1. Car

1-1. 역할

# 차에서 발생하는 각종 raw data 들을 인접한 교차로에 위치한 신호등 컨트롤러로 전송한다.

# 신호등 컨트롤러로부터 현재 신호의 남은시간, 다음 신호 유지시간 정보를 수신한다.

# 자신의 차량의 상태를 인접한 차량과 공유하여 응급상황이나 교통정보를 제공한다. (미정)

1.2 흐름

getStatus() 를 통하여 차량의 실시간 상태를 측정하여 Car 구조체 멤버로 담고, sendStatus() 로 신호등 컨트롤러에게 차량의 상태(Car 구조체)를 전송한다.

이 후, 차량은 Sharing() 을 통하여 자신의 차량의 현재 상태를 공유하고, recvSigInfo() 로 현재 신호 및 다음 신호의 정보를 신호등 컨트롤러로부터 수신한다. 최종적으로 개별 차량은 현재 교차로에서 신호와 다른 차량의 상태를 포함한 전체 교통정보를 getRoadInfo() 를 통하여 확인한다.

1.3 모듈 설계

* Car 구조체 멤버 (typedef struct carData)
* int id // 차량 고유 번호
* int lane\_num // 도로 번호
* int curX // gps 상 x좌표
* int curY // gps 상 y좌표
* int speed // 현재 속도
* int distance // 정지선까지의 거리
* int emergency // 응급상황의 정도
* char direction // 차량 주행방향
* char cur\_time // 현재 시간
* int wait\_time // 정지차량의 대기시간
* Car-client function
* void getStatus(Car \*) // 차량 상태 실시간 측정
* void sendStatus(int \*, Car\* ) // 차량의 raw data 전송

# GetStatus() 와 sendStatus() 는 Car 구조체 멤버변수를 활용한다.

* sigInfo recvSigInfo() // 신호 정보 수신

\* 신호 정보 구조체(sigInfo) 수신 및 반환

* void getRoadInfo(sigInfo \*) // 전체 교통 정보 확인
* Car-client attribute
* struct sockaddr\_in ctrIP // 현재 교차로의 신호등 컨트롤러의 IP 주소
* carData \*Car // 차량 정보 구조체 포인터 변수
* sigInfo \*Sig // 신호 정보 구조체 포인터 변수
* int sock // 차량 고유 소켓 변수

2. Controller

2-1. 역할

# 인접 차량들의 carData를 수신하고, 수신한 데이터를 서버로 전송한다.

# 서버로부터 신호요구치를 수신한다. // 신호요구치는 wxxyzz로 이루어져있다.

w는 0~2번으로 구성되고, 응급상황의 정도를 의미한다.

(0 : 평시, 1: 혼잡상황, 공사 등, 2: 자연재해 또는 응급차량 발생시)

xx 는 01~12번으로 구성되고, 숫자는 교차로의 중심을 기준으로한 신호등의 위치를 의미한다.

y 는 1~2번으로 구성되고, 숫자는 신호등의 수신호를 의미한다 (1: 직진, 2: 좌회전)

z 는 0(초) 부터 99초까지 정의 되어 있으며, 신호를 유지시켜줄 시간을 의미한다.

# 기본적으로 신호요구치가 들어온 순서대로 신호등(led)을 제어하되, 응급상황 정도가 높은 순으로 우선순위를 가진다.

# 인접 차량에게 현재 신호의 남은시간과 다음 신호의 유지시간을 전송한다.

2-2. 흐름

sendCarData() 를 통하여 차량들의 정보를 서버로 전송한 후, 서버에서 자체 알고리즘을 통하여 도출된 신호요구치를 recvSigValue() 에서 수신한다. 이렇게 도출된 값은 응급상황 정도가 높은 순을 우선으로하여 차례대로 신호큐(Queue)에 저장되며, controlSig() 에서 신호큐(Queue)에 저장된 신호값을 토대로 신호등을 제어한다.

별도로 sendSigInfo() 함수를 통하여 서버에서 받은 다음 신호에 대한 값을 자체적으로 정제하여 차량들에게 전송한다.

2-3. 모듈 설계

* sigInfo 구조체 멤버 (typedef struct sigInfo)
* int cur\_sig // 현재 신호
* int cur\_sig\_time // 현재 신호의 남은 시간
* int next\_sig // 다음 신호
* int next\_sig\_time // 다음 신호의 유지 시간
* sigQueue 구조체 멤버 (typedef struct sigQueue)
* int count // 큐 개수
* sig front // 앞 데이터 신호(노드) 주소
* signal 구조체 멤버 (typedef struct signal)
* int sig\_val // 신호 요구치
* struct signal \*next // 다음 신호(노드) 주소
* Controller function
* createSigQueue() // 신호 큐를 생성한다.
* void sendCarData(Car \*, sockaddr\_in) // 차량들의 정보를 서버로 전송한다.
* int recvSigValue(int, sigQueue \*) // 서버로부터 신호값을 받는다. 신호 값 반환
* void controlSig(int, sigQueue \*) // 신호값에 따라 신호등(led)을 제어한다.
* void sendSigInfo(int, sigInfo \*) // 신호정보를 차량들에게 전송한다.
* Controller attribute
* struct sockaddr\_in servIP // 서버의 IP 주소
* struct sockaddr\_in ClntAddr[128] // 현재 컨트롤러와 연결되어있는 차량들의 ip 주소 (최대 128대)
* typedef signal \* sig; // 신호 구조체의 포인터 변수

3. Web Server

4. DB Server