KSW 2021 Fall Program

IoT Farm Firefighting System using Simplified Data Transmission with UAV and LoRa

Ricardo Gonzalez
Purdue University

Jiawei Chang
Purdue University

Hyoungjoo Lee
Dankook University

Yunji Kim
Dongguk University

Jason Su
Purdue University

Shixuan Mao
Purdue University

Youngseo Kang
Dongguk University

Sieun Choi
Dongguk University

CONTENTS

01

Introduction

Literature Review

Problem Statement

System Flow

Problem Statement - Detection

Literature Review

Previous Studies

Image & Video


- big data size
- data processing time ↑
- early detection X
- initial facility cost ↑

Proposed System

Integer

- small data size
- data processing time ↓
- early detection O
- initial facility cost ↓

Problem Statement - Suppression

Literature Review

Previous Studies

Sprinkler

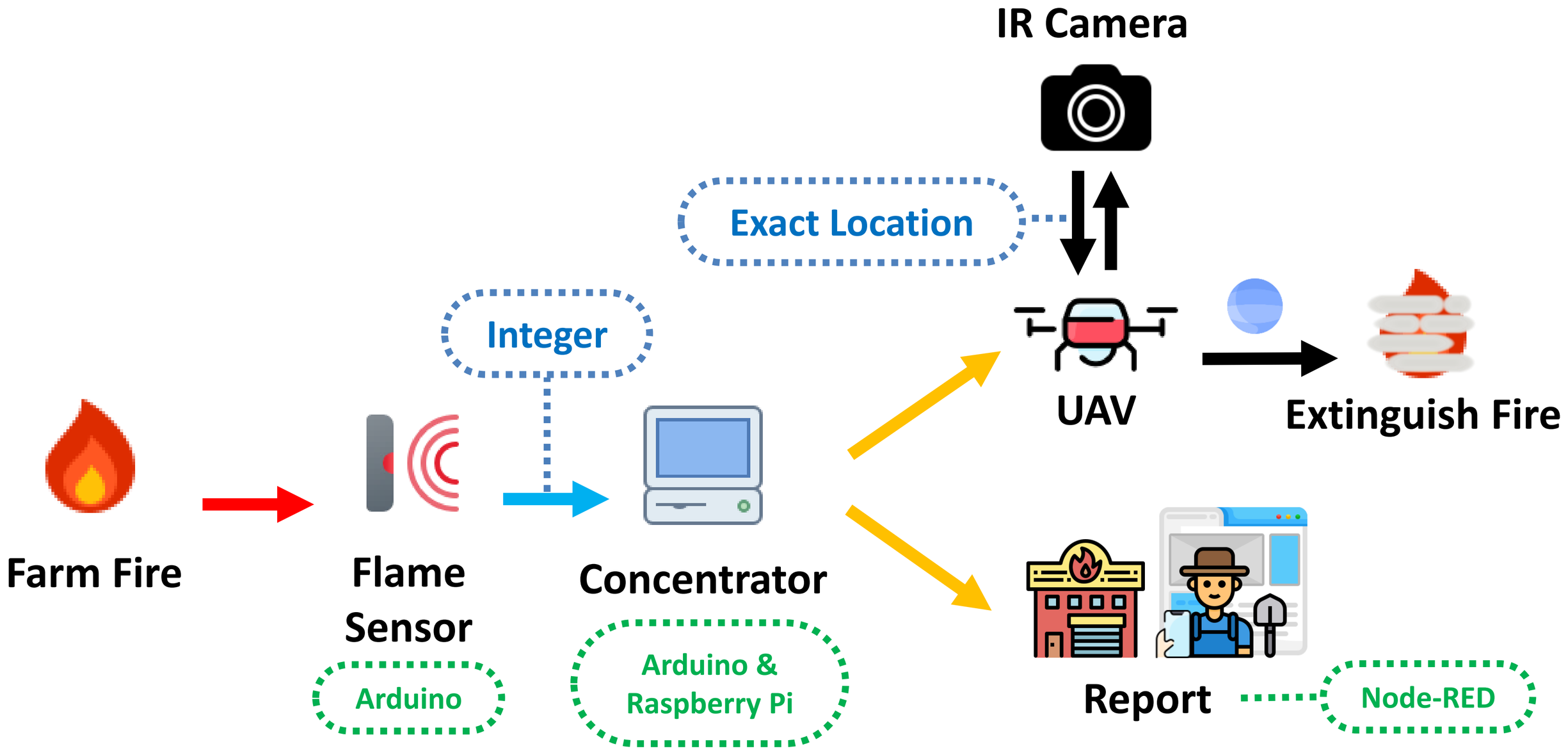
- $6,000~$10,000 per acre
- installed all over the farm
- require continuous maintenance (pump & pipeline)

Proposed System

UAV

- go to the specific fire location
- maintenance cost ↓
- efficient fire suppression

System Flow


IR Camera
Exact Location
Integer
UAV
Extinguish Fire
Farm Fire
Flame Sensor
Concentrator
Arduino
Arduino & Raspberry Pi
Report
Node-RED

