|  |
| --- |
| **(이스케이프 파이어 룸) 개발보고서** |

이름 : 김진희, 양나영, 이예빈

|  |  |  |
| --- | --- | --- |
| **1** |  | **개발 목표와 특징** |

□ 각 팀에서 개발하고자 하는 제안시스템의 특징을 반영한 개발목표를 기술

[목표]

: 화재 발생 시 빠른 대피 안내와 생존자를 빠르게 대피시키자!

[시스템 특징]

1. 생존자 구조 로봇
   1. 화재 발생 시 투입되어 생존자를 찾는다.
   2. 찾은 후 대피로 이동 및 안내한다.
   3. 안내하며 표식 (LED)을 남긴다.
   4. 온습도 및 가스 농도, 대피 안내 시각화
2. 화재 대피 안내 로봇
   1. 화재 발생 감지
   2. 화재 감지 시 사이렌과 안내방송 시작
   3. 정해진 구간에서 안내 방송 무한 재생

|  |  |  |
| --- | --- | --- |
| **2** |  | **개발 필요성** |

**□** 왜 본 시스템 개발이 필요한지 기술적 환경 및 수요 배경을 중심으로 기술

저희 조는 항상 변수들이 존재하는 화재 현장 및 화재 대피 실패 요인에 대해 알아보았습니다. 실패 요인을 인지, 반응, 대피 3단계로 나누었을 때, 인지 단계에서는 화재에 대한 인지가 지연되어 **대피 시간 확보**가 불가했습니다. 반응 단계에서는 피난시설을 알지 못하여 본능적으로 **문을 열고 회피하여 연소가 확대**되거나 밝은 곳을 향하는 반응을 보여 창문에서 추락하는 경우도 발생했습니다. 마지막으로 대피 단계에서는 **대피 경로를 확보하지 못하**거나 친숙한 경로인 **승강기를 이용하여 대피**하는 경우도 있었습니다. 또한 화재 현장에서 소방관분들이 순직하시는 경우는 생존자들을 구조하는 과정에서 제일 많이 발생했습니다.

실패 요인들을 보며 **대피 시간 및 경로 확보, 생존자 구조에 초점을 맞춰서 이전보다 성공률이 높인 화재 대피 시스템이 필요**하다고 생각했습니다. 따라서 **인명 피해를 최소한으로 줄이고 효율적인 대피 경로로 안전하게 생존자들을 인솔하고, 화재 현장 탐색을 통해 생존자를 구조하는 시스템 제작을 계획**했습니다.

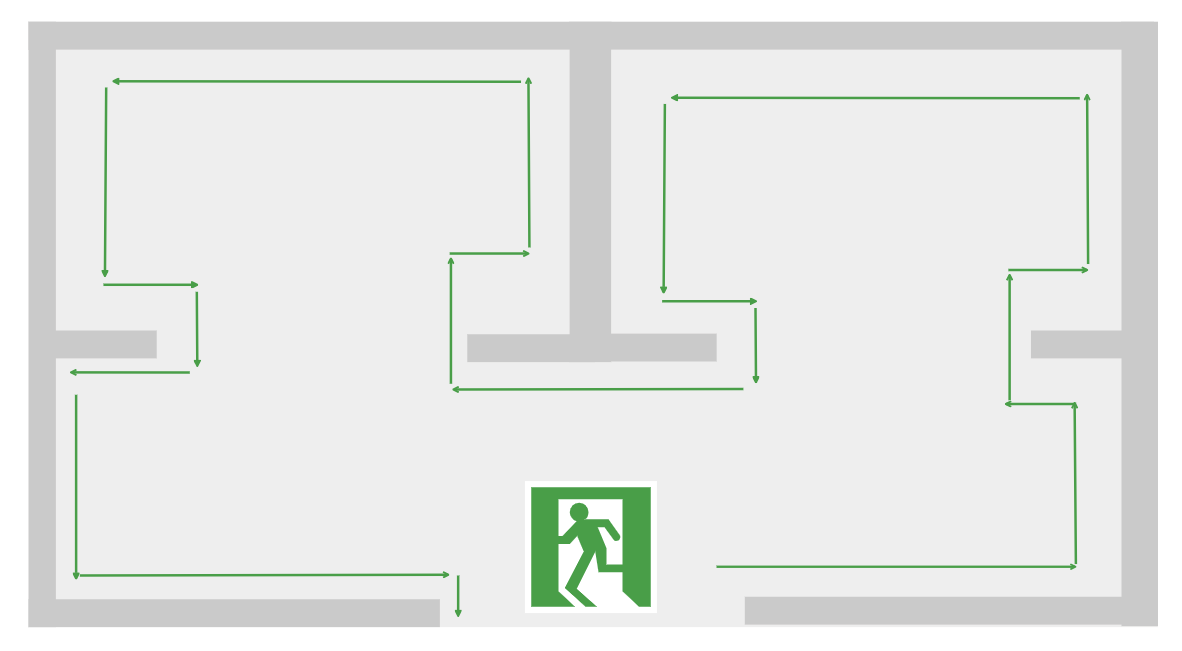
(add . 인력이 부족한 상황에서 쉽고 빠르게 생존자를 대피시키기 위한 자율주행 기능 계획 )

