SUMMARY REPORT

<SAFE ROAD>

캡스톤 디자인 (다)반 1,2조	
20160433	김민정
20160458	김지우
20160459	김초원
20160521	유영미
20160589	조윤지

INDEX

개요

요새 운전자 없이 스스로 운전을 하는 자율주행 자동차가 각광을 받고 있다. 자율주행 자동차들에게 있어서 앞에 장애물이 존재하는지를 판단하는 능력뿐만 아니라 도로상황에 대한 정보를 빠르게 받아들이는 능력 또한 매우 중요하다. 신호등에서 가장 이상적인 방법은 신호등과 자동차가 서로 상호작용하여 지나다니는 차가 없을 때에는 사람이 바로 횡단보도를 건널 수 있게 신호를 바뀌고, 차가 많이 지나다닐 때에는 일반적인 신호등의 신호체계를 이용하여 신호가 바뀌는 것이다. 그러나 신호등의 알고리즘을 좀 더 효율적으로 바꿔서 사람과 차량이 대기하는 시간을 줄이는 방법은 쉽지 않다. 새로운 방법에 어떠한 결함이 잠재되어 있는 지를 알지 못하기 때문에 안정성을 최우선으로 두고 오랫동안 사용해 온 검증된 방법 위주로 구현되어야 하기 때문이다. 따라서 신호등의 신호체계를 바꾸기 전에 차량으로 중점을 두고 교통신호를 제어하는 방법을 이용하여 변화를 시작해보는 것이 좋겠다는 생각으로 구현을 시도해보았다. 우리 팀이 이번에 구현해 본 제품은 간단하게 미래의 도로를 구현해 본 것이다. 먼저 신호등의 신호를 클라우드를 이용하여 차량에게 송신한다. 차량은 횡단보도를 감지한 후 직면하게 될 신호등의 신호와 현재 횡단보도에 보행자가 존재하는지에 대한 정보를 클라우드에서 수신한다. 이 정보를 이용하여 빨간 불일지라도 차량이 계속 주행할지 아니면 멈추어 기다려야 할 지를 판단한다.

일정

일	월	화	수	목	금	토
	11 **주제선정**	12	13	14	15	16
17	18	19	20	21 **회로 설계**	22	23
24	25	26	27	28 **회로 설계**	29	30
1 **시험기간**	2	3	4	5	6	7
8	9	10	11 **자동차 및 도로 회로 구현**	12	13 **외관 만들기 및 소프트웨어 구현**	14
15	16 **기능 검토**	17 **보고서 및 PPT 작성**	18	19 **발표**		

역할

	이름	역할
도로	김민정	도로 신호등이 움직이는 부분과 온습도 모듈의 OLED 출력 부분 Software 설계 및 구현을 담당함. 또한 도로 외관을 꾸밈.
	김지우	도로 Hardware 에서 신호등 LED 와 네오픽셀 LED 회로 설계 및 구현을 담당함. 또한 Cloud software 구현으로 자동차와 신호등간의 통신이 가능하도록 함. 또한 도로 외관을 만듦. 도로 부분 및 전체적인 보고서 작성. PPT 작성.
	조윤지	도로 Hardware 에서 회로 설계 및 구현을 담당함. 온습도 모듈을 이용하여 OLED 에 출력되도록 하고, 초음파 센서를 이용하여 횡단보도에 사람이 있는지 판단하는 값을 내도록 함. 또한 도로 외관을 만듦.
자동차	김초원	자동차 Software 부분을 담당함. Cloud (ThingSpeak)를 통하여 자동차와 신호등간의 통신이 이루어지도록 software 를 설계하고 구현함. 또한 도로 외관을 만듦. 자동차 부분 보고서 작성.
	유영미	자동차 Hardware 부분을 담당함. 모터 드라이버를 이용하였고, 조도 센서 및 LED 를 이용하여 자동차를 움직이고 멈추는 것을 컨트롤 할 수 있도록 회로를 설계하고 구현함. 자동차 부분 보고서 작성.

기능 설명

일정한 시간 단위로 보행자 신호등과 자동차 신호등이 동시에 바뀐다. 보행자 신호등이 초록 불일 때는 자동차 신호등은 빨간 불이, 보행자 신호등이 빨간 불일 때는 자동차 신호등은 파란 불이 된다. 이 신호등 값을 Cloud로 전송하여 자동차가 계속 주행할지 말지를 판단하는 요소가 되도록 한다.

횡단보도에 사람이 존재하는지에 대한 정보를 얻기 위해 초음파 센서를 쓴다. 초음파 센서가 횡단보도 전체를 다 감지할 수 있도록 도보의 턱에 설치하여 반대편 도보 턱까지의 거리 정보를 잰다. 횡단보도 거리보다 초음파 센서로 측정된 값을 Cloud를 통해 보내고, 자동차가 Cloud를 통해 값을 받는다. 이 값이 횡단보도 거리 값보다 작으면 사람이 있다고 자동차는 판단 할 수 있다.

자율주행 자동차는 카메라를 사용하여 정보를 얻어 횡단보도 앞 정차라인이 어디인지를 알 수 있지만, 그 방법은 지금 구현하기 어려움이 많아서 차량이 한 대 밖에 없다는 가정을 두고 조도센서를 이용하기로 했다. 횡단보도 앞 바닥에 LED를 달고 자동차의 바닥에 조도센서를 단다. 자동차는 이동을 하고 있던지 정차해 있던지 앞으로 만나게 되는 신호등의 상황에 대한 data(신호등 정보와 초음파 센서가 측정한 거리 정보)를 Cloud에서 실시간으로 받아오게 된다. 그러다 LED로 인하여 빛을 감지하면 현재의 data들을 토대로 계속해서 주행을 할지 멈춰야 할지 정하게 된다.

또한, 도로에서 온도와 습도 정보를 전광판에 띄워 주는 것을 모티브로 하여, 현재의 온도와 습도를 온습도 센서와 OLED 패널을 이용해 실시간으로 OLED에 표시해주는 기능을 구현했다.