02

Experimental Field & Location Algorithm

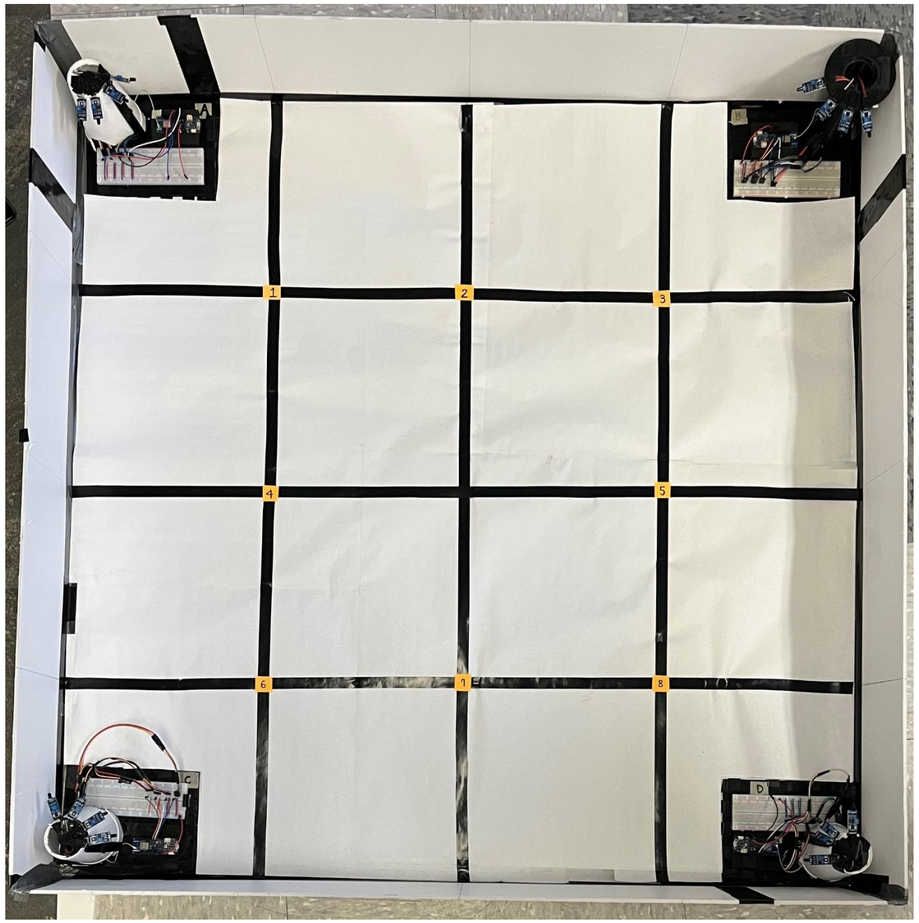
Experimental Field & Sensor Board

Arduino Location Decision Algorithm

Demo Video

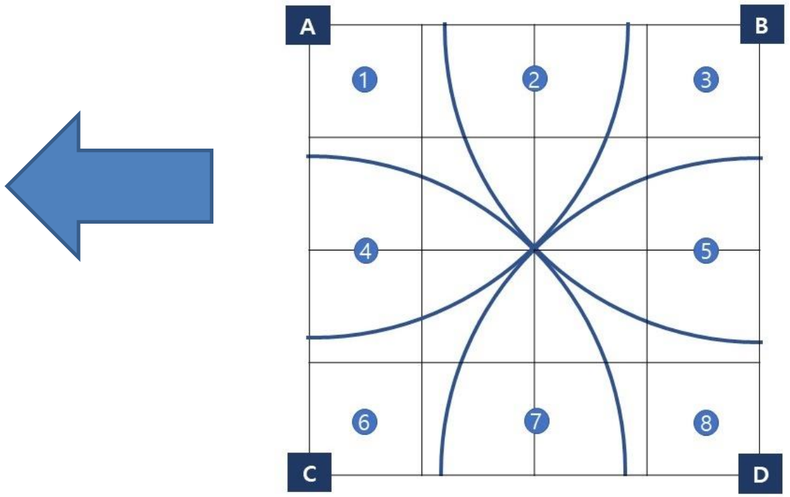
Experimental field

To test flame detection indoor



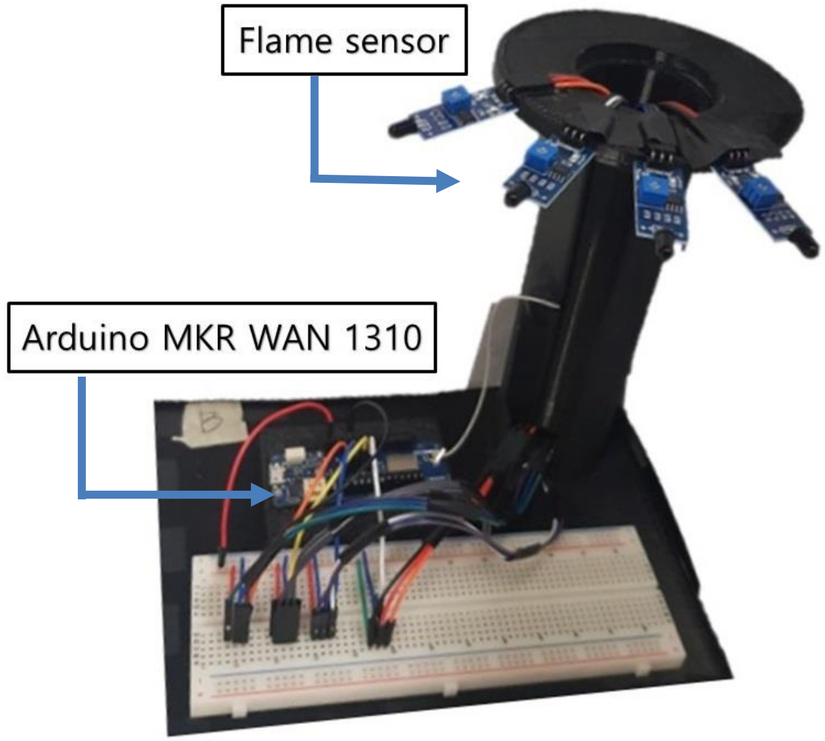
To test flame detection

a sensor board will be installed on the edge to detect a full range of fires in the experimental field.
Consists of square with side lengths of 48 inches
with grid of 4 by 4 lines on the field.



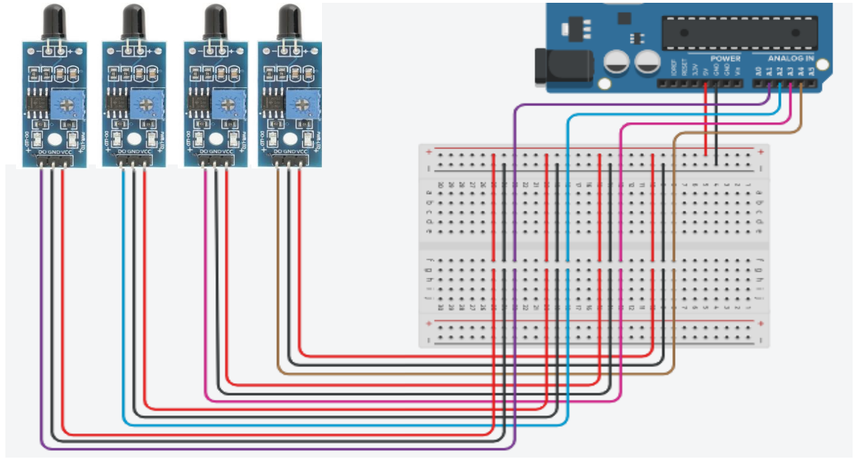
Sensor board

Arduino attached flame sensor to detect flame in the experimental field



Arduino controls the sensor board.
Arduino MKR WAN 1310 capable of LoRa communication.
flame sensors are connected to the Arduino on the sensor board to detect the flame and know the occurrence of a fire.
Send sensor board’s name to the central Arduino when detect fire.

< Sensor board Arduino circuit >



Detection Sensing range using Flame sensor

Depending on flame sensor’s number

Problem :

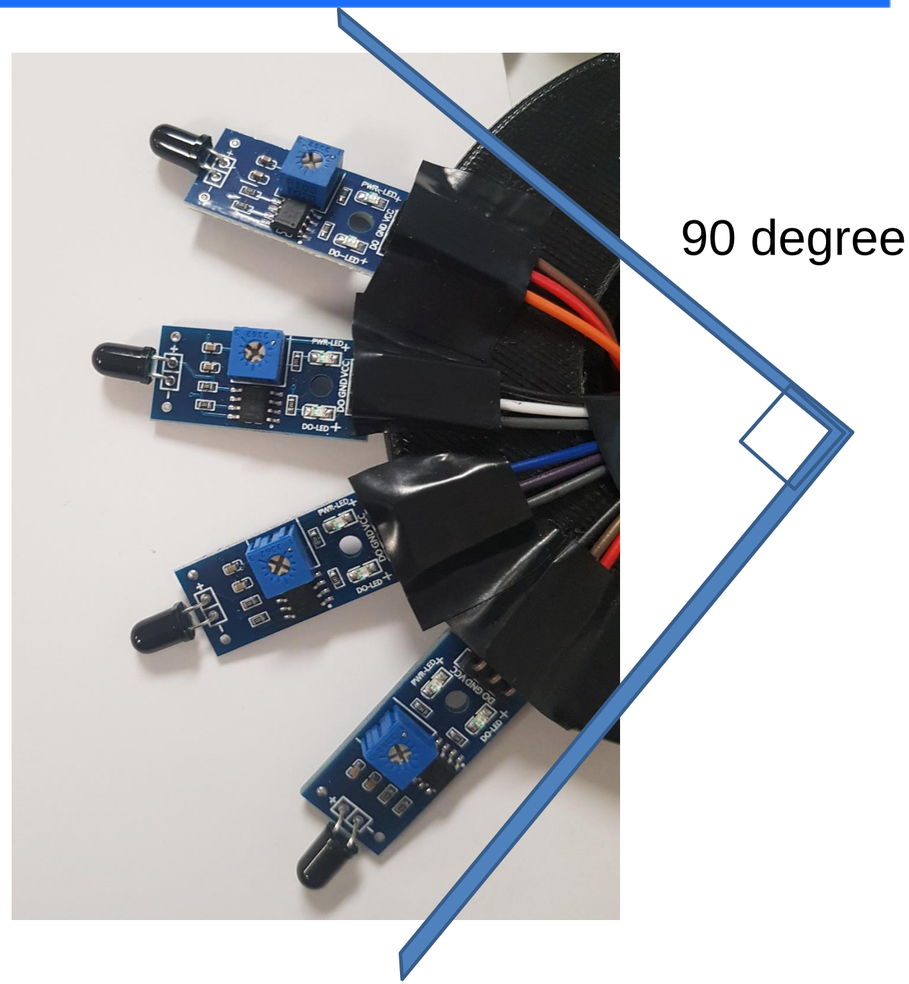
The detection radius of the flame sensor is 60 degrees.
However, as the distance of the flame increases,
it has a narrower sensing radius than 60 degrees.

Solution :

we assumed 90 degrees and tested for
at least how many flame sensors are needed
to detect all fires within 34 inches.
At that time, a total of **4 flame sensors** were needed per sensor board.

