전쟁중인 저개발국가 위험구역 개척(자폭) 로봇

1. 프로젝트 개요

개척(자폭) 로봇은 현대 개발도상국이 전쟁 중인 지역이 많다는 것을 고려해 그들에게 현실적으로 도움을 줄 수 있는 방법을 고안해 만들어낸 임베디드 시스템입니다. 개발 도상국이라는 위치에 있는 현실적인 요건들을 최대한 고려해서 효율성 높은 시스템을 만드는 것을 가장 큰 목적으로 정했습니다. 때문에 다양한 기능이 달린 부품이 아닌 목적을 가장 충실하게 수행해 낼 수 있는 최소한의 부품인, 임베디드, 서브모터, 바퀴, 적외선 센서, 몇 개의 전선과 블루투스 모듈하고 소량의 폭약만이 탑재됩니다.

2. 개발 배경 및 필요성

전쟁 중인 국가에서는 살상을 위해 대량의 폭약이 탑재되는 폭발형 무기를 사용합니다. 그 무기들이 이상 없이 본래의 소기 목적을 달성한다면 민간인들에게는 전쟁으로 인한 육체적 간접 피해가 상당히 줄었을 겁니다. 하지만 전쟁은 정밀함을 추구하는 것보다 빠르고 대량으로 무기를 양산하여, 물량으로 밀어붙이는 전술이 현대전에 가장 좋은 전술로 채택되고 있습니다. 때문에 불량품인 살상 무기들이 민간 지역까지 흘러가 크나큰 피해를 남기고 있습니다. 이런 피해를 줄이고자 저는 위험 지역 개척(자폭) 로봇이 필요하다 생각하였습니다.

3. 시스템 아키텍처

ESP32 (임베디드 보드)
역할:
1. 스마트폰으로부터 블루투스 명령 수신(전진, 후진, 좌/우, 무장, 기폭)
2. 적외선 센서 데이터 처리(장애물 감지)
3. 모터 드라이버에 신호를 보내 바퀴 구동

4. 페이로드(폭약) 시스템 제어(안전장치 해제, 기폭 신호 전송)

DC 기어 모터 4개 + 바퀴 4개
L298N 모터 드라이버

역할:
ESP32의 명령을 받아 모터를 정방향/역방향으로 구동합니다.
두 바퀴의 속도를 다르게 하여 방향을 전환합니다.

적외선(IR) 근접 센서

역할:
로봇 전방의 단순 장애물 감지에 사용됩니다.

HC-06

역할:
운영자의 스마트폰과 로봇간의 무선 연결을 담당합니다.
스마틒노에서 MIT 앱 인벤터 등으로 만든 간단한 앱이나,
플레이스토어의 블루투스 시리얼 터미널 앱을 통해 명령을 전송합니다.

기폭 시스템(개별 장착)

4. 핵심 기능

1. 4륜 구동 원격 조종 기능
- 스마트폰 앱을 통해 로봇을 전/후/좌/우로 실시간 조종.
- 4개의 모터를 이용해 비포장지나 작은 장애물에 대한 주파 능력 확보

2. 전방 장애물 자동 정지 기능
- IR 센서가 전방 일정 거리 내의 장애물을 감지

- 조종 실수로 인한 충동을 방지하기 위해 자동으로 모터를 정지시키고 운영자에게 경고 알림.

5. 프로토타입 개발 일정(총 4주)