도로 부분

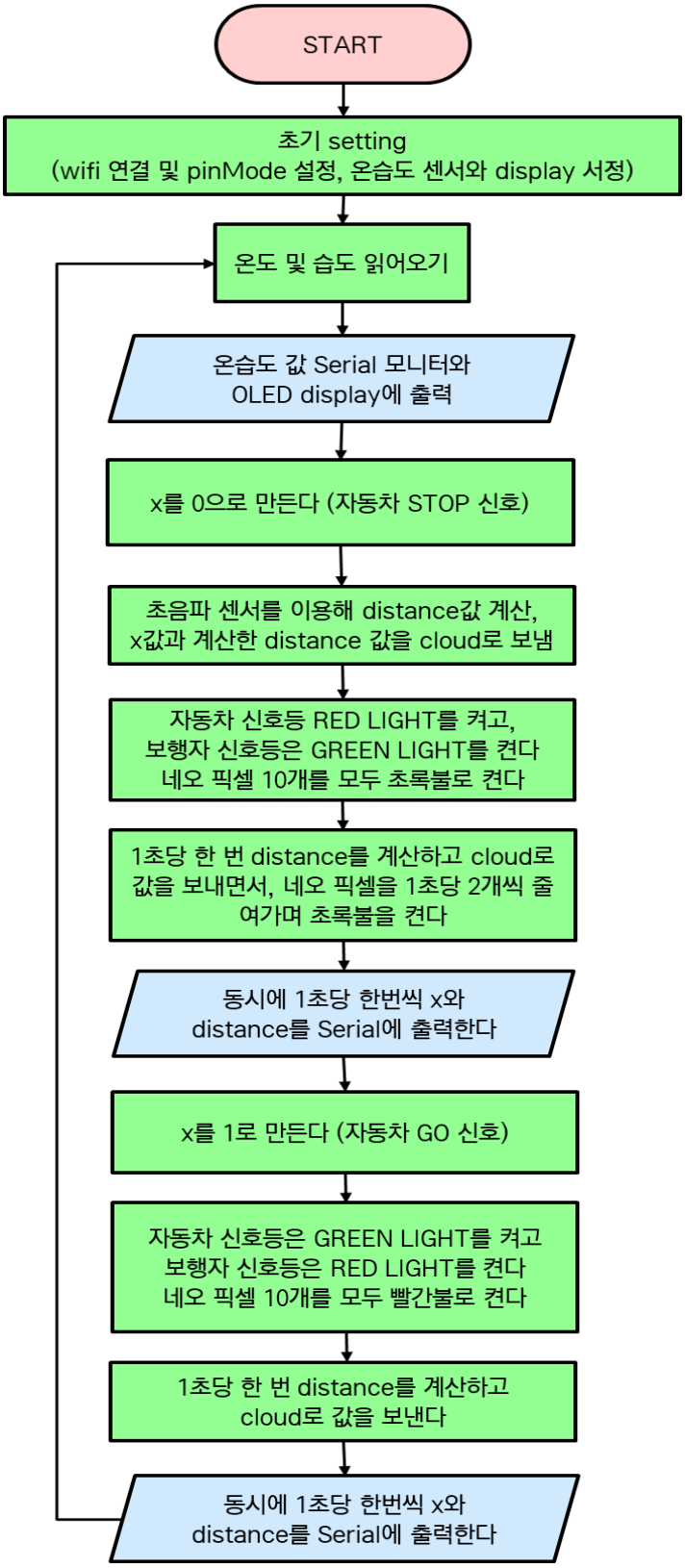
담당

김민정

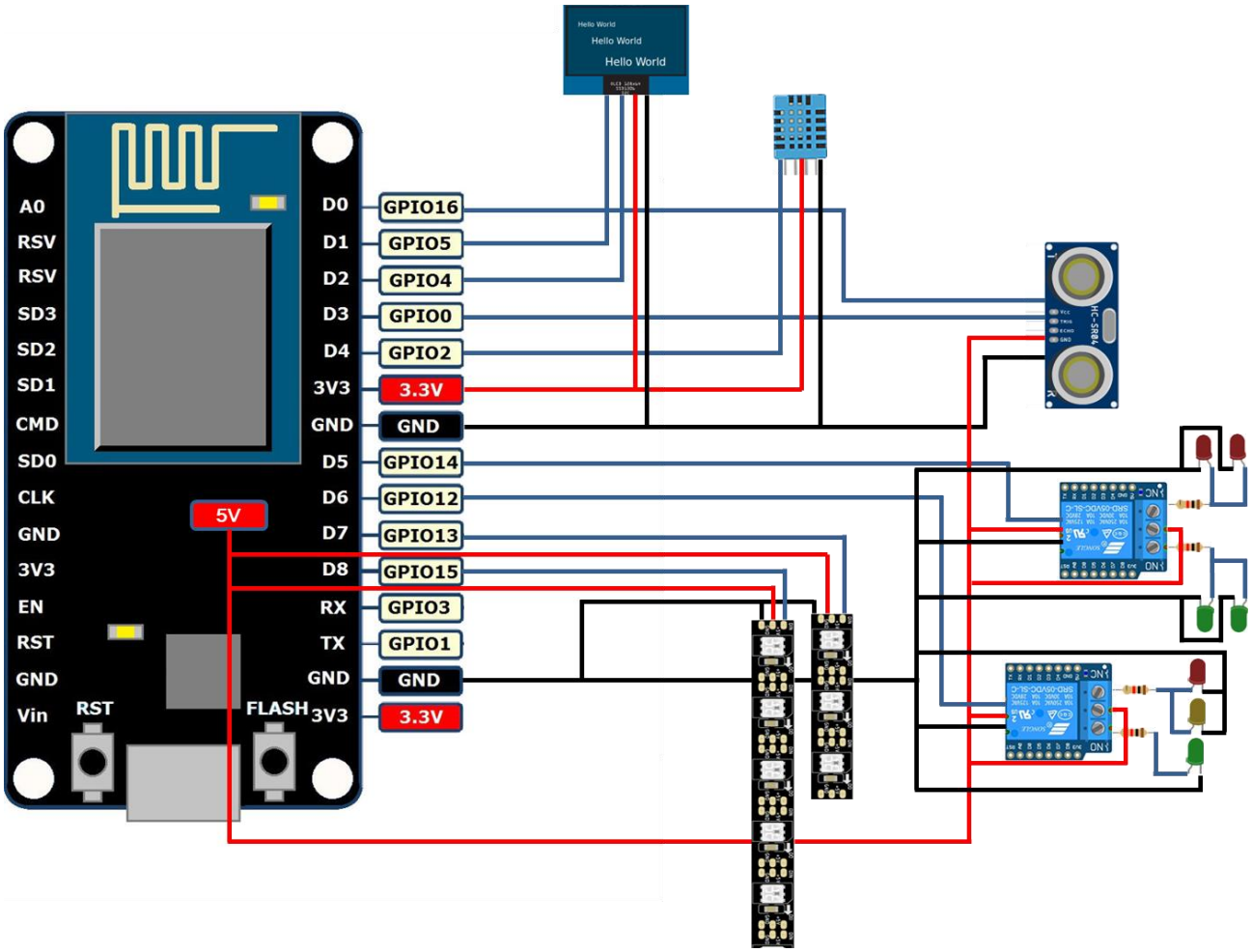
김지우

조윤지

Flow Chart


START
초기 setting
(wifi 연결 및 pinMode 설정, 온습도 센서와 display 서정)
온도 및 습도 읽어오기
온습도 값 Serial 모니터와
OLED display에 출력
x를 0으로 만든다 (자동차 STOP 신호)
초음파 센서를 이용해 distance값 계산,
x값과 계산한 distance 값을 cloud로 보냄
자동차 신호등 RED LIGHT를 켜고,
보행자 신호등은 GREEN LIGHT를 켠다
네오 픽셀 10개를 모두 초록불로 켠다
1초당 한 번 distance를 계산하고 cloud로
값을 보내면서, 네오 픽셀을 1초당 2개씩 줄
여가며 초록불을 켠다
동시에 1초당 한번씩 x와
distance를 Serial에 출력한다
x를 1로 만든다 (자동차 GO 신호)
자동차 신호등은 GREEN LIGHT를 켜고
보행자 신호등은 RED LIGHT를 켠다
네오 픽셀 10개를 모두 빨간불로 켠다
1초당 한 번 distance를 계산하고
cloud로 값을 보낸다
동시에 1초당 한번씩 x와
distance를 Serial에 출력한다

회로도



신호등 및 네오 픽셀

핀이 모자란 것을 방지하기 위해 LED 제어는 모두 릴레이 모듈을 사용했다. 첫 번째 릴레이 모듈은 보행자 신호등을 제어하기 위한 것으로, D5(GPIO 14)에 연결되어있다. 횡단보도에서 서로 마주보고 있는 두 보행자 신호등 LED는 빨강은 빨강끼리, 초록은 초록끼리 동시에 켜지므로 서로 저항과 병렬로 연결되어 있다. 두 번째 릴레이 모듈은 자동차 신호등을 제어하기 위한 것으로, D6(GPIO 12)에 연결되어 있다. 보행자 신호등이 빨간 불일 때는 자동차 신호등은 파란 불이고, 보행자 신호등이 파란 불일 때는 자동차 신호등은 빨간 불과 노란 불이 동시에 켜진다. 횡단보도를 건널 시간이 어느 정도

남았는지 알려주는 역할을 하는 네오 픽셀(10개짜리)는 D8(GPIO 15)와 연결되어있다. 릴레이 모듈 하나로 보행자와 도로 신호등을 제어할 수 있었지만, 초반 프로젝트 기획서에는 위급 상황일 때(예를 들면 구급차가 지나가는 상황) 보행자 신호등을 강제로 초록불이 되지 않게 하려고 했었기 때문에 그를 위해 릴레이 모듈을 두 개 쓰게 되었다.

초음파와 온습도, OLED display

초음파센서의 Trig를 D0(GPIO 16)에 연결하고 echo를 D3(GPIO 0)에 연결한다. Vcc는 5V에 연결하고 GND는 ground에 연결해준다. 정차 라인 앞에 설치하는 LED(NeoPixel)는 Vcc는 5V를 쓰며, GND는 ground에 연결해 준다. NeoPixel Din은 D7(GPIO 13)에 연결해 준다. 이 NeoPixel은 자동차 조도 센서가 인식해야 하기 때문에 항상 밝게 빛나야 한다. 온습도 센서는 D4(GPIO 2)에 연결해주고 3.3V와 ground에도 각각 연결해준다. OLED패널은 SCL은 D1(GPIO 5)에 SDA는 D2(GPIO 4)에 연결해준다. 3.3V와 ground에도 마찬가지로 각각 연결해준다. 5V 전원을 쓰는 모듈은 각각 초음파, 릴레이 모듈, NeoPixel이고 나머지는 3.3V 전원을 쓴다.

