0225 회의록 및 활동 기록

⚫ 자율주행 코드

▼ 정상 소스 코드

```
#include <SoftwareSerial.h>
#include <AFMotor.h>
AF_DCMotor motor_L(1);
AF_DCMotor motor_R(4);

int Lspeed = 170;            //좌측 모터 속도
int Rspeed = 200;            //우측 모터 속도

//초음파센서 출력핀(trig)과 입력핀(echo), 변수, 함수 선언//
int TrigPin = A0;
int EchoPin = A1;
long duration, distance;

void Obstacle_Check();
void Distance_Measurement();
void Forward();
void Backward();
void Right();
void Left();
void Stop();

void setup() {
  Serial.begin(9600);              // PC와의 시리얼 통신속도
  Serial.println("Eduino Smart Car Start!");

  pinMode(EchoPin, INPUT);   // EchoPin 입력
  pinMode(TrigPin, OUTPUT);  // TrigPin 출력

  motor_L.setSpeed(Lspeed);              // 왼쪽 모터의 속도
  motor_L.run(RELEASE);
  motor_R.setSpeed(Rspeed);              // 오른쪽 모터의 속도
  motor_R.run(RELEASE);
}
```

```
void loop() {
  Forward();
  delay(50);
  Obstacle_Check();
}

//////////장애물 확인 및 회피 방향 결정//////////
void Obstacle_Check() {
  int val = random(2);
  Distance_Measurement();
  delay(50);

  Serial.println(distance);

  while (distance < 300) {
    if (distance < 150) {
      Backward();
      delay(800);
      Stop();
      delay(50);
      Distance_Measurement();
      delay(100);
    }
    else {
      if (val == 0) {
        Right();
        delay(400);
      }
      else if (val == 1) {
        Left();
        delay(400);
      }
      Distance_Measurement();
      delay(100);
    }
  }
}


////////거리감지//////////
void Distance_Measurement() {
  digitalWrite(TrigPin, LOW);
  delay(2);
  digitalWrite(TrigPin, HIGH);  // trigPin에서 초음파 발생(echoPin도 HIGH)
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  duration = pulseIn(EchoPin, HIGH);    // echoPin 이 HIGH를 유지한 시간을 저장 한다.
  distance = ((float)(340 * duration) / 1000) / 2;
  delay(50);
}


//////////모터 제어 함수//////////
void Forward() {
  motor_L.run(FORWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}
```

```
void Backward() {
  motor_L.run(BACKWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Right() {
  motor_L.run(FORWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed*0.5);
}

void Left() {
  motor_L.run(BACKWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed*0.5);  motor_R.setSpeed(Rspeed);
}

void Stop() {
  motor_L.run(RELEASE);       motor_R.run(RELEASE);
  motor_L.setSpeed(0);  motor_R.setSpeed(0);
}
```

▼ 코드 개요

위 자율주행 코드는 울트라소닉 센서를 사용하여 장애물을 감지하고 회피하는 아두이노 기반의 스마트 카를 구현한 것으로, 스마트 카는 전진하며, 장애물이 감지될 때까지 이동한다. 장애물이 감지되면 스마트 카는 임의로 왼쪽 또는 오른쪽으로 회피하고, 물체와의 거리가 150 이하일 때는 장애물을 피하기 위해 800ms간 후진한다.

본 코드는 AFMotor 라이브러리를 사용하여 스마트 카의 모터를 제어하며, SoftwareSerial 라이브러리를 사용하여 컴퓨터와의 시리얼 통신을 설정하였다. 초음파 센서에 사용되는 핀은 코드의 맨 앞부분에서 A0과 A1로 정의된다. 장애물과의 거리는 pulseIn 함수를 사용하여 측정하며, 왼쪽 및 오른쪽 모터 속도에 대한 값은 각각 Lspeed 및 Rspeed로 정의하였다.

코드의 메인 루프는 Forward() 및 Obstacle_Check() 함수로 구성된다. Forward() 함수는 왼쪽 및 오른쪽 모터를 지정된 속도로 전진하도록 설정하였다. Obstacle_Check() 함수는 Distance_Measurement() 함수를 사용하여 장애물과의 거리를 확인하고, 거리가 300 이하인 경우 while 루프로 들어간다. 만일 물체와의 거리가 150 이하이면 Backward() 함수를 사용하여 스마트 카가 800ms간 후진하도록 설정하였다. 반면 거리가 150에서 300 사이인 경우, 스마트 카는 Left() 및 Right() 함수를 사용하여 임의로 왼쪽 또는 오른쪽으로 회전한다. 이때 동작 간 딜레이 시간은 400ms로 설정되며, 스마트 카는 다시 Distance_Measurement() 함수를 사용하여 거리를 다시 확인한다.