1주 차: 계획 및 부품 확보
2주 차: 하드웨어 조립 및 기본 구동
3주 차: 핵심 기능 프로그래밍
4주 차: 통합 테스트 및 디버깅

6. 코드

Arduino IDE

```
#include <Servo.h>

Servo servoLeft;
Servo servoRight;

char command = 'S'; // 기본 명령: 정지

void setup() {
  tone(4, 3000, 1000);
  delay(1000);
  Serial.begin(9600);

  servoLeft.attach(13);
  servoRight.attach(12);

  stopMotors();
```

```
}

void loop() {
  // 시리얼 통신으로 명령 수신
  if (Serial.available() > 0) {
    command = Serial.read();
    executeCommand(command);
  }
}

void executeCommand(char cmd) {
  switch(cmd) {
    case 'F': // Forward (전진)
      moveForward();
      break;
    case 'B': // Backward (후진)
      moveBackward();
      break;
    case 'L': // Left (왼쪽 회전)
      turnLeft();
      break;
    case 'R': // Right (오른쪽 회전)
      turnRight();
      break;
    case 'S': // Stop (정지)
      stopMotors();
      break;
    default:
      stopMotors();
      break;
  }
```

```
}

void moveForward() {
  servoLeft.writeMicroseconds(1700);    // 왼쪽 서보 반시계방향
  servoRight.writeMicroseconds(1300);   // 오른쪽 서보 시계방향
}

void moveBackward() {
  servoLeft.writeMicroseconds(1300);    // 왼쪽 서보 시계방향
  servoRight.writeMicroseconds(1700);   // 오른쪽 서보 반시계방향
}

void turnLeft() {
  servoLeft.writeMicroseconds(1300);    // 왼쪽 서보 시계방향
  servoRight.writeMicroseconds(1300);   // 오른쪽 서보 시계방향
}

void turnRight() {
  servoLeft.writeMicroseconds(1700);    // 왼쪽 서보 반시계방향
  servoRight.writeMicroseconds(1700);   // 오른쪽 서보 반시계방향
}

void stopMotors() {
  servoLeft.writeMicroseconds(1500);    // 정지
  servoRight.writeMicroseconds(1500);   // 정지
}
```

Processing

```
import processing.serial.*;
```

```
Serial arduinoPort;
Serial bluetoothPort;

String arduinoPortName = "COM3";  // Arduino 포트 (Mac: /dev/cu.usbserial)
String bluetoothPortName = "COM4"; // 블루투스 포트 (Mac: /dev/cu.HC-05)

void setup() {
  size(400, 300);

  // 사용 가능한 포트 출력
  println("사용 가능한 포트:");
  printArray(Serial.list());

  // Arduino 연결
  try {
    arduinoPort = new Serial(this, arduinoPortName, 9600);
    println("Arduino 연결 성공: " + arduinoPortName);
  } catch (Exception e) {
    println("Arduino 연결 실패: " + e.getMessage());
  }

  // 블루투스 연결
  try {
    bluetoothPort = new Serial(this, bluetoothPortName, 9600);
    println("블루투스 연결 성공: " + bluetoothPortName);
  } catch (Exception e) {
    println("블루투스 연결 실패: " + e.getMessage());
  }
```

```
  background(200);
  textAlign(CENTER, CENTER);
  textSize(16);
}

void draw() {
  background(200);
  fill(0);
  text("로봇 제어 브릿지", width/2, 50);
  text("상태: 실행 중", width/2, 100);

  // 블루투스에서 데이터 수신
  if (bluetoothPort != null && bluetoothPort.available() > 0) {
    char command = bluetoothPort.readChar();
    println("받은 명령: " + command);

    // Arduino로 명령 전송
    if (arduinoPort != null) {
      arduinoPort.write(command);
      displayCommand(command);
    }
  }

  // 키보드로 테스트
  if (keyPressed) {
    char cmd = getCommandFromKey(key);
    if (cmd != 'X' && arduinoPort != null) {
      arduinoPort.write(cmd);
      displayCommand(cmd);
    }
```

```
  }
}

void displayCommand(char cmd) {
  fill(0, 150, 0);
  String cmdText = "";
  switch(cmd) {
    case 'F': cmdText = "전진 ▲"; break;
    case 'B': cmdText = "후진 ▼"; break;
    case 'L': cmdText = "좌회전 ◄"; break;
    case 'R': cmdText = "우회전 ►"; break;
    case 'S': cmdText = "정지 ■"; break;
  }
  text("현재 명령: " + cmdText, width/2, 150);
}

char getCommandFromKey(char k) {
  switch(k) {
    case 'w': case 'W': return 'F';
    case 's': case 'S': return 'B';
    case 'a': case 'A': return 'L';
    case 'd': case 'D': return 'R';
    case ' ': return 'S';
    default: return 'X';
  }
}

void keyReleased() {
  // 키를 떼면 정지
  if (arduinoPort != null) {
    arduinoPort.write('S');
```

```
    }
}
```