Ultrasonic Sensor-Based Autonomous

Haesung Oh

Abstract

Currently, drones are commercialized and used in various fields. This paper presents a drone that explores dangerous disaster sites for humans to enter directly. It added autonomous flight to avoid obstacles using ultrasonic sensors for situations where direct control is difficult, or radio communication is impossible. When an obstacle appears within a range in the ultrasonic sensor, it autonomously drives with an algorithm that moves to the left or right. In addition, this paper suggests a returning algorithm for easy drone withdrawal. This function works when the drone cannot drive during autonomous flight or exceeds half of the maximum driving distance.

I. Introduction

1. Necessity and Research Purpose

Small drones can move flexibly in any space because they can adjust their altitude as well as front, rear, left, and right. In a disaster, the initial reaction is insufficient due to various factors. Therefore, the goal was to grasp the disaster site with a drone. Since indiscriminate input of rescue teams without the absence of identification of disaster sites can cause secondary damage, it aims to be an autonomous drone that enables safe and fast topographic identification without relying on human control.

2. Features and Advantages

In the event of a collapse, there are numerous obstacles, so it can be hard to control directly. In addition, assuming a situation in which communication is difficult due to radio wave obstruction materials (rebar, concrete), the autonomous driving function was added to automatically identify the terrain and return to the way it came to facilitate withdrawal.

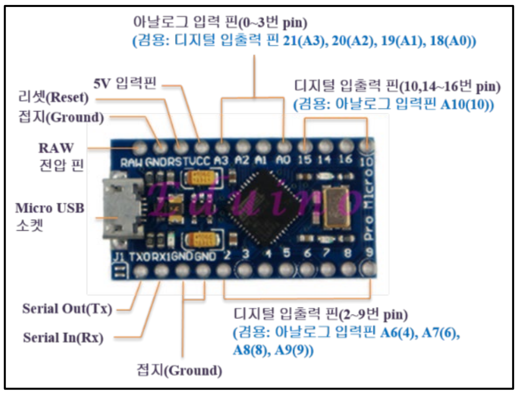
II. Research contents

1. Hardware

This study uses commercially available Arduino-based drone kits to ease the addition and coding of ultrasonic sensors. The kit uses the Arduino Pro Micro board as the motherboard. Figure 1 shows the specifications of this motherboard. Also, Figure 2 shows the pins used by the Arduino Pro Micro in this kit.

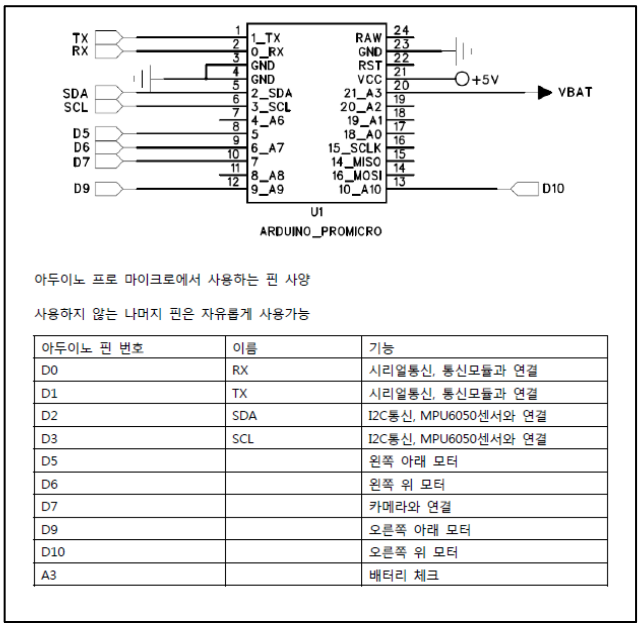
Ultrasonic sensors with minimized size and weight are in favor. Therefore, this study uses the SRF01 model (Figure 3). This sensor is the most tin ultrasonic sensor. It also has the advantage of parallel connecting on a single digital pin by assigning different address values to each sensor (Figure 4).

All four SRF01 sensors connect parallelly to the digital pin15 that remains empty in the completed drone kit and is attached in the front-rear, left-right, and left-right directions (Figure 5). In conclusion, the structure shown in Figure 6 controls the hardware system.



