|  |
| --- |
| **( Estella ) 개발계획서** |

이름: 루즈한, 김현률, 아비스

|  |  |  |
| --- | --- | --- |
| **1** |  | **개발 목표와 특징** |

□ 각 팀에서 개발하고자 하는 제안시스템의 특징을 반영한 개발목표를 기술

이 아두이노 코드는 움직이는 두 개의 서보모터로 자동차와 같은 로봇을 제어하

는 것으로 보입니다. 이 로봇은 장애물 감지를 위한 적외선 센서, 빛 세기 센서를

갖추고 있으며 블루투스 명령을 통해 제어가 가능합니다. 주요 로직은 수신된 블

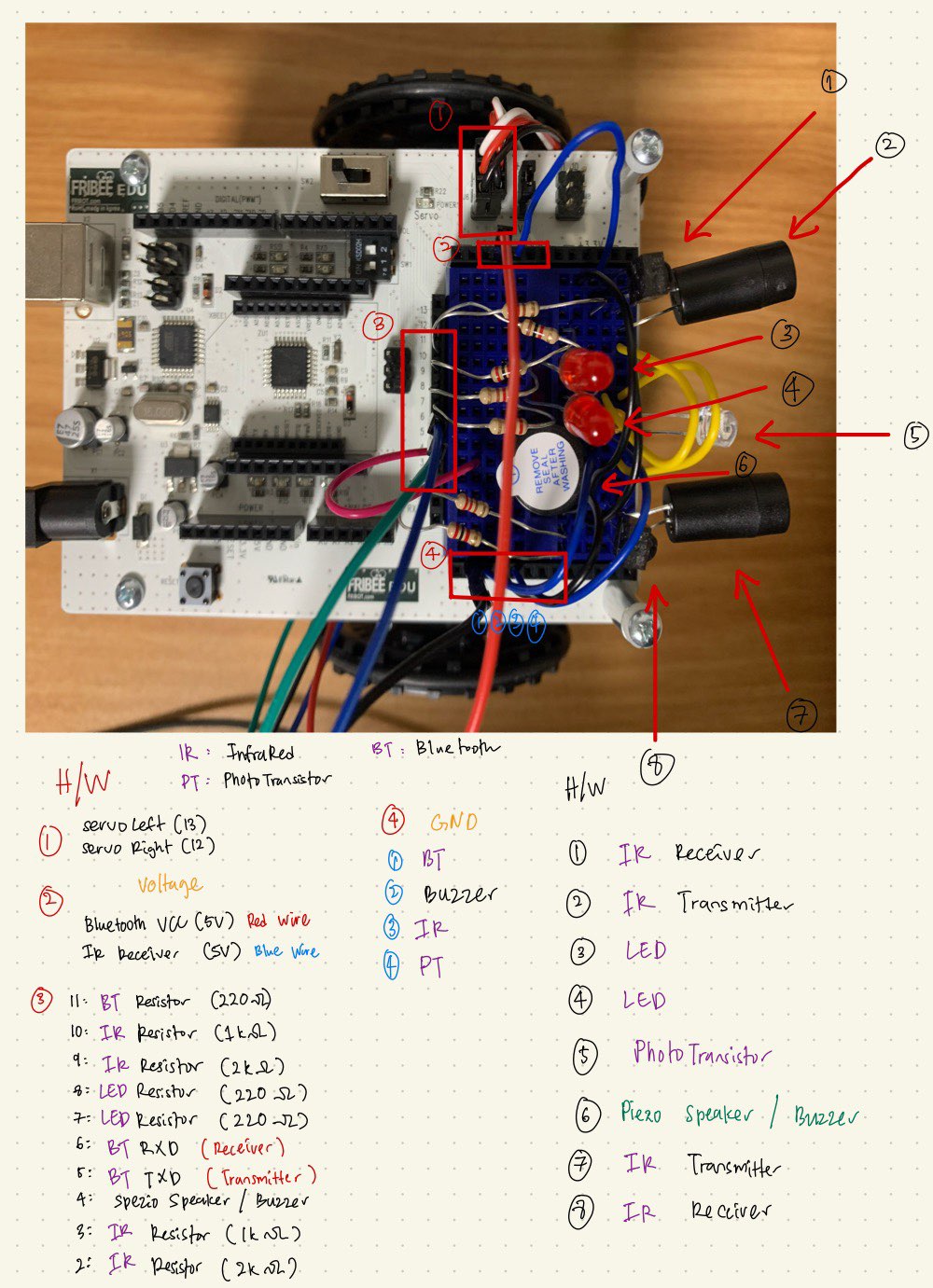
루투스 명령을 기반으로 장애물 회피,빛 세기 측정, 서보모터 제어 등이 있습니다.

