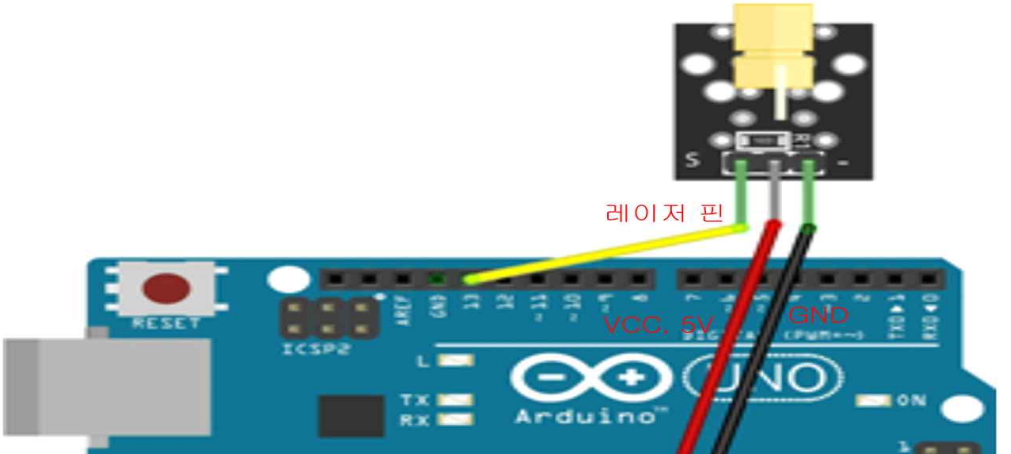
레이저 센서와 타겟추적

설계도



레이저 코드

```
int laser=13;

void setup() {
  // put your setup code here, to run once:
pinMode(laser, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(laser, HIGH);
delay(1000);
digitalWrite(laser, LOW);
delay(1000);
}
```

레이저와 자동 타겟 코드(1)

```
#include <Servo.h>
const int trigPin = 9;
const int echoPin = 10;
const int laser = 13;
const int servoPin = 6;
Servo myServo;
long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(laser, OUTPUT);
  myServo.attach(servoPin);
  Serial.begin(9600);
}
```

레이저와 자동 타겟 코드(1)

```
void loop() {
 // 서보 모터가 0도부터 180도까지 회전하면서 거리 측정
 for (int angle = 0; angle <= 180; angle += 5) {
  myServo.write(angle);
  delay(200); // 모터가 움직일 시간 줌
  // 초음파 거리 측정
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // cm
  Serial.print("Angle: ");
  Serial.print(angle);
  Serial.print(" - Distance: ");
  Serial.println(distance);
```

레이저와 자동 타겟 코드(1)

```
if (distance > 0 && distance < 10) { // 30cm 이내에 물체가 있으면
    digitalWrite(laser, HIGH);
    delay(3000); // 레이저 3초간 ON
    digitalWrite(laser, LOW);
   }
  }
}
```

레이저와 자동 목표물 추적 코드

```
#include <Servo.h>

const int trigPin = 9;
const int echoPin = 10;
const int laser = 13;
const int servoPin = 6;

Servo myServo;

long duration;
int distance;
int currentAngle = 90;
const int initialAngle = 90;
bool targetFound = false;
```

레이저와 자동 목표물 추적 코드

```
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(laser, OUTPUT);

  myServo.attach(servoPin);
  myServo.write(currentAngle);
  Serial.begin(9600);
}
```

```
void loop() {
  if (!targetFound) {
    // 탐색 모드
    int detectedAngle = scanForTarget();
    if (detectedAngle != -1) {
      smoothMove(currentAngle, detectedAngle);
      currentAngle = detectedAngle;
      targetFound = true;
      digitalWrite(laser, HIGH);
      Serial.println("Target acquired. Tracking...");
    } else {
      if (currentAngle != initialAngle) {
        smoothMove(currentAngle, initialAngle);
        currentAngle = initialAngle;
      }
      digitalWrite(laser, LOW);
      Serial.println("No target found. Scanning...");
    }
```

레이저와 자동 목표물 추적 코드

```
} else {
  // 추적 모드
  int bestAngle = trackTargetAround(currentAngle);
  if (bestAngle != -1) {
    smoothMove(currentAngle, bestAngle);
    currentAngle = bestAngle;
    digitalWrite(laser, HIGH);
    Serial.print("Tracking... New angle: ");
    Serial.println(currentAngle);
  } else {
    // 타겟 놓침
    targetFound = false;
    digitalWrite(laser, LOW);
    Serial.println("Lost target. Switching to scan.");
  }
 }
 delay(100);
}
```

```
// 전체 범위에서 타겟을 탐색
int scanForTarget() {
 int detectedAngle = -1;
 int minDistance = 1000;

 for (int angle = 0; angle <= 180; angle += 5) {
  myServo.write(angle);
  delay(50);

  int dist = measureDistance();
  if (dist > 0 && dist < 10 && dist < minDistance)
{
    minDistance = dist;
    detectedAngle = angle;
  }
 }

 return detectedAngle;
}
```

레이저와 자동 목표물 추적 코드

```
// 현재 각도 주변에서만 추적
int trackTargetAround(int centerAngle) {
  int detectedAngle = -1;
  int minDistance = 1000;
  int range = 15; // ±15도 범위로 타겟 탐색
  int startAngle = max(0, centerAngle - range);
  int endAngle = min(180, centerAngle + range);
  for (int angle = startAngle; angle <= endAngle; angle += 2) {
    myServo.write(angle);
    delay(30);

    int dist = measureDistance();
    if (dist > 0 && dist < 10 && dist < minDistance) {
      minDistance = dist;
      detectedAngle = angle;
    }
  }

  return detectedAngle;
}

// 거리 측정 함수
int measureDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH, 20000);
  return duration * 0.034 / 2;
}
```

레이저와 자동 목표물 추적 코드

```
 // 서보 부드럽게 이동
void smoothMove(int fromAngle, int toAngle) {
  if (fromAngle == toAngle) return;

  float angle = fromAngle;
  float step = 0.5;      // 더 작게 움직이기 → 부
드러움 향상
  int delayTime = 15;    // 느긋하게 움직이기

  while (abs(angle - toAngle) >= step) {
    angle += (toAngle > angle) ? step : -step;
    myServo.write((int)angle);
    delay(delayTime);
  }

  myServo.write(toAngle); // 최종 각도 정확히
}
```

수고하셨습니다