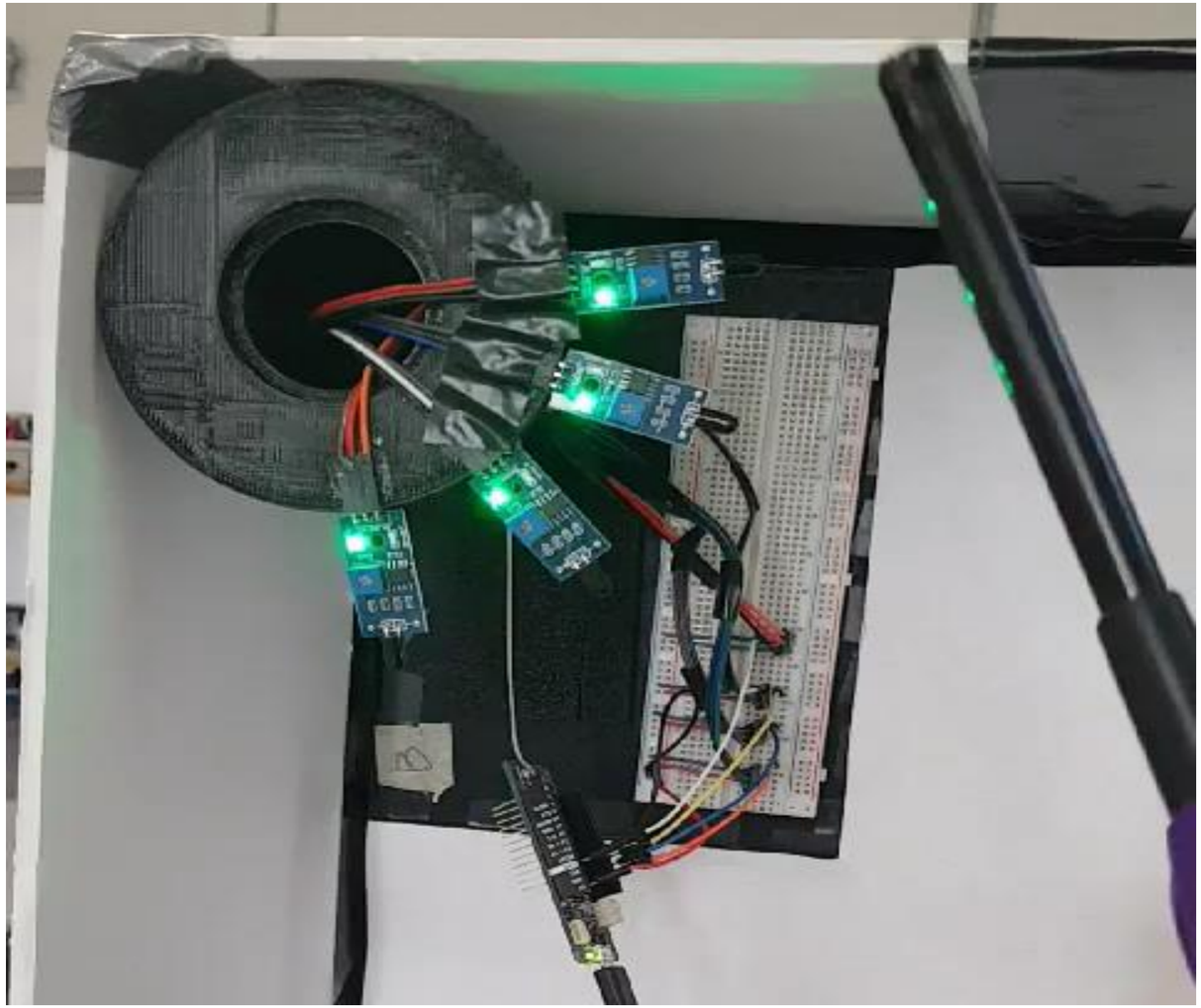
Conclusion :

Using 4 Flame sensor



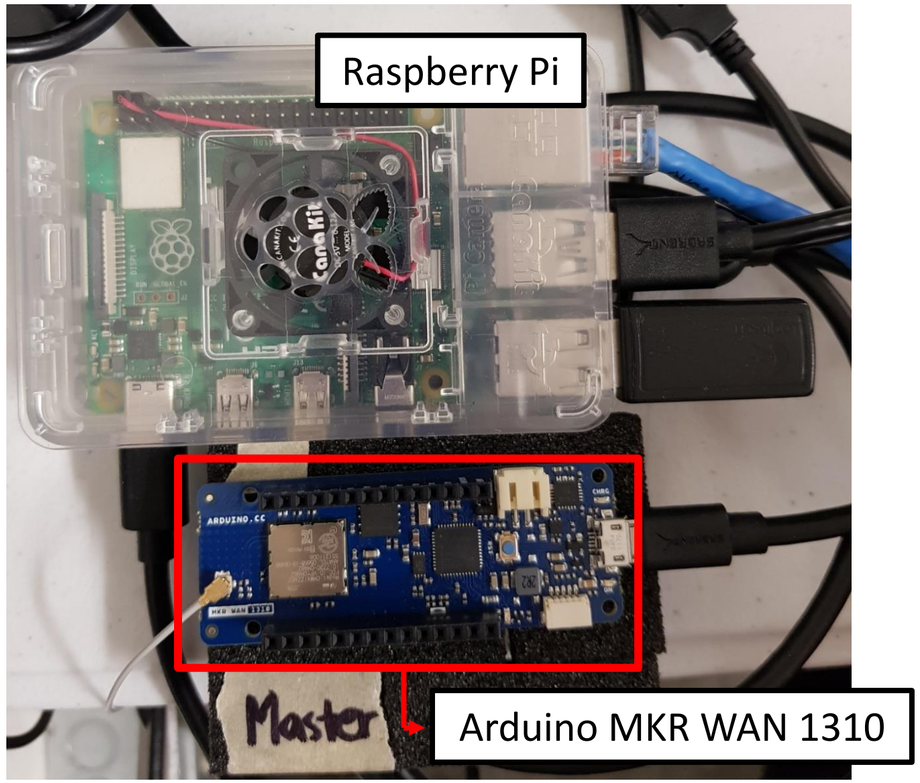
Detection Sensing range using Flame sensor

Depending on flame sensor’s number



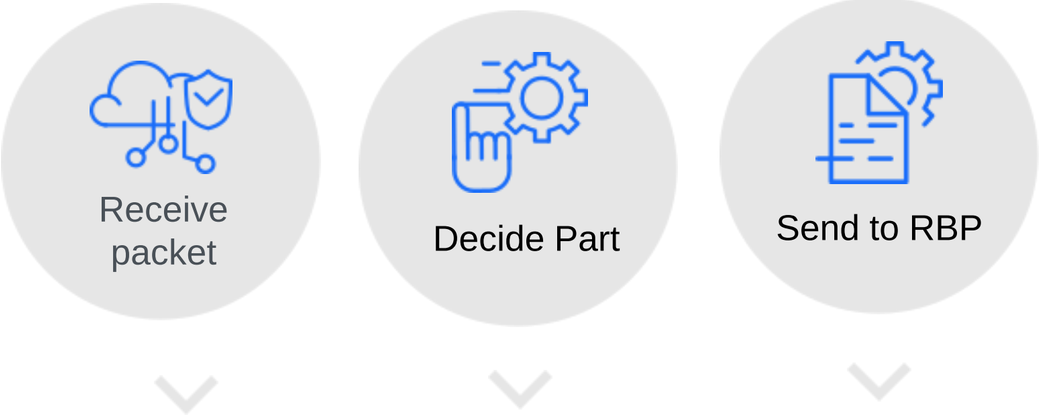
Central Arduino

Decide flame location part



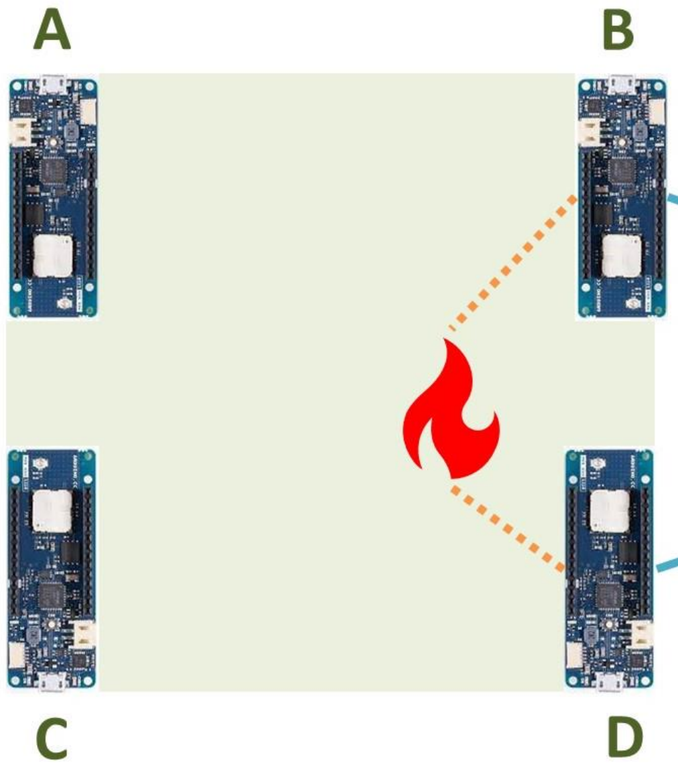
Arduino MKR WAN 1310
Central Arduino is connected to the Raspberry Pi by port.
Central Arduino delivers only the label of the sensor to the Raspberry Pi server through the organizing algorithm,
by removing these errors and confusion.

