[팀 프로젝트]

보고서

과 목 명: 기초공학설계

프로젝트: 아두이노로 장애물 떨어뜨리기

과목명	기초공학설계	
교수명	신석훈 교수님	
학과	소프트웨어학과	
학번/이름	2016125003	구영민
	2016125019	김한섭
	2016125029	서민철
조	7조	
제출일	2016년 12월 8일	

1. 문제 분석

1-1. 문제에 대한 분석

1차 시험

테이블(50cm × 60cm) 위에 놓인 **하나**의 물체를 자동차를 이용하여 테이블 아래로 밀어내는 문제이다. 자동차가 테이블 아래로 떨어지면 실격이고 물체를 밀어내고 종료신호를 출력할 때까지의 시간을 측정한다. 도전은 총 세 번 가능하며 빨리 성공할수록 점수가 높고 같은 시도에서 성공하였다면 더 짧은 기록을 가진 팀이 더 높은 점수를 받는다.

2차 시험

테이블 위에 놓인 **여러 개**의 물체를 자동차를 이용하여 테이블 아래로 밀어내는 문제이다. 대부분의 규칙은 1차 시험과 동일하나 물체를 무조건 자동차의 정면으로 밀어내야 인정된다.

1-2. 해결 방안

1차 시험

테이블의 크기가 50cm × 60cm이기 때문에, 대각선의 길이를 계산해 보면 대략 78cm이다. 자동차의 전후 길이와 물체의 크기를 고려해 보면 센서가 감지해야 하는 물체까지의 거리는 약 60cm이다. 따라서, 초음파 센서와 적외선 센서 모두 센서의 스펙상으로는 물체 감지 범위 내에 있었기 때문에 물체를 감지하는 데는 문제가 없었다. 따라서 어떤 센서가 실제로 물체를 더 정확하고 정밀하게 감지하는지를 테스트하였다. 그 결과, 초음파 센서의 경우 휴지와 같은 흡음제의 경우 제대로 측정이 불가능하여 정면 센서로 두기에는 부적합하였다. 두 센서 모두 값이 튀는 모습을 보여주었으나, 적외선 센서가 조금 더 안정적으로 처리하였다. 전면 센서와 하단 센서 모두 적외선 센서로 처리하였다.

센서의 장착을 마친 후 펌웨어를 작성하였다. 물체가 감지 범위에 있는지를 확인하려면 analogRead 함수를 사용하면 되는데, 아두이노 공식 문서에서 이 함수를 사용할 경우 0.0001초가 걸린다고 되어 있었다. 따라서 소프트웨어적으로 50번을 연속으로 읽은 후 40번 이상 범위에 있을 경우 가까이 있는 것으로 간주하였고, 코드로 옮겨 테스트해보니 정상적인 작동을 확인할 수 있었다. 자동차를 움직이는데 사용된 간단한 알고리즘은 다음과 같다: 현재 있는 위치에서는 제자리 회전을 한다. 물체를 두 번 연속 감지하면 감지하지 못할 때까지 일단 회전한 후, 물체를 감지하였던 시간부터 감지하지 못했던 시간의 정확히 절반만큼 회전시키고 직진한다. 만약 땅바닥을 보거나 물체를 감지하지 못하면 잠시 뒤로 와서 재탐색한다. 만약 재탐색에 실패하면 종료 사인을 내고 끝낸다.

