해 미션을 제대로 수행하지 못했다. 이러한 문제 때문에 해상도를 낮춤으로써 실시간으로 영상처리가 가능하도록 했다.

4.8 직진 주행 문제

여러 자세를 실험해보면서 로봇의 영점이 많이 바뀜으로써 몸이 조금씩 틀어지는 문제가 생겼다. 그래서 로봇이 직진하는 동안 제대로 직진하지 못하고 점점 몸이 옆으로 틀어지는 문제가 발생했다. 이를 해결하기 위해 로봇의 영점을 다시 설정함으로써 문제를 해결했다.

4.9 두 로봇간의 영점 차이

동일한 로봇이고 같은 동작을 수행하도록 시켰음에도 불구하고 두 로봇의 영점이 달라서 둘의 모션에 차이가 있었다. 이 때문에 두 로봇을 같은 동작을 수행하는데도 불구하고 코드를 다르게 짜는 방법으로써 문제를 해결했다.

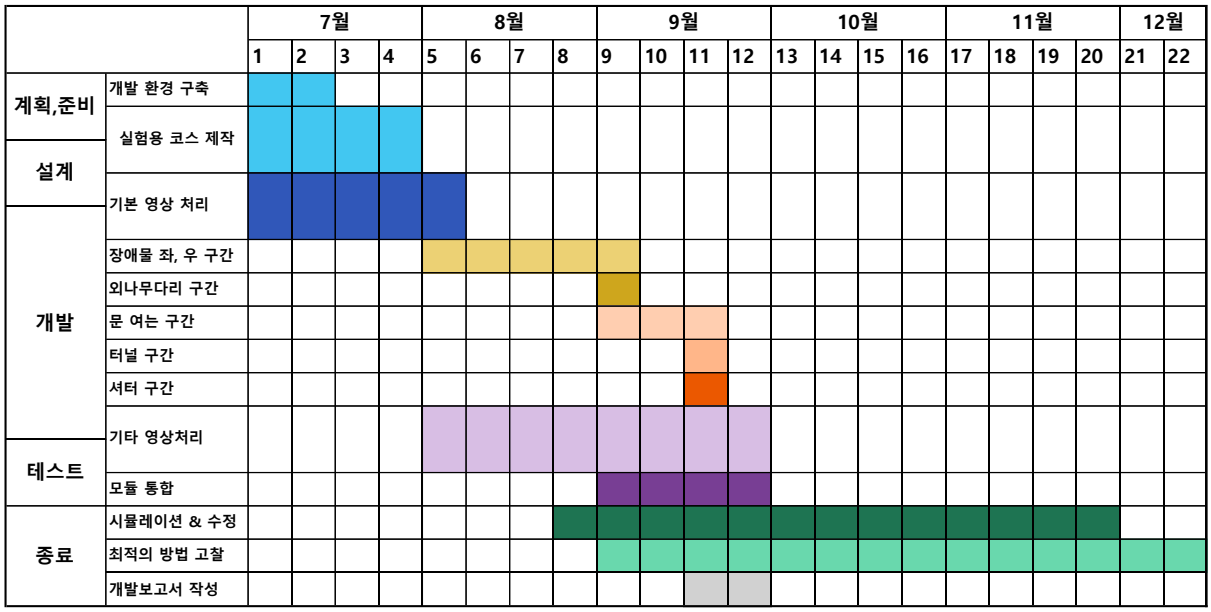
5. 개발결과물의 차별성

휴머노이드 로봇은 실제 인간의 신체와 매우 유사한 형태와 모양을 지닌 로봇이다. 이 로봇은 인간의 팔, 다리, 손, 발, 머리, 몸통 등과 같은 신체 부위와 매우 비슷한 모양과 관절을 가지며 위치 또한 적절한 위치에 있어 인간의 행동을 가장 유사하게 묘사할 수 있다. 이러한 특징을 가지고 있기 때문에 실제 인간이 처한 상황에 투입되어 적절한 대응 및 수월한 활동을 할 수 있다.

이번 지능형 휴머노이드 대회의 미션인 '재난 상황 속에서 시민 구출하기'는 재난 상황에서 실제 인간이 가장 수행하기 쉬운 구조와 상황이기 때문에 휴머노이드 로봇을 사용하는 것이 가장 바람직하다. 예를 들어, 탈출경로에 놓여진 장애물을 제거하는 동작은 인간과 비슷한 신체조건을 이용하기 때문에 다른 로봇들과 비교할 때 가장 적절하며, 비상구와 셔터 등 여러 장애물을 로봇의 손을 이용하여 효과적으로 탈출할 수 있다. 휴머노이드는 손, 발과 여러 관절 역할을 하는 모터와, 인간의 눈 역할을 하는 카메라를 활용하여 인간을 대신해 재난 상황 속에서 미션을 수행할 수 있다. 또한 인간의 모습과 유사하게 생긴 휴머노이드 로봇은 몸을 구부리는 동작으로 다른 자동차 로봇에 비해 여러 장애물 구간을 통과할 수 있다. 특히 자신의 키보다 작은 터널구간을 통과할 때

몸을 자유자재로 움직여 통과할 수 있다는 점에서 다른 로봇들에 비에 협소한 공간에서 미션을 수행할 수 있다. 특히 이번 작품을 통해 휴머노이드의 장점을 장애물 옮기기, 터널 옮기기, 셔터를 올리는 미션을 통해 잘 나타나게 해주었다. 인간과 같이 관절을 이용해 손과 발을 이용하는 점과 몸을 구부릴 수 있다는 특징을 통해 기존 로봇이 가지고 있는 한계성을 해결한 예라고 볼 수 있다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장



번호	이름	대학	학과	학년	역할	담당업무
1	김재현	숭실대학교	전자정보공학부	4학년	팀장	알고리즘 구성, 영상처리
2	김종연	숭실대학교	전자정보공학부	4학년	팀원	알고리즘 구성, 로봇동작, 보드간 통신
3	박범규	숭실대학교	전자정보공학부	4학년	팀원	알고리즘 구성, 영상처리
4	서지연	숭실대학교	전자정보공학부	4학년	팀원	알고리즘 구성, 영상처리, 보드간 통신
5	안주영	숭실대학교	전자정보공학부	4학년	팀원	알고리즘 구성, 영상처리