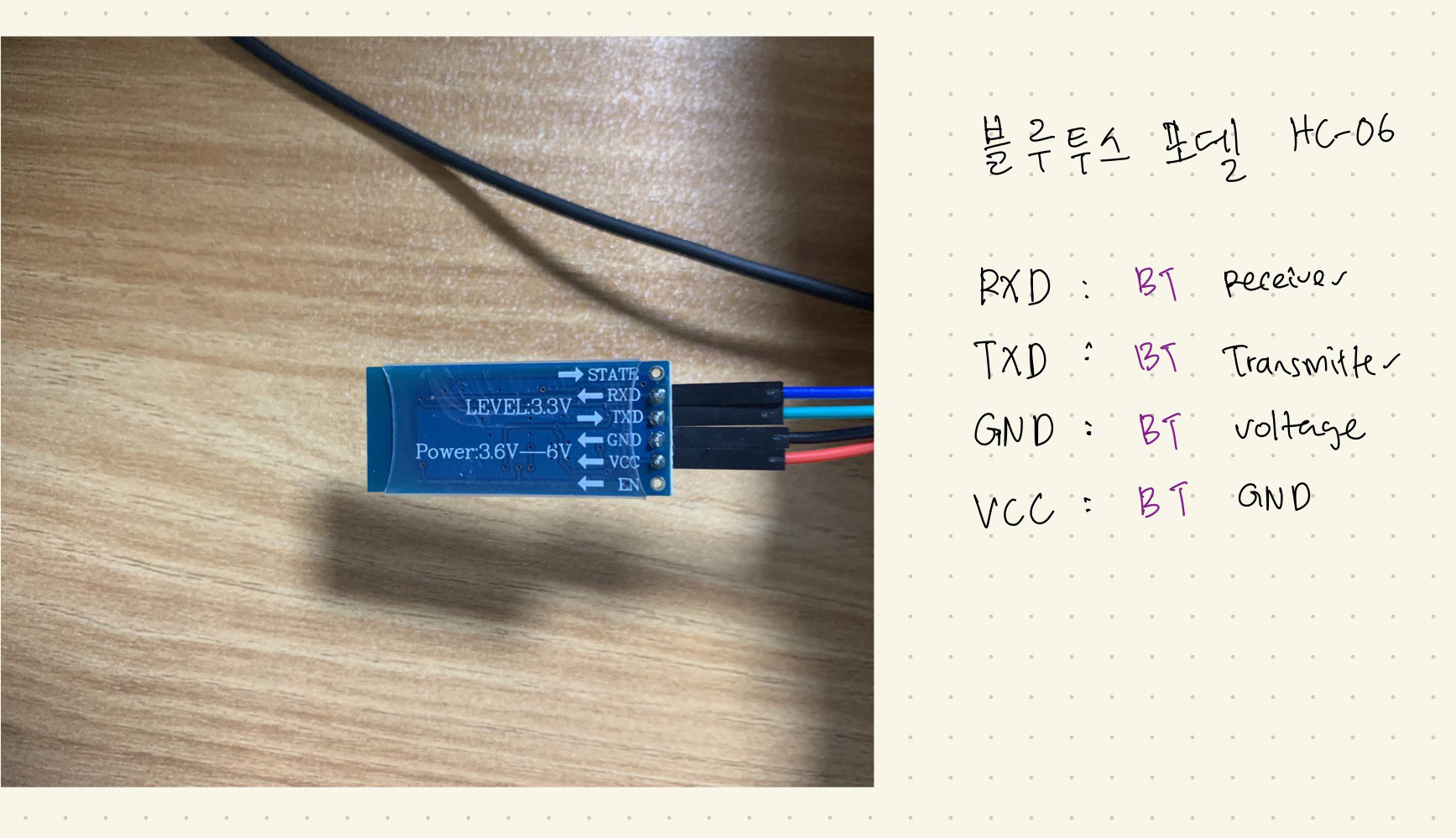
|  |  |  |
| --- | --- | --- |
| **2** |  | **개발 필요성** |

**□** 왜 본 시스템 개발이 필요한지 기술적 환경 및 수요 배경을 중심으로 기술

적외선 센서 시스템을 사용하여 물체를 감지하여 위반 사항이 발생하지 않도록 합니다. 그런 다음 블루투스 시스템을 사용하여 차량의 사용자 장치로부터 신호를 수신하여 장애물을 피하라는 명령을 받습니다. 또한 포토트랜지스터를 사용하여 차량 주변의 밝기 레벨을 사용하여 차량의 움직임을 제어할 수 있으며 어두워지면 차량이 움직이고 밝을 때는 차량이 정지합니다. 이 기능은 특히 야간에 신호등과 같은 조명이 있는 경우 구현할 수 있습니다. 다음으로 LED의 활용은 차량 앞에 장애물이 있으면 사용자가 신호와 부저가 소리를 냅니다. 따라서 사용자는 차량 앞에 있는 장애물을 인식하게 됩니다.