2차 시험

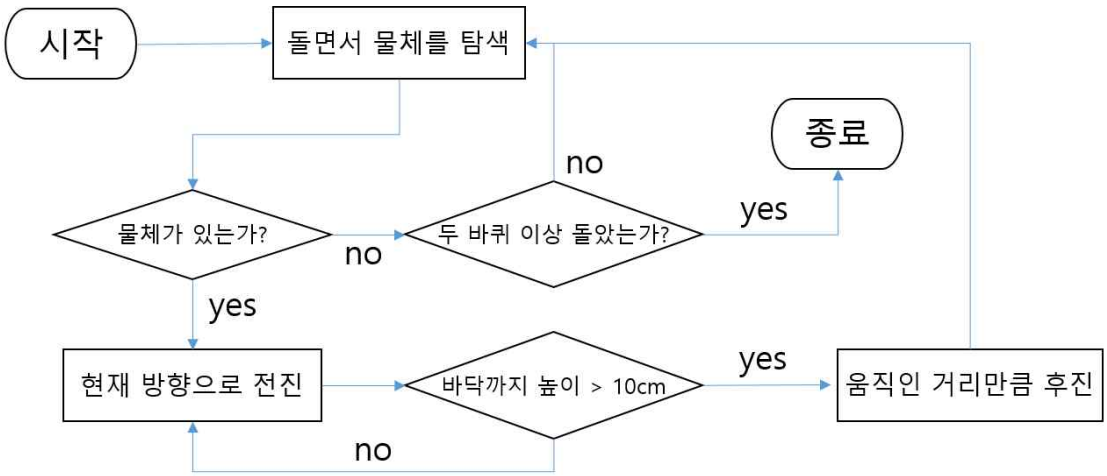
1차 때와 달라진 것은 자동차의 위치, 테이블의 크기, 장애물의 갯수이다. 테이블의 크기

가 커졌지만 자동차가 테이블의 중앙에 있다고 조건이 주어졌기 때문에, 물체를 떨어뜨리러 갔다가 다시 되돌아오도록 알고리즘을 수정하여 적외선 센서의 감지 범위를 유지할 수 있었다. 또한, 모서리에 떨어지는 것을 방지하기 위하여 앞의 양 쪽에 초음파 센서를 부착하여 1차 시험보다 더 안전하게 모서리에서 멈출 수 있도록 부품 설계를 수정하였다.