코드에는 또한 스마트 카를 후진, 오른쪽 또는 왼쪽으로 회전하도록 하는 함수 및 모터를 정지하는 함수가 포함되어 있는데, 이러한 함수들은 Obstacle_Check() 함수에서 필요할 때 호출된다.

▼ **코드 취약점**

이 코드는 Eduino Smart Car를 제어하기 위한 아두이노 코드로, 초음파 센서를 사용하여 차량의 전방에 장애물이 있을 경우 장애물을 회피하도록 구현되었다.

하지만 이 코드에는 사용자 인증이나 암호화 기능이 없기 때문에, 누구나 이 코드를 업로드하여 차량을 제어할 수 있다. 따라서 악의적인 사용자가 이 코드를 이용하여 차량을 제어하거나 센서 값을 조작할 가능성이 있다.

1. **무작위 방향 변경:** 코드는 장애물을 만날 때 로봇의 방향을 무작위로 변경한다. 이로 인해 예측할 수 없는 동작을 일으키고, 로봇이 다른 장애물과 충돌할 가능성이 존재한다. 따라서 장애물의 위치와 거리에 따라 로봇이 이동할 수 있는 최적의 방향을 결정하기 위해 보다 견고한 알고리즘을 사용할 필요가 있다.
2. **장애물 메모리 없음**: 위의 코드는 이전에 만난 장애물의 위치를 기억하지 않는다. 이로 인해 로봇이 같은 방향으로 계속 회전하고, 주변 환경을 효과적으로 탐색하지 못할 수 있다. 따라서 보다 정교한 접근 방식을 사용할 필요가 있다.
3. **보정 없음:** `Distance_Measurement()` 함수에서 거리 계산은 고정된 음속을 가정하는데, 이는 모든 조건에서 정확하지 않을 수 있다. 또한 이 함수는 센서 읽기의 보정이나 유효성 검사를 수행하지 않는데, 이로 인해 거리가 부정확하게 측정되거나 장애물 회피 작업이 원활하게 이루어지지 않을 수 있다.
4. **오류 처리 없음:** 본 코드는 오류 처리나 장애 허용 메커니즘을 포함하지 않아 오류가 발생할 경우 IoT기기의 대처가 불가하다. 예를 들어 초음파 센서가 작동하지 않거나 응답을 멈춘 경우, 차량은 장애물을 감지할 수 없으며 이로 인해 충돌할 가능성이 있다.
5. **제한된 센서 범위:** 위의 코드는 제한된 범위(300cm) 내의 장애물만 검사한다. 이 검사 범위는 모든 환경에 대해 충분하지 않을 수 있으며, 더 멀리 위치한 장애물과 충돌할 위험성이 있다.