The rule of Central Arduino

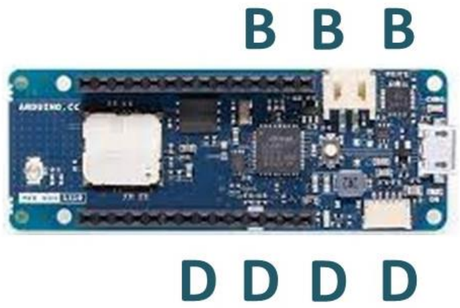


Location Part Decision Algorithm : 1

How to collect packet , send to raspberry PI

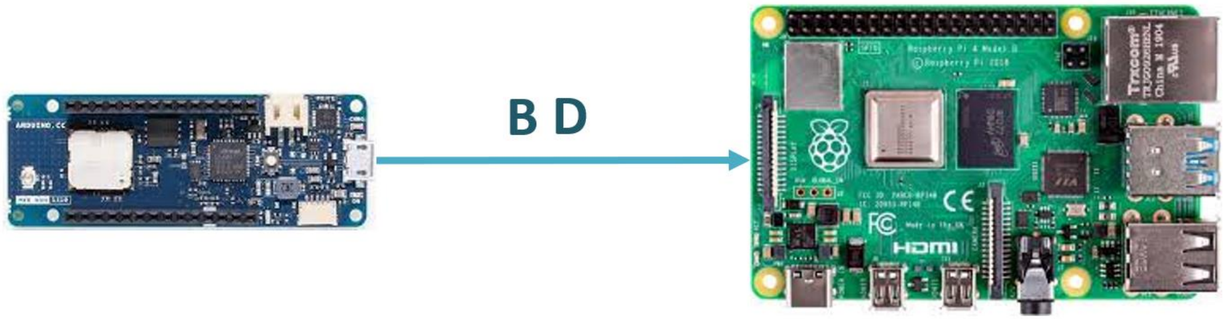


First, central Arduino receives packets sent by Arduino on all sensor boards for a certain period of time.
(13 seconds)



check how many packets were sent for each sensor board A, B, C, and D.

Location Part Decision Algorithm : 1



decide which part it is and send it to the Raspberry Pi

Location Part Decision Algorithm : 2

Central Arduino code : packet receive function

Receive Packet function

```
18 void recieve_packet(){
19
20   // get packets until 10 second
21   int t = 10000;
22   while(t-->0){
23     int packetSize = LoRa.parsePacket();       // try to parse packet
24     char node_name;
25     if (packetSize) {
26
27       Serial.print("Received packet '");       // received a packet
28                                                 // read packet
29       while (LoRa.available()) {                //
30         node_name = (char)LoRa.read();          // stare char value
31         //Serial.print(node_name);
32         if(node_name == 'A'){ A++; }            // count packet
33         else if(node_name == 'B'){ B++; }
34         else if(node_name == 'C'){ C++; }
35         else if(node_name == 'D'){ D++; }
36       }
37       Serial.print("' with RSSI ");
38       Serial.println(LoRa.packetRssi());
39     }
40   }
41 }
```

Location Part Decision Algorithm : 2

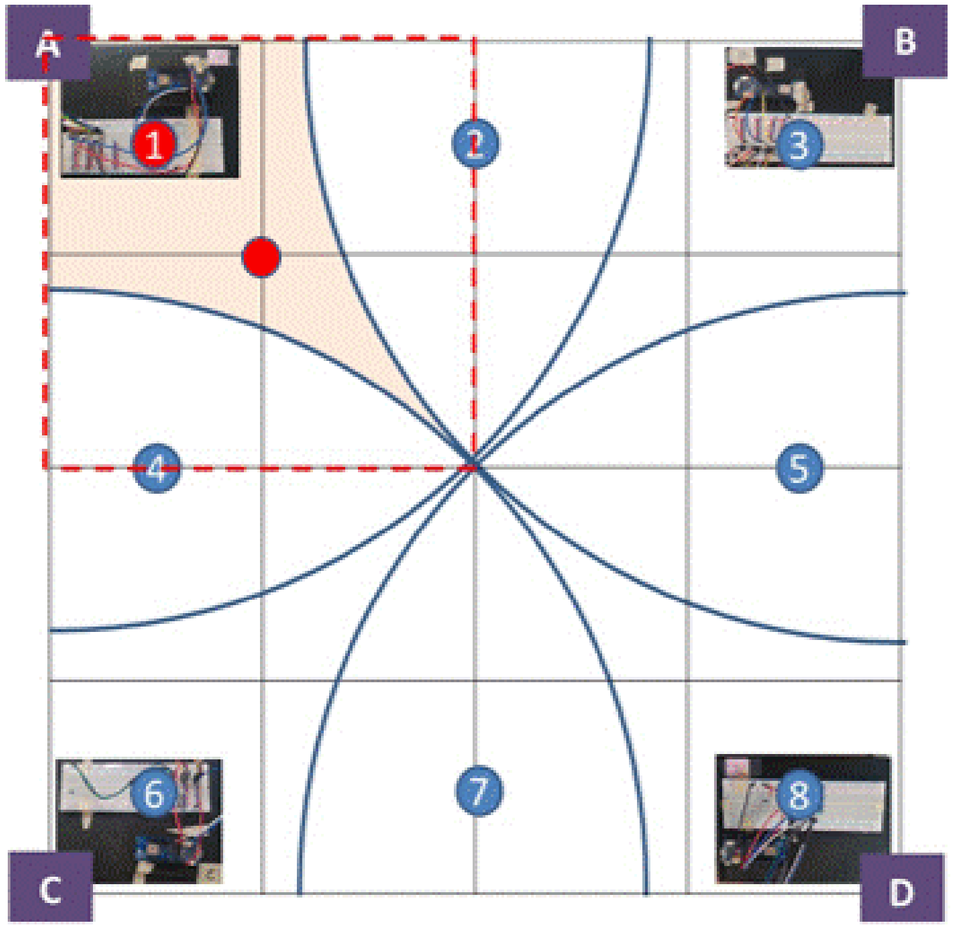
Central Arduino code : decision


```
54 void loop() {
55
56   recieve_packet();
57
58   //Serial.println("result");
59   Serial.print(A);
60   Serial.print(B);
61   Serial.print(C);
62   Serial.println(D);
```

```
64  //part decition depending on case
65  int result = 0;
66  if( A == 0 ){
67    if( B == 0 ){
68      if( C == 0 ){
69        if ( D != 0 ){ result = 8; }            // D   ( part 8 )
70      }
71      else {// C != 0
72        if ( D == 0 ) {result = 6;}             // C   ( part 6 )
73        else {result = 7;}                      // CD  ( part 7 )
74      }
75    }
76    else{
77      if ( D == 0 ) {result = 3;}               // B   ( part 3 )
78      else { result = 5;}                       // BD  ( part 5 )
79    }
80  }
81  else {
82    if( B == 0 ) {
83      if( C == 0 ){ result = 1;}                // A   ( part 1 )
84      else { result = 4; }                      // AC  ( part 4 )
85    }
86    else { result = 2; }                        // AB  ( part 2 )
87  }
88
```

