제17회 임베디드SW경진대회
개발완료보고서[주니어 임베디드SW챌린저]

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 20 page 이내
※ 작성 양식 (폰트 : 맑은 고딕 / 폰트 크기 : 10pt / 자간 : 0% / 장평 : 100% / 줄 간격 : 130%)
※ 제출 포맷 : pdf

1. 팀 정보

팀명	CrunchRobo	팀장	조성준
팀원	신종환	팀원	채지훈

2. 개발완료보고서

1. 개요

1.1. 작품 개요

경기장에서 교차로의 교통상황을 인식하여 출발점에서 도착점에 이르는 가장 빠른 경로를 구하여, 로봇이 그 경로를 주행하여 출발점에 도착하도록 할 수 있는 로봇과 프로그램을 만들었다.

교통상황의 인식은 컬러센서를 이용하며, 가장 빠른 경로를 찾아 로봇이 그 경로를 따라 이동하며, 주행 중 돌발상황이 발생하면, 그 상황을 고려하여 다시 가장 빠른 경로를 찾도록 하였다..

1.2. 개발 목표

1. 경기장 위를 안정적으로 주행할 수 있는 모형을 제작한다. 모형의 핵심은 컬러센서이며, 이를 통해 교차로의 정보, 교차로 인식, 주행 등 모든 것을 결정할 것이다.

2. 출발점 앞의 교통상황(실제로는 1.9 mm 폭의 연속된 색깔 테이프 9개)을 안정적으로 읽고 저장한다.

3. 저장된 교차로 정보에 따라 가장 빠른 경로를 찾아낸다. 찾아내는 방법은 다익스트라 알고리즘을 재귀적 방법으로 이용하여 구현한다..

2. 개발 환경 설명 (최대한 자세하게 기술)

2.1. Software 구성

microPython을 Visual Studio Code를 사용하여 프로그래밍하였다.

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

필요한 기능은 3가지이다. 교차로의 소요시간을 읽어오는 것, 그 정보를 토대로 최단 경로를 도출하는 것, 그리고 주행 중 사고가 발생했을 때 다른 경로를 찾는 것.

경로를 찾는 알고리즘이 두번째와 세번째에 비슷하게 사용되기 때문에 field라는 클래스를 만들어 경기장의 정보를 저장하고, 경로를 찾는 함수와 대안 경로를 찾는 함수를 만들어 경기장의 정보, 경로 정보를 함수에서 편하게 접속하고 저장할 수 있도록 하였다.

2.3. Software 기능 (알고리즘 설명 포함)

주어진 경기장의 크기, 출발점과 도착점의 위치는 프로그램에 내에 반영하고, 각 교차로에서의 소요 시간에 대한 정보가 입력되면 자체적으로 최단 경로를 구하고 그 경로를 따라 로봇이 주행한다.

교차로의 소요시간을 읽어오는 것은 여러 방법 테스트를 통하여 최종적으로는, 테이프 전체 길이

를 센서로 읽은 후 테이프 숫자만큼 나누어, 나누어진 부분의 색 데이터를 읽어 각 교차로 정보로 인식하였다.

로봇이 읽어온 교차로 시간 정보까지 입력되면, 로봇이 갈 수 있는 모든 경우의 수를 구하고, 도착점에 도달하지 못하는 경우는 폐기, 그렇지 않은 경우는 시간을 계산해 시간 순으로 저장하거나 폐기한다. 로봇은 그 중 최소 시간이 소요되는 경로에 따라 주행하며, 만약 주행 중 사고 발생 시 사고 발생 직전에 들렀던 교차로를 시작점으로 설정, 다시 도착점까지 도달하는 모든 경우의 수를 구하게 된다. 이 때 사고가 발생하는 방향으로 가야하는 경로는 모두 제외하고, 나머지 경로 중 최소 시간이 소요되는 경로로 로봇은 주행하게 된다. 또한 어느 교차로로 가는 중에 사고가 났는지를 따로 저장하여, 여러 곳에서 사고가 나더라도 잘 피해가도록 구현했다

2.4. 프로그램 사용법 (Interface)

프로그램을 EV3 다운로드하여 출발점에서 출발시킨다.

2.5. 개발환경 (언어, Tool, 사용시스템 등)

Visual Studio Code에 설치된 Extension 은 LEGO Mindstorms EV3 MicroPython, Python, Korean Language Pack for Visual Studio code 이다.

개발 언어는 Python 3.7.3

에디터는 Visual Studio Code 1.38.