|  |  |  |
| --- | --- | --- |
| **3** |  | **H/W 구현** |

□ H/W를 어떻게 구현했는지를 그림, 사진 등으로 설명, 사용 센서, 부품들을 나열

* 화재 대피 안내 로봇
  + 대피 사이렌 방송(화재 감지 시)
    - 불꽃 감지 센서 모듈
    - 피에조 부저
  + 대피 안내 방송
    - 사용자 녹음 모듈 + 스피커(ISD 1820)
  + 로봇 제어 및 복도 이동 반복
    - 적외선 센서(IR)
* 생존자 구조 로봇
  + 대피 경로 밝히기(시야 확보)
    - LED, 수은 건전지
    - 서보모터
  + 생존자 감지 및 구조
    - 인체 감지 센서
    - ESP 32 Cam
  + 화재 현장 상태 분석 및 시각화
    - 일산화탄소 가스 감지 센서 (MQ-7)
    - 온/습도 센서 (DFT11)
    - LCD



* + 화재 현장 탈출 및 로봇 이동
    - 적외선 센서
    - 초음파 센서
* 대피 경로 제작
  + 폼보드, 하드보드지 

[하드웨어 구현 사진]

|  |  |  |
| --- | --- | --- |
|  | 생존자 구조 | 화재 대피 안내 |
| 앞 |  |  |
| 뒤 |  |  |
| 오른쪽 |  |  |
| 왼쪽 |  |  |
| 위 |  |  |

|  |  |  |
| --- | --- | --- |
| **4** |  | **S/W 구현** |

□ S/W를 어떻게 구현했는지를 설명

* 생존자 구조 로봇

> 기본 주행 : 우측 벽을 따라 주행

> 생존자 발견시 : LCD에 따라오라는 말과 가스 농도 및 온도 출력 후 LED를 떨어뜨리며 주행

* 화재 대피 안내 로봇

> 화재시 사이렌 소리를 울린 후 주행하면서 ‘ 대피하세요 ‘ 라는 말을 반복적으로 나타냄

→ 자세한 사항은 맨 마지막장을 참고하시길 바랍니다.