함수 설명

void **oled_display**(int h, int t)	온도(t)와 습도(h)를 받아 OLED display에 온습도를 출력하는 함수
long **distance_cal**()	초음파 함수에서 값을 받아와 distance를 계산하여 반환하는 함수
void **data_to_cloud**(int x, int distance)	x값과 distance값을 받아 cloud(ThingSpeak)로 write하는 함수
void **print_x_distance**(int distance, int x)	Serial 모니터에 x값과 distance값을 출력해주는 함수
void **setStripRed**()	NeoPixel 10개를 모두 빨간 불이 출력되도록 만들어주는 함수
void **setStripGreen**(int num)	NeoPixel 10개 중 num에 해당하는 개수만 초록 불이 출력되도록 만들어주는 함수

코드 설명 (data_to_cloud.ino)

```
/* header include */
#include <DHT.h>  //온습도 센서
#include <ESP8266WiFi.h> //WiFi header include
#include <Wire.h>
#include <Adafruit_GFX.h>
#include  <Adafruit_SSD1306.h>  //OLED  header include
#include  <Adafruit_NeoPixel.h>  //NeoPixel  header include
/* Cloud를 위한 WiFi 설정 */
 String apiKey = "P4GTBD9A0LD15HK0";    // Write API key from ThingSpeak
 const char *ssid =  "capstone";  // wifi ssid and wpa2 key
 const char *pass =  "87654321";
 const char* server = "api.thingspeak.com";
```

```
/* 손쉽게 알아보기 위한 Name 정의*/
 #define DHTPIN 2  //온습도센서 D4(GPIO 2)
 #define LIGHT_P 14 //보행자 신호등 GPIO 14
 #define LIGHT_C 12 //차 도로 신호등 GPIO 12
 #define TRIG 16 //초음파센서 trigger pin D0
 #define ECHO 0 //초음파센서 echo pin D3

 #define SCREEN_WIDTH 128 // OLED display width, in pixels
 #define SCREEN_HEIGHT 64 // OLED display height, in pixels

 #define NEOPIN 15 //relay NeoPixel Pin GPIO 15
 #define NUM_LEDS 10 //NeoPixel 개수는 10개이다.
 #define OLED_RESET LED_BUILTIN //OLED RESET 4
 // SCL는 GPIO 4에, SDA GPIO 5에 꽂는다.
```

```
DHT dht(DHTPIN, DHT11); //온습도 센서 함수
 WiFiClient client;         // WiFiClient로 선언
 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
 Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, NEOPIN, NEO_GRB + NEO_KHZ800);
//OLED display와 NeoPixel 함수 setting

 void setup()
 {
/* pinMode setting*/
  //초음파 pinMode setting
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  //TRAFFIC LIGHT pinMode setting
  pinMode(LIGHT_C,OUTPUT);
  pinMode(LIGHT_P,OUTPUT);
 //Serial setting
  Serial.begin(115200);
//OLED display 작동
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C ) ;
  display.clearDisplay( ) ;
  display.setTextColor(WHITE) ;
//온습도 센서 작동
  dht.begin();
//NeoPixel 작동
  strip.begin();
  strip.show();
/*WiFi 연결 코드*/
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);   // WiFi 연결
  while (WiFi.status() != WL_CONNECTED )  {
    delay(500);
    Serial.print(".");//  만약  연결이  안되면  ...을 출력한다.
   }
  Serial.println("");
Serial.println("WiFi connected");}
 // 연결이 잘 되었다면 연결 되었다고 출력한다.
```

```
void loop()  {
   /* 온습도 읽어오기 */
  float h = dht.readHumidity(); //습도를 읽어온다
  float t = dht.readTemperature(); //온도를 읽어온다
  Serial.print("온도 : ");
  Serial.print(t);
  Serial.print(", 습도 : ");
  Serial.println(h); //온습도 Serial에 출력

  oled_display(h,t); //OLED에 온습도 값 출력

  long distance = distance_cal(); //초음파 센서를 이용해 distance를 구한다
  int x=0; //자동차 STOP 보행자 GREEN
  data_to_cloud(x, distance); //x 정보와 distance 정보를 cloud로 보낸다.
  digitalWrite(LIGHT_C,HIGH); //자동차 RED LIGHT ON
  digitalWrite(LIGHT_P,LOW); //보행자 신호등 GREEN LIGHT ON
  setStripGreen(10); //NEOSTRIP GREEN LIGHT ON
  print_x_distance(distance, x); //x값과 distance 값 시리얼 모니터에 출력

/* 아래 코드는 1초마다 초음파 센서에서 distance 값을 읽어오고, 그 값을 Cloud에 보내며 동시에 시리얼에 프린트 하는 코드다. 또한 보행자 NeoPixel LED를 2개씩 줄여가며 파란 불을 출력한다. */
  distance = distance_cal();
  data_to_cloud(x, distance);
  setStripGreen(8); //8개만 GREEN LIGHT ON
  print_x_distance(distance, x);

  distance = distance_cal();
  data_to_cloud(x, distance);
  setStripGreen(6); //6개만 GREEN LIGHT ON
  print_x_distance(distance, x);
```

```
distance = distance_cal();
  data_to_cloud(x, distance);
  setStripGreen(4); //4개만 GREEN LIGHT ON
  print_x_distance(distance, x);

  distance = distance_cal();
  data_to_cloud(x, distance);
  setStripGreen(2); //2개만 GREEN LIGHT ON
  print_x_distance(distance, x);

x=1; //자동차 GO 보행자 RED
  distance = distance_cal();
  //초음파 센서를 이용해 distance를 구한다
  data_to_cloud(x, distance);
  //x 정보와 distance 정보를 cloud로 보낸다.
  digitalWrite(LIGHT_C,LOW);
  //자동차 GREEN LIGHT ON
  digitalWrite(LIGHT_P,HIGH);
  //보행자 RED RIGHT ON
  setStripRed();//NEOPixel LED 10개
  모두 RED LIGHT ON
  print_x_distance(distance,  x);  //x값과 distance 값
시리얼 모니터에 출력

/* 아래 코드는 1초마다 초음파 센서에서 distance
값을 읽어오고, 그 값을 Cloud에 보내며 동시에
시리얼에 프린트 하는 코드다. */
  distance = distance_cal();
  data_to_cloud(x, distance);
  print_x_distance(distance, x);

  distance = distance_cal();
  data_to_cloud(x, distance);
  print_x_distance(distance, x);

  distance = distance_cal();
  data_to_cloud(x, distance);
  print_x_distance(distance, x);
```

```
  distance = distance_cal();
  data_to_cloud(x, distance);
  print_x_distance(distance, x);

 }
void oled_display(int h, int t){
      display.clearDisplay( ) ;
      display.setTextSize(2) ;
      //OLED Display에 온도 출력
      display.setCursor(2,2) ;
      display.print("Temp") ;
      display.setCursor(60,2) ;
      display.print(t);
      // OLED Display에 습도 출력
      display.setCursor(2,30) ;
      display.print("Hum") ;
      display.setCursor(60,30) ;
      display.print(h);
      //OLED 보이기
      display.display( ) ;
   return;
 }
 long distance_cal() {
  long duration, distance;
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH); //TRIGGER 올리기
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  duration = pulseIn(ECHO, HIGH); // EchoPin핀에서
펄스값을 받아온다.
   distance = (duration/2)/29.1; //갔다 온 거리이기
때문에 2로 나누고
   //cm로 만들기 위해 29.1로 나누어 준다.
  return distance;
//나중에 자동차에서 distance 값을 받아 24~26이면
사람 없음으로 판단하고, 그 이하면 사람 있음으로
판단하여 자동차가 멈추게 된다.
 }
```

```
void data_to_cloud(int x, int distance){
    if (client.connect(server,80))  {    // connect to server (api.thingspeak.com) port 80
          String postStr = apiKey;
// apiKey += field1 += field2 ....
          postStr +="&field1=";
 // &field1 --> Address of field1
          postStr += String(x);
// 자동차 신호등 정보
          postStr +="&field2=";
          postStr += String(distance); //횡단보도 사람 유무정보
          postStr += "₩r₩n₩r₩n";
       // Sending Data to server
          client.print("POST /update HTTP/1.1₩n");
          client.print("Host: api.thingspeak.com₩n");
          client.print("Connection: close₩n");
          client.print("X-THINGSPEAKAPIKEY: "+apiKey+"₩n");
          client.print("Content-Type: application/x-www-form-urlencoded₩n");
          client.print("Content-Length: ");
          client.print(postStr.length());
          client.print("₩n₩n");
          client.print(postStr);
  //ThingSpeak에 Data를 전달했음을 알리기 위해 시리얼 모니터에 다음 문장 출력
          Serial.println("go to cloud");
        }

        client.stop();
        Serial.println("Waiting...");
          // thingspeak needs minimum 15 sec delay between updates

    return;
 }
```

```
void print_x_distance(int distance, int x){
   Serial.print("x : ");
   Serial.print(x);
   Serial.print(", distance : ");
   Serial.println(distance);
   delay(1000);
   return;
 }

/* strip.setPixelColor(0, 255, 0, 0);에서, 첫번째 파라메터는 LED 의 번호이고, 2,3,4번째 파라메터는 RGB 값이다. 0~255까지 지정할수 있다. setPixelColor로 설정한 값은. strip.show(); 를 실행할때, 네오픽셀에 실제로 나타난다. */

 void setStripRed() {
    int i;
    for(i=0; i<NUM_LEDS; i++){
     strip.setPixelColor(i, 255, 0, 0);
    }   //NeoPixel 설정. 0부터 Neopixel 개수(10개)만큼 모두 빨간색으로 만든다.
    strip.show();  //NeoPixel에 설정한 대로 불을 켠다.
   return;
 }
void setStripGreen(int num){
    int i;
    for(i=0; i<num; i++){
       strip.setPixelColor(i, 92, 209, 229);
          //NeoPixel 설정. 0부터 num개수 만큼 모두 초록색으로 만든다.
    }

    for(i=num; i<NUM_LEDS; i++){
       strip.setPixelColor(i, 0, 0, 0);
 //남은 NeoPixel들은 모두 OFF한다.
    }
    strip.show();//NeoPixel에 설정한 대로 불을 켠다.
    return;
 }
```