Location Part Decision Algorithm : 3

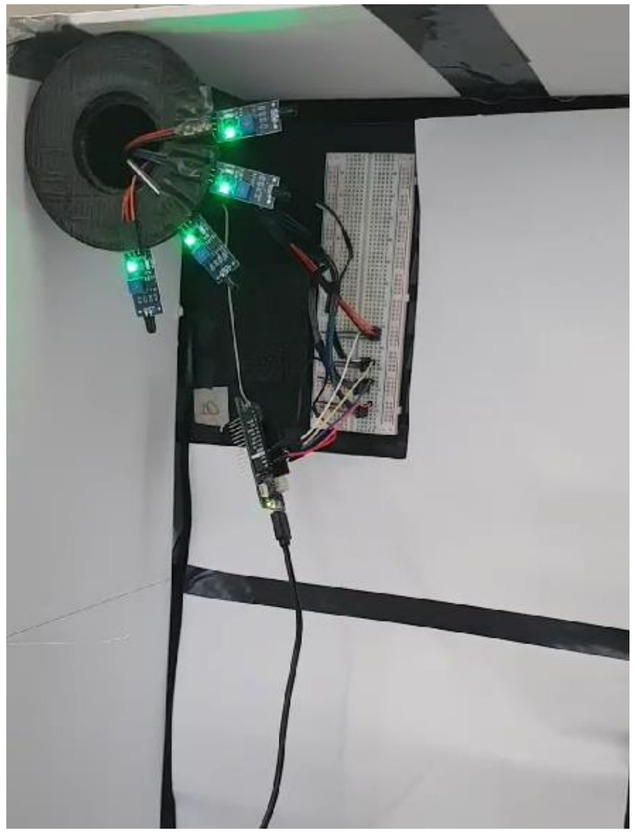
Decide which part occur fire

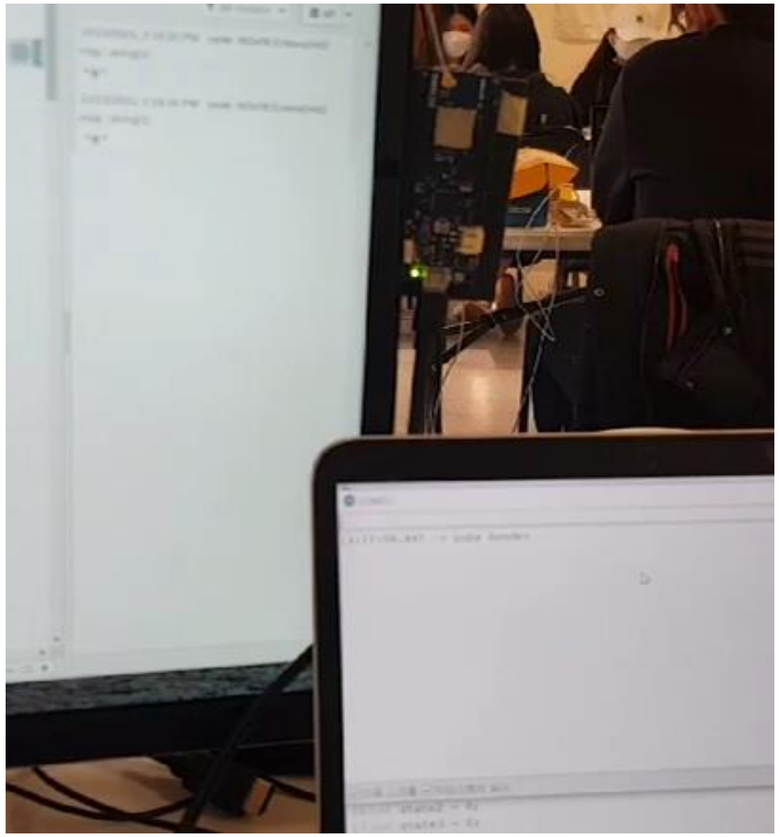


choose parts
send coordinates for each part to the drone.

Location Part Decision Algorithm : 4

Demo video : sending & receiving packet and getting result in the Raspberry Pi

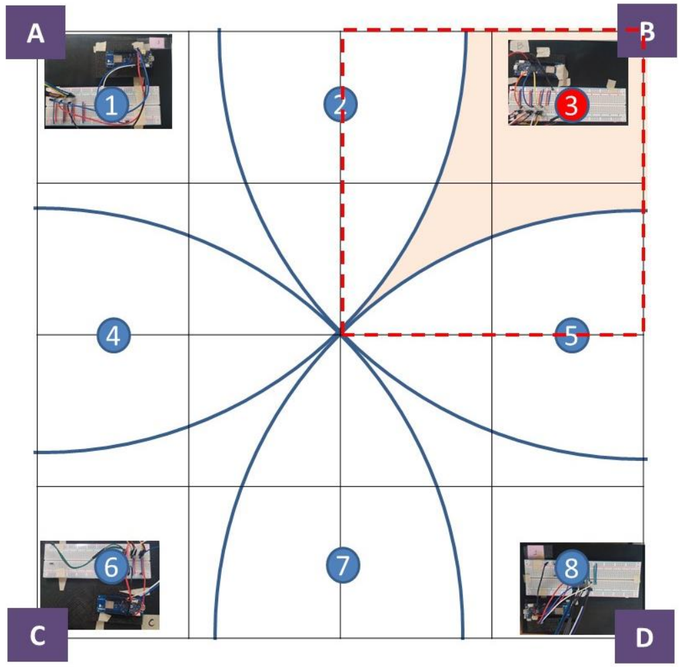


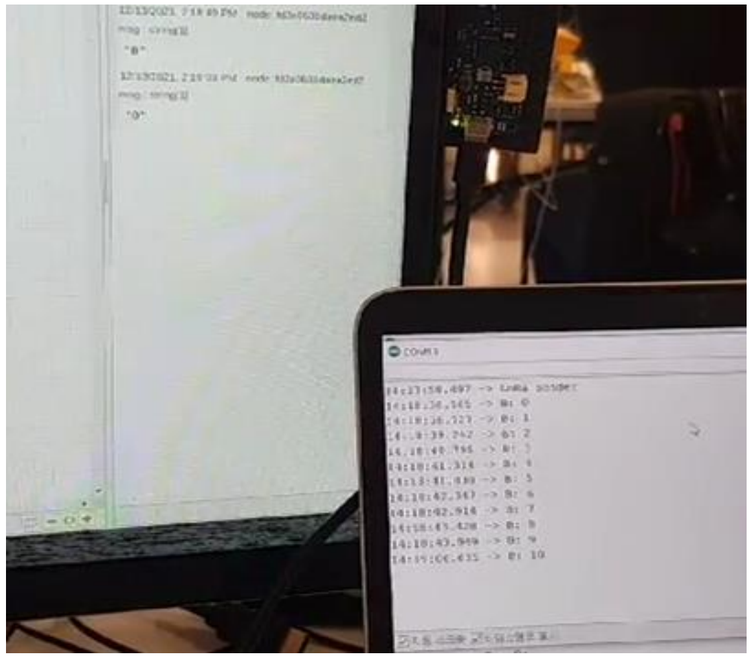


When only sensor B detects a fire,
The sensor board send packets and
Central Arduino receive packets

Location Part Decision Algorithm : 4

Demo video : sending & receiving packet and getting result in Raspberry Pi





represents the part where the fire broke out to be sent to the drone after receiving the result of which part.