|  |  |  |
| --- | --- | --- |
| **5** |  | **구현 소감** |

□ 개발 과정에 있었던 일과 느낌을 소개

* 사용할 센서가 너무 많아 로봇 한 개로 사용하기에는 무리가 있어 두 개의 로봇을 사용했습니다. 로봇을 안내와 구조의 기능으로 분리해 구현한 덕분에 계획한 로봇보다 더 효율적인 개발을 한 것 같습니다.
* 서보 모터 사용 시 센서 간의 충돌이 발생하여 코드 실행에 오류가 있었지만 조건문의 세밀한 조절과 다른 센서에 대해 더 깊이 있게 알아보면서 결국 해결할 수 있었습니다.
* 일반 바닥과 자체 제작한 맵 바닥의 환경이 달라 바퀴 회전도 제어와 속력 제어 수치가 유동적이었습니다. 세밀한 조건문 분리와 delay를 사용해 많은 시행 착오 끝에 로봇이 맵을 제대로 인지해 움직이도록 했습니다.(다만 환경 및 배터리 양에 따라 값이 변동되는 문제가 있습니다.)
* 다양한 센서를 사용하면서 전력 소모가 빨리 진행되면서 배터리가 빨리 방전되었습니다. 따라서 배터리를 자주 교체해야 하는 번거로움이 있었습니다.
* 다양한 종류의 센서를 사용해보면서 그 센서들이 어떻게 실행되고 어떤 역할을 하는지 배울 수 있었습니다. 오류를 겪으며 센서들의 값도 수정하면서 원하는 로봇으로 구현하는 과정이 흥미로웠습니다.
* 주제를 선정할 때 창의적이면서 다양한 아이디어가 나와서 계획 시 도움이 많이 되었고 매주 회의를 통해 부족하거나 기능에 대한 의견을 들으며 피드백을 지속했습니다. 1학기 개인 프로젝트보다 팀 프로젝트 진행은 개발자로서 필요한 역량인 소통 및 협력을 강화하며 성장할 수 있는 시간이었습니다. 팀원들과 역할을 분담하고 어려운 부분은 함께 고민하고 그 오류를 해결하면서 개발에 대한 자신감을 얻을 수 있었습니다.

|  |  |  |
| --- | --- | --- |
| **6** |  | **향후 보완 사항** |

□ 향후 보완이 필요하다고 생각되는 사항 기술

* 더 섬세한 센서 값 조절을 통해 로봇이 이동할 때 보다 정확하고 세밀한 이동을 할 수 있게 소프트웨어 구현을 하면 더 좋을 것 같습니다.
* 배터리 소모가 빨랐기에 전력 소모가 적은 센서를 사용하거나 더 큰 전력을 사용할 방법이 있다면 로봇을 구동하는데 도움이 될 것 같습니다.
* 전선을 더 깔끔하게 정리한다면 회로의 흐름을 더 쉽게 이해할 수 있을거라고 생각합니다.

|  |  |  |
| --- | --- | --- |
| **7** |  | **참고 문헌** |

□ 개발 과정에 참고한 도서, 사이트 등을 기술

카메라 모듈 사용 참고 :

[ESP32-CAM 소개와 사용방법 :: 남보공방 (makernambo.com)](https://makernambo.com/109)

MQ-7 가스센서 사용 라이브러리 및 동작 원리 참고 :

[에듀이노-코딩교육 전문 쇼핑몰 (eduino.kr)](https://eduino.kr/product/detail.html?product_no=119)

이외 IoT 프로그래밍 및 1학기 수업 시간에 배운 내용 기반 구현

|  |  |  |
| --- | --- | --- |
| **8** |  | **스케치 소스 코드** |

□ 스케치 코드를 첨부하세요. 가져 온 소스는 표시하세요. (파란색으로 표시)

[생존자 구조 로봇 코드]

#include <Servo.h>

#include <MQ7.h>

#include <DHT11.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

Servo servoLeft;

Servo servoRight;

int motor\_control = 7; // 서보모터

Servo servo;

int inputPin = 11; // 인체 감지 센서

int pirState = LOW; // 센서 초기상태에는 움직임이 없음을 가정

int val = 0; // 센서 신호의 판별을 위한 변수

int trig = 2;

int echo = 3;

int pin=4;

DHT11 dht11(pin);

