|  |
| --- |
| **(작품명) 주간진행보고서** |

**2학년 YA반 2023. 12. 07**

|  |  |  |
| --- | --- | --- |
| **1** |  | **주간 진행 내용 (11/29~12/06)** |

구현 내용, H/W 사진, S/W 스케치 등

This Arduino code appears to control a car-like robot with two servo motors for movement. The robot is equipped with infrared sensors for obstacle detection, and light intensity sensors, and is controllable via Bluetooth commands. The main logic includes obstacle avoidance, light intensity measurement, and servo motor control based on received Bluetooth commands.

이 아두이노 코드는 움직이는 두 개의 서보모터로 자동차와 같은 로봇을 제어하는 것으로 보입니다. 이 로봇은 장애물 감지를 위한 적외선 센서, 빛 세기 센서를 갖추고 있으며 블루투스 명령을 통해 제어가 가능합니다. 주요 로직은 수신된 블루투스 명령을 기반으로 장애물 회피, 빛 세기 측정, 서보모터 제어 등이 있습니다.

1 장애물 감지 후 정지

2 장애물 감지 후 정차 후 우회전

3 차량이 감지되고 메뉴가 우측에 표시되면 차량이 멈춥니다

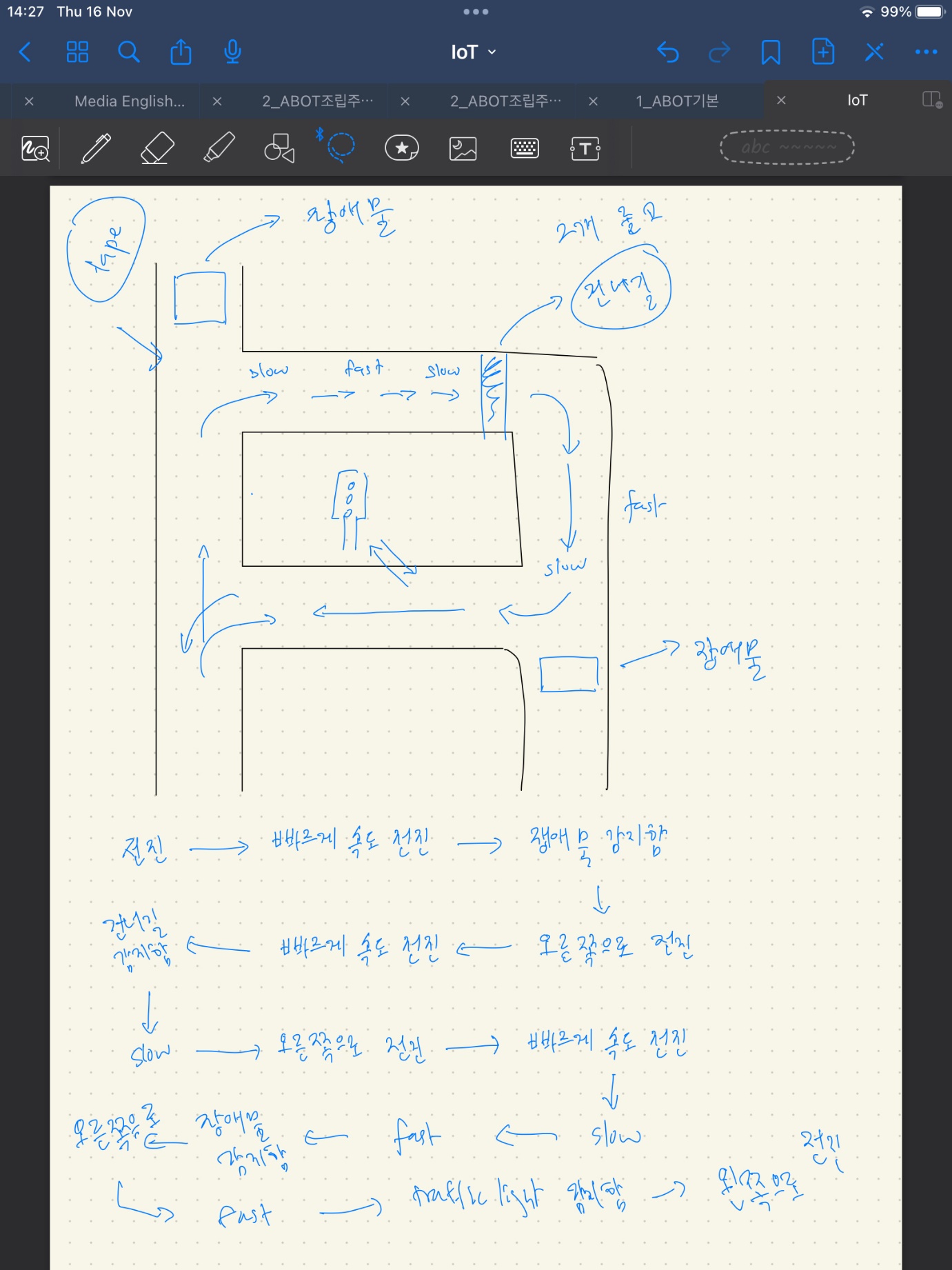
4 필요한 항목(도로) 및 참고자료 발굴

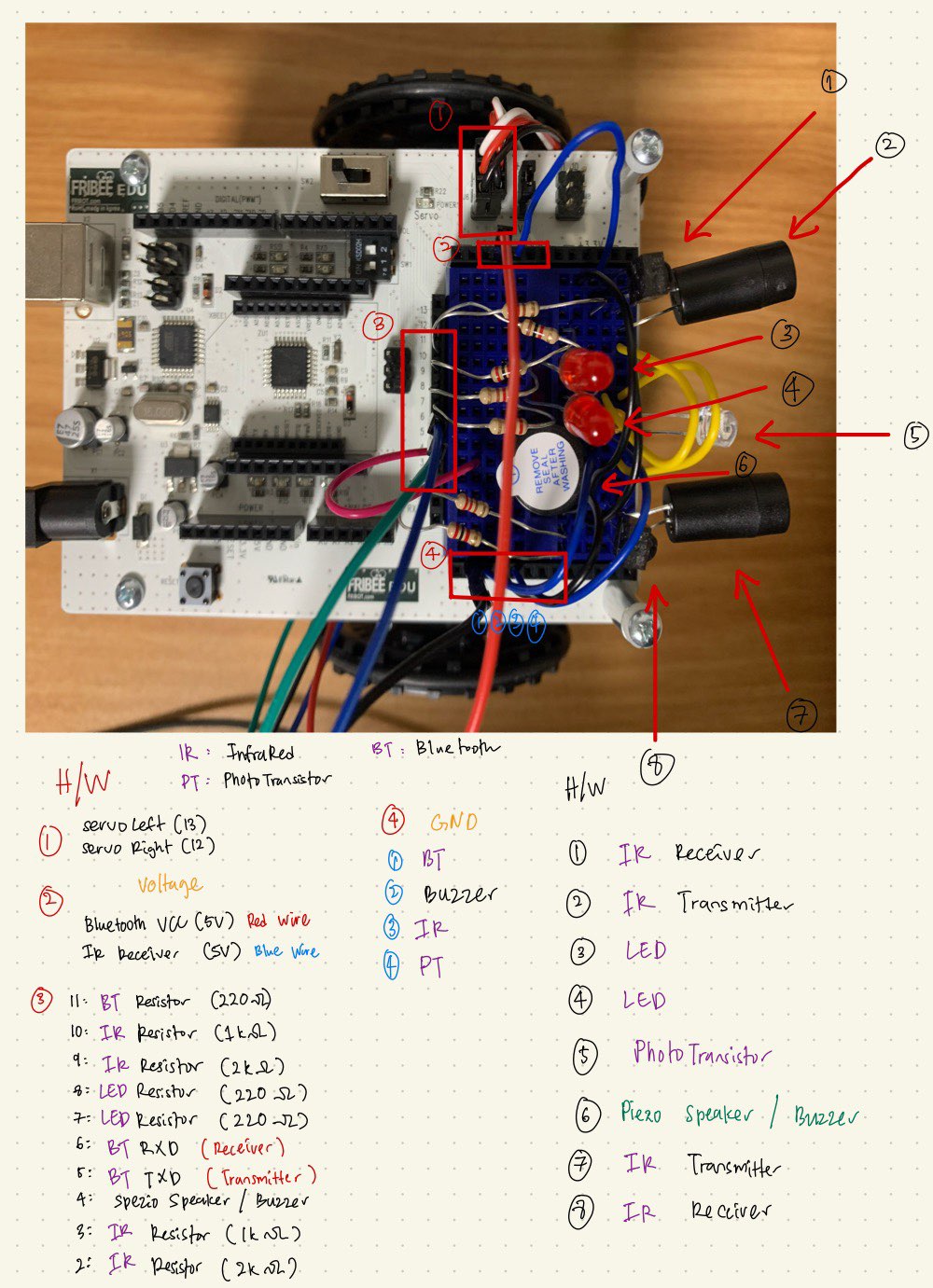
5 차량에 블루투스 페어링 장치 추가