03

Suppression & Reporting

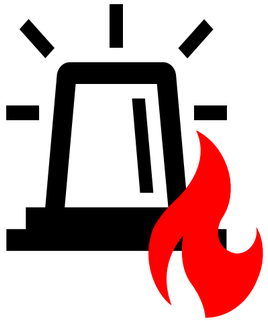
Limits & Effects

Problem & Future Plan

Review

The parts were divided into detection & suppression with reporting

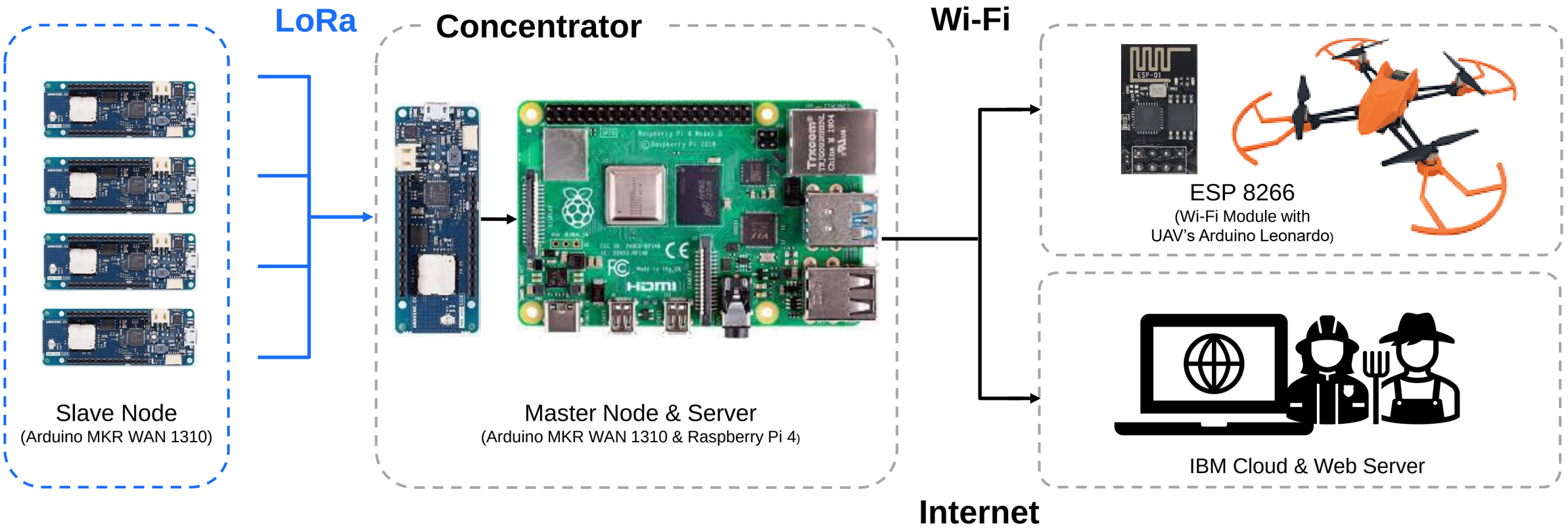
Detection



Suppression & Reporting

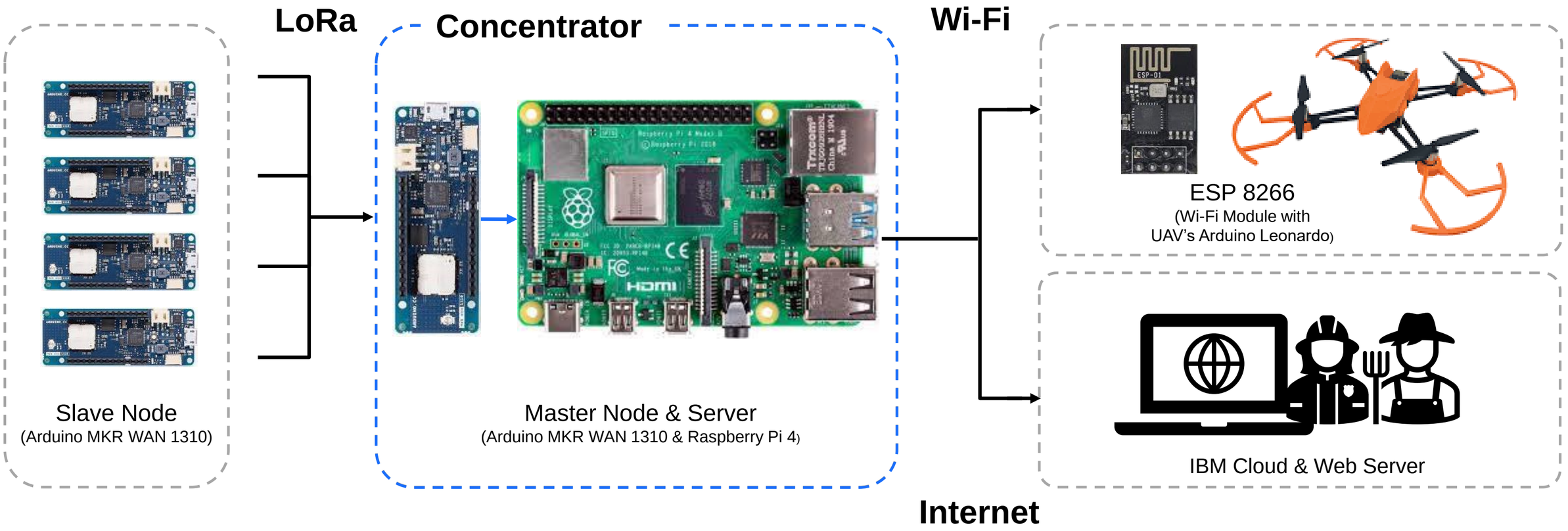
Review

Detection information is received from each slave node to the master node



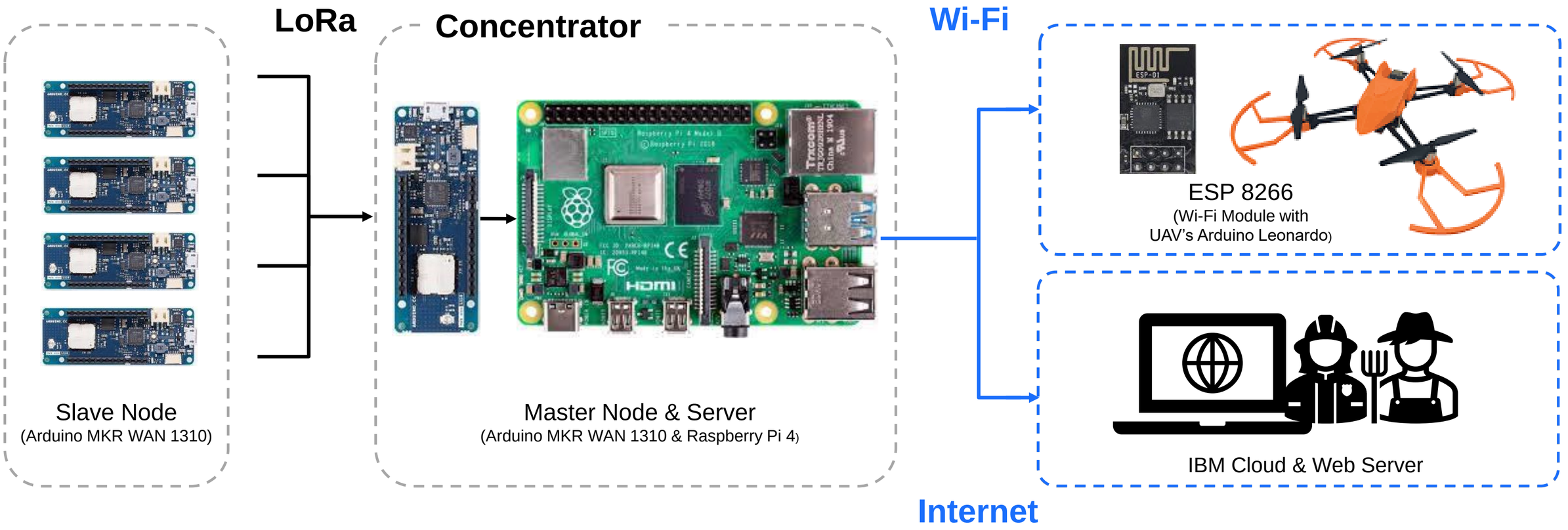
Review

The information is transferred from the master node to RPI and a section value is derived on the Node-RED



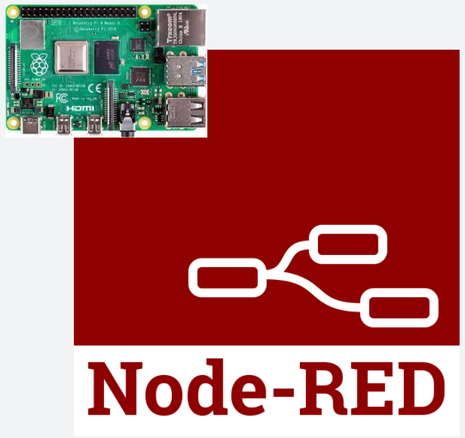
Review

Research has been conducted on the suppression part & reporting part



Suppression

Using Wi-Fi communication to deliver section values to the UAV



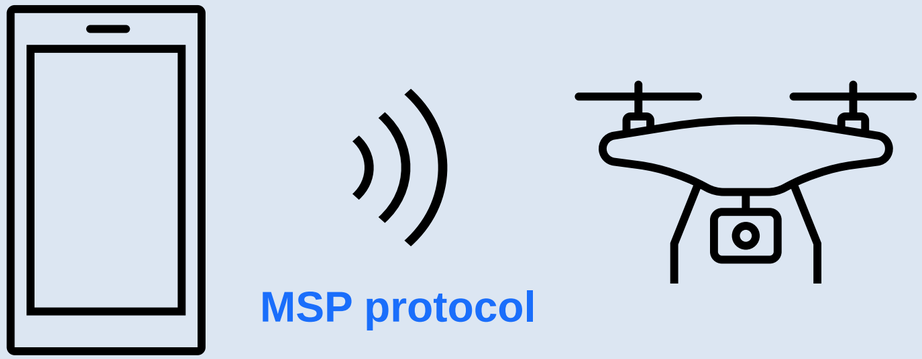
MQTT

Distribute information efficiently
Increase scalability
Reduce network bandwidth consumption
Reduce update rates
Suitable for remote sensing & control