|  |  |  |
| --- | --- | --- |
| **3** |  | **H/W 구현** |

□ H/W를 어떻게 구현했는지를 그림,사진 등으로 설명, 사용 센서, 부품들을 나열



**적외선 센서:** 장애물 감지.

**LED:** 신호를 물리적으로 사용자에게 표시합니다.

**포토트랜지스터:** 정지할 것인지 이동할 것인지, 빛의 밝기를 감지합니다.

**부저:** 장애물이 있을 때 소리를 냅니다.

**블루투스:** 장치에서 명령을 내리는 중입니다.

|  |  |  |
| --- | --- | --- |
| **4** |  | **S/W 구현** |

□ S/W를 어떻게 구현했는지를 설명

이 코드는 Arduino 플랫폼을 사용하여 자동차를 제어하는 간단한 소프트웨어를 구현한 것입니다. 여러 기능이 통합되어 있으며, 다음과 같은 주요 구현 내용이 있습니다.

**라이브러리 및 핀 설정:**

Servo 및 SoftwareSerial 라이브러리를 가져왔습니다.

서보 모터 및 소프트웨어 시리얼 통신을 위한 핀 설정을 초기화합니다.

#include <Servo.h>

#include <SoftwareSerial.h>

Servo servoLeft;

Servo servoRight;

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

**핀 및 블루투스 설정:**

디지털 및 아날로그 핀을 입력 또는 출력으로 설정합니다.

서보 모터를 연결하고 초기 설정을 수행합니다.

소프트웨어 시리얼 통신을 초기화합니다.

void setup() {

  // ... (핀 모드 및 초기 설정)

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

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

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

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

}

**빛 센서 및 빛 강도 측정:**

Rc 타임 메서드를 사용하여 빛 센서로부터 빛의 강도를 측정합니다.

float tLight = float(rcTime(11));   // 빛 강도 측정

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

**자동차 동작 제어:**

빛의 강도에 따라 자동차를 시작하거나 정지시킵니다.

if (tLight < 2000) {

  stopCar();

} else {

  moveForward();

}

**블루투스 명령 처리:**

블루투스로부터 명령이 도착하면 해당 명령을 처리합니다.

if (BT.available()) {

  char command = BT.read();

  handleBluetoothCommand(command);

}

**장애물 회피:**

적외선 거리 센서를 사용하여 장애물을 감지하고, 장애물이 감지되면 자동차를 정지시킵니다.

int irLeft = irDistance(9, 10);

int irRight = irDistance(2, 3);

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

  stopCar();

  // 장애물 감지 시 소리 및 LED 표시 후 장애물 명령을 기다림

  obstacleDetected = true;

  while (obstacleDetected) {

    if (BT.available()) {

      char command = BT.read();

      handleBluetoothCommand(command);

      obstacleDetected = false;  // 명령 처리 후 플래그를 재설정

    }

  }

}

**서보 모터 및 자동차 동작 함수:**

서보 모터를 사용하여 자동차의 동작을 구현한 함수입니다.

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

void moveForward();

void stopCar();

void turnRight();

void turnLeft();

void moveBackward();

이러한 구현을 통해 빛 강도, 블루투스 명령, 장애물 회피 등 다양한 입력에 따라 자동차의 동작을 제어할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| **5** |  | **구현 소감** |

□ 개발 과정에 있었던 일과 느낌을 소개(개인별)

**루즈한:** 처음에는 로봇 시스템을 이동하기가 힘들지만, 이 IOT 프로그래밍 중이기 때문에 로봇 시스템을 취급하기 때문에 로봇 시스템을 처리하는 것에 대해 많이 배웠어요.간단한 운동에서 힘든 운동 .차량의 중심은 차량의 centering 매우 중요합니다.