LEGOMindstormEV3.

3. 개발 프로그램 설명 (최대한 자세하게 기술)

3.1. 파일 구성

개발 편의 상 하나의 파일에 라이브러리 불러오기, 함수 정의, 로봇 주행 명령문 등을 모두 작성하였다.

3.2. 함수별 기능

buttonWait: 피브릭 버튼을 누를 때까지 대기하고, 버튼을 누르면, 소리를 내고 다음으로 넘어간다.

pCont:라인을 따라갈 때 로봇의 움직임을 컨트롤한다.

searchRoute:로봇이 출발점에서 시작해 도착점까지 도달할 수 있는 모든 경로 구하는 함수. 들르는 교차로의 번호가 저장되는 node_route 객체와 로봇의 이동 방향이 저장되는 direc_route 객체를 구분한다. 경로를 구하는 행위, 완전한 경로를 메모리에 저장하는 행위는 모두 searchRoute 함수가 호출하는 재귀 함수인 _getRoute 함수에서 이루어진다.

_getRoute:로봇이 출발점에서부터 인접한 교차로로 움직였다고 가정, 움직인 교차로의 번호와 움직인 방향을 인자로 받음. 만약 제일 마지막으로 도착한 교차로가 도착점과 일치한다면, 인자로 받은 데이터를 getTime 함수에 넘기고 재귀를 끝낸다. 가끔 도착점에 도착해 재귀를 끝냈는데도 계속 경로를 구하는 경우가 있어, 움직인 교차로 데이터에 도착점이 포함된 경우에도 재귀를 끝내도록 하였다. 앞 두 경우를 제외하면 아직 로봇이 경기장 중간에서 이동 중인 경우이므로, 위, 아래, 오른쪽, 왼쪽 방향으로 갈 수 있는지, 이미 지나온 교차로는 아닌지를 확인한다. 로봇이 물리적으로 갈 수 있고, 해당 방향의 교차로를 지나지 않았다면, 해당 방향의 교차로 번호와, 해당 방향을 각각 인자로 받은 데이터에 추가, 다시 재귀적으로 자기 자신을 호출한다.

isAccident:로봇이 실제로 주행 중일 때 응급 상황인지 확인하기 위하여 실행된다. 로봇이 가는 중에 사고가 났음을 확인했다면, 다시 돌아가서 해당 교차로에서부터 도착점까지의 새로운 경로들을 찾고, 경로들 중 사고가 난 방향에 있는 교차로가 있는 경로는 제외한다.

getTime:각 경로의 소요 시간을 구함. 교차로 번호 리스트와 이동 방향 리스트를 인자로 받음

3.3. 주요 함수의 흐름도

3.4. 기술적 차별성

최단경로를 구하는 방법에 있어서, 재귀 함수를 통해 로봇이 경기장에서 갈 수 있는 모든 가능성을 확인하고, 모든 경로를 일단 저장한 후 시간에 따라 가장 적은 시간이 걸리는 경로들만 저장

하는 것. 사고가 발생했을 때 그 교차로를 출발점으로 설정하여 다시 최단 경로를 구하여 문제를 해결하는 것 등, 결선경기에서의 경기장 변형에 대한 여러가지 가능성에 대비하였다.

출발점 앞의 교차로 정보 인식을 위하여 여러 방법과 실험을 통하여 그 정확성을 높이기 위하여 노력하였다.

4. 개발 중 장애요인과 해결방안

최단거리 찾는 개발 과정이 가장 어려웠다. 팀원들과 회의도 많이 하였고, 선생님의 도움도 받았고, 여러 책을 참고하기도 하였다. 그런 과정에서 새롭게 많은 개념을 알게 되었고, 결국 구현하게 되었다.

5. 개발결과물의 차별성

주행을 가끔 성공하는 것이 아닌, 항상(또는 대부분의 테스트에서) 성공할 수 있는 모형과 프로그램을 만들고 싶었다. 처음 시작할 때와 달리 지금은 그 목표에 반 이상은 도달한 느낌이다. 남은 기간 더 노력하여 빠르고도 안정적으로 항상 최단 거리를 찾아 도착하는 모형을 완성하고 싶다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

해커톤 대회까지는 팀원끼리 힘을 합쳐 모형과 프로그램을 작성하여, 다행하게도, 결선진출하였으나, 결선일이 기말고사와 겹쳐 팀원 2명은 불참할 의사를 밝혔다. 1명으로도 충분히 대회는 참가할 수 있으나, 안따까운 마음이다