아날로그 입력 핀(0~3번 pin)
(겸용: 디지털 입출력 핀 21(A3), 20(A2), 19(A1), 18(A0))
5V 입력핀
리셋(Reset)
접지(Ground)
디지털 입출력 핀(10,14~16번 pin)
(겸용: 아날로그 입력핀 A10(10))
RAW 전압 핀
Micro USB 소켓
Serial Out(Tx)
Serial In(Rx)
디지털 입출력 핀(2~9번 pin)
(겸용: 아날로그 입력핀 A6(4), A7(6), A8(8), A9(9))
접지(Ground)

figure 1 Arduino Pro Micro Pin map



아두이노 프로 마이크로에서 사용하는 핀 사양

사용하지 않는 나머지 핀은 자유롭게 사용가능

아두이노 핀 번호	이름	기능
D0	RX	시리얼통신, 통신모듈과 연결
D1	TX	시리얼통신, 통신모듈과 연결
D2	SDA	I2C통신, MPU6050센서와 연결
D3	SCL	I2C통신, MPU6050센서와 연결
D5		왼쪽 아래 모터
D6		왼쪽 위 모터
D7		카메라와 연결
D9		오른쪽 아래 모터
D10		오른쪽 위 모터
A3		배터리 체크

Figure 2 Arduino Pro Micro Pin usage

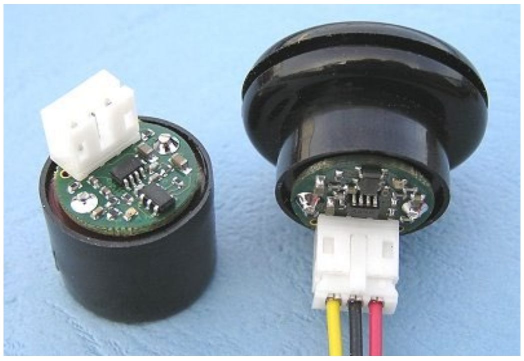


Figure 3 SRF01 Ultrasonic Sensor

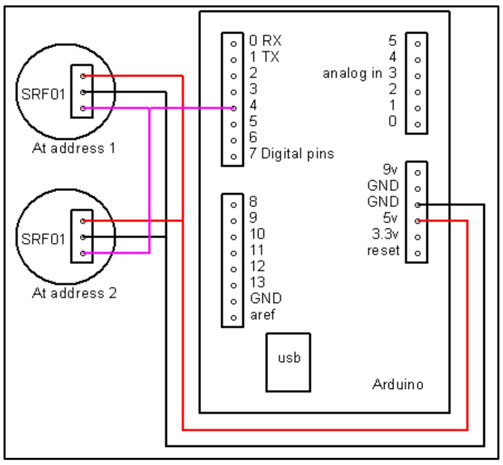


Figure 4 Parallel Connecting of SRF01

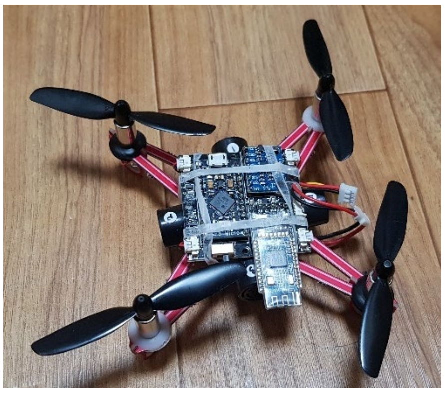


Figure 5 Assembled Drone

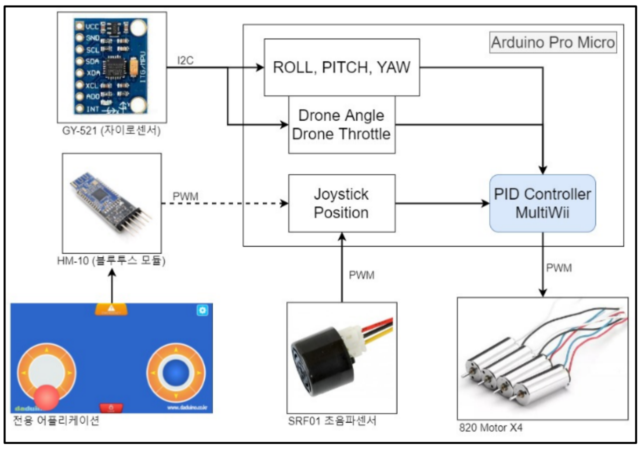


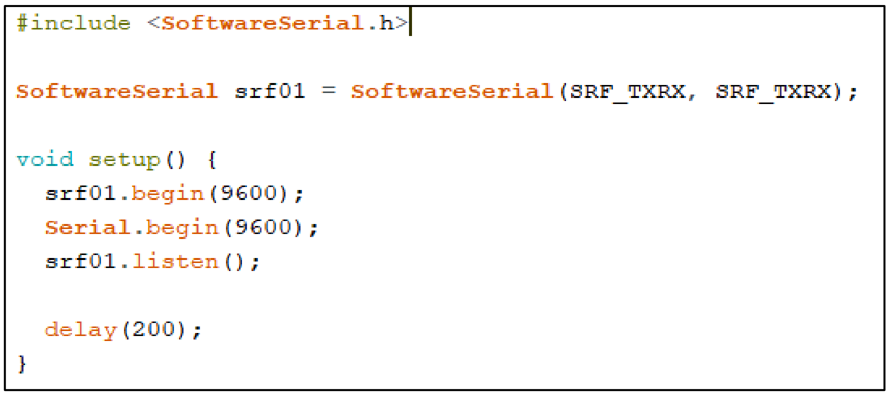
Figure 6 Overview of Hardware Control

2. Software

A. SRF01 Ultrasonic Sensor

i. Serial Communications