LiquidCrystal\_I2C lcd(0x27,16,2);

byte tem[8] = {B00100,B00100,B00100,B00100,B01110,B10001,B10001,B01110};

byte d[8] = {B00100,B01010,B00100,B00000,B00000,B00000,B00000,B00000};

byte du[8] = {B00000,B00000,B10101,B01110,B11111,B01110,B10101,B00000};

byte smile[8] = {B00000,B01010,B01010,B01010,B00000,B11111,B01110,B00100};

byte soso[8] = {B00000,B01010,B01010,B01010,B00000,B11111,B00000,B00000};

byte bad[8] = {B00000,B01010,B01010,B01010,B00000,B01110,B10001,B00000};

MQ7 mq7(A0, 5.0);

void setup()

{

pinMode(10, INPUT); pinMode(9, OUTPUT); //Left IR LED & Receiver

pinMode(8, INPUT); pinMode(7, OUTPUT); //Right IR LED & Receiver

delay(1000); // Delay to finish tone

servoLeft.attach(13);

servoRight.attach(12);

pinMode(inputPin, INPUT); // 인체감지 센서

Serial.begin(9600);

pinMode(trig, OUTPUT);

pinMode(echo, INPUT);

lcd.init();

lcd.backlight();

lcd.home();

lcd.createChar(0, tem);

lcd.createChar(1, du);

lcd.createChar(2, smile);

lcd.createChar(3, soso);

lcd.createChar(4, bad);

lcd.createChar(5, d);

servo.attach(motor\_control);

}

void loop()

{

int err;

float temp, humi;

int irLeft = irDetect(9, 10, 41100);

int irRight = irDetect(7, 8, 41000);

int cntRight=7;

int cntLeft=7;

int i;

servo.write(0);

val = digitalRead(inputPin);

digitalWrite(trig, HIGH);

delay(10);

digitalWrite(trig,LOW);

float duration = pulseIn(echo,HIGH);

float distance = duration\*340/2/10000;

Serial.print(irLeft);

Serial.println(irRight);

Serial.print("\nDistance : ");

Serial.print(distance);

Serial.println("cm\n");

if(val == HIGH){

Serial.print("인체 감지됨");

for(i=0; i<90;i+=10){

servo.write(i);

}

}

if((irLeft==1)&&(irRight == 1))

{

Serial.print(irLeft);

Serial.println(irRight);

Serial.print("\nDistance : ");

Serial.print(distance);

Serial.println("cm\n");

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

delay(380);

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1700); // 우회전

delay(540);

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

delay(750);

}

if((irLeft == 0)&&(irRight=1))

{

Serial.print(irLeft);

Serial.println(irRight);

Serial.print("\nDistance : ");

Serial.print(distance);

Serial.println("cm\n");

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

delay(75);

servoLeft.writeMicroseconds(1500);

servoRight.writeMicroseconds(1300); // 좌회전

delay(1150);

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

delay(50);

}

if((irLeft==1)&&(irRight==0))

{

Serial.print(irLeft);

Serial.println(irRight);

Serial.print("\nDistance : ");

Serial.print(distance);

Serial.println("cm\n");

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

if(distance >= 5.5){

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1450); //우

delay(3);

}

else {

servoLeft.writeMicroseconds(1550);

servoRight.writeMicroseconds(1330); //좌

delay(3);

}

}

if((irLeft==0)&&(irRight==0))

{

Serial.print(irLeft);

Serial.println(irRight);

Serial.print("\nDistance : ");

Serial.print(distance);

Serial.println("cm\n");

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

delay(75);

servoLeft.writeMicroseconds(1500);

servoRight.writeMicroseconds(1300); // 좌회전

delay(1150);

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300); // 전진

delay(50);

}

if((err=dht11.read(humi, temp))==0)

{

Serial.print("temperature:");

Serial.print(temp);

Serial.print(" humidity:");

Serial.print(humi);

Serial.println();

}

else

{

Serial.println();

Serial.print("Error No :");

Serial.print(err);

Serial.println();

}

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Escape!!");

lcd.setCursor(0,1);

lcd.write(0);

lcd.print((int)temp);

lcd.write(5);

lcd.print("C");

lcd.print(" ");

lcd.write(1);

lcd.print((int)mq7.getPPM());

lcd.print("ppm");

lcd.print(" ");

if((int)mq7.getPPM() > 9){

lcd.write(4);

