



라인트레이서의 진화 더 똑똑한 자율주행을 향해

기존 라인트레이서는 정면 장애물을 인식하는 데 한계가 있습니다.

이로 인해 보행자나 장애물과의 충돌 위험이 지속됩니다.

실제 환경에서의 자율주행을 위해서는
더 정교한 인식 기술이 필요합니다.

라인트레이서+장애물 감지 사례 조사



시중 아두이노 키트는 초음파와 적외선 센서, 드라이버 모듈을 조합하여 사용합니다.

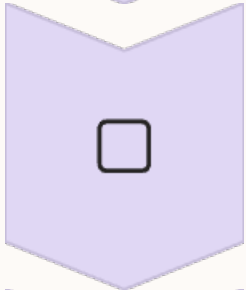
유튜브등 인터넷자료등을 참고했으며 장애물 감지 후 경고음을 울리며 즉시 정지하는 방식을 채택했습니다.

우리만의 해결방식: 경고+정지 로직



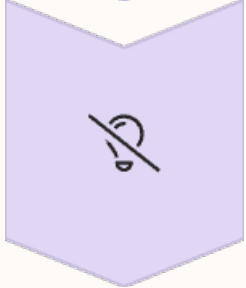
경고음 발생

장애물 감지 시 부저로 경고음을 발생시킵니다.



정지 동작

위험 상황에서 모터 브레이크로 즉시 정지합니다.



시각적 경고

LED 경고등으로 상황을 시각적으로 알립니다.

단순 회피가 아닌, 상황별로 차별화된 대응을 설계했습니다.

거리와 방향에 따라 실시간으로 판단하고 대응합니다.

아이디어 평가 및 최적안 도출

센서 조합

- 초음파 센서: 정확한 거리 측정
- IR 센서: 방향 감지
- 부저: 경고 알림

센서 배치

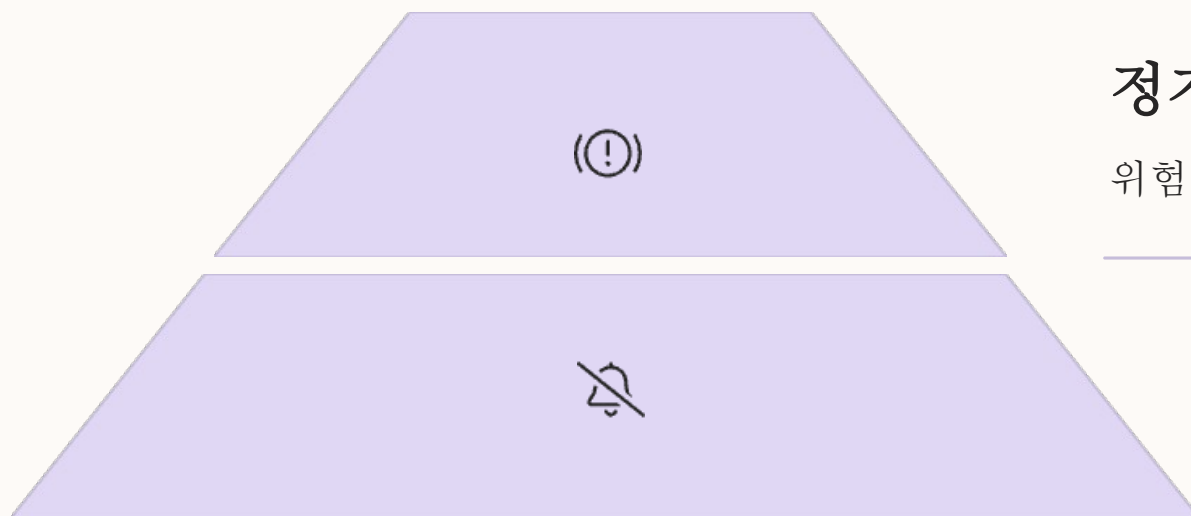
- 중앙 설치: 정면 장애물 감지
- 측면 설치: 좌우 인식 범위 확대

대응 효과

- 정지: 충돌 방지
- 회피: 자율 주행 유지
- 경고: 사용자 알림



최적 조합 선정: 시스템 설계



정지

위험 거리에서 모터 제어로 완전 정지

경고

장애물 감지 시 부저로 초기 경고

중앙에 초음파 센서, 좌우에 IR 센서와 부저를 배치했습니다.
라인 이탈 시 자동으로 감지하고 복귀하는 로직도 추가했습니다.