🔴 공격용 악성 코드

▼ **Code 1**_거리가 0일때 후진하도록 조작한 소스 코드 ⇒ 장애물 인지 기능을 무력화하여 장애물과 충돌하도록 조작

```
#include <SoftwareSerial.h>
#include <AFMotor.h>
AF_DCMotor motor_L(1);
AF_DCMotor motor_R(4);

int Lspeed = 170;            //좌측 모터 속도
int Rspeed = 200;            //우측 모터 속도

//초음파센서 출력핀(trig)과 입력핀(echo), 변수, 함수 선언//
int TrigPin = A0;
int EchoPin = A1;
long duration, distance;

void Obstacle_Check();
void Distance_Measurement();
void Forward();
void Backward();
void Right();
void Left();
void Stop();

void setup() {
  Serial.begin(9600);              // PC와의 시리얼 통신속도
  Serial.println("Eduino Smart Car Start!");

  pinMode(EchoPin, INPUT);   // EchoPin 입력
  pinMode(TrigPin, OUTPUT);  // TrigPin 출력

  motor_L.setSpeed(Lspeed);              // 왼쪽 모터의 속도
  motor_L.run(RELEASE);
  motor_R.setSpeed(Rspeed);              // 오른쪽 모터의 속도
  motor_R.run(RELEASE);
}

void loop() {
  Forward();
  delay(50);
  Obstacle_Check();
}

///////////장애물 확인 및 회피 방향 결정//////////
void Obstacle_Check() {
  int val = random(2);
  Distance_Measurement();
  delay(50);

  Serial.println(distance);

  while (distance <= 0) {   //요 부분 수정
    Backward();
    delay(800);
    Stop();
    delay(50);
    Distance_Measurement();
    delay(100);
  }
```

```
  while (distance < 300) {
    if (distance < 150) {
      Backward();
      delay(800);
      Stop();
      delay(50);
      Distance_Measurement();
      delay(100);
    }
    else {
      if (val == 0) {
        Right();
        delay(400);
      }
      else if (val == 1) {
        Left();
        delay(400);
      }
      Distance_Measurement();
      delay(100);
    }
  }
}

void Distance_Measurement() {
  digitalWrite(TrigPin, LOW);
  delay(2);
  digitalWrite(TrigPin, HIGH);  // trigPin에서 초음파 발생(echoPin도 HIGH)
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  duration = pulseIn(EchoPin, HIGH);    // echoPin 이 HIGH를 유지한 시간을 저장 한다.
  distance = ((float)(340 * duration) / 1000) / 2;
  delay(50);
}

void Forward() {
  motor_L.run(FORWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Backward() {
  motor_L.run(BACKWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Right() {
  motor_L.run(FORWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed*0.5);
}

void Left() {
  motor_L.run(BACKWARD);
```

▼ **Code 2**_Distance_Measurement() 함수(=장애물 거리 측정 함수)의 반환값을 항상 400이 되도록 조작한 소스 코드 ⇒ 장애물과의 실제 거리를 반영하지 못하게 함으로써 장

애물과 충돌하도록 조작

```
#include <SoftwareSerial.h>
#include <AFMotor.h>
AF_DCMotor motor_L(1);
AF_DCMotor motor_R(4);

int Lspeed = 170;            //좌측 모터 속도
int Rspeed = 200;            //우측 모터 속도

//초음파센서 출력핀(trig)과 입력핀(echo), 변수, 함수 선언//
int TrigPin = A0;
int EchoPin = A1;
long duration, distance;

void Obstacle_Check();
void Distance_Measurement();
void Forward();
void Backward();
void Right();
void Left();
void Stop();

void setup() {
  Serial.begin(9600);              // PC와의 시리얼 통신속도
  Serial.println("Eduino Smart Car Start!");

  pinMode(EchoPin, INPUT);   // EchoPin 입력
  pinMode(TrigPin, OUTPUT);  // TrigPin 출력

  motor_L.setSpeed(Lspeed);              // 왼쪽 모터의 속도
  motor_L.run(RELEASE);
  motor_R.setSpeed(Rspeed);              // 오른쪽 모터의 속도
  motor_R.run(RELEASE);
}

void loop() {
  Forward();
  delay(50);
  Obstacle_Check();
}

///////////장애물 확인 및 회피 방향 결정///////////
void Obstacle_Check() {
  int val = random(2);
  Distance_Measurement();
  delay(50);

  Serial.println(distance);

  while (distance < 300) {
    if (distance < 150) {
      Backward();
      delay(800);
      Stop();
      delay(50);
```

```
      Distance_Measurement();
      delay(100);
    }
    else {
      if (val == 0) {
        Right();
        delay(400);
      }
      else if (val == 1) {
        Left();
        delay(400);
      }
      Distance_Measurement();
      delay(100);
    }
  }
}


////////거리감지//////////
void Distance_Measurement() {
  //digitalWrite(TrigPin, LOW);
  //delay(2);
  //digitalWrite(TrigPin, HIGH);  // trigPin에서 초음파 발생(echoPin도 HIGH)
  //delayMicroseconds(10);
  //digitalWrite(TrigPin, LOW);
  //duration = pulseIn(EchoPin, HIGH);    // echoPin 이 HIGH를 유지한 시간을 저장 한다.
  //distance = ((float)(340 * duration) / 1000) / 2;
  //delay(50);
  distance = 400; //요 부분 수정
}


//////////모터 제어 함수//////////
void Forward() {
  motor_L.run(FORWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Backward() {
  motor_L.run(BACKWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Right() {
  motor_L.run(FORWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed*0.5);
}

void Left() {
  motor_L.run(BACKWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed*0.5);  motor_R.setSpeed(Rspeed);
}

void Stop() {
  motor_L.run(RELEASE);       motor_R.run(RELEASE);
  motor_L.setSpeed(0);  motor_R.setSpeed(0);
}
```