The SRF01 sensor shall use serial communication to communicate with the Arduino board. However, the Arduino Pro Micro board has only two serial communication pins and even uses them to communicate with Bluetooth communication modules, making it difficult to use the serial communication pins. Therefore, this study uses software serial communication to enable digital pins to virtual serial communication pins. The following figure shows the basic setting.



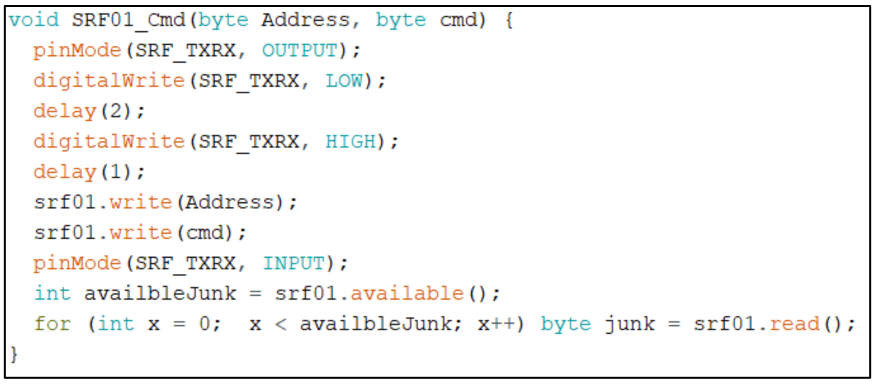
```
#include <SoftwareSerial.h>

SoftwareSerial srf01 = SoftwareSerial(SRF_TXRX, SRF_TXRX);

void setup() {
  srf01.begin(9600);
  Serial.begin(9600);
  srf01.listen();

  delay(200);
}
```

Figure 7 Software Serial Communications

This work sets the SRF_TXRX value to 15 and uses the digital pin15 as the software serial communication pin.

ii. Input Command

SRF01 sensors use digital pins on Arduino. So the digitalWrite() function on the Arduino code is used. For example, the figure below is a function of inputting a command line to the SRF01 sensor.