자동차 부분

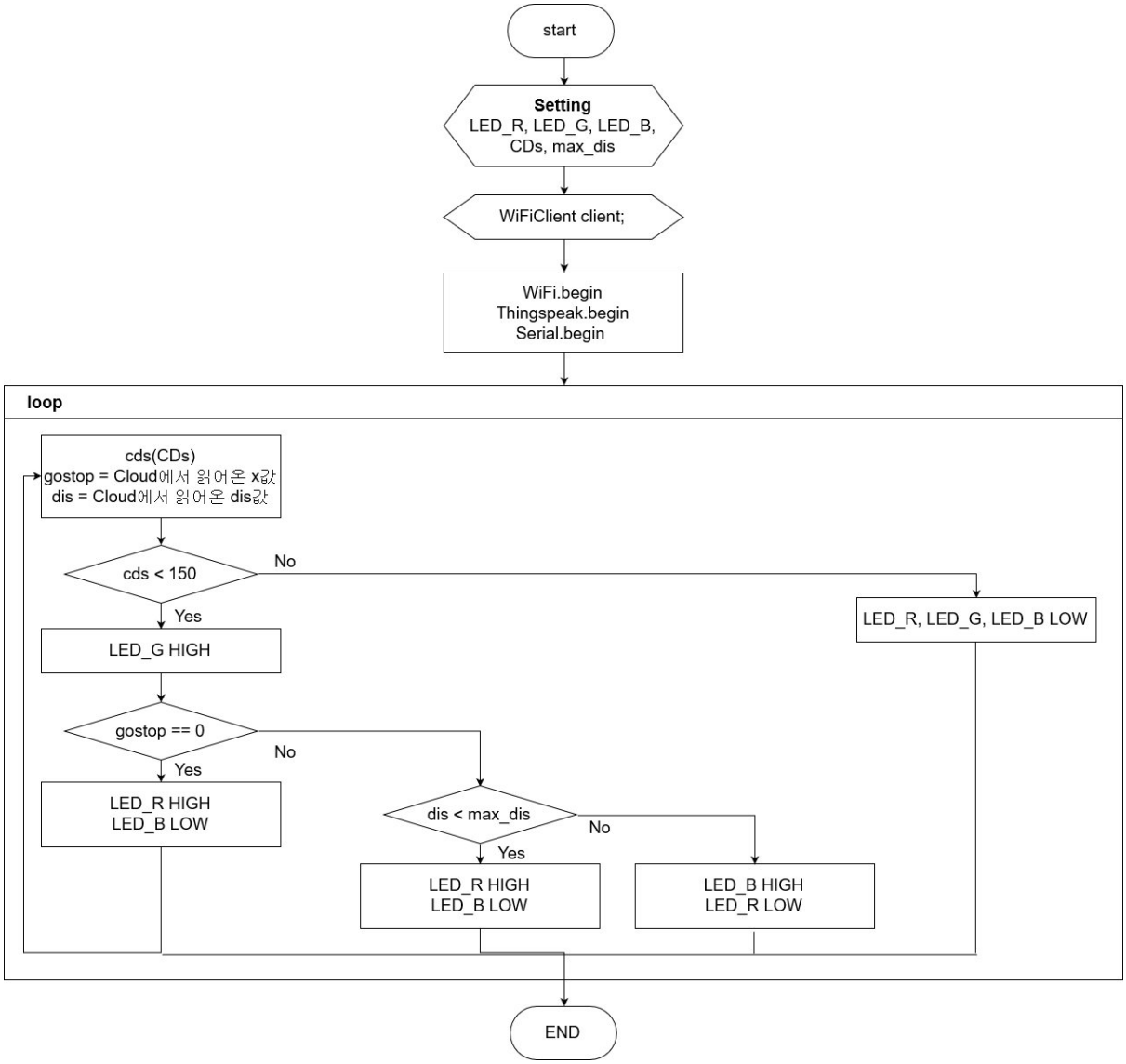
담당

김초원

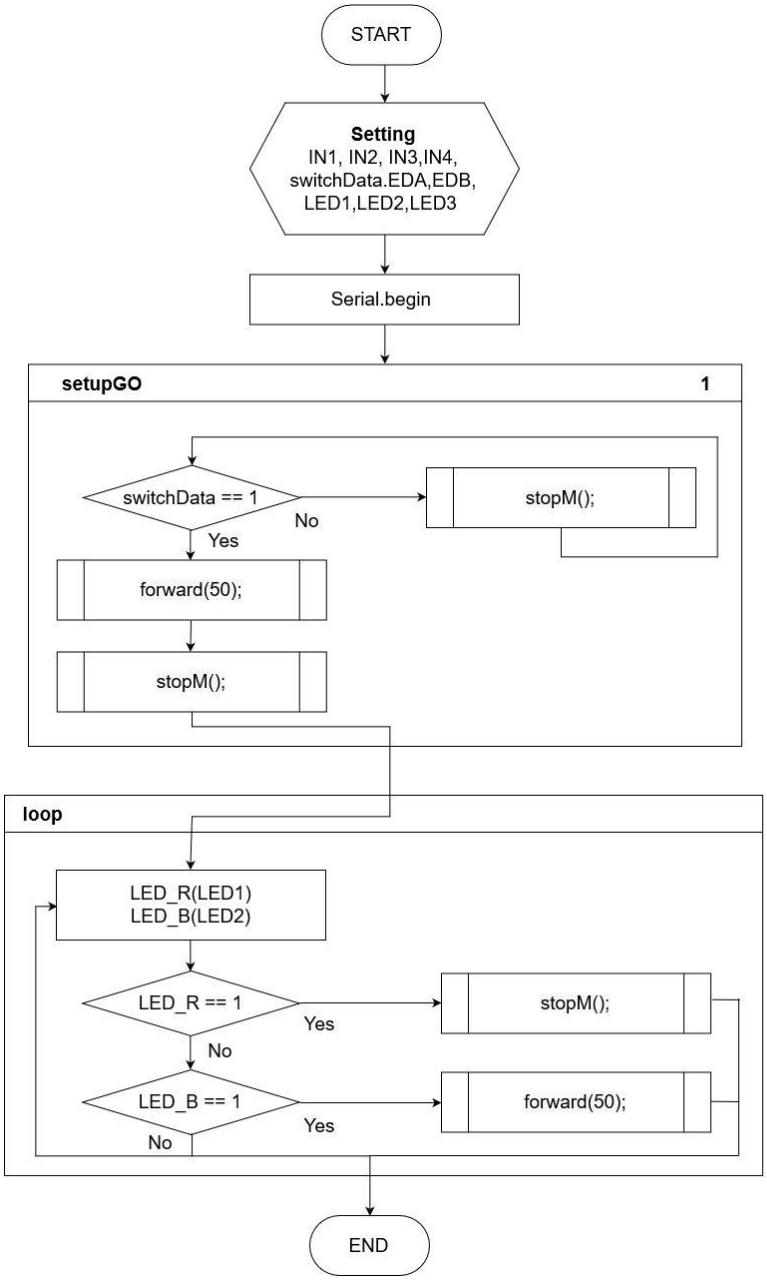
유영미

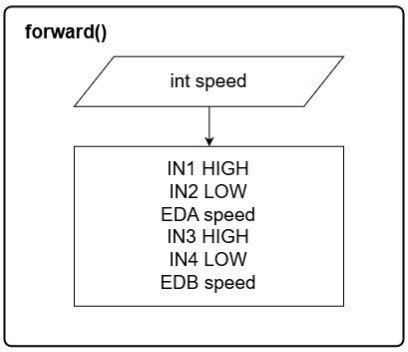
Flow Chart

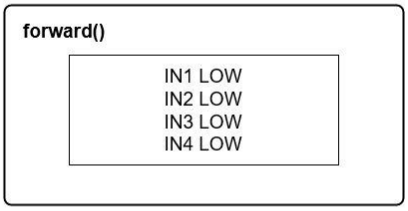
1) Node MCU (Main board)



