시행착오

하드웨어

- 소켓 위치 잘못 설계하면 가려져서 못 꽂음. 완성 상태 생각하고 소켓 설치할 것.
- 이번에 라즈베리파이 꽂을 때 메인보드 ADC 위치랑 라즈베리파이 CSI핀 위치가 겹쳐서 끝까지는 안 꽂혔음. 부품 모양 제대로 확인하고 배치할 것.
- 부품 배치할 공간 부족해서 쓰레기통 새로 삼.
- 부품을 어떻게 설치할지 생각해놨어야 함. (부품 규격 등)
- 서포터 없이는 제작이 힘들 것(특히 다층 구조)
- 회로 설계 시 전원이 여러 개면 그라운드 전부 통일할 것. 특히 노트북과 연결하여 사용할 시 노트북 그라운드도 제대로 다 연결했는지 확인할 것.
- 모터드라이버 사용 시 성능 확인 및 유니폴라/바이폴라 작동방식 정확히 숙지할 것.
- 센서 수광/발광 파장 생각하고 구입할 것(이번에는 조금밖에 차이 안 나서 괜찮았음.)
- 입출력부분에서 제대로 작동하지 않을 경우 프로세서 고장 의심할 것. 아두이노에서 ADC사용할 때, 그 핀(A0핀)이 고장나서 바이어스 전압 생겨서 측정이 제대로 안 됐음.
- 모터 상 순서 확실히 알아볼 것.
- 모터 제대로 작동하지 않을 시 접촉불량 확인해볼 것.
- 플라스틱에 구멍뚫을 때 인두기로 뚫으면 편함.
- 하드웨어 디버깅할 때 반드시 오실로스코프로 찍어볼 것.
- 파워 모터드라이버에서 끌어썼더니 제대로 작동 안 함. 파워는 제대로 된 거 하나 만들어서 쓰자.
- 전원 방향은 항상 유의하자.

소프트웨어

- 파이썬은 속도가 좀 느려서 탈조가 잘 일어났음.(속도 때문인지 불확실, 여러 오류 있었음)
- SLA7026사용 시 phase를 반전해서 줘야 함.
- 라즈베리파이에서 gpio가 8bit가 아니라 32bit라서 memory map 사용 시char*로 포인터를 주면 안 되고 int*로 줘야 함.
- 라즈베리 1과 2,3의 gpio base address가 서로 다름
- clock_gettime함수의 tv_nsec값은 1,000,000,000ns=1sec마다 초기화됨. 현재 시간에서 시작 시간을 뺐을 때 음수가 나오는 경우는 버퍼 오버플로우로 인해 발생하는 현상이 아니므로, unsigned를 쓰거나 int의 최댓값을 더하거나 하면 안 되고 1000000000을 더해주어야 함. 애초에 사용 식이(Tc-Ts)<dt인데, Ts+dt>1000000000이면 문제가 된다. 이럴 경우 Ts에서 1000000000을 빼 주면 될 듯하다.
- memory map 함수 mmap 사용하려면 superuser권한 있어야 함. 아니면 아마도 segmentation error 날 것,
- segmentation error는 mmap에서 할당한 것 외에 접근했을 때 발생함.
- printf함수가 호출된 후 segmentation error가 발생했다 하더라도 printf결과가 무시될 수 있음. 아마도 버퍼가 날아가서 그런 듯.

- 라즈베리파이 GPIO 레지스터는 write-only인 듯. 값을 변경하여 출력을 변화시킬 때, 실제로 출력이 변화되더라도 읽은 값은 일정하였음. 따라서 읽은 GPIO값에 기반한 수정은 하면 안 됨.
- 프로그램 실행이 종료되어도 GPIO레지스터 값 그대로 남아 있음. 아마도 껐다 켜도 마찬가지. 반드시 초기화 루틴 / 강제 종료 시 초기화하는 루틴 필요.
- gpio 핀 mmap으로 할당받아놓고 실수로 이상한 범위 초기화하면 ssh가 끊겼다. 아마 인터넷 관련 부품 건드린 듯. mmap쓸 때는 데이터시트 보고 정확한 범위만을 할당받자.
- 만약 인터넷이 연결되지 않을 때에는 아래의 커맨드를 순서대로 입력해보자.

