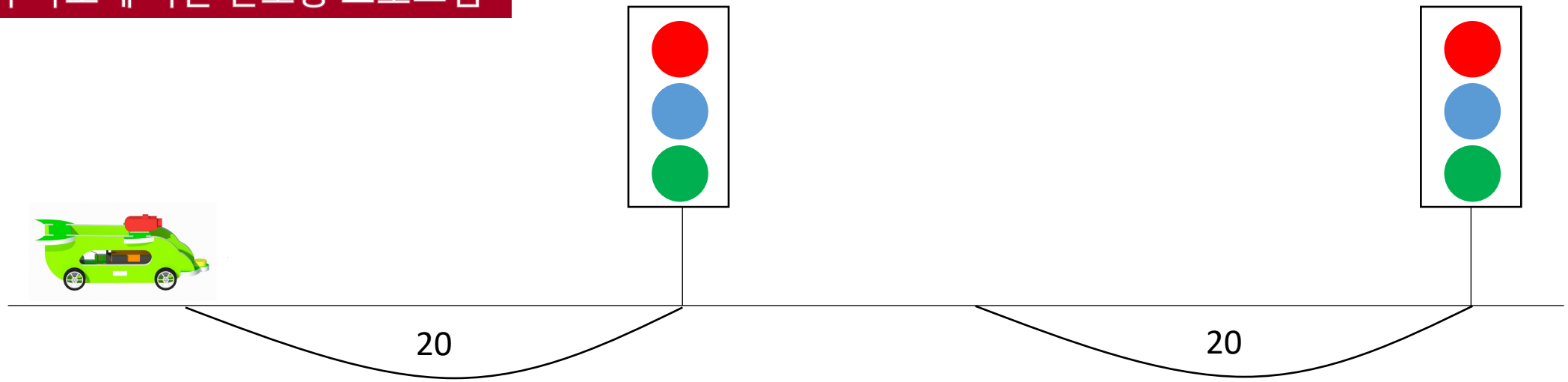
IOT 프로그래밍

Multi Processing을 이용한 센서 제어 프로그램

오승은 201758074

시나리오

차 속도에 따른 신호등 프로그램



- 신호등의 주황색은 파란색으로 대체함
- 거리 센서는 다가오는 차의 거리를 측정 -> 현재 차가 신호등을 지나가면 다음 오는 차의 거리에 의해 신호가 바뀜
- 신호등의 거리 센서는 차 하나만 지나간다고 가정하고 측정
- 거리가 20cm보다 멀리 있으면 빨강 불, 20cm이면 파란 불, 20cm보다 가까우면 초록 불
- 1번째 신호등에 측정되는 차의 속도가 3cm/s이상이면 경고 버저를 울림
- 1번째 신호등에 측정되는 차의 20초 동안의 평균 속도를 계산 -> 2번째 신호등에서의 차의 거리와 앞에서 구한 차의 평균 속도를 사용하여 2번째 신호등이 초록불이 켜져야 되는 시간을 예측하여 계산 (초록불이 켜져야 되는 시간 = 현재 시간 + (현재 차의 거리 – 20) / 평균 속도)

시스템 구성도 (서버 <-> 클라이언트1)

Server - 부모

- 20초 평균 속도 구하기
- 거리 별 명령어 확인
- 클라이언트에 명령어 지시

Server – 자식1

- 클라이언트 1 송수신 전용

차 속도에 따른 신호등 프로그램

클라이언트 1 접속

차의 거리를 서버(자식)에 전달

서버(자식)이 서버(부모)에게 거리 전달

거리에 따른 명령어를 서버(자식)에게 전달

서버(자식)은 자신에게 연결된 클라이언트에게 명령어 전달

클라이언트는 받은 명령어 수행

Client - 부모

- 센서를 통해 거리 값 받기
- 발신 전용

Client - 자식

- 서버로 온 명령어 수행
- 수신 전용

시스템 구성도 (서버 <-> 클라이언트2)

Server - 부모

- 20초 평균 속도 구하기
- 거리별 명령어 확인
- 클라이언트에 명령어 지시

Server – 자식2

- 클라이언트 2 송수신 전용

차 속도에 따른 신호등 프로그램

클라이언트 2 접속

서버(자식)으로부터 클라이언트 1의 20초간 평균 속도를 수신

차의 거리 확인

평균속도에 따른 시간을 계산하여 그 시간이 지난 후에 초록불을 킴

평균속도에 따른 거리 20을 지나가는 시간이 지난 후에 빨강불을 킴

Client - 부모

- 센서를 통해 거리값 받기
- 발신 전용

Client - 자식

- 평균 속도를 통해 초록불과 빨강불이 켜질 시간 확인 후 수행
- 수신 전용

Server - 부모

```
if (fork_num == 2)
{
    while (1)
    {
        read(fd1[0], buffer, sizeof(buffer));
        sp = atof(buffer);
        write(fd2[1], buffer, sizeof(buffer));
        read(fd2[0], buffer, sizeof(buffer));
        red_order(buffer, sp);
        puts(buffer);
        write(fd2[1], buffer, sizeof(buffer));
    }
}
```

void red_order(char *buffer, double sp) 함수

```
void red_order(char *buffer, double sp)
{
    int time = 0;
    double dis = atof(buffer);
    time = (dis - 100) / sp;
    sprintf(buffer, "%f", sp);
    puts(buffer);
}
```