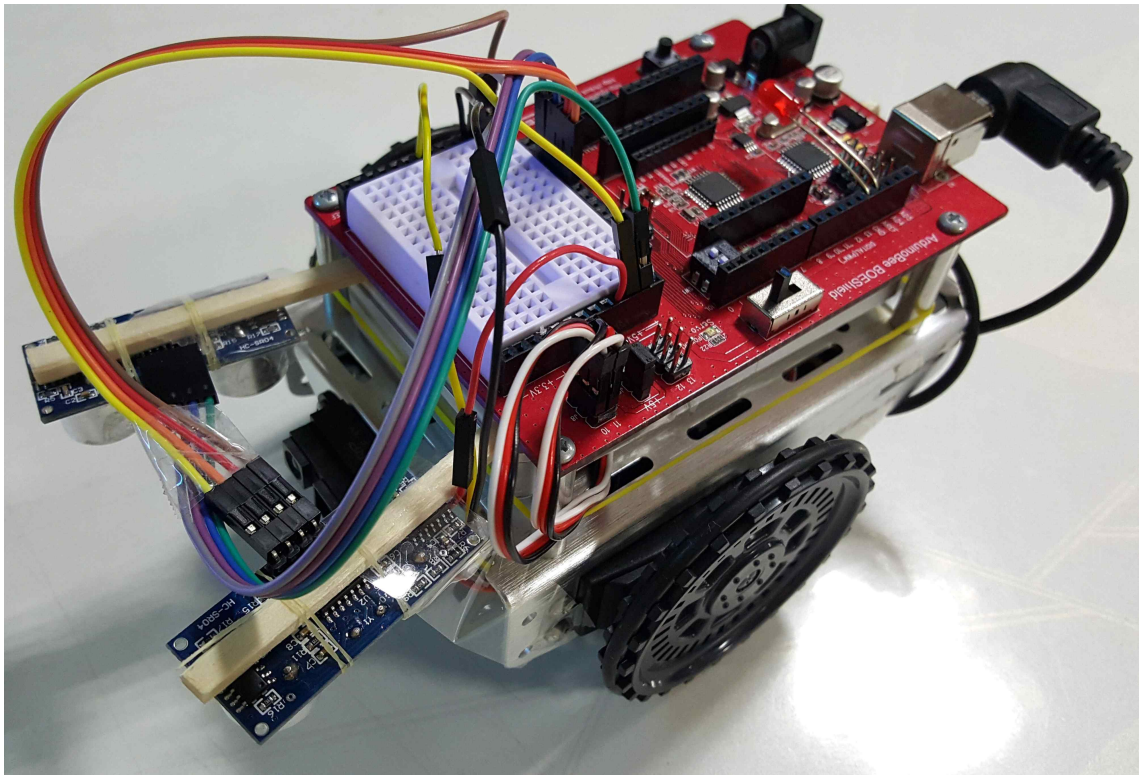
2. 설계

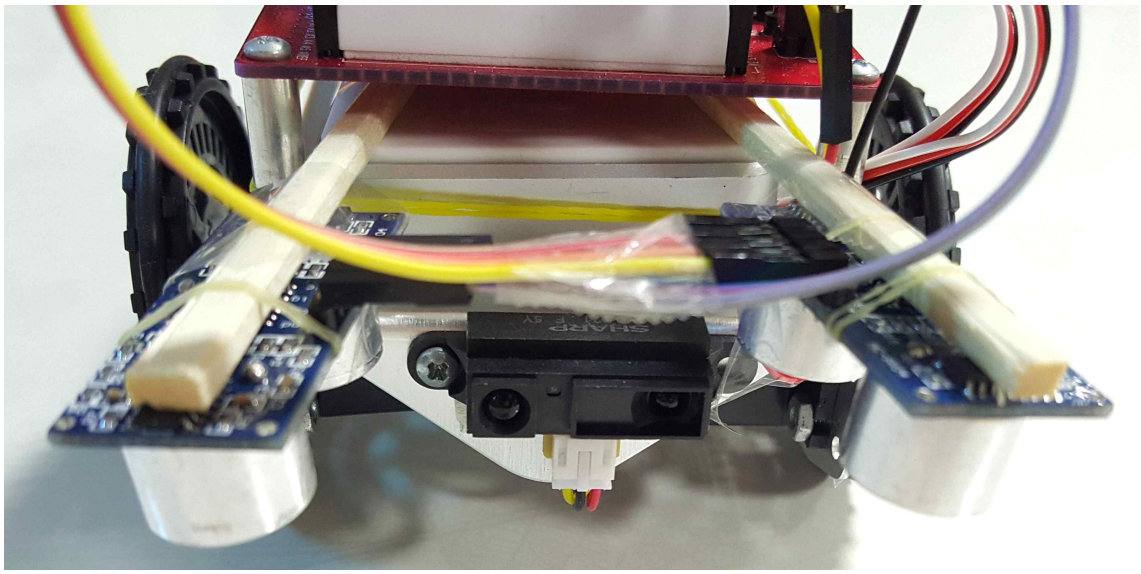
2-1. 문제 해결 방법과 알고리즘

상기한 2차 시험 때 펌웨어의 알고리즘을 이해하기 편리하도록 순서도로 시각화하였다.



2-2. 하드웨어 디자인





2차 시험 시 설계한 하드웨어의 모습이다.