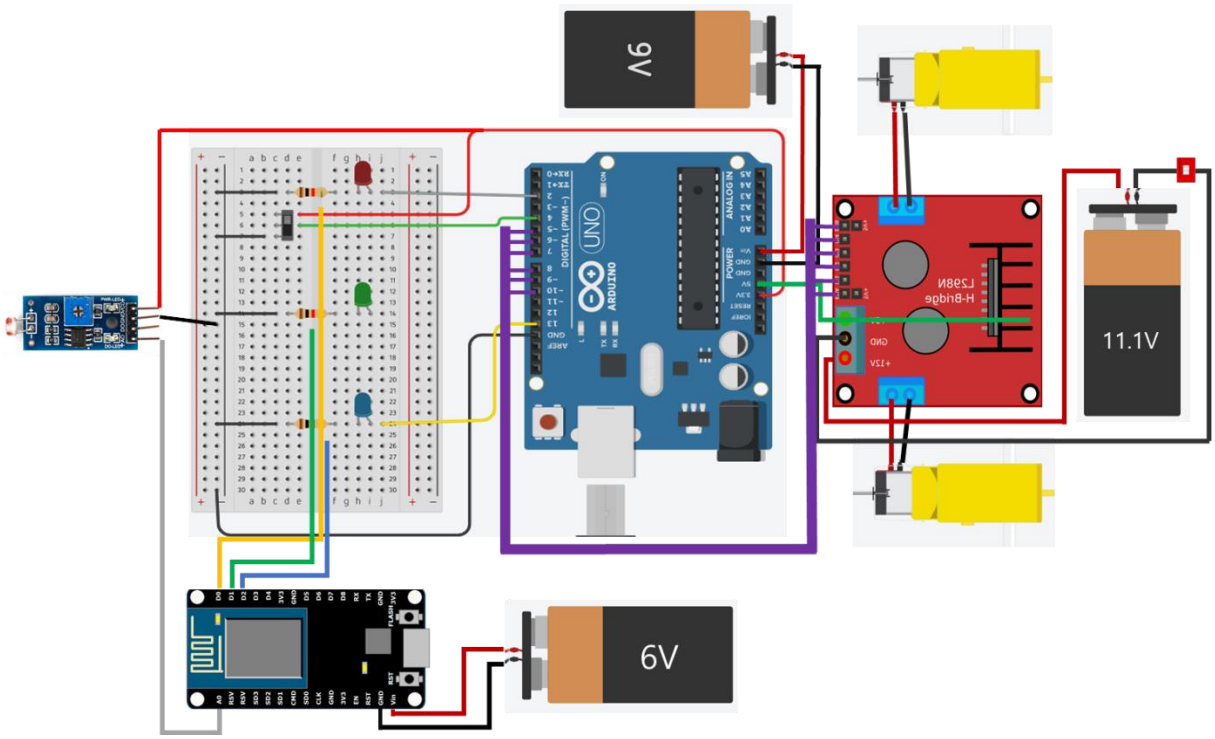
2) Aduino Uno (Sub board)


START
Setting
IN1, IN2, IN3,IN4,
switchData.EDA,EDB,
LED1,LED2,LED3
Serial.begin
setupGO
1
switchData == 1
No
Yes
stopM();
forward(50);
stopM();
loop
LED_R(LED1)
LED_B(LED2)
LED_R == 1
Yes
No
stopM();
LED_B == 1
Yes
No
forward(50);
END


forward()
int speed
IN1 HIGH
IN2 LOW
EDA speed
IN3 HIGH
IN4 LOW
EDB speed


forward()
IN1 LOW
IN2 LOW
IN3 LOW
IN4 LOW

회로도



Node mcu 는 클라우드에 올라간 두개의 data 를 받아오고 그에 따라 LED RED 와 BLUE 의 on off 를 조절한다. 또한, 조도센서의 값을 측정하고 그 값에 따라 LED GREEN 의 on off 를 조절한다. 즉, Node mcu 의 기능은 다음과 같다.

1. 클라우드에 올라간 두 신호 (x, distance)를 받아온다.
2. 조도센서의 값을 받아온다.
3. 1 번과 2 번의 신호를 통해 3 개의 led 의 on off 를 조절한다.

아두이노는 두 개의 RED 와 GREEN LED 의 on off 상태를 확인하여 그 값에 따라 차를 움직일지 말지를 정한다. RED LED 가 on 이 되면 차가 멈추고 BLUE LED 가 on 이 되면 차가 달리게 된다. 따라서 조도센서가 없는 구간에서는 차는 이동하거나 멈추거나 기존의 동작을 유지하게 되지만, 실제로는 nodemcu 를 통해 앞으로 만나게 될 신호등의 상황에 대한 정보를 계속해서 update 를 하고 있다. 그 값은 조도센서가 있는 특정구간에서만 사용하게 된다. 그 구간은 신호등의 상황에 따라 최소한 멈춰야 하는 커트라인인 셈이다. 차를 노트북에 연결한 채로 달릴 수 없기 때문에 외부전원을 사용하였다. 모터 드라이버용 11.1V 짜리 외부전원, 아두이노와 nodemcu 용 각각 9V, 6V 전원이 사용되었다.

함수 설명

1) Node MCU (Main board)

void **loop()**	Cloud에서 읽어온 x값을 변수 gostop에, distance값을 변수 dis로 저장하고, 조도 센서의 값이 150 이하이면 green led를 키며 세 값에 따라 LED 3개를 제어하는 함수

2) Arduino Uno (Sub board)

void **forward(int speed)**	int speed의 속도로 모터 2개를 제어해서 전진하는 함수
void **stopM()**	모터 2개를 멈추는 함수
void **setupGO()**	변수 SwitchData값이 1이 되면 (Switch on) 일정한 시간만큼 모터가 동작하고 멈추는 함수
void **loop()**	LED의 값을 읽어와서 blue led가 켜지면 차를 전진시키고 red led가 켜지면 차를 정지시키는 함수

코드 설명

1) Node MCU (Main board)

```
/* header include */
#include "ThingSpeak.h"   //ThingSpeak 를 사용하기 위한 header
#include <ESP8266WiFi.h>   //WiFi header include

/* 손쉽게 알아보기 위한 Name 정의*/
#define LED_R 16   //자동차 Red LED pin D0
#define LED_G 5   //자동차 Green LED pin D1
#define LED_B 4   //자동차 Blue LED pin D2
#define CDs A0   //조도센서 pin A0
#define max_dis 24   //사람이 있을 때의 최대 거리
```

```
/* 계속 loop하는 함수 */
void loop() {
    int cds=analogRead(CDs);
    //조도센서의 값을 읽어와서 cds 에 저장한다.
    //이때 조도센서는 analog 로 값을 전달하기 때문에 analog 신호를 Read 한다.
    int gostop = ThingSpeak.readIntField(myChannelNumber, 1) ;
    int dis = ThingSpeak.readIntField(myChannelNumber, 2) ;
```

```
/* Cloud를 위한 WiFi 설정 */
char ssid[] = "kcw" ;  // your network SSID (name)
char pass[] = "87654321" ;  // your network password
unsigned long myChannelNumber = 935565;  // CH ID
const char *myreadAPIKey = "P7PBXKK0V3TNXHHY" ; //read API Key
WiFiClient client;  // WiFiClient로 선언

/* 기본 Setting */
void setup() {
  WiFi.begin (ssid, pass) ;  //Initialize WiFi
  Serial.begin(115200);       // Initialize serial
   ThingSpeak.begin(client);   // Initialize ThingSpeak
  //자동차 LED(Red, Green, Blue) pin setting
  pinMode(LED_R,OUTPUT);
  pinMode(LED_G,OUTPUT);
  pinMode(LED_B,OUTPUT);
}
```