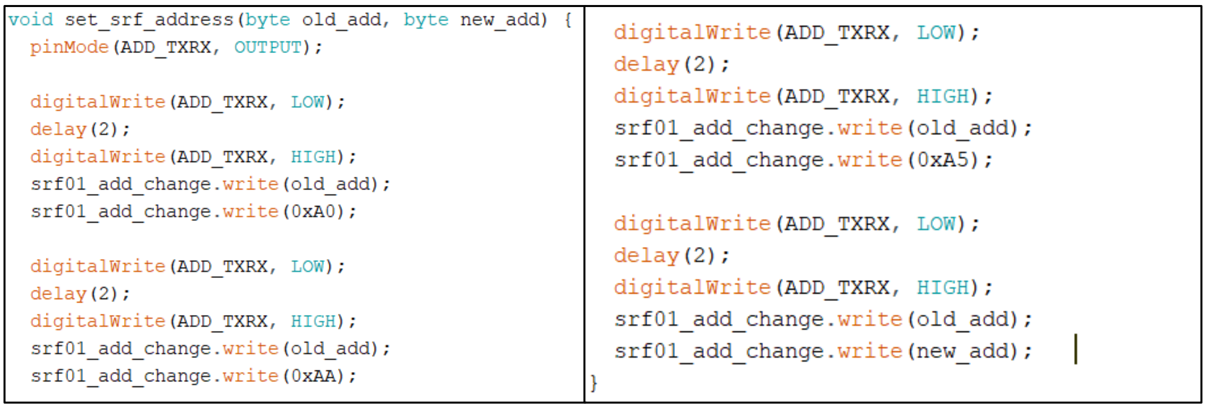
```
void SRF01_Cmd(byte Address, byte cmd) {
  pinMode(SRF_TXRX, OUTPUT);
  digitalWrite(SRF_TXRX, LOW);
  delay(2);
  digitalWrite(SRF_TXRX, HIGH);
  delay(1);
  srf01.write(Address);
  srf01.write(cmd);
  pinMode(SRF_TXRX, INPUT);
  int availbleJunk = srf01.available();
  for (int x = 0;  x < availbleJunk; x++) byte junk = srf01.read();
}
```

Figure 8 SRF01 Sensor Command Input Function

The digital pin receives commands in the order of LOW and HIGH. Then it gets the address value of the sensor and the command line through a predefined serial. After that, it goes through a process of removing noise.

iii. Address Assignment

The SRF01 sensor can store numbers from 1 to 16 in the built-in board and use them as address values. This feature allows the parallel connecting of multiple sensors to one digital pin to use them simultaneously. In addition, the system can distinguish different sensors by receiving an address value from the above command function. This address value is set to 1 as the factory default but can be modified with the following procedure:



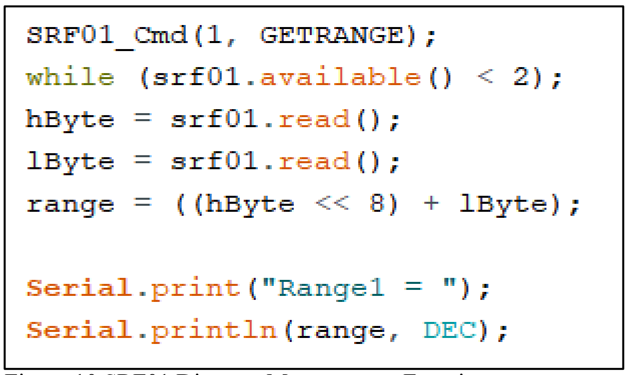
```
void set_srf_address(byte old_add, byte new_add) {
  pinMode(ADD_TXRX, OUTPUT);

  digitalWrite(ADD_TXRX, LOW);
  delay(2);
  digitalWrite(ADD_TXRX, HIGH);
  srf01_add_change.write(old_add);
  srf01_add_change.write(0xA0);

  digitalWrite(ADD_TXRX, LOW);
  delay(2);
  digitalWrite(ADD_TXRX, HIGH);
  srf01_add_change.write(old_add);
  srf01_add_change.write(0xAA);
  digitalWrite(ADD_TXRX, LOW);
  delay(2);
  digitalWrite(ADD_TXRX, HIGH);
  srf01_add_change.write(old_add);
  srf01_add_change.write(0xA5);

  digitalWrite(ADD_TXRX, LOW);
  delay(2);
  digitalWrite(ADD_TXRX, HIGH);
  srf01_add_change.write(old_add);
  srf01_add_change.write(new_add);
}
```

Figure 9 SRF01 Sensor Address Assignment Function

Entering 0xA0, 0xAA, 0xA5(new address value) as a command line in order sets the board to the new address value. In this study, four sensors were placed and used as address values of 1 (front), 2 (right), 3 (rear), and 4 (left), respectively.