⚫ 취약점 보완 코드

▼ **취약점 보완 코드 1**_장애물을 만날 때 로봇의 방향 번경 전 앞에 장애물이 있는지를 검사하는 과정 추가

—> 장애물이 있는 경우 장애물을 피할 방향을 결정하고, 장애물이 없는 방향으로 변경하여 이동한다. 이를 위해 `Left()`와 `Right()`함수를 이용하여 로봇을 각 방향으로 회전시키고, `Distance_Measurement()`함수를 이용하여 회전한 방향으로부터의 거리를 측정한다. 이후 거리가 더 긴 방향으로 회전하여 이동한다.

```
#include <SoftwareSerial.h>
#include <AFMotor.h>
AF_DCMotor motor_L(1);
AF_DCMotor motor_R(4);

int Lspeed = 170;            //좌측 모터 속도
int Rspeed = 200;            //우측 모터 속도

//초음파센서 출력핀(trig)과 입력핀(echo), 변수, 함수 선언//
int TrigPin = A0;
int EchoPin = A1;
long duration, distance;

void Obstacle_Check();
void Distance_Measurement();
void Forward();
void Backward();
void Right();
void Left();
void Stop();

void setup() {
  Serial.begin(9600);              // PC와의 시리얼 통신속도
  Serial.println("Eduino Smart Car Start!");

  pinMode(EchoPin, INPUT);   // EchoPin 입력
  pinMode(TrigPin, OUTPUT);  // TrigPin 출력

  motor_L.setSpeed(Lspeed);              // 왼쪽 모터의 속도
  motor_L.run(RELEASE);
  motor_R.setSpeed(Rspeed);              // 오른쪽 모터의 속도
  motor_R.run(RELEASE);
}

void loop() {
  Forward();
  delay(50);
  Obstacle_Check();
}
```

```
//////////장애물 확인 및 회피 방향 결정//////////
void Obstacle_Check() {
  int val = random(2);
  Distance_Measurement();
  delay(50);

  Serial.println(distance);

  while (distance < 300) {
    // 장애물이 있을 경우
    if (distance < 150) {
      Backward();
      delay(800);
      Stop();
      delay(50);
      Distance_Measurement();
      delay(100);

      // 장애물을 피할 방향 결정
      if (val == 0) {
        Right();
        delay(400);
      }
      else if (val == 1) {
        Left();
        delay(400);
      }

      // 변경된 방향으로 이동
      Distance_Measurement();
      delay(100);
      continue;
    }

    // 장애물이 없는 경우
    Forward();
    Distance_Measurement();
    delay(100);

    // 변경할 방향 검사
    int left_distance, right_distance;
    Left();
    delay(300);
    Distance_Measurement();
    left_distance = distance;
    Stop();
    delay(50);

    Right();
    delay(300);
    Distance_Measurement();
    right_distance = distance;
    Stop();
    delay(50);

    // 장애물이 없는 방향으로 변경
    if (left_distance > right_distance) {
```

```
      Left();
      delay(400);
    }
    else {
      Right();
      delay(400);
    }

    Distance_Measurement();
    delay(100);
  }
}


////////거리감지//////////
void Distance_Measurement() {
  digitalWrite(TrigPin, LOW);
  delay(2);
  digitalWrite(TrigPin, HIGH);  // trigPin에서 초음파 발생(echoPin도 HIGH)
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  duration = pulseIn(EchoPin, HIGH);    // echoPin 이 HIGH를 유지한 시간을 저장 한다.
  distance = ((float)(340 * duration) / 1000) / 2;
  delay(50);
}


//////////모터 제어 함수//////////
void Forward() {
  motor_L.run(FORWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Backward() {
  motor_L.run(BACKWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Right() {
  motor_L.run(FORWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed*0.5);
}

void Left() {
  motor_L.run(BACKWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed*0.5);  motor_R.setSpeed(Rspeed);
}

void Stop() {
  motor_L.run(RELEASE);       motor_R.run(RELEASE);
  motor_L.setSpeed(0);  motor_R.setSpeed(0);
}
```