Ref. https://inductiveautomation.com/resources/article/what-is-mqtt

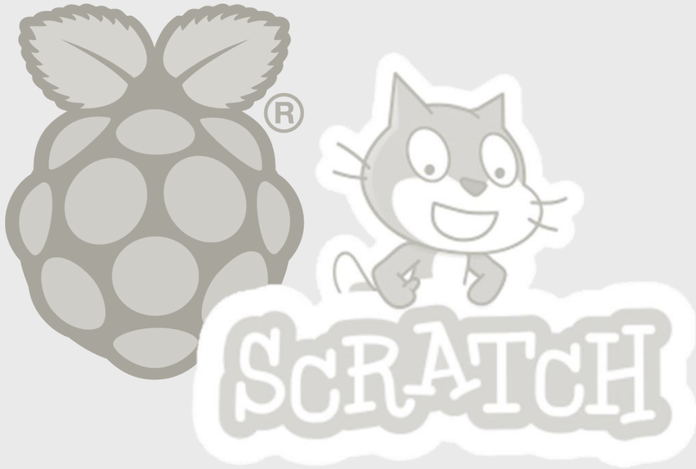
Problem

UAV must receive continuous values through the MSP protocol from the mobile phone application



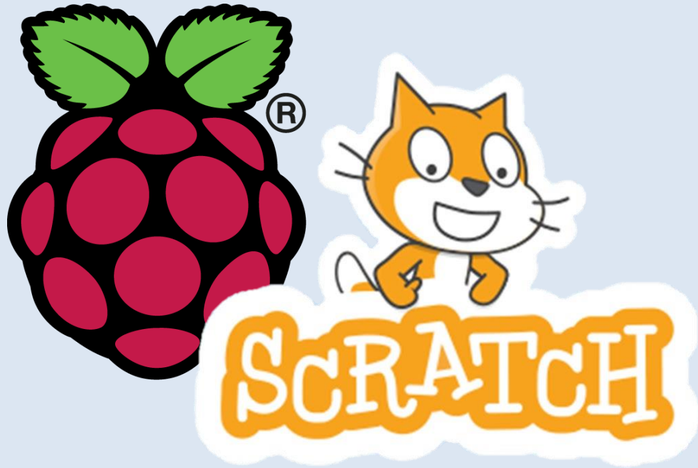
Multiwii Serial Protocol

Binary message based protocol
used for control, telemetry and sensors



Trial

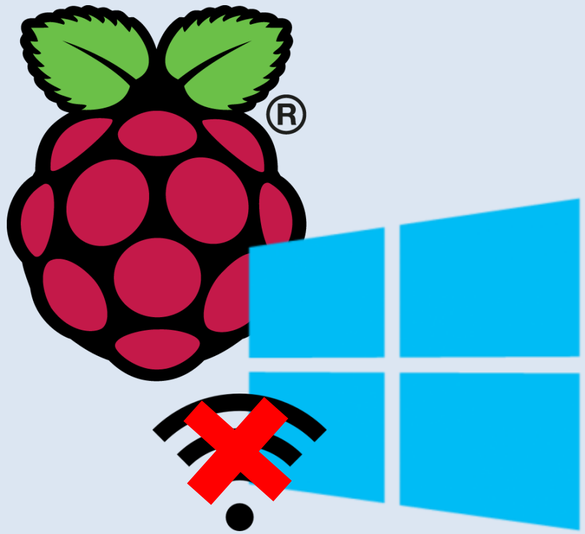
Install scratch2 on Raspberry Pi for automatic flight



Tried to use scratch to send MSP protocol
and drive way point

Problem

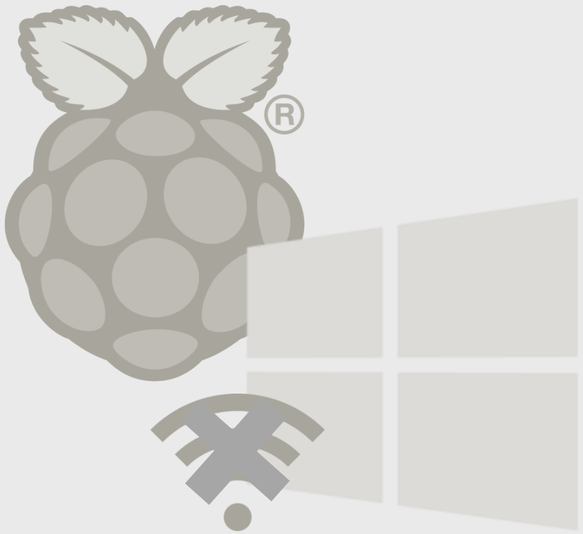
Windows-based files had to be executed so that Windows OS had to be operated on Raspberry Pi



Windows OS supported by Raspberry Pi does not afford Wi-Fi communication

Trial

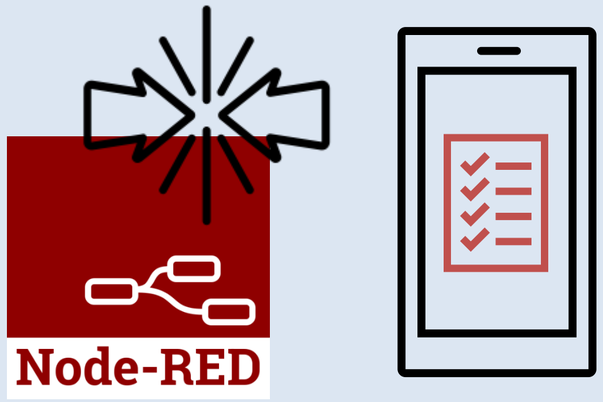
Raspberry Pi continuously transmits an MSP protocol signal



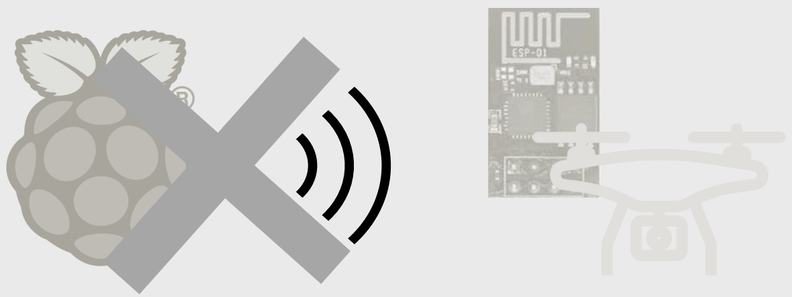
Raspberry Pi artificially generated
the MSP protocol to deceive ESP8266

Problem

It is difficult to determine the exact transmission method and frequency of the MSP protocol

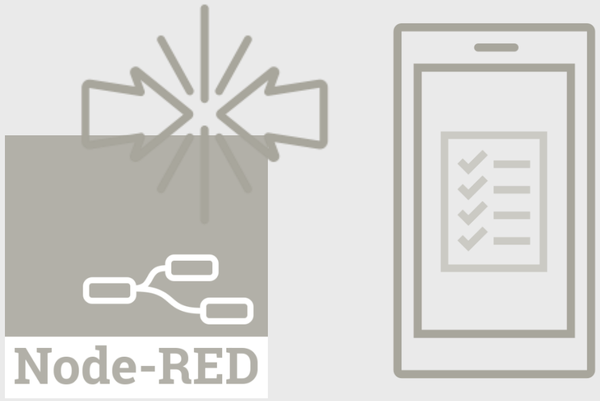


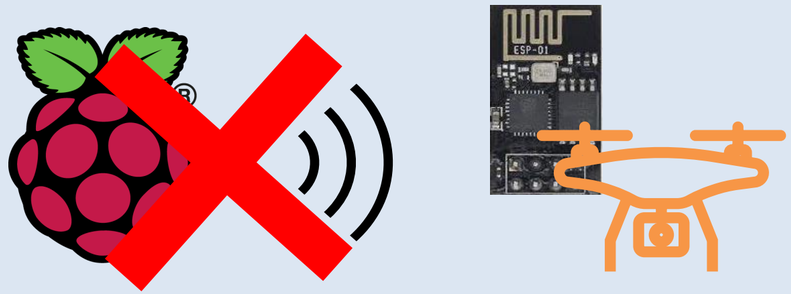
Collision occurred when installing MSP nodes, and it is difficult to properly grasp the format of protocols transmitted from existing apps