- 20초간의 평균 속도를 구함

Server – 자식1 (클라이언트1)

```
if (fork_num == 1)
{
    str_len = read(y, buffer, bufsiz);
    printf("receive message1: %s₩n", buffer);
    if (i <= 20)
    {
        dis[i] = atoi(buffer);
        i++;
    }
    if (i == 20)
    {
        sp = speed(dis);
        sprintf(message, "%f", sp);
        write(fd1[1], message, sizeof(message));
        strcpy(message, "₩0");
    }
    if (atoi(buffer) == 0)
    {
        i = 0;
    }
    order(buffer);
    write(y, buffer, sizeof(buffer));
}
```

20초가 지나지 않았을 때 (빨강 박스)

- 거리 값을 배열에 저장함
- 거리에 대한 명령을 클라이언트1에게 전달

20초가 지났을 때 (파랑 박스)

- 서버(부모)에게 20초 동안의 거리 값을 전달
- 거리에 대한 명령을 클라이언트1에게 전달

Server – 자식2 (클라이언트2)

```
else
{
    read(fd2[0], buffer, sizeof(buffer));
    write(y, buffer, sizeof(buffer));
    str_len = read(y, buffer, bufsiz);
    printf("receive message2: %s₩n", buffer);
    write(fd2[1], buffer, sizeof(buffer));
    read(fd2[0], buffer, sizeof(buffer));
    usleep(10);
    write(y, buffer, sizeof(buffer));
}
```

20초가 지났을 때

- 서버(부모)가 계산한 평균 속도를 클라이언트2에게 전달

Client1 - 부모

```
while(1)
{
    digitalWrite(trigPin, LOW);
    usleep(2);
    digitalWrite(trigPin,HIGH);
    usleep(20);
    digitalWrite(trigPin,LOW);

    while (digitalRead(echoPin) == LOW);
    startTime = micros();
    while (digitalRead(echoPin) == HIGH);
    travelTime = micros()-startTime;

    distance = travelTime * 17 / 1000;
    //printf("Distance: %d cm₩n", distance);

    sprintf(message, "%d", distance);

    if (write(x, message, strlen(message) + 1) == -1)
    {
        perror("client: write");
        break;
    }
    sleep(1);
    strcpy(message, "₩0");
}
```

- 센서를 통해 거리를 측정한다
- 거리를 서버(자식)에게 전송한다

Client1 - 자식

```
else if (pid == 0)
{
    while (1)
    {
        int str_len = read(x, message, sizeof(message));
        if (str_len == -1)
        {
            perror("parent client: read");
            break;
        }
        if (str_len == 0)
            break;
        puts(message);
        light(message);
        strcpy(message, "₩0");
    }
    close(x);
    return 0;
}
```

- 서버(자식)으로부터 온 명령어를 수신한다
- light() 함수를 통해 명령어를 실행한다

Client2 - 부모

```
while (1)
{
    digitalWrite(trigPin, LOW);
    usleep(2);
    digitalWrite(trigPin,HIGH);
    usleep(20);
    digitalWrite(trigPin,LOW);

    while (digitalRead(echoPin) == LOW);
    startTime = micros();
    while (digitalRead(echoPin) == HIGH);
    travelTime = micros()-startTime;

    distance = travelTime * 17 / 1000;
    printf("Distance: %d cm₩n", distance);

    sprintf(message, "%d", distance);
    write(fd1[1], message, sizeof(message));

    if (write(x, message, strlen(message) + 1) == -1)
    {
        perror("client: write");
        break;
    }
}
```

- 센서를 통해 거리를 측정한다
- 거리를 클라이언트(자식)에게 전달한다

Client2 - 자식

```
while (1)
{
    int str_len = read(x, message, sizeof(message));
    if (str_len == -1)
    {
        perror("parent client: read");
        break;
    }
    if (str_len == 0)
        break;
    puts(message);
    time = atof(message);
    read(fd1[0], message, sizeof(message));
    distance = atoi(message);
    time = (distance - 20) / time;
    sleep(time);
    digitalWrite(Red, 0);
    digitalWrite(Blue, 1);
    sleep(1);
    digitalWrite(Blue, 0);
    digitalWrite(Green, 1);
    strcpy(message, "₩0");
    time = 20 / time;
    sleep(time);
    digitalWrite(Green, 0);
    digitalWrite(Red, 1);
}
```

- 서버(자식)으로부터 온 속도와 클라이언트(부모)로부터 온 거리를 통해 초록불과 빨강불의 시간을 계산한다
- 시간에 맞춰 신호등을 수행한다