iv. Distance Measurement

To measure the distance in cm, enter the 0x54 command, cut the bits output from the sensor's board into 8 units, and read them twice. And the algorithm can determine the distance by combining the two bytes.



```
SRF01_Cmd(1, GETRANGE);
while (srf01.available() < 2);
hByte = srf01.read();
lByte = srf01.read();
range = ((hByte << 8) + lByte);

Serial.print("Range1 = ");
Serial.println(range, DEC);
```

Figure 10 SRF01 Distance Measurement Function

B. Directional Control Algorithm

The user can directly control the drone kit used in this study with a dedicated smartphone application. This application has two joysticks. The left joystick manipulates the drone's altitude and rotation of the YAW axis, and the right joystick manipulates the forward, backward, left, and right movements in the XY plane.

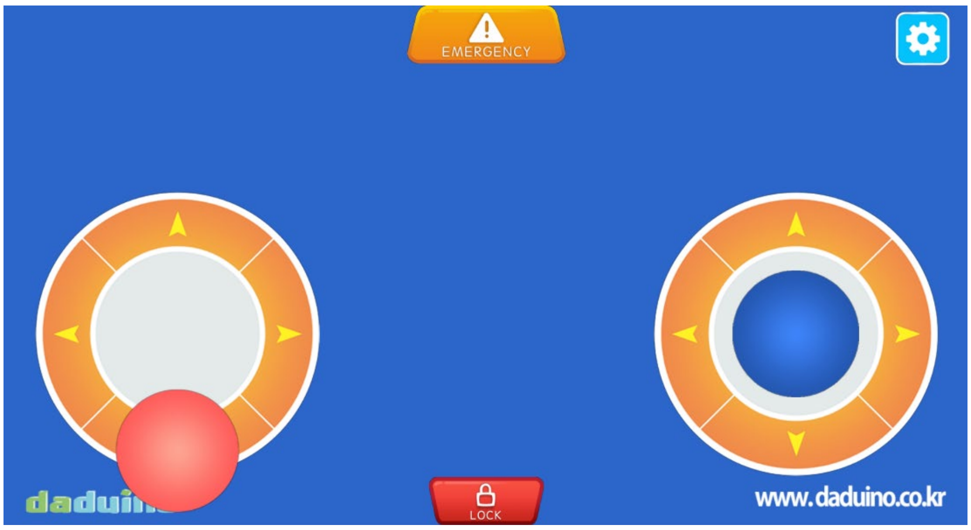


Figure 11 Drone Control Application

An array called rcData within the drone's code stores this joystick's movement. For example, rcData[0] and rcData[1] reserve the x and y coordinates of the left joystick, and rcData[2] and rcData[3] do the x and y coordinates of the right joystick, respectively. In this study, the drone was manipulated by arbitrarily changing the rcData array to ignore the application joystick and force it to change direction.

C. Autonomous Flight Algorithm

i. Priority and Flow Chart of Autonomous Flight Algorithm

Figure 11 shows the flow chart of autonomous flight algorithm.