  ifconfig wlan0 up
  sudo wpa_cli reconfigure
  sudo systemctl restart wpa_supplicant
  sudo systemctl daemon-reload
  sudo systemctl restart dhcpcd
- 만약 노트북 등에서 핫스팟을 켜서 라즈베리파이를 접속시키려 하는 경우라면, 5G를 사용하는 환경이 문제였을 수 있다.
- 커스텀 라이브러리와 표준 라이브러리를 한 프로세스 내에서 함께 사용하지 말자. 오류 나더라.
- 센서 프로세스와 모터 프로세스 한 번에 실행시키니까 오류 난다. 적당히 텀을 두는 것이 좋을 듯하다.

탐색 알고리즘

1. DFS 탐색
   - 추정 오차 조정: 초기 버전은 오차를 x, y를 따로 축적하였다. 그리고 이미 방문한 노드에 이웃한 노드를 방문하게 된다면 현재의 추정 오차를 갱신한다. 그런데, 오차를 x, y 따로 관리하게 된다면, 오차를 갱신한 뒤 다음 정점을 방문할 때 (x, y 중 하나의 변수에 대하여) 로봇이 인식하는 위치에 누적된 오차보다 갱신한 오차가 너무 작아 정점을 인식하지 못하는 경우가 발견되었다. 이를 해결하기 위하여 오차의 누적은 x, y를 따로 관리하지 않고 원의 반지름으로 표현하여 저장한다.
   - 방문한 노드가 이전에 방문한 노드인지, 아닌지를 판단하는 로직의 정확도를 높이기 위한 노력
     1) 정점의 이전 방문 여부를 판단하는데 누적 오차를 줄이는 것이 중요한 역할을 한다. 이 값이 너무 크다면 방문한 적이 없는 정점도 방문한 정점으로 판단하게 된다. 따라서 이 값을 최대한 합리적으로 줄이는 것이 필요하다. 그러므로 로봇이 정점을 방문하는 방향을 최대한 원을 그리듯이 방문하도록 설정함으로써, 이미 방문한 정점의 오차와 비교하여 오차를 보정한다.
     2) 이전에 찾은 정점이 지금 찾은 정점과 같은지를 확인할 때 단지 위치만 생각하는 것이 아니라 그 노드의 Junction 또한 같은지 확인한다.
2. 최단 경로 탐색 알고리즘
3. 웹 서버와의 통신 및 비동기 구현

    - std::async를 호출할 때 레퍼런스 형 변수를 받는 함수를 비동기로 호출하려면 std::ref로 인자를 감싸야한다.
4. 리눅스에서 c++ 컴파일 하기: g++ *.cpp -o output -Wall -lm -static -std=gnu++17 -lstdc++fs -pthread
5. 어라 저거 위에거 했는데, std 라이브러리가 안 먹더라. 그래서 sudo g++ *.cpp -o algorithm -std=gnu++17 -lstdc++fs -pthread 씀.

웹 어플리케이션

- Javascript에서 무언가 잘 작동하지 않을 때 Type 검사를 해 보는 것이 좋다.
- json을 주고 받을 때에는 string으로 많이 주고 받는데, json으로 받은 데이터를 javascript에서 사용할 때 type을 신경쓰자.
- parseInt나 parseFloat 함수는 string을 특정 리터럴로 바꾸어주는데, 웹에서 사용하는 100px이나 100%와 같은 string도 숫자로 바꾸어 준다. 정말 편하다.
- 웹에서 서버로 json파일을 POST할 때 request.body가 undefined로 나올 수 있다. 이 때 body-parser을 설치하여 해결하면 된다.
- request나 response할 때에는 꼭 content type을 명시하자. 그냥 header 잘 구성하자.
- javascript를 쓸 때에는 비동기임을 명심하고, 타입을 잘 파악하자.