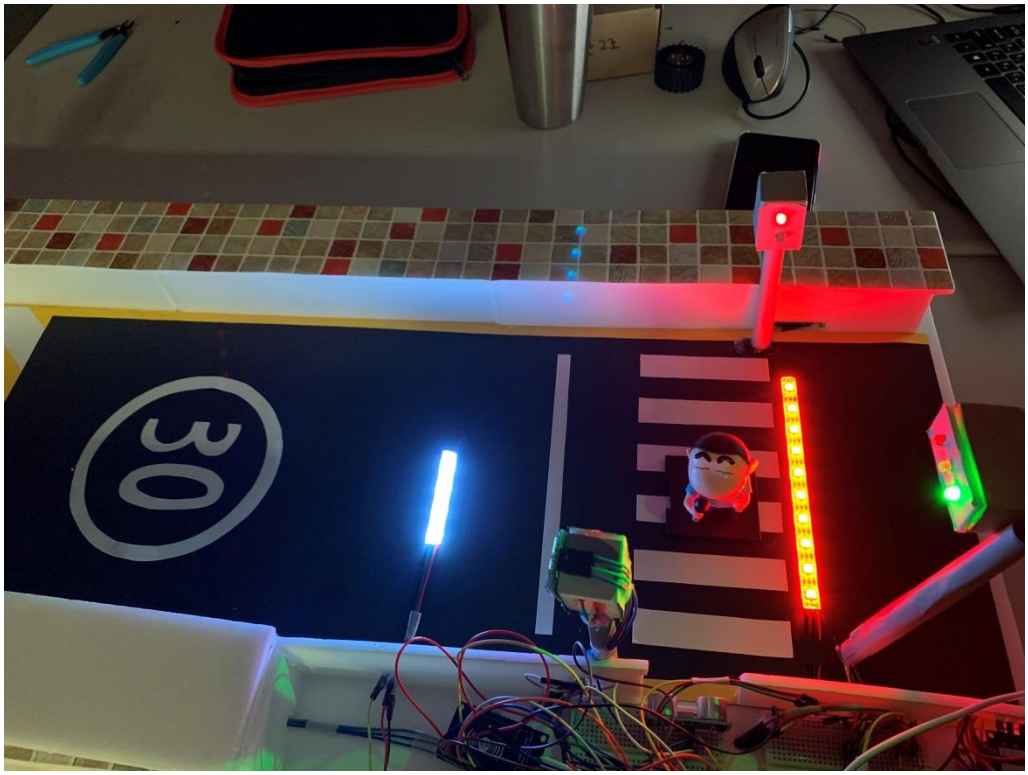
```
    if(cds<150){  //도로위에 있는 cds 값을 읽어옴
     digitalWrite(LED_G,HIGH);  //Green LED ON
     if(gostop == 0) {   //자동차신호가 빨간불이면
      digitalWrite(LED_R, HIGH);  //Red LED ON
      digitalWrite(LED_B, LOW);  //Blue LED OFF
      }
     else
     {
        if(dis<max_dis)
        //Cloud 를 통해 들어온 dis 값이 특정거리보다 작으면(특정거리 안에 사람이 있으면)
        {
          digitalWrite(LED_R, HIGH);  //Red LED ON
          digitalWrite(LED_B, LOW);  //Blue LED OFF
        }
        else
        {
          digitalWrite(LED_B, HIGH);  //Blue LED ON
          digitalWrite(LED_R, LOW);  //Red LED ON
        }
     }
   }
   else  //조도센서에 값을 받지 못할 때
   {
    digitalWrite(LED_R, LOW);  //Red LED OFF
    digitalWrite(LED_G, LOW);  //Green LED OFF
    digitalWrite(LED_B, LOW);  //Blue LED OFF
   }
  delay(1000);
}
```

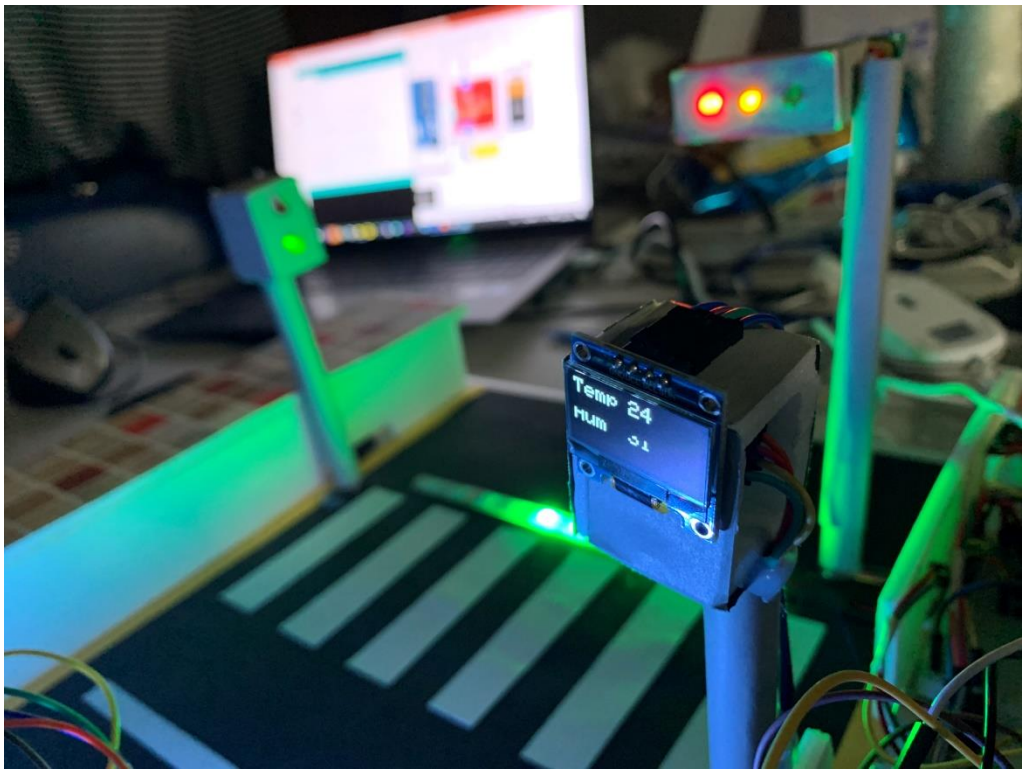
2) Arduino (Sub board)

```
/* 손쉽게 알아보기 위한 Name 정의*/
#define IN1 9 //모터 A 의 1 제어
#define IN2 8//모터 A의 2 제어
#define IN3 7//모터 B의 1 제어
#define IN4 6//모터 B의 2 제어
#define switchData 4 //스위치상태 Readpin
#define EDA 10 //모터 A의 speed제어 pin
#define EDB 5 //모터 B의 speed제어 pin
#define LED1 2 //빨강 LED
#define LED2 13 //파랑 LED
#define LED3 12 //초록 LED
/* 자동차 GO함수*/
void forward(int speed) {
//전진 : 모터 두 개를 모두 정회전 시킴
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(EDA,speed);//모터 A 속도 조절
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
        analogWrite(EDB,speed);//모터 B 속도 조절
}
/* 자동차 stop 함수*/
void stopM() { // 정지 : 2개의 모터 모두 회전 멈춤
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, LOW);}
/* 기본 Setting */
void setup() {
  pinMode(IN1, OUTPUT); // A 모터 1
  pinMode(IN2, OUTPUT); // A 모터 2
  pinMode(IN3, OUTPUT); // B 모터 1
  pinMode(IN4, OUTPUT);   // B 모터 2
  pinMode(EDA, OUTPUT);
  pinMode(EDB, OUTPUT);
  pinMode(switchData, INPUT_PULLUP);
  // 스위치
  Serial.begin(115200);
  setupGO(); //setGO 함수 실행
  delay(2000);}
```