}

if(((int)mq7.getPPM() > 2) && ((int)mq7.getPPM() <= 9)){

lcd.write(3);

}

if((int)mq7.getPPM() < 2){

lcd.write(2);

}

}

int irDetect(int irLedPin, int irReceiverPin, long frequency)

{ tone(irLedPin, frequency, 8);

delay(1);

int ir = digitalRead(irReceiverPin);

delay(1);

return ir; }

[화재 대피 안내 로봇 코드]

//대피안내로봇

#include <Servo.h>

#include <SoftwareSerial.h>

Servo servoLeft;

Servo servoRight;

//대피 안내 관련 핀 번호

int rec = 5;

int playe = 6;

//사이렌 관련 핀 번호

int piezo = 4;

//화재 감지

int flame = 11;

int state = 0;

//블루투스 모듈

int bluetoothTx = 8;

int bluetoothRx = 7;

SoftwareSerial bluetooth(bluetoothTx,bluetoothRx);

void setup()

{

Serial.begin(9600);

//적외선

pinMode(10, INPUT); pinMode(9, OUTPUT);

pinMode(3, INPUT); pinMode(2, OUTPUT);

servoLeft.attach(13);

servoRight.attach(12);

//불꽃 감지 센서

pinMode(flame, INPUT);

//블루투스

bluetooth.begin(9600);

}

void loop()

{

state = digitalRead(flame);

int irLeft = irDetect(9, 10, 42000);

int irRight = irDetect(2, 3, 42000);

if(state == 0){ // 불꽃이 감지 되었을 때

siren();

siren();

delay(100);

noTone(piezo);

delay(100);

digitalWrite(playe, HIGH);

digitalWrite(playe, LOW);

}

if((state > 0) || (state < 0)) { //불꽃이 감지 되지 않을 때

noTone(piezo);

delay(100);

}

Serial.println("HAha");

Serial.println(irLeft);

Serial.println(irRight);

//블루투스로 조종

char cmd;

if(bluetooth.available())

{

cmd = (char)bluetooth.read();

Serial.print("Command = ");

Serial.println(cmd);

if(cmd == '1')

{

Serial.println("OK");

//대피 방송 및 주행

while(1)

{

while(1)

{

int irLeft = irDetect(9, 10, 42000);

int irRight = irDetect(2, 3, 42000);

if((irLeft == 0) && (irRight == 0))

{

Serial.println("Sound on");

digitalWrite(playe, HIGH);

digitalWrite(playe, LOW);

delay(100);

Serial.println("sound OFF");

backward(1000);

turnLeft(1080);

servoLeft.writeMicroseconds(1500);

servoRight.writeMicroseconds(1500);

delay(100);

break;

}

else

{

forward(1);

Serial.println("NO");

Serial.println(irLeft);

Serial.println(irRight);

if(cmd == '2'){

break;

}

}

}

}

}

}

if(cmd == '2')

{

servoLeft.writeMicroseconds(1500);

servoRight.writeMicroseconds(1500);

delay(5000);

}

}

int irDetect(int irLedPin, int irReceiverPin, long frequency)

{

tone(irLedPin, frequency, 8);

delay(1);

int ir = digitalRead(irReceiverPin);

delay(1);

return ir;

}

//사이렌 소리 출력 함수

void siren()

{

for (int hz = 300; hz <= 750; hz++)

{

tone(piezo, hz);

delay(5);

}

for (int hz = 750; hz >= 300; hz--)

{

tone(piezo, hz);

delay(5);

}

}

//주행 관련 함수

void forward(int time)

{

servoLeft.writeMicroseconds(1700);

servoRight.writeMicroseconds(1300);

delay(time);

}

void turnLeft(int time)

{

servoLeft.writeMicroseconds(1300);

servoRight.writeMicroseconds(1300);

delay(time);

}

void backward(int time)

{

servoLeft.writeMicroseconds(1300);

servoRight.writeMicroseconds(1700);

delay(time);

}