▼ **취약점 보완 코드 2**_;보안 코드 1’에 이전에 만난 장애물의 위치를 기억하게 하는 코드 추가

—> `prev_distance`라는 전역 변수를 추가하고, `Obstacle_Check()` 함수에서 현재 거리(`distance`)와 이전 거리(`prev_distance`)를 비교하여, 이전에 만난 장애물의 위치를 기억하게 하는 코드를 추가하였다.

```
#include <SoftwareSerial.h>
#include <AFMotor.h>

AF_DCMotor motor_L(1);
AF_DCMotor motor_R(4);

int Lspeed = 170;            //좌측 모터 속도
int Rspeed = 200;            //우측 모터 속도

//초음파센서 출력핀(trig)과 입력핀(echo), 변수, 함수 선언//
int TrigPin = A0;
int EchoPin = A1;
long duration, distance, prev_distance;

void Obstacle_Check();
void Distance_Measurement();
void Forward();
void Backward();
void Right();
void Left();
void Stop();

void setup() {
  Serial.begin(9600);              // PC와의 시리얼 통신속도
  Serial.println("Eduino Smart Car Start!");

  pinMode(EchoPin, INPUT);   // EchoPin 입력
  pinMode(TrigPin, OUTPUT);  // TrigPin 출력

  motor_L.setSpeed(Lspeed);              // 왼쪽 모터의 속도
  motor_L.run(RELEASE);
  motor_R.setSpeed(Rspeed);              // 오른쪽 모터의 속도
  motor_R.run(RELEASE);
}

void loop() {
  Forward();
  delay(50);
  Obstacle_Check();
}

///////////장애물 확인 및 회피 방향 결정///////////
void Obstacle_Check() {
  int val = random(2);
  Distance_Measurement();
  delay(50);

  Serial.println(distance);

  while (distance < 300) {
    // 장애물이 있을 경우
```

```
    if (distance < 150) {
      Backward();
      delay(800);
      Stop();
      delay(50);
      Distance_Measurement();
      delay(100);

      // 이전에 만난 장애물의 위치를 저장
      prev_distance = distance;

      // 장애물을 피할 방향 결정
      if (val == 0) {
        Right();
        delay(400);
      }
      else if (val == 1) {
        Left();
        delay(400);
      }

      // 변경된 방향으로 이동
      Distance_Measurement();
      delay(100);
      continue;
    }

    // 장애물이 없는 경우
    Forward();
    Distance_Measurement();
    delay(100);

    // 변경할 방향 검사
    int left_distance, right_distance;
    Left();
    delay(300);
    Distance_Measurement();
    left_distance = distance;
    Stop();
    delay(50);

    Right();
    delay(300);
    Distance_Measurement();
    right_distance = distance;
    Stop();
    delay(50);

    // 장애물이 없는 방향으로 변경
    if (left_distance > right_distance) {
      Left();
      delay(400);
    }
    else {
      Right();
      delay(400);
    }
```

```
    Distance_Measurement();
    delay(100);
  }
}

////////거리감지//////////
void Distance_Measurement() {
  digitalWrite(TrigPin, LOW);
  delay(2);
  digitalWrite(TrigPin, HIGH);  // trigPin에서 초음파 발생(echoPin도 HIGH)
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  duration = pulseIn(EchoPin, HIGH);    // echoPin 이 HIGH를 유지한 시간을 저장 한다.
  distance = ((float)(340 * duration) / 1000) / 2;
  delay(50);
}


//////////모터 제어 함수//////////
void Forward() {
  motor_L.run(FORWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Backward() {
  motor_L.run(BACKWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed);
}

void Right() {
  motor_L.run(FORWARD);  motor_R.run(BACKWARD);
  motor_L.setSpeed(Lspeed);  motor_R.setSpeed(Rspeed*0.5);
}

void Left() {
  motor_L.run(BACKWARD);  motor_R.run(FORWARD);
  motor_L.setSpeed(Lspeed*0.5);  motor_R.setSpeed(Rspeed);
}

void Stop() {
  motor_L.run(RELEASE);       motor_R.run(RELEASE);
  motor_L.setSpeed(0);  motor_R.setSpeed(0);
}
```