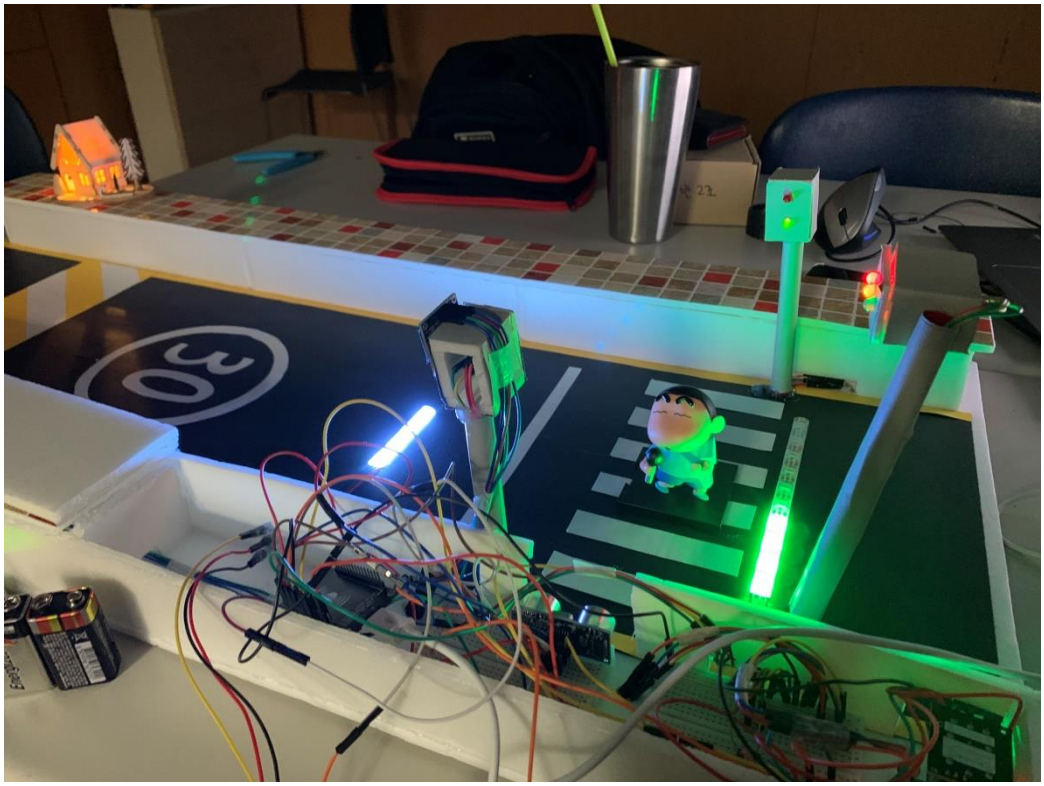
```
/* 손쉽게 알아보기 위한 Name 정의*/
void setupGO(){
//스위치data가 1이 될 때까지 무한 대기
//값이 1이 들어오면 2초간 자동차 이동 후 멈춤
//시뮬레이션 용 함수
  while(1){
    if(digitalRead(switchData))
    { forward(50);
      delay(2000);
      stopM();
      break;
    }
    else {
       stopM(); //멈춘 상태로 대기
      }
  }
}
/* 계속 loop하는 함수 */
void loop() {
 //R LED와 B LED의 상태를 확인하여 멈추거나 이동하는 함수
//LED의 상태 확인
 int LED_R = digitalRead(LED1);
 int LED_B = digitalRead(LED2);

// 이동 결정
 if(LED_R == 1) stopM();//RED ON이면 멈춤
 else if(LED_B == 1) forward(50); //BLUE ON이면 이동
 else{}
}
```

결과

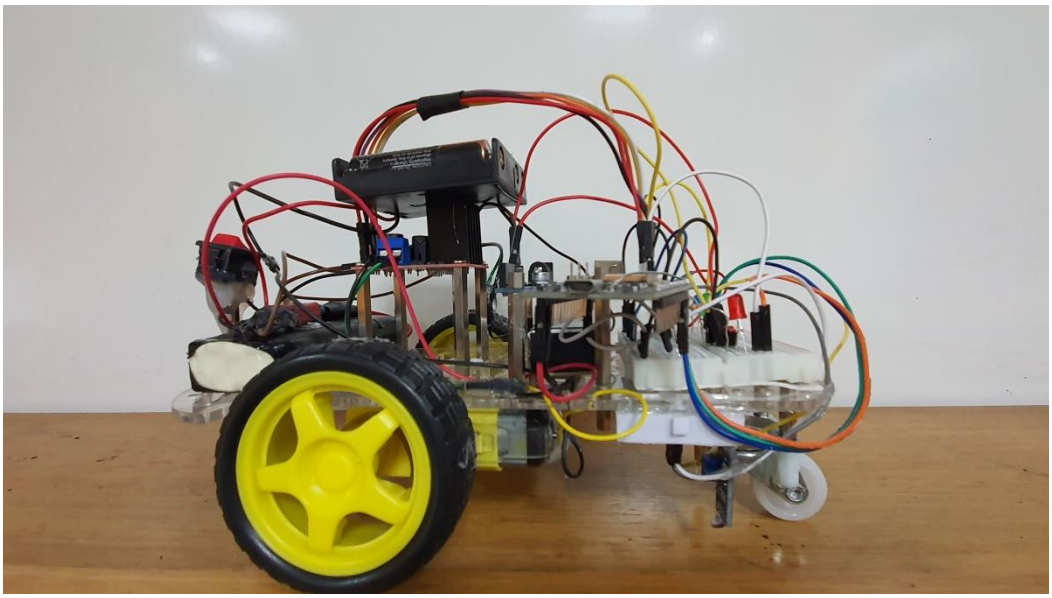




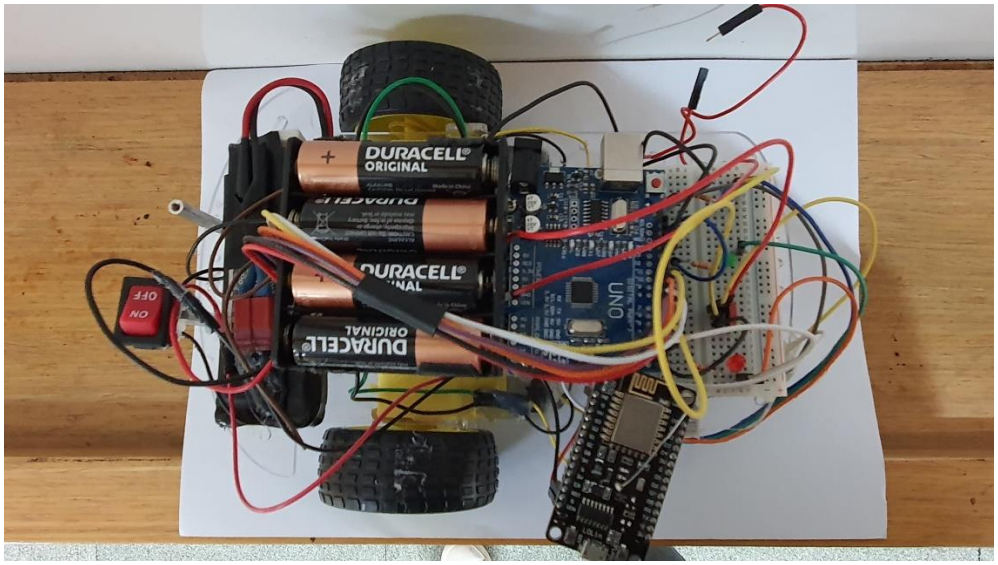
설계대로 완성된 도로, 신호등과 OLED Display에 실시간으로 표시되는 온도와 습도.

