3차 과제 보고서

20171620 문성찬

– 과제 이름
ㄱㄴ 장애물 통과

– 과제 개요
go_any라는 전진/후진을 담당하는 모듈과 TurnModule라는 스윙턴과 포인트턴을 담당하는 모듈과 ultraModule라는 대상과의 거리를 측정하는 초음파를 담당하는 모듈을 이용해 1차 주행은 오른쪽으로 스윙 턴을 먼저 하고 포인트 턴을 나중에 하는 코드를, 2차 주행은 왼쪽으로 포인트 턴을 먼저 하고 스윙 턴을 나중에 하는 코드를 작성하는 것입니다.

– 요구사항 분석
전진/후진을 담당하는 go_any 모듈에서는 이번 과제에서 필요로 하는 제한 시간 없이 전진/후진을 하는 동작 코드를 구현해야 하고, TurnModule 모듈에서는 이번 과제에서 핵심 동작이라 할 수 있는 한쪽 바퀴가 멈추고 다른 쪽 바퀴만 움직임으로써 회전하는 스윙턴과 양쪽 바퀴가 서로 반대로 돌면서 회전하는 포인트턴을 구현해야 했으며, ultraModule 모듈에서는 앞 대상과의 거리를 초음파를 통해 측정해 거리를 결과로 반환하는 코드를 구현해야 합니다.
메인 주행 코드에서는 총 장애물을 2개 통과할 것인데 각 장애물에서 수행해야 할 회전 형태가 다르고 2개의 장애물을 지나면 제한 시간 없이 계속 전진하는 것이 아닌 약 50의 속도로 3초 주행하게끔 구현해야 합니다.

– 프로그램 설계 & 코드 주요 부분 설명
go_any 모듈에서 전진/후진 코드는 지난번에 초반 삼륜 구동체의 전진/후진 코드를 이용해 파라미터를 수정하고 코드를 수정함으로써 구현했고, TurnModule 모듈에선 스윙턴 같은 경우 한쪽 바퀴의 속도를 0으로 설정하고 다른쪽 바퀴의 속도만 speed 파라미터로 받아오게끔 설계했고 포인트턴 같은 경우 한쪽 바퀴를 전진으로 하고 speed 파라미터를 받아왔다면 반대쪽의 바퀴는 후진으로 설정하고 speed를 파라미터로 받아오게끔 설계했습니다.
UltraModule 모듈에서는 time함수를 이용해 왕복시간을 계산하고 거리 = 속도*시간 공식을 이용해 해당 거리를 리턴해서 받아오도록 설계했습니다.
메인 코드에서는 장애물과 구동체 사이의 제한 사거리를 먼저 선언해주고, 구동체와 장애물과의 거리가 제한 사거리 보다 클 때에는 직진을 하게끔 아닌 경우에는 회전 동작을 수행하게끔 설계했습니다. 그리고 각 장애물마다의 다른 패턴의 회전과 두 장애물을 모두 지나고 마지막으로 약 50의 속도로 3초 직진하는 동작을 수행하게끔 하기 위해 True/False의 값을 갖는 변수를 이용해 특정 조건이 충족될 때마다 다음 단계의 동작을 수행하게끔 설계했습니다.

– 프로그램 구현 및 평가 결과

처음 코드를 완성하자마자 구동체가 원하는데로 동작하지 않았습니다.
바닥의 재질에 따라 바퀴의 굴러감도 달랐고 할때마다의 배터리의 용량도
모터의 세기에 영향을 많이 끼쳐서 다소 정확하게 동작시키는데 어려움이 있었던 것으로 생각합니다. 일단은 두 모터의 세기가 다소 달라 직진부터 똑바로 안되었기 때문에 go_any파일에 있는 전진/후진 코드의 잘 안 굴러가는 바퀴의 속도를 반대쪽 바퀴의 속도보다 조금 높게 설정함으로써 직진을 똑바로 가게끔 조정했습니다.
이 3차 과제에서의 무엇보다 중요한 점은 90도를 얼마나 정확히 도는 점인데, 그 부분에 대해서 많은 고민을 한 결과, 90도를 한번에 돌려고 하는 것보다 조금 더 세밀하게 턴의 동작을 나눠줌으로써 구동체의 회전을 구현하는 것이 좋다고 결론을 내렸고 0.1초씩 약 5개의 회전수행 코드를 넣어줌으로써 회전 동작을 구현했습니다.
정작 시험 때, 배터리의 용량에 다소 신경 쓰지 못한 신경이 있어 완벽히 동작을 수행하진 못하였지만, 다음부터는 배터리의 용량이 충분한 상태에서 구동체의 동작을 통해 코드를 수정함으로써 좀 더 실제 시험에서 완벽하게 수행할 수 있도록 노력해야겠다고 다짐했습니다.