프로토타입 제작 및 실험



제작

아두이노 우노, TB6612FNG 모터 드라이버,
IR 센서, 초음파센서 등 조합



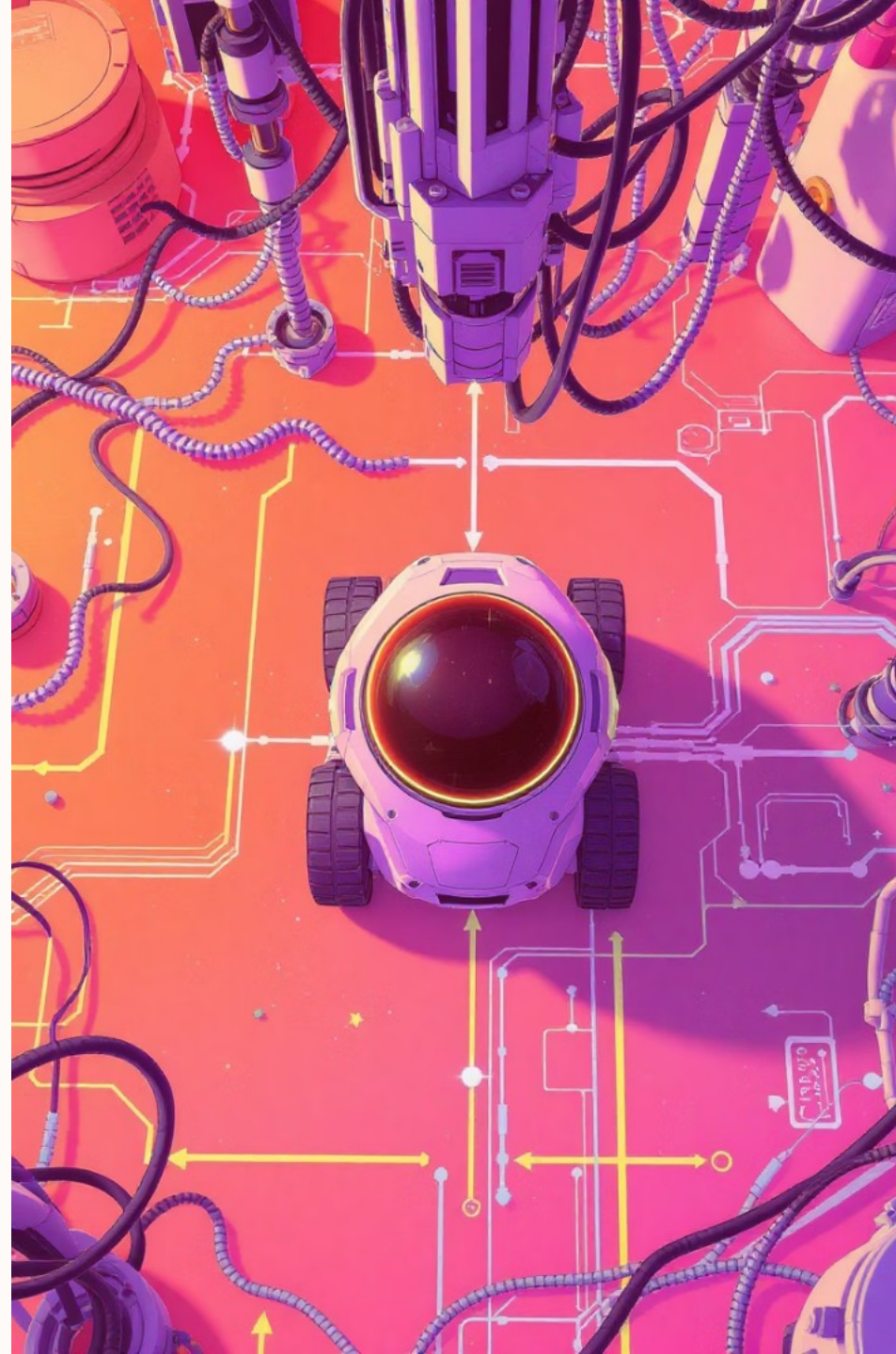
시연 및 테스트

코드의 오류를 제외하고 실제테스트의 문제 미발생



거리별 반응

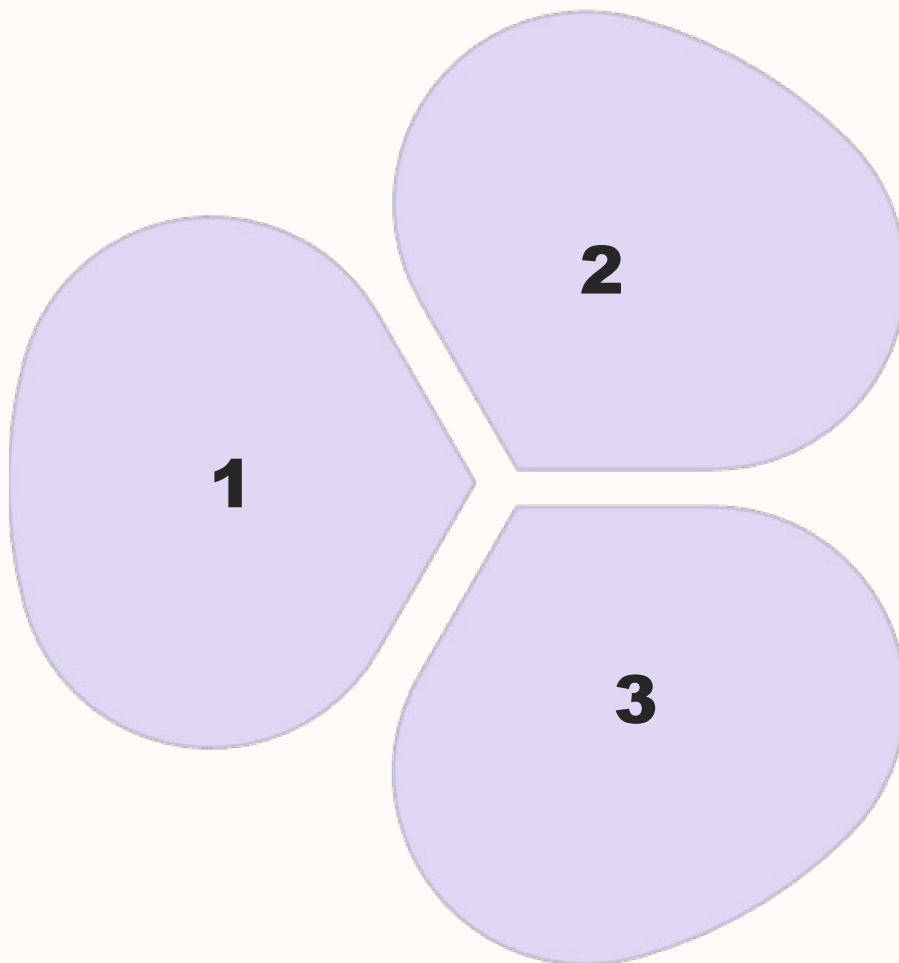
15cm/5cm 거리에서의 반응 속도 측정



확인된 한계와 문제점

센서 오작동

특정 조명이나 바닥 조건에서
IR 센서가 오작동합니다.



사각지대

특정 각도에서는 센서가 장애물을
인식하지 못합니다.

타이밍 문제

연속된 장애물이나 곡선 라인에서
정지 타이밍에 한계가 있습니다.