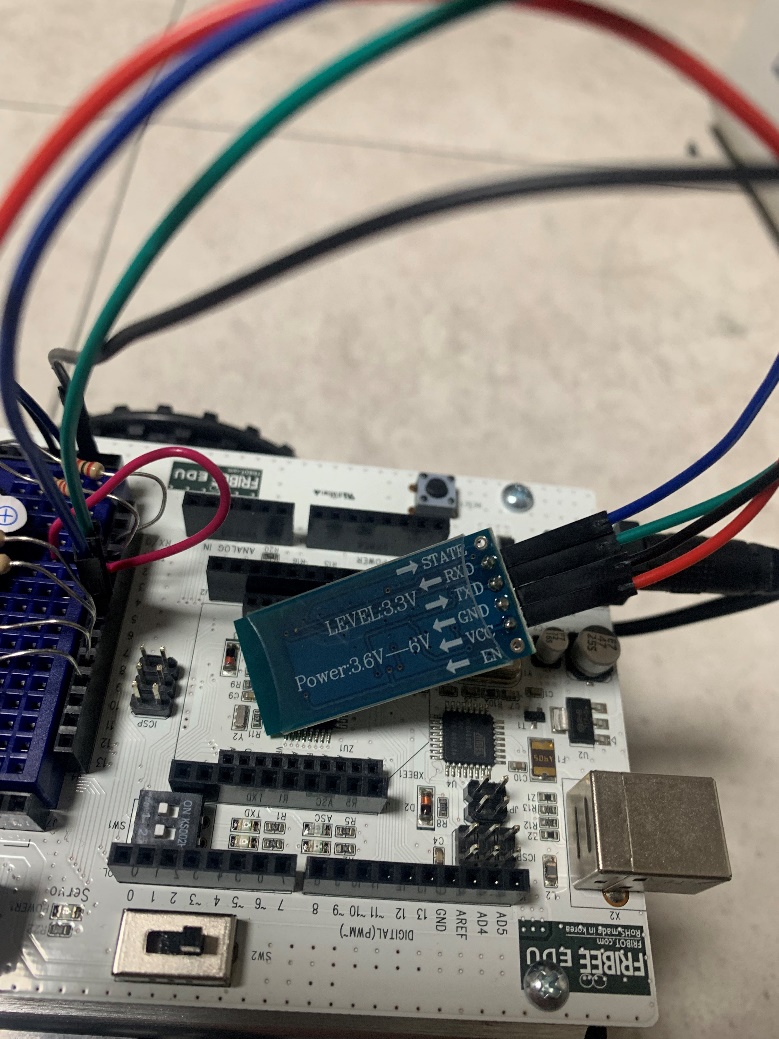
6 원격으로 차량을 제어합니다

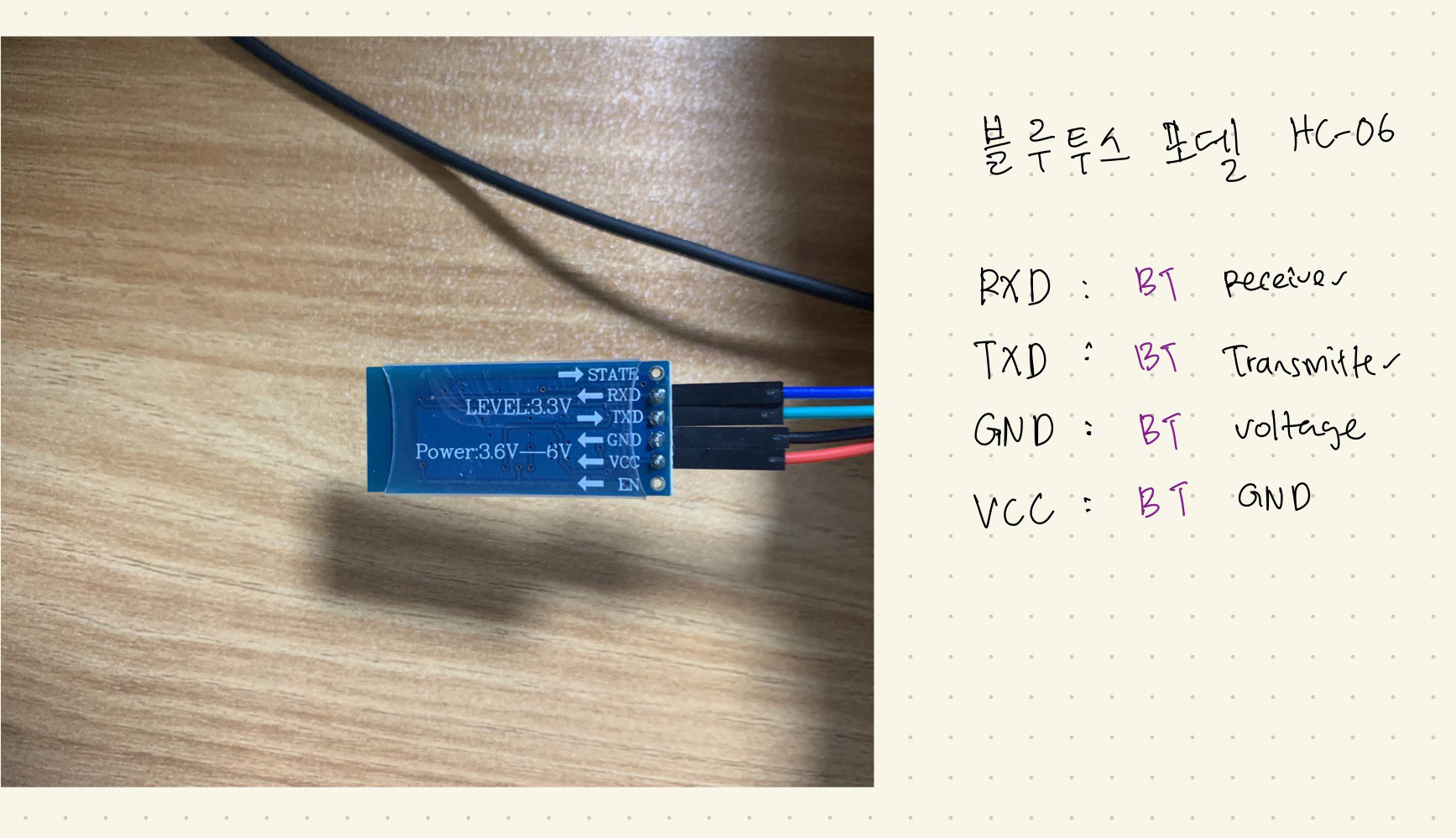
7 포토트랜지스터, 빛이 있으면 멈출 것이고, 빛이 없으면 계속 앞으로 나아갈 것이다

8 적외선 센서, 하나의 센서만 감지되면 방향을 반대하여 장애물을 피하고 계속해서 앞으로 나아갑니다









//Software Estella

#include <Servo.h>

#include <SoftwareSerial.h>

Servo servoLeft;

