Line Tracer Team Project Document

2021038131 장준혁, 201902788 신기섭

1. API 개요

프로그램의 개발 편의성 및 유지, 보수를 용이하게 하기 위해서 여러 단계의 API로 추상화하였다. API는 총 3단계로 구성된다: 레지스터, IO 입출력 등 머신과 가까운 로직을 다루는 로우레벨 API; 로우레벨 API를 바탕으로 n (cm) 직진, n (도) 회전, 곡선을 이탈하지 않고 따라가기, 교차로 까지 직진 등 자주 사용되는 기능들을 구현한 하이레벨 API; 마지막으로 하이레벨 API를 바탕으로 각 장애물 구간을 통과할 수 있도록 구현한 트랙 주행함수이다.

1-1. 로우레벨 API

레지스터, IO 입출력 등 머신과 가까운 로직을 다룬다. 대부분의 함수는 실습에서 제공한 함수들을 그대로 사용했다. 적외선 센서 함수는 편의성을 위해 각 센서의 값을 서로 다른 8개의 전역 변수에 저장하도록 개선하였다.

대표적인 함수 목록은 다음과 같다.

`void ir_init(), void moter_init(void)` 등등: 기계를 초기화하는 함수들

`int ir_read()`: 센서들을 읽는 함수. 읽은 센서의 값은 8개의 전역변수에 저장되어 어디서든 접근할 수 있다.

`void move(uint16_t leftDuty, uint16_t rightDuty)`: 왼쪽 모터의 속도를 `leftDuty`, 오른쪽 모터의 속도를 `rightDuty`로 설정한다.

1-2. 하이레벨 API

로우레벨 API를 바탕으로 n (cm) 직진, n (도) 회전, 곡선을 이탈하지 않고 따라가기, 교차로까지 직진 등 자주 사용되는 기능들을 구현하였다.

대표적인 함수들은 다음과 같다.

`void rotate(int degree)`: 제자리에서 `degree`도 만큼 회전한다. `Degree`가 양수면 시계방향, 음수면 반시계 방향으로 회전한다.

`void rotate_solely(int degree)`: 한 쪽 바퀴는 정지한 상태에서 한 쪽 바퀴만을 사용하여, `degree`도 회전한다. 로봇의 위치가 바뀔 수 있다. 마찬가지로 `degree`로 양수가 주어지면 시계 방향으로, 음수가 주어지면 반시계 방향으로 회전한다.

`void MJSt(int speed, int distance)`: 주어진 거리를 주어진 속도로 직진한다. 경로는 보정되지 않는다.

`void moveFAR(int speed, int checkObstacle, int time, int degree, int ignore_cnt, int strict)`: `speed` 인자로 주어진 속도로 곡선 경로를 따라가다가, 회전해야 할 방향의 센서에 검은색 선이 입력으로 들어오면, `degree` 인자만큼의 각도만큼 회전한다. 이 때, `ignore_cnt`로 0보다 더 큰 값이 전달된다면, 그 횟수만큼 회전해야 할 방에 검은 선이 감지되어도 그것을 무시한다. 이 함수는 경로를 조금 이탈했을 때 좌우측 바퀴의 속도를 조절함으로써 경로를 보정한다. 경로 보정 알고리즘은 하위 항목에서 자세히 서술되어 있다. 다만, 곡률이 너무 심하다면 경로를 이탈할 가능성이 있다. `strict` 매개변수로는 0이나 1이 전달될 수 있는데, 1일 경우 회전해야한다는 판단을 더 쉽게 내린다. (다시 말해, 회전해야 한다고 결정을 내리는 조건이 느슨하고, 민감도가 높다.) 반대로 0일 경우에는 직진해야한다는 판단을 더 쉽게 내린다. (다시 말해, 회전해야 한다고 결정을 내리는 조건이 더 까다롭다.)

`void moveFARWithoutAdjust(int speed, int checkObstacle, int time, int degree, int ignore_cnt, int strict)`: `speed` 인자로 주어진 속도로 곡선 경로를 따라가다가, 회전해야 할 방향의 센서에 검은색 선이 입력으로 들어오면, `degree` 인자만큼의 각도만큼 회전한다. 이 때, `ignore_cnt`로 0보다 더 큰 값이 전달된다면, 그 횟수만큼 회전해야 할 방에 검은 선이 감지되어도 그것을 무시한다. 이 함수는 경로를 조금 이탈했을 때 소폭 회전 및 소폭 이동을 함으로써 경로를 보정한다. 경로 보정 알고리즘은 하위 항목에서 자세히 서술되어 있다. 다만, 곡률이 너무 심하다면 경로를 이탈할 가능성이 있다. `strict` 매개변수로는 0이나 1이 전달될 수 있는데, 1일 경우 회전해야한다는 판단을 더 쉽게 내린다. (다시 말해, 회전해야 한다고 결정을 내리는 조건이 느슨하고, 민감도가 높다.) 반대로 0일 경우에는 직진해야한다는 판단을 더 쉽게 내린다. (다시 말해, 회전해야 한다고 결정을 내리는 조건이 더 까다롭다.)