– 결론 & 감상문

구동체 동작의 편리함과 정확성을 위해서 4륜 구동체에서 3륜 구동체로 구동체를 개조했다. 3륜 구동체로 바꿀 때 전선을 끼는 과정에서 전선 끼는 부분이 다소 어려운 곳에 위치해 있어서 애를 먹었으나, 곧 잘 조립해 나갔고 억지로 좁은 곳에 전선을 끼우는 방법 보다 그 부분만 따로 조립을 해체해서 하는 방식으로 조금 더 효율적으로 조립을 했었다. 3륜 구동체로 바꿈과 동시에 기존에 LCD모니터를 동원하여 라즈베리파이에 접속하는 것에서 발전해서 개인 랩탑에서 원격으로 라즈베리파이에 접속하는 VNC를 이용해서 조금도 코드 작성이나 파일 전송하는 것에 쉽게 다가설 수 있었다.
이번에 교수님이 내주신 3차 과제는 간단하게 설명하자면 구동체가 두개의 장애물을 마주칠 것인데, 처음에 전진하다가 첫번째 장애물과 마주치면 한쪽 바퀴는 멈추고 다른쪽 바퀴만 움직여서 회전하는 스윙턴의 방식으로 장애물을 90도로 통과하는 것이고 그렇게 다시 전진하다가 두번째 장애물과 마주치면 두 바퀴가 각자 다른 방향으로 돌면서 회전하는 포인트턴의 방식으로 장애물을 90도로 통과하는 과제였다. 처음에는 구동체에 내장되있던 다른 메서드나 기능들에 대해서 몰랐으나 이번 과제를 함으로써 조금은 내장되어있던 메서드의 기능들이나 그 메서드들을 어디에 사용하는 지 알 수 있는 기회가 되었다.
교수님이 한 개의 코드 안에 모든 함수와 메서드를 집어넣어서 복합적으로 구현해도 된다고 하셨지만 import를 통해 같은 디렉터리내에 다른 메서드들이나 기능들을 불러오는 기능을 사용하고 싶은 나는 특정 수행코드를 모듈로 만들어봄으로써 과제 코드를 구현하려 노력했다. 첫 번째로 구현한 모듈은 go_any 모듈로 구동체의 전진과 후진을 담당하는 모듈이었다. 이 과제에서 필요한 기능은 특정 제한된 시간없이 계속 전진/후진하는 코드와 특정 시간동안 전진/후진하는 코드였다.
이 두개의 구현된 코드에서의 차이는 다르지 않았지만 전진과 후진을 설정하고 속도를 설정함으로써 많은 기능들을 요했고 또 나는 그것들을 세밀하게 이해해보려고 노력했다. 두번째로 구현한 모듈은 TrunModule 모듈로 위에 말한 구동체의 스윙턴과 포인트턴의 동작을 포함한 코드라고 할 수 있다. 이 모듈도 go_any코드와 마찬가지로 좌측/우측 모터의 전진/후진의 설정을 정하고 속도를 정함으로써 턴의 유형에 따라 알맞게 코드를 구현하였다.
마지막으로 구현한 모듈은 ultraModule 모듈로 이번 과제에서 나에게 가장 생소하게 왔던 기능들부터 조금은 이해하기 힘든 메서드까지 포함하고 있었다.

간단하게 말하자면 이 모듈은 장애물과의, 즉 앞에 있는 대상과의 거리를 초음파를 통해 측정하고 그 거리를 결과로 반환해오는 모듈이다. 나는 그 코드를 천천히 정독하고 옆에 주석을 넣음으로써 천천히 이해해갈 수 있었다.
암튼 이 세개의 모듈을 구현하고 메인 과제 수행 코드를 작성했을 때, 세개의 모듈에서 특정 메서드를 가져오는 것은 문제가 없었지만, 과제에서는 특정 조건에 따라 수행되어야할 메서드가 다 달랐기 때문에 그것들을 코딩을 통해 설정해 줘야하는데 그 부분에서 많이 고민한 것 같았다. 몇몇 친구들의 조언을 들어본 결과, 그 친구들은 obstacle, 즉 장애물의 갯수를 반환하는 변수를 설정한 뒤 그 값이 바뀜에 따라 수행되어야할 메서드도 다르게 구현했다는 것이다. 처음에는 솔깃했으나 조금 더 효율적인 방법을 모색한 나는 장애물의 갯수로 변수를 설정한 것이 아닌 참과 거짓의 boolean형태의 변수를 몇개 잡아줘서 이 값이 True나 False면 특정 메서드가 실행되게끔 구현했다.
그 결과 해당 과제에 대한 코드를 쉽게 작성할 수 있었다. 또한 나는 다른 친구들이 스윙턴이나 포인트턴으로 90도를 한번에 돌려고 할 때, 조금 더 세밀하게 턴의 동작을 나눠줌으로써 구동체의 회전을 구현했다.
하지만 코딩을 다했다고 해서 구동체가 원할히 움직이지 않았다. 예를 들어 일직선 전진을 휘게 전진한다는지 90도 회전을 60도 회전한다는지 수정해야할 부분들이 많았고 개선해야할 문제점들이 많았다. 모터 쪽 바퀴를 바꿔보기도 했고 코드 중 일부를 수정해보기도 했으며 구동체가 원활히 굴러가게끔 열심히 노력한 과제였다고 생각한다.