Servo servoRight;

const int setpoint = 2;    // 세트포인트를 위한 출력 핀 (용도는 완전히 명확하지 않음)

const int kpl = -50;        // 왼쪽 서보에 대한 비례 제어 이득

const int kpr = -50;        // 오른쪽 서보에 대한 비례 제어 이득

int speed = 0;              // 자동차의 속도를 나타내는 변수 (현재 사용되지 않음)

const int threshold = 1;    // 장애물 감지를 위한 거리 임계값

SoftwareSerial BT(5, 6);    // 소프트웨어 시리얼 통신 핀 (RX: 5, TX: 6)

void setup() {

  pinMode(10, INPUT);        // 핀 10을 입력으로 설정

  pinMode(9, OUTPUT);        // 핀 9를 출력으로 설정

  pinMode(3, INPUT);         // 핀 3을 입력으로 설정

  pinMode(2, OUTPUT);        // 핀 2를 출력으로 설정

  pinMode(8, OUTPUT);        // 핀 8을 출력으로 설정

  pinMode(7, OUTPUT);        // 핀 7을 출력으로 설정

  pinMode(setpoint, OUTPUT); // 세트포인트 핀을 출력으로 설정

  servoLeft.attach(13);      // 왼쪽 서보를 핀 13에 연결

  servoRight.attach(12);     // 오른쪽 서보를 핀 12에 연결

  tone(4, 3000, 1000);       // 핀 4에서 1초 동안 소리를 생성

  delay(1000);               // 1초 동안 대기

  // 블루투스 설정

  BT.begin(9600);            // 9600의 보레이트로 소프트웨어 시리얼 통신 시작

  Serial.begin(9600);        // 9600의 보레이트로 시리얼 통신 시작

}

bool shouldMoveForward = false;  // 자동차가 전진해야 하는지를 나타내는 플래그

bool shouldResumeCommands = true;  // 블루투스 명령 실행을 재개해야 하는지를 제어하는 플래그

void loop() {

  // 블루투스 제어

  if (BT.available()) {

    char command = BT.read();

    handleBluetoothCommand(command);

  }

  // 장애물 회피 로직

  int irLeft = irDistance(9, 10);    // 왼쪽 적외선 센서를 사용하여 거리 측정

  int irRight = irDistance(2, 3);    // 오른쪽 적외선 센서를 사용하여 거리 측정

  digitalWrite(8, !irLeft);          // 핀 8을 왼쪽 적외선 센서의 읽기 값에 기반하여 설정

  digitalWrite(7, !irRight);         // 핀 7을 오른쪽 적외선 센서의 읽기 값에 기반하여 설정

  // 빛 센서로 측정

  float tLight = float(rcTime(11));   // 오른쪽 빛 강도를 측정하고 float로 변환

  float ndShade = (tLight / 1000) - 0.5; // 정규화된 빛 차이 계산

  // 헤딩 표시

  Serial.println("ndShade     tRight");

  Serial.print(tLight);             // tRight 값을 표시

  Serial.print("       ");         // 공백을 표시

  Serial.print(ndShade);            // ndShade 값을 표시

  Serial.println();                 // 추가 줄바꿈

  delay(500);                       // 1초 지연

  if (irLeft < threshold && irRight < threshold) {

    tone(4, 3000, 1000);

    delay(1000);

    stopCar();

    digitalWrite(8, LOW);

    digitalWrite(7, LOW);

    // 장애물에 대한 기기 알림 및 명령을 기다림

    BT.println("장애물 감지! '1'을 보내면 우회전, '2'를 보내면 정지 후 종료.");

    while (true) {

      if (BT.available()) {

        char command = BT.read();

        handleBluetoothCommand(command);

        break; // 명령 처리 후 루프 탈출

      }

    }

  } else if (irLeft < threshold) {

    turnSlightRight();

    moveForward();

  } else if (irRight < threshold) {

    turnSlightLeft();

    moveForward();

  } else if(tLight < 3000){

    stopCar();

  } else if(tLight > 3000){

    moveForward();

  }

  // 플래그가 설정된 경우 자동으로 전진

  if (shouldMoveForward) {

    moveForward();

    shouldMoveForward = false;     // 전진 후 플래그 재설정

    shouldResumeCommands = true;   // 명령 실행 재개

  }

}

// 적외선 거리를 측정하는 함수

int irDistance(int irLedPin, int irReceivePin) {

  int distance = 0;

  for (long f = 41000; f <= 42000; f += 500) {

    distance += irDetect(irLedPin, irReceivePin, f);