1-2-1. 경로 보정 알고리즘.

센서의 번호는 아래의 그림1과 같다.

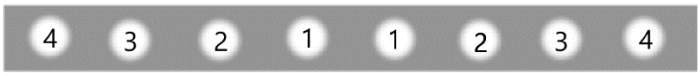

그림 1. 센서 번호

2번 센서는 경로 보정을 위해 사용한다. 3번센서는 경로를 심하게 이탈했을 때 강한 경로 보정을 하기 위해 사용된다. 4번 센서는 교차로 및 장애물 체크를 위해 사용된다.

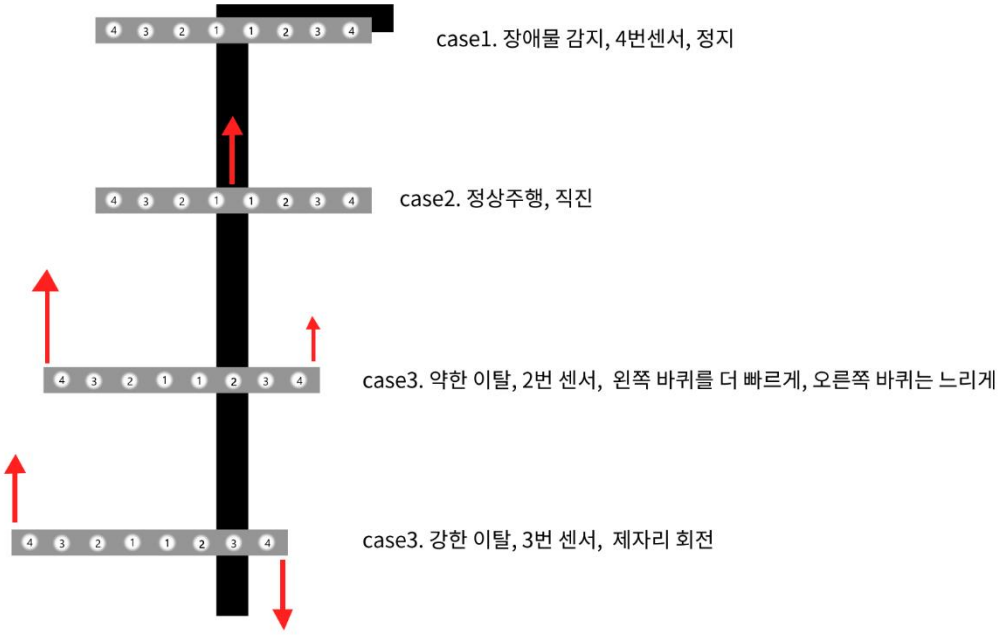

그림 2. 주행 알고리즘

그림 2의 case 1과 같이 회전하는 방향과 같은 쪽의 4번 센서에 검은 선이 감지되면 장애물 또는 교차로 판단하고 정지한다.

그림 2의 case 2과 같이 2, 3, 4번 센서에 검은 선이 감지되지 않으면 직진한다.

그림 2의 case 3과 같이 2번 센서에 검은 선이 감지되면 경로를 이탈한 것으로 판단하고 `moveFAR(...)` 함수의 경우 왼쪽 바퀴를 더 빠르게 오른쪽 바퀴를 느리게 설정하여 천천

히 우회전한다. `moveFARwithoutAdjust(...)` 함수의 경우에는 시계 방향으로 약간 회전한 뒤 짧은 거리를 앞으로 이동한다.

그림 2의 case 4과 같이 3번 센서에 검은 선이 감지되면 경로를 심하게 이탈한 것으로 판단하고 제자리에서 우회전한다.

1-3. 트랙 주행 함수.

실제 트랙을 주행하기 위해 최종적으로 사용되는 함수이다. 하이레벨 API를 바탕으로 각 장애물 구간을 통과할 수 있도록 구현했다.

`track1(), track2(), …, track6()` 으로 총 6개의 작은 구간으로 전체 트랙을 분할했다.

2. 업무 분담

Name	Contribution (100%)
장준혁	100 %
신기섭	100 %

3 단계 API 설계 및 전체 프로그램의 초안은 장준혁 학생이 작성하였다. 초안 알고리즘을 기준으로 처음에 3분 50초를 기록했다. 그 후 신기섭 학생이 기록 개선을 위해 속도 파라미터를 최적화하고 하이레벨 API의 핵심적인 로직을 수정했고 하이레벨 API의 변경에 따라 트랙 주행 함수도 수정하여 최종적으로 3분 31초를 기록했다.

3. 기타