코드

```
#define AIN1_PIN 4 // 좌 모터 방향1
#define AIN2_PIN 3 // 좌 모터 방향2
#define PWMA_PIN 5 // 좌 모터 속도(PWM)

#define LED_WARN_PIN 7 // 거리감지 led

#define BIN1_PIN 10 // 우 모터 방향1
#define BIN2_PIN 2 // 우 모터 방향2
#define PWMB_PIN 6 // 우 모터 속도(PWM)

#define IRL_PIN A3 // 왼쪽 IR 센서
#define IRR_PIN A2 // 오른쪽 IR 센서

#define TRIG_PIN 13 // 초음파 Trig
#define ECHO_PIN A0 // 초음파 Echo
#define BUZZER_PIN 11 // 부저 tone

// 라인센서 임계값
const int thrL = 750;
const int thrR = 750;

// 속도 및 거리
const int speed = 80;
const int STOP_DIST = 5; // 즉시 정지 거리
const int WARN_DIST = 12; // 경고음 발생 거리
```

```
// 상태 변수
bool obstacleDetected = false;
bool isBeeping = false;
unsigned long lastBeepTime = 0;

void setup() {

    pinMode(LED_WARN_PIN, OUTPUT);

    pinMode(AIN1_PIN, OUTPUT);
    pinMode(AIN2_PIN, OUTPUT);
    pinMode(PWMA_PIN, OUTPUT);

    pinMode(BIN1_PIN, OUTPUT);
    pinMode(BIN2_PIN, OUTPUT);
    pinMode(PWMB_PIN, OUTPUT);

    pinMode(IRL_PIN, INPUT);
    pinMode(IRR_PIN, INPUT);

    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);

    pinMode(BUZZER_PIN, OUTPUT);

    Serial.begin(9600);
    Serial.println("STBY=HIGH");
}
```

```
// 거리 측정 (cm)
long measureDistance() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    long duration = pulseIn(ECHO_PIN, HIGH, 60000);
    if (duration == 0) return 0;
    return duration / 58;
}

// 모터 구동
void drive(int spd, bool leftFwd, bool rightFwd) {
    digitalWrite(AIN1_PIN, leftFwd ? HIGH : LOW);
    digitalWrite(AIN2_PIN, leftFwd ? LOW : HIGH);
    analogWrite(PWMA_PIN, spd);

    digitalWrite(BIN1_PIN, rightFwd ? HIGH : LOW);
    digitalWrite(BIN2_PIN, rightFwd ? LOW : HIGH);
    analogWrite(PWMB_PIN, spd);
}
```

코드

```
// 장애물 감지 처리
void handleObstacle(long dist) {
    unsigned long now = millis();

    if (dist > 0 && dist <= WARN_DIST) {
        obstacleDetected = true;
        digitalWrite(LED_WARN_PIN, HIGH); // LED ON

        int interval = map(dist, STOP_DIST, WARN_DIST, 150, 500);

        if (!isBeeping && now - lastBeepTime >= interval) {
            tone(BUZZER_PIN, 1000, 150); // 1kHz, 150ms
            isBeeping = true;
            lastBeepTime = now;
        }

        if (isBeeping && now - lastBeepTime >= 150) {
            noTone(BUZZER_PIN);
            isBeeping = false;
        }

        if (dist <= STOP_DIST) {
            drive(0, true, true); // 즉시 정지
        }
    } else {
        digitalWrite(LED_WARN_PIN, LOW); // 🚫 LED OFF
        noTone(BUZZER_PIN);
        isBeeping = false;
        obstacleDetected = false;
    }
}
```

```
void loop() {
    // 1. 거리 측정
    long dist = measureDistance();

    // 2. 장애물 감지 및 경고 처리
    handleObstacle(dist);

    // 3. 라인트레이싱 (장애물 없을 때만 동작)
    int vL = analogRead(IRL_PIN);
    int vR = analogRead(IRR_PIN);
    bool onL = (vL > thrL);
    bool onR = (vR > thrR);

    if (!obstacleDetected) {
        if (onL && onR) drive(speed, true, true); // 직진
        else if (onL)   drive(speed, false, true); // 좌회전
        else if (onR)   drive(speed, true, false); // 우회전
        else             drive(0, true, true); // 정지
    }

    // 4. 디버깅 출력
    Serial.print("vL="); Serial.print(vL);
    Serial.print(" vR="); Serial.print(vR);
    Serial.print(" | L="); Serial.print(onL);
    Serial.print(" R="); Serial.print(onR);
    Serial.print(" | Dist="); Serial.print(dist);
    Serial.print("cm | Obstacle="); Serial.println(obstacleDetected);

    delay(10);
}
```



시스템 개선 및 추가 아이디어

1 IR 감도 보정

다양한 환경에서도 안정적으로 작동하도록 센서 감도를 보정합니다

2 Millis 함수 추가

LiDAR 센서를 추가하면 인식률이 30% 이상 향상될 것으로 기대됩니다.



결론 및 시사점



안전성 강화

똑똑한 라인트레이서는 충돌 위험을 최소화합니다.



실용성 향상

차별화된 경고+정지+복귀 로직으로 실환경 적용 가능성이 높아집니다.



지속적 개선

현장 실험과 피드백을 통해 계속 발전시켜 나갑니다.