**김현률:** 프로그램은 오류가 없는 것 같고 컴퓨터랑 연결되어 있으면 잘 작동하는데 선을 제거하면 오류가 많이 나고, 동작이 잘 되지 않아 원하는 것을 더 하지 못해 아쉽다.

**아비스:** 트랜지스터, 블루투스 모듈, 적외선 센서를 이용하여 자율주행 RC카를 구현하는 것은 흥미로운 도전이었습니다. 트랜지스터를 이용하여 모터를 제어하고 블루투스로 원격통신을 구현하며 적외선 센서로 주변을 감지하는 것이 가능했습니다. 이러한 다양한 센서와 모듈을 통합하여 RC카가 주행 중 주변을 감지하고 사용자 명령을 수행할 수 있도록 했습니다. 이 프로젝트는 나에게 iot와 프로그래밍에 대한 풍부한 경험을 주었습니다.

|  |  |  |
| --- | --- | --- |
| **6** |  | **향후 보완 사항** |

□ 향후 보완이 필요하다고 생각되는 사항 기술

신호등의 존재를 인지하고 멈추는 것을 구현하며, 그 외에도, 차량에 더 많은 센서를 장착하여 멀리서 사람들이 도로를 건너는 것인지하고, 차량이 천천히 멈추도록 해 볼 것이다. 식으로

|  |  |  |
| --- | --- | --- |
| **7** |  | **참고 문헌** |

□ 개발 과정에 참고한 도서, 사이트 등을 기술

How To Train Yourself Arduino Robot: <https://loadofprogrammer.tistory.com/20>

Bluetooth: <https://www.youtube.com/watch?v=-E9J9ExdZ20>

Infra-Red Sensors: <https://www.youtube.com/watch?v=gADIb1Xw8PE>

Internet of Things의 관련: 1주차-7주차의 PowerPoint

|  |  |  |
| --- | --- | --- |
| **8** |  | **스케치 소스 코드** |

□ 스케치 코드를 첨부하세요. 가져온 소스는 표시하세요.

//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 bluetoothCommandReceived = false;

bool obstacleDetected = false; // 장애물 감지 플래그

void loop() {

  // 빛 센서 로직을 사용하여 자동차 시작

  if (bluetoothCommandReceived) {

    // 명령이 수신된 경우 명령을 실행하고 플래그를 재설정합니다

    bluetoothCommandReceived = false;

  } else {

    // 명령이 수신되지 않으면 광 센서 로직을 진행합니다

    float tLight = float(rcTime(11));   // 빛 강도 측정

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

    // 빛 센서에 기반하여 시작 또는 정지

    if (tLight < 2000) {

      stopCar();

    } else {

      moveForward();

    }

  }

  // 블루투스 명령 확인

  if (BT.available()) {

    char command = BT.read();

    handleBluetoothCommand(command);

  }

  // 장애물 회피 로직

  int irLeft = irDistance(9, 10);

  int irRight = irDistance(2, 3);

  digitalWrite(8, !irLeft);

  digitalWrite(7, !irRight);

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

    stopCar();

    tone(4, 3000, 1000);

    delay(150);

    digitalWrite(8, LOW);

    digitalWrite(7, LOW);

    obstacleDetected = true;  // 장애물이 감지되면 플래그를 true로 설정

    // 블루투스 명령을 기다림

    while (obstacleDetected) {

      if (BT.available()) {

        char command = BT.read();

        handleBluetoothCommand(command);

        obstacleDetected = false;  // 명령 처리 후 플래그를 재설정

      }

    }

  }

}

// 적외선 거리를 측정하는 함수

int irDistance(int irLedPin, int irReceivePin) {

  int distance = 0;

  for (long f = 39000; 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;

}

// 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);

}

void handleBluetoothCommand(char command) {

  bluetoothCommandReceived = true;  // 명령을 받았을 때 플래그 설정

  switch (command) {

    case '1':

      moveForward();  // 차량을 즉시 정지시킵니다.

      break;

    case '2':

      //delay(150);

      turnRight();  // 우회전 명령을 수행합니다.

      break;

    case '3':

      //delay(150);

      turnLeft();  // 좌회전 명령을 수행합니다.

      break;

    case '4':

      moveBackward();  // 차량을 즉시 후진시킵니다.

      break;

    case '5':

      stopCar();  // 차량을 즉시 정지시킵니다.

      break;

  }

}

// 커패시터의 방전 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;

}

// 자동차를 전진시키는 함수

void moveForward() {

  maneuver(200, 200, 500);

}

// 자동차를 정지시키는 함수

void stopCar() {

  maneuver(0, 0, 250);

  delay(150);

}

// 자동차를 우회전시키는 함수

void turnRight() {

  maneuver(200, -200, 500);

}

// 자동차를 좌회전시키는 함수

void turnLeft() {

  maneuver(-200, 200, 600);

}

// 자동차를 후진시키는 함수

void moveBackward() {

  delay(150);

  maneuver(-200, -200, 600);

}

파일 이름에 본인 이름과 작품명을 넣으세요.