3. 구현

3-1. 소스 코드와 주석

2차 시험 시의 코딩 결과이다.

```
#include <Servo.h>
#define trigPin_Left 5
#define echoPin_Left 4
#define trigPin_Right 3
#define echoPin_Right 2

Servo servoLeft, servoRight;

void move(int left, int right, int duration) {
    servoLeft.writeMicroseconds(left);
    servoRight.writeMicroseconds(right);
    delay(duration);
}

char sensorFront = A0;
int trigPin = 8;
int echoPin = 9;

int status = 1;
long last_turn_time = -1;

// 초음파 센서에서 반환된 마이크로초를 cm으로 변환해주는 함수
long microsecondsToCentimeters(long microseconds) {
    return microseconds / 29 / 2;
}

// 적외선 센서에서 값을 읽어와 cm으로 변환해주는 함수
float infrared_rays_sensor(char sensor) {
    return (24.61 / (map(analogRead(sensor), 0, 1023, 0, 5000) - 0.1696)) * 1000;
}

// 총 시행 횟수만큼 적외선 센서에서 값을 읽어와 측정 범위 내에 있는 확률을 반환
float in_range_probability(char sensor, double threshold) {
```

```
    int count = 0;
    for (int j = 0; j < 50; j++) {
        float infrared_rays = infrared_rays_sensor(sensor);
        // Serial.println(infrared_rays);
        if (1.0 <= infrared_rays && infrared_rays <= threshold) {
            count++;
        }
    }
    return (float) count / 50;
}

// 초음파 센서에서 값을 읽어와 cm로 반환
long ultrasonic_waves_sensor(int trig, int echo) {
    long duration, cm;

    digitalWrite(trig, LOW);
    delayMicroseconds(2);

    digitalWrite(trig, HIGH);  //초음파 발생
    delayMicroseconds(10);
    digitalWrite(trig, LOW);

    duration = pulseIn(echo, HIGH);  //시간 측정
    cm = microsecondsToCentimeters(duration);

    return cm;
}

// 초음파 센서에서 값을 읽어와 범위 내에 있는지 여부를 반환
bool in_range_from_bottom(int trig, int echo, long threshold) {
    long ultrasonic_waves = ultrasonic_waves_sensor(trig, echo);
    // Serial.print(ultrasonic_waves);
    // Serial.print(" ");
    // Serial.println(0 <= ultrasonic_waves && ultrasonic_waves <= threshold);
    return 0 <= ultrasonic_waves && ultrasonic_waves <= threshold;
}

// 장애물이 있는지를 제자리 회전하며 확인 후 존재 시 중앙으로 회전하고 true 반환, 없으면 false 반환
bool check_hurdle() {
    int count = 0;
```

```
    for (int i = 0; i < 70; i++) {
        double d = in_range_probability(sensorFront, 80.0);
        if (d >= 0.8) {
            digitalWrite(13, HIGH);
            delay(50);
            digitalWrite(13, LOW);

            count++;
        }
        else {
            if (count >= 2) {
                move(1530, 1530, 200 * (count / 2 + 1));
                move(1500, 1500, 0);

                return true;
            }
            count = 0;
        }
        move(1470, 1470, 150);

        if (millis() / 1000 > 110) return false; // 110초 이상이면 종료
    }
    return false;
}

void setup() {
    // 초기화
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(13, OUTPUT);
    pinMode(trigPin_Left, OUTPUT);
    pinMode(echoPin_Left, INPUT);
    pinMode(trigPin_Right, OUTPUT);
    pinMode(echoPin_Right, INPUT);
    servoLeft.attach(10);
    servoRight.attach(11);
    digitalWrite(13, LOW);

    int no_answer = 1;
    int warning = 0;
```

```
    // 차체 구동하는데 핵심적인 부분
    status = check_hurdle();
    int count = 0;

    while (status) {
        if (in_range_from_bottom(trigPin_Left, echoPin_Left, 10) == 0 ||
            in_range_from_bottom(trigPin_Right, echoPin_Right, 10) == 0) {
            // 바닥을 봤을 때

            move(1500, 1500, 500);
            move(1400, 1600, 10 * count);
            move(1470, 1470, 1000);
            count = 0;
            status = check_hurdle();
        }
        else {  // 가까이 있을 때
            move(1600, 1400, 10);
            count++;
        }

        if (millis() / 1000 > 110) status = 0; // 110초 경과 시 끝냄
    }

    move(1500, 1500, 0);
    digitalWrite(13, HIGH);
}

void loop() {}
```