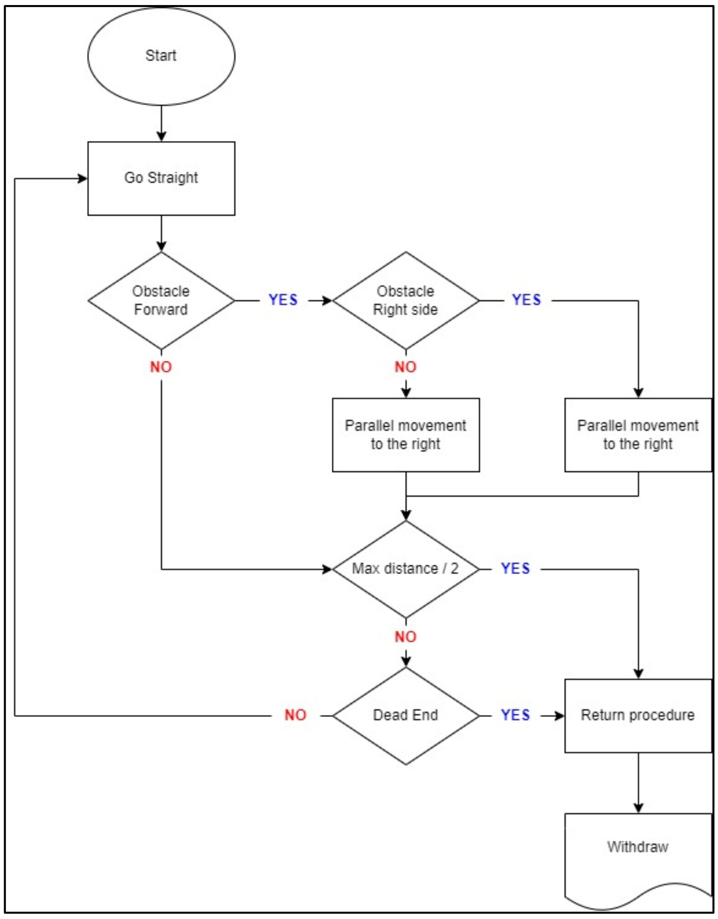


Figure 12 Autonomous Flight Algorithm Flow Chart

In this algorithm, the priorities are as follows:

1. Return after driving half of the maximum mileage.
2. Go straight if there are no obstacles toward.
3. If there is no obstacle on the right, move parallel to the right.
4. Parallel movement to the left.
5. Return when the drone encounters a dead end.

ii. Returning Algorithm

The checkBattery() function in the drone code returns the current battery remaining in % units. Therefore, with the above algorithm, if the checkBattery() value becomes 50 at any time during the autonomous tour, the algorithm immediately returns to the way it came.

Since it is necessary to reverse the way it came, a stack structure stores the values of the drone's direction as a rcData array and the travel distance using the gyro sensor. Then when returning, it popped to return the path in reverse order.