  }

  return distance;

}

// 특정 주파수에서 적외선 거리를 감지하는 함수

int irDetect(int irLedPin, int irReceiverPin, long frequency) {

  tone(irLedPin, frequency, 8);

  delay(1);

  int ir = digitalRead(irReceiverPin);

  delay(1);

  return ir;

}

// 커패시터가 방전하는 데 걸리는 시간을 측정하는 함수 (빛 강도 측정)

long rcTime(int pin) {

  pinMode(pin, OUTPUT);

  digitalWrite(pin, HIGH);

  delay(5);

  pinMode(pin, INPUT);

  digitalWrite(pin, LOW);

  long time = micros();

  while (digitalRead(pin));

  time = micros() - time;

  return time;

}

// 서보 모터를 사용하여 기동을 수행하는 함수

void maneuver(int speedLeft, int speedRight, int msTime) {

  servoLeft.writeMicroseconds(1500 + speedLeft);

  servoRight.writeMicroseconds(1500 - speedRight);

  if (msTime == -1) {

    servoLeft.detach();

    servoRight.detach();

  }

  delay(msTime);

}

// 블루투스 명령을 처리하는 함수

void handleBluetoothCommand(char command) {

  switch (command) {

    case '1':

      turnRight();

      moveForward();

      shouldMoveForward = true;

      break;

    case '2':

      turnLeft();

      moveForward();

      shouldMoveForward = true;

      break;

    case '3':

      moveBackward();

      shouldResumeCommands = true;

      break;

    case '4':

      moveForward();

      shouldResumeCommands = true;

      break;

    case '5':

      stopCar();

      shouldResumeCommands = false;

      break;

    // 필요에 따라 더 많은 명령을 추가

  }

}

// 자동차를 전진시키는 함수

void moveForward() {

  maneuver(200, 200, 500);

}

// 자동차를 정지시키는 함수

void stopCar() {

  maneuver(0, 0, 250);

  delay(500);

}

// 자동차를 오른쪽으로 회전시키는 함수

void turnRight() {

  maneuver(200, -200, 600);

}

// 약간 오른쪽으로 회전하는 함수

void turnSlightRight() {

  maneuver(100, -100, 1000);

}

// 자동차를 왼쪽으로 회전시키는 함수

void turnLeft() {

  maneuver(-200, 200, 600);

}

// 약간 왼쪽으로 회전하는 함수

void turnSlightLeft() {

  maneuver(100, -100, 1000);

}

// 자동차를 후진시키는 함수

void moveBackward() {

  maneuver(-200, -200, 600);

}

|  |  |  |
| --- | --- | --- |
| **2** |  | **진행 중 어려운 점, 문의 사항** |

Difficulties in progress, inquiries

**11/29**

장애물을 감지할 때 어려움이 있습니다. irDetect로 인해 빠르게 감지되지 않습니다. 그 외에도 갑자기 길을 건너는 사람이 로봇이 감지하지 못하면 여전히 시도하고 있습니다.

**12/06**

현재 우리는 도로 개발에 문제가 있고 적외선 센서의 감도가 여전히 우리를 만족시키지 못했습니다.

|  |  |  |
| --- | --- | --- |
| **3** |  | **다음주 계획 (12/07~12/13)** |

**11/29**

다음 주 계획은 자동차의 일부 부품을 고치고, 도로와 건물을 짓는 것 등을 시도하는 것입니다

좀 더 구체적인 S/W 스케치를 , H/W 사진그리고 좀 더 시행착오를 합니다

**12/06**

다음 주 계획은 자동차의 일부 부품을 고치고, 도로와 건물을 짓고, 센서를 고치고, 만족할 수 있을 때까지 시도하고, 비디오를 녹화하고, 프레젠테이션 슬라이드를 만드는 것입니다. 저는 자동차를 멈추게 하기 위해 도로 LED인 하드웨어를 하나 더 추가할 수도 있습니다.

좀 더 구체적인 S/W 스케치를 , H/W 사진그리고 좀 더 시행착오를 합니다

파일 이름에 이름과 작품명을 넣으세요.