4. 프로젝트 보고와 후기

4-1. 팀 구성과 담당 업무

구영민

- 전체적인 펌웨어 코딩
- 센서 배치 총괄
- 보고서 검토

김한섭
- 보고서 작성
- 알고리즘 구상

서민철
- 센서 배치
- 2차 펌웨어 코딩

4-2. 후기

구영민

컴퓨터상에서 돌아가는 C나 파이썬을 코딩할 때에는 하드웨어를 직접 제어하는 일은 없었다. 아두이노를 처음 보고서는 C로 돌아가는 언어니 컴퓨터로 코딩할 때처럼 하면 되겠다는 생각이 들었다. 아두이노라는 임베디드 기기의 펌웨어를 직접 코딩해 보니 생각이 달라졌다. 우리가 사용하는 컴퓨터와는 다르게 CPU 성능도 좋지 않고 디버거도 물릴 수 없어 신중에 신중을 가하며 코딩해야 했다. 그런데도 생각한 대로 제대로 작동하지 않으면 센서가 문제인지, 연결 문제인지, 코드의 버그인지 시리얼 포트 출력을 이용해 알아내야 했다. 또한, 센서도 생각보다 정확하지 않아 오차를 보정해 가며 코드를 만들어야 했다. 프로그래밍 실습을 해 보며 메모리 등의 자원이 제한되고 속도가 느린 하드웨어에서 펌웨어를 만들 때 필요한 여러 가지 점을 느낄 수 있었다.

김한섭

처음에 팀 배정을 받았을 때는 팀원들이 코딩에 능한 친구들이라서 이번 프로젝트는 순탄하게 갈 줄 알았다. 하지만 컴퓨터로 코딩할 때와는 다르게 하드웨어를 제어해야하는 프로젝트다 보니 많은 시행착오가 있었다. 우선 문제가 생겼을 때, 그 문제가 하드웨어상의 문제인지 코드의 오류인지를 알아야 했다. 이 문제를 해결하는 일은 시간이 매우 오래 걸리기 때문에 조립을 하기 전에 부품 하나하나를 체크해야했고 코드를 짤 때에는 가능하면 문제가 생기지 신중하게 짜야했다. 조립을 하고난 이후에 분명히 문제가 없는 것 같던 정면 적외선 센서가 문제가 생겼다. 이상한 값을 반환하여 물체를 인식을 못하는 문제였다. 이러한 문제를 해결하기 위해 정면 센서를 초음파 센서로 교체했으나 시험에 쓰이는 장애물이 무엇인지 몰랐던 상태였던 우리는 반사가 되지 않아 초음파 센서로 잘 읽히지 않는 물체가 나올 수도 있었기 때문에 적외선 센서를 이용하여 다른 방법을 모색하기로 했다. 적외선 센서를 여러 번 측정하여 평균치를 구하는 방식으로 측정방식을 바꾸었더니 오차가 훨씬 적어졌다. 여러 차례 수정을 거쳐 노력했지만 탐지 거리를 너무 짧게 한 결과 시험 당시에 물체가 예상보다 멀어 측정이 되지 않는 문제가 발견되었다. 결국 1차 시도는 실패를 했지만 2차 시도 때에는 오류를 수정하여 통과할 수 있었다.

서민철

처음에 고려했던 알고리즘은 제자리에서 돌다가, 앞에 부착된 센서가 일정 거리를 반환하면 그 방향을 향해서 쭉 직진하는 방식이었다. 두 가지의 센서(초음파, 적외선)를 전부 이용하여 제작했으며 장애물을 인식하기 위해 앞에 부착할 센서로는 초음파 센서를 사용했고, 추락방지를 위해 하단에 부착할 센서로 적외선을 사용했다. 적외선은 하드웨어 자체의 버그인지 아닌지 잘 모르겠지만 가끔 이상한 값을 반환하여 앞에 부착했을 시에 물체를 찾았음에도 기기가 제대로 작동하지 않았다. 때문에 초음파를 앞에다가 부착하고 적외선은 추락방지를 위해 바닥에 장착했는데, 초음파를 전방에 장착했을 경우 생길 수 있는 문제점은 휴지나 옷과 같이 음파를 반사하기 힘든 물체는 탐지가 어렵다는 점이었다. 장애물이 뭐가 나올지 모른다는 불안감에 저희는 앞에 달린 초음파를 적외선으로 교체하고 가끔 이상한 값을 반환하는 경우를 어떻게 처리할 것인가를 생각해 보았다. 그 결과, 이 문제를 해결할 수 있는 방법은 '확률'을 이용하는 것이었으며 50번의 측정값 중 40번 정도가 물체를 찾았다고 판단될 만한 경우를 생각했다. 그렇게 우리 조는 아두이노 실습 준비를 끝마치고, 비록 1번의 실수가 있었지만 2차 시도에서 무난하게 통과함으로써 무사히 마쳤다.