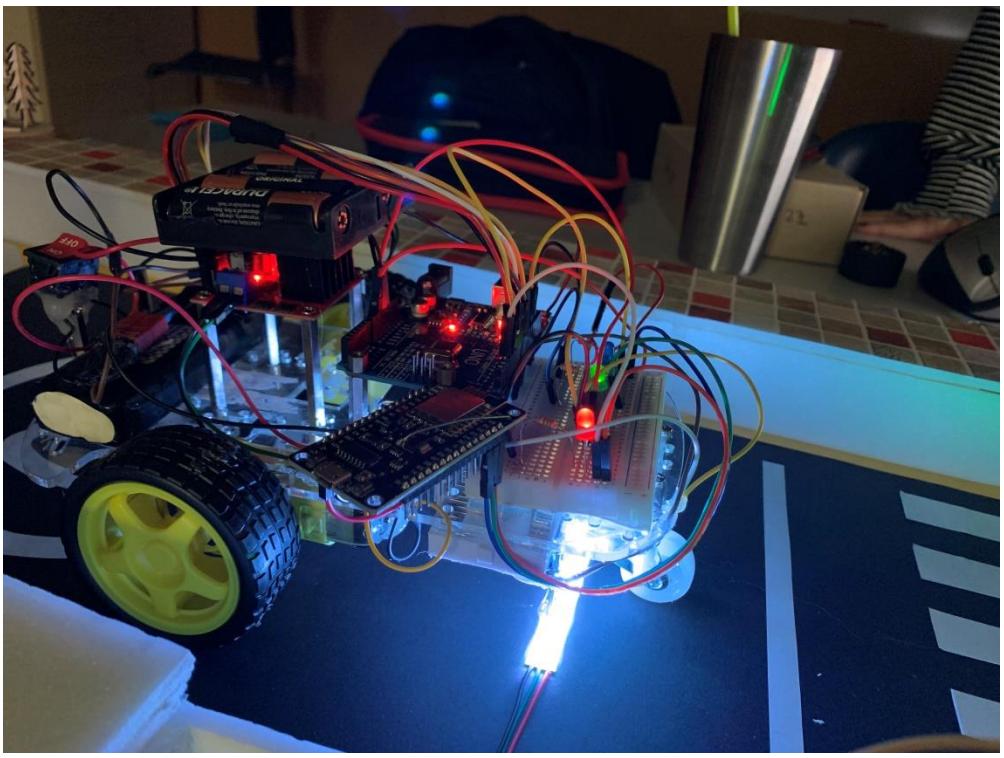


보행자 신호등이 초록불일 때, 남은 시간을 알리기 위해 점점 줄어드는 NeoPixel ON LED 개수



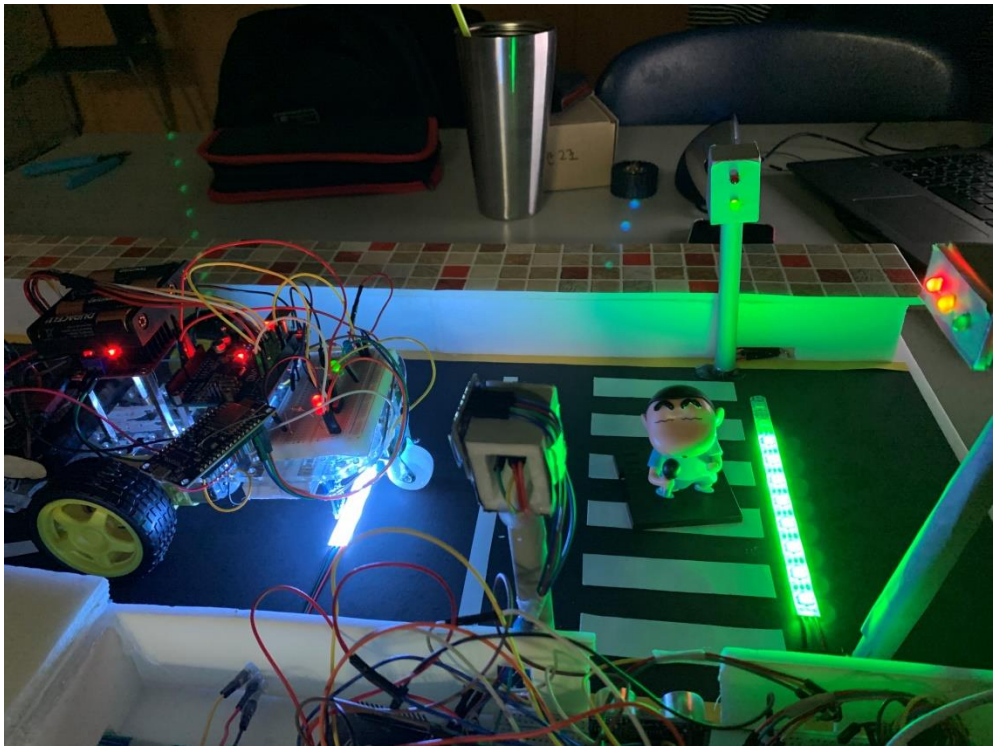
모터드라이브를 이용하여 설계대로 완성된 차체



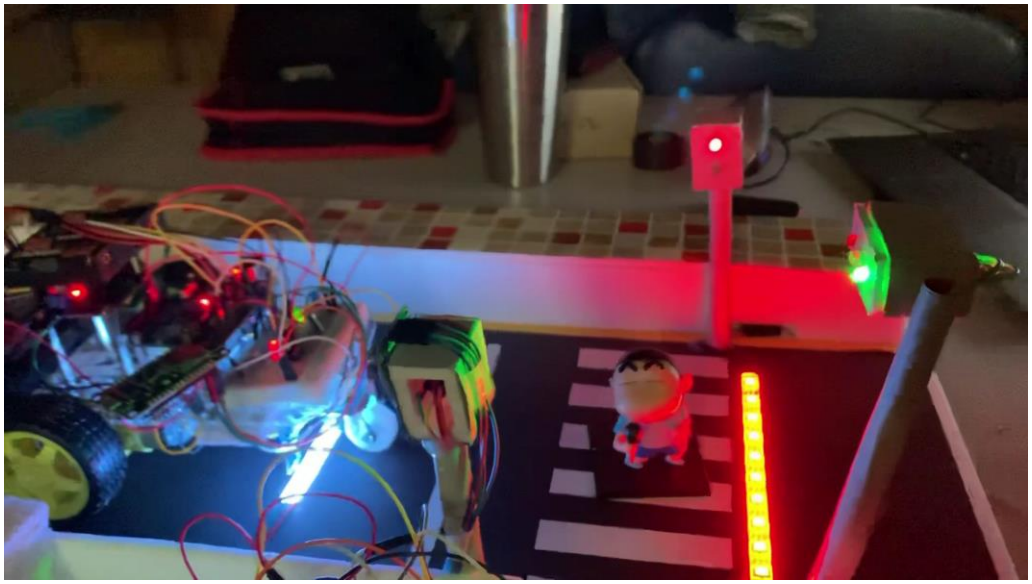


조도 센서를 이용하여 정차선 근처에서 빛을 탐지하여 자신이 계속 가야할지, 멈춰야 할지를 결정한다. 위의 사진은 조도 센서가 빛을 감지하고(차의 GREEN LED ON), 멈춤 신호를 받아(RED LED ON) 멈춰있는 상태이다.

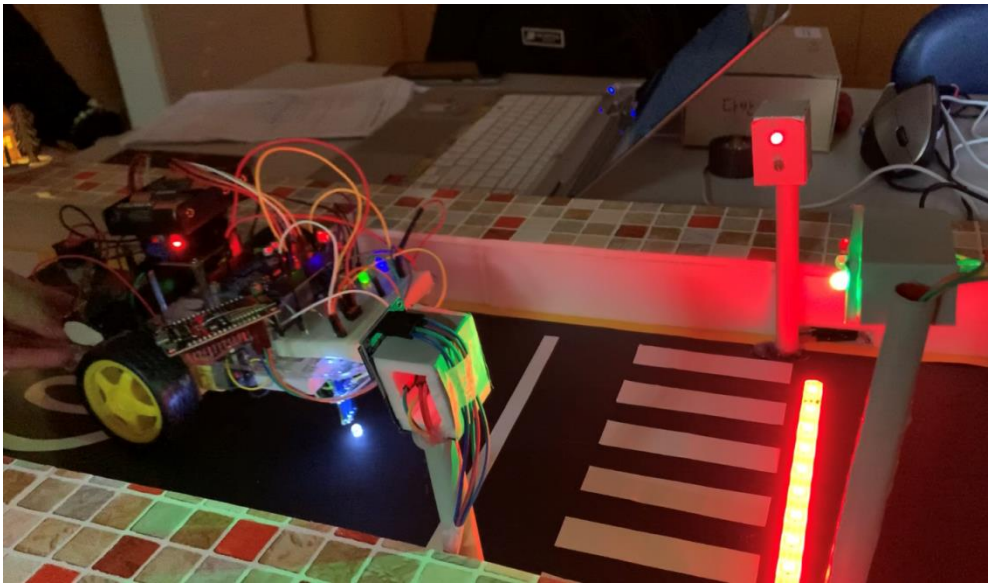
차의 RED LIGHT가 ON(멈춤 신호 받음)하여 차가 멈추는 경우는 두 가지인데 아래 사진에 덧붙여 설명하도록 한다.



자동차 신호등이 빨간 불일 경우.



자동차 신호등이 파란 불이지만, 사람이 횡단보도에 있는 경우.



위의 사진은 횡단보도에 사람이 없고, 자동차 신호등이 초록 불이므로 가도 된다는 신호를 받아(BLUE LED ON) 계속 주행하는 모습이다.

```
온도 : 22.80, 습도 : 53.00
go to cloud
Waiting...
x : 0, distance : 26
go to cloud
Waiting...
x : 0, distance : 26
go to cloud
Waiting...
x : 0, distance : 15
go to cloud
Waiting...
x : 0, distance : 14
go to cloud
Waiting...
x : 0, distance : 14
go to cloud
Waiting...
x : 1, distance : 14
x : 0, distance : 26
go to cloud
Waiting...
x : 1, distance : 26
go to cloud
Waiting...
x : 1, distance : 26
go to cloud
Waiting...
x : 1, distance : 26
go to cloud
Waiting...
x : 1, distance : 9
go to cloud
Waiting...
x : 1, distance : 9
온도 : 23.70, 습도 : 61.00
go to cloud
Waiting...
x : 0, distance : 9
```

위의 사진은 시리얼 모니터를 캡쳐한 것이다. distance가 26인 경우는 사람이 횡단보도에 없는 경우이다. 사람이 들어오면 distance가 줄어드는 것을 볼 수 있다. x가 0이면 차는 stop신호를 받게 되고(자동차 신호등 빨간 불) x가 1이면 차는 go신호를 받게 된다(자동차 신호등 초록 불) 단, x가 1일 때도 distance를 받아 사람이 있는 경우라면 stop 신호를 받게 된다.