Trial

Raspberry Pi continuously transmits an MSP protocol signal

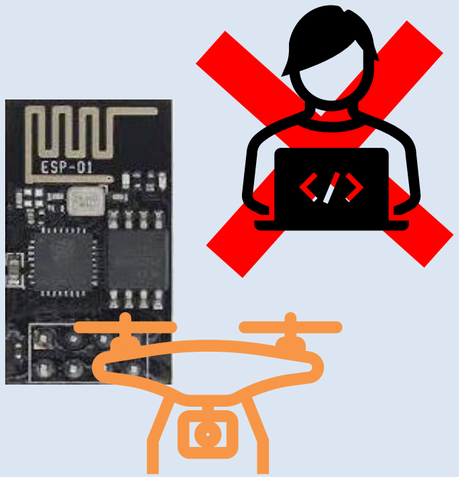




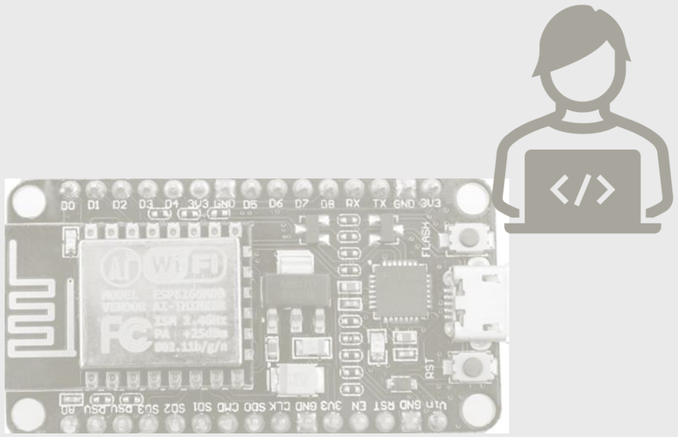
Raspberry Pi artificially generated
the MSP protocol to deceive ESP8266

Problem

Modify the flight code of the UAV and use MQTT by ignoring MSP protocol

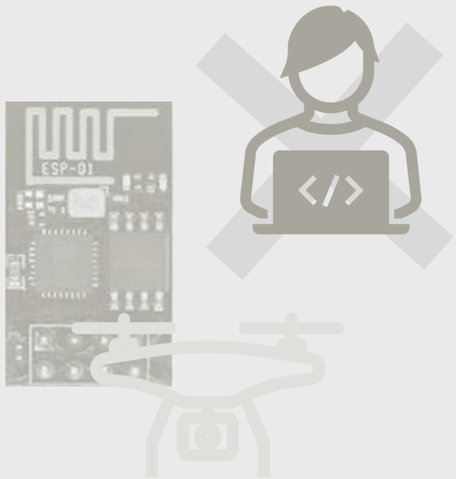


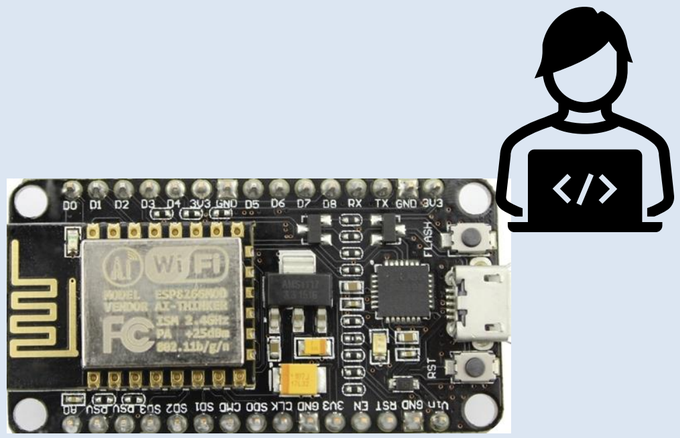
The esp8266-01 model had a built-in code
that could communicate in advance,
but could not upload a new code to the model.



Trial

Using another esp8266 model

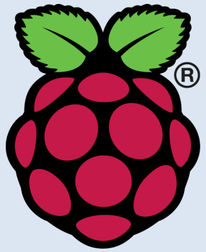


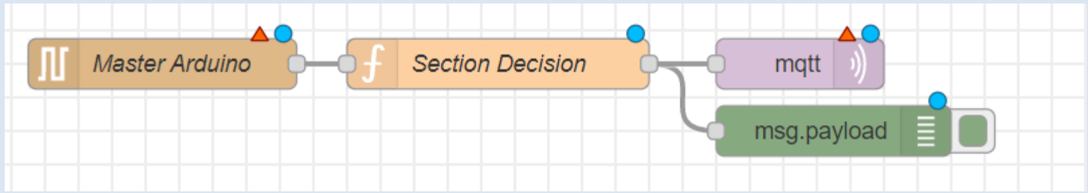


Decided to transmit data to another model of esp8266 other than the ESP-01 attached to the UAV

Final

Using another esp8266 model

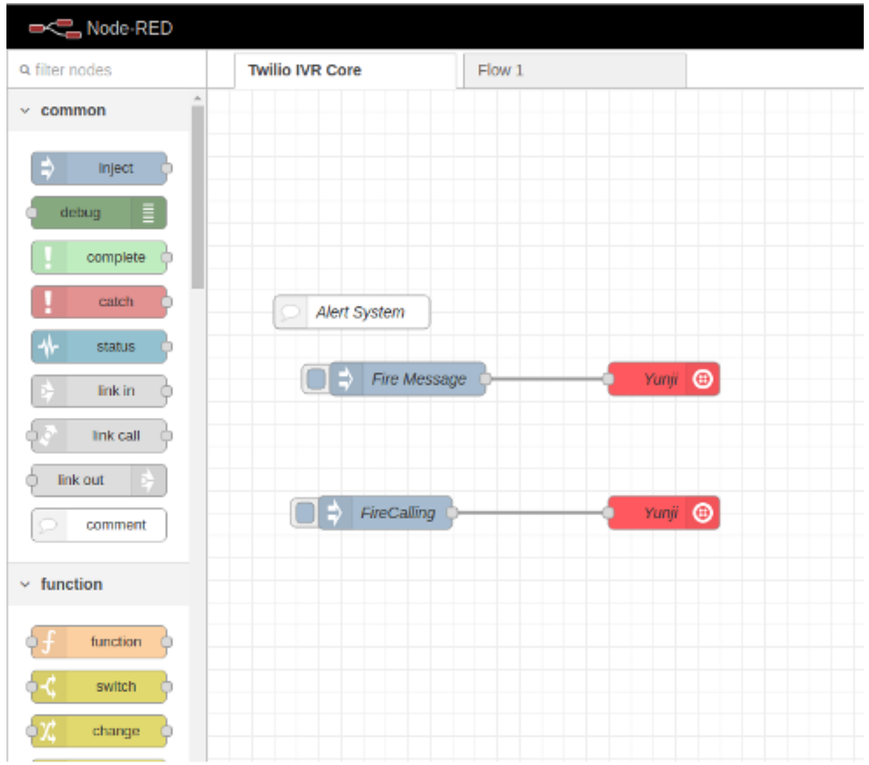


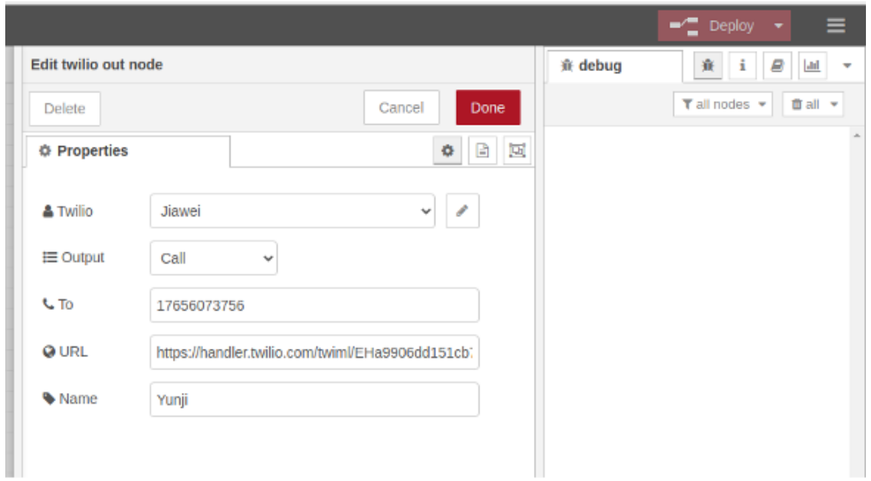


Section values were sent to esp8266 using MQTT node of Node-RED.

Report