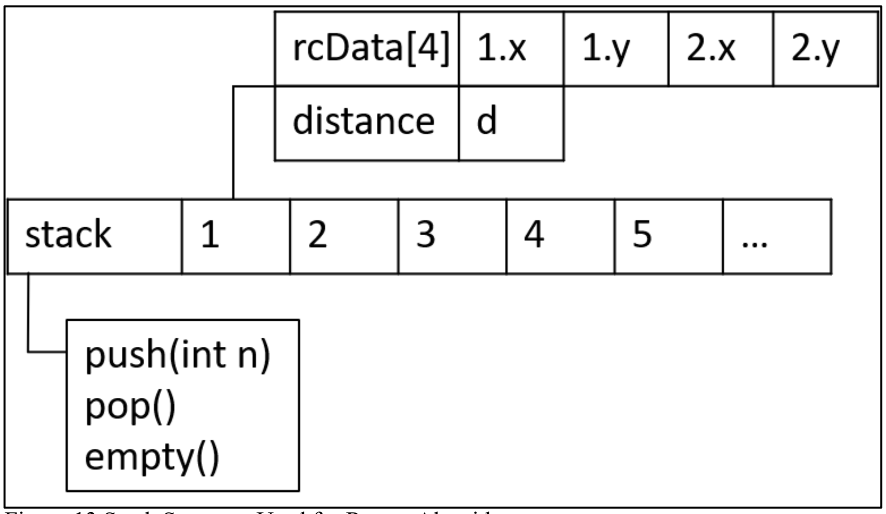


Figure 13 Stack Structure Used for Return Algorithm

III. Results

1. Experimental results

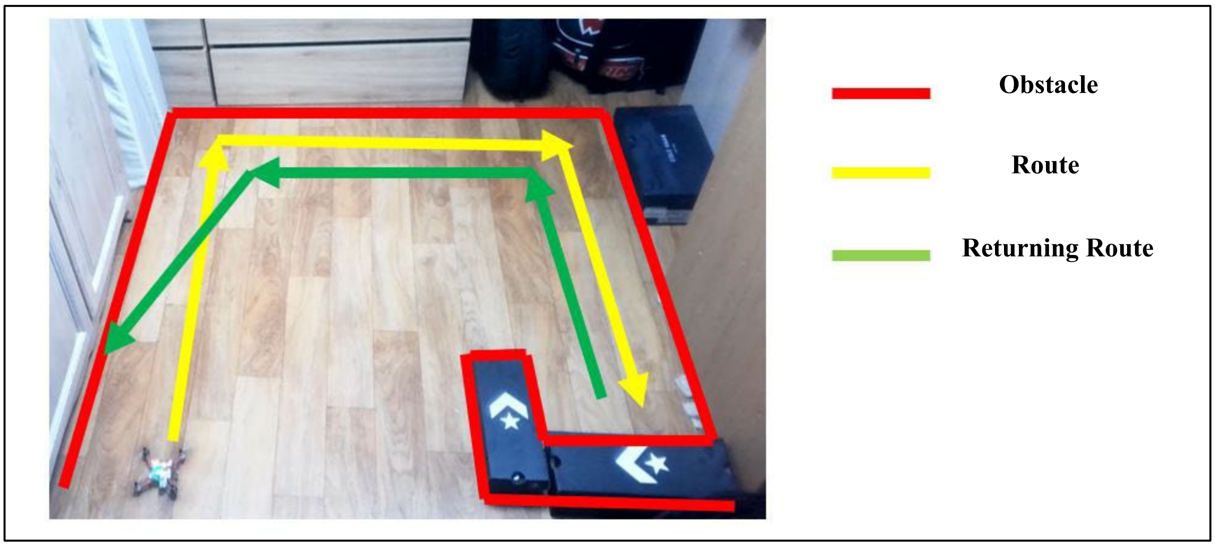


Figure 14 Obstacles and Routes in the Experiment

Experiments showed that the drone followed the intended algorithm to avoid obstacles. However, it was unstable along the YAW axis, so when returning to the reverse driving, it could not return to the starting point and hit the wall while driving.

2. Conclusion

This study devises a drone to help the rescue team carry out operations by quickly and accurately checking the internal situation. Putting the drone into a place where people enter from a disaster/disaster site, such as a collapse of a structure, should help this. The drone uses ultrasonic sensors to identify obstacles in the local hardware rather than GPS, which leads to independent self-driving. Ultimately, we eliminate cases where we miss the golden time when we cannot grasp the initial stage in disaster situations. In a narrow space where humans are challenging to control, or communication is difficult, an algorithm is also implemented to drive autonomously and return the path to the starting point for the safe withdrawal of drones. This autonomous driving algorithm might be helpful in other fields.

IV. Future research and complementary points

In terms of hardware, the maximum motor output of the drone kit was somewhat small. In addition, since the experiment was conducted in a narrow room to prove the algorithm's precision, the throttle of the motors was reduced. So, I could confirm that it was flying at a relatively low height. Therefore, in future studies, it is necessary to find a way to set the throttle high enough to increase the altitude but keep the parallel movement speed.

In addition, the three axes of ROLL, PITCH, and YAW were all unstable. As a result, maintaining the same altitude was even a difficulty. Therefore, in the following study, stable stabilization that endures the same height and does not move parallel should be implemented first.

In this study, camera attachment was omitted due to hardware limitations, but it can be used to identify actual terrain by attaching cameras and SD cards. Furthermore, rather than grasping the topography with a two-dimensional image as a camera, the LiDAR sensor can be used to identify the surrounding environment in three dimensions in real time, monitor it, and use it for autonomous driving. In this way, terrain monitoring and autonomous driving can be performed simultaneously.

In terms of software, not only parallel movement on a two-dimensional plane but also z-direction elevation control should be added to the autonomous driving algorithm to aim for more flexible autonomous driving. In addition, in this study, the GPS function was excluded because the drone is for where communication was impossible, but it is expected to be used for broader coverage.

V. Reference

<Deep Reinforcement Learning 기반의 자율비행 드론 알고리즘 비교 및 분석> (황인용 외 3명, 한국통신학회 학술대회논문집, 2018)

<PPO 기반 강화학습을 이용한 드론의 자율비행> (박성관, 김동환, 제어로봇시스템학회 논문지, 2020)

<다중센서를 이용하는 쿼드롭터 자율항법 알고리즘> (변상민 외 2명, 울산대학교 전기공학부, 2013)

<실내 자율비행 드론을 위한 위치 및 고도 제어> (윤혜인 외 4명, 대한전자공학회 학술대회, 2020)

<심층강화학습 기반 환경 인식 및 자율비행> (이덕진, 기계저널, 2019)

아두이노 기반 드론 키트 다두이노 : http://daduino.co.kr/

SRF01 센서 데이터시트:

https://wiki.dfrobot.com/SRF01_Ultrasonic_sensor__SKU_SEN0004_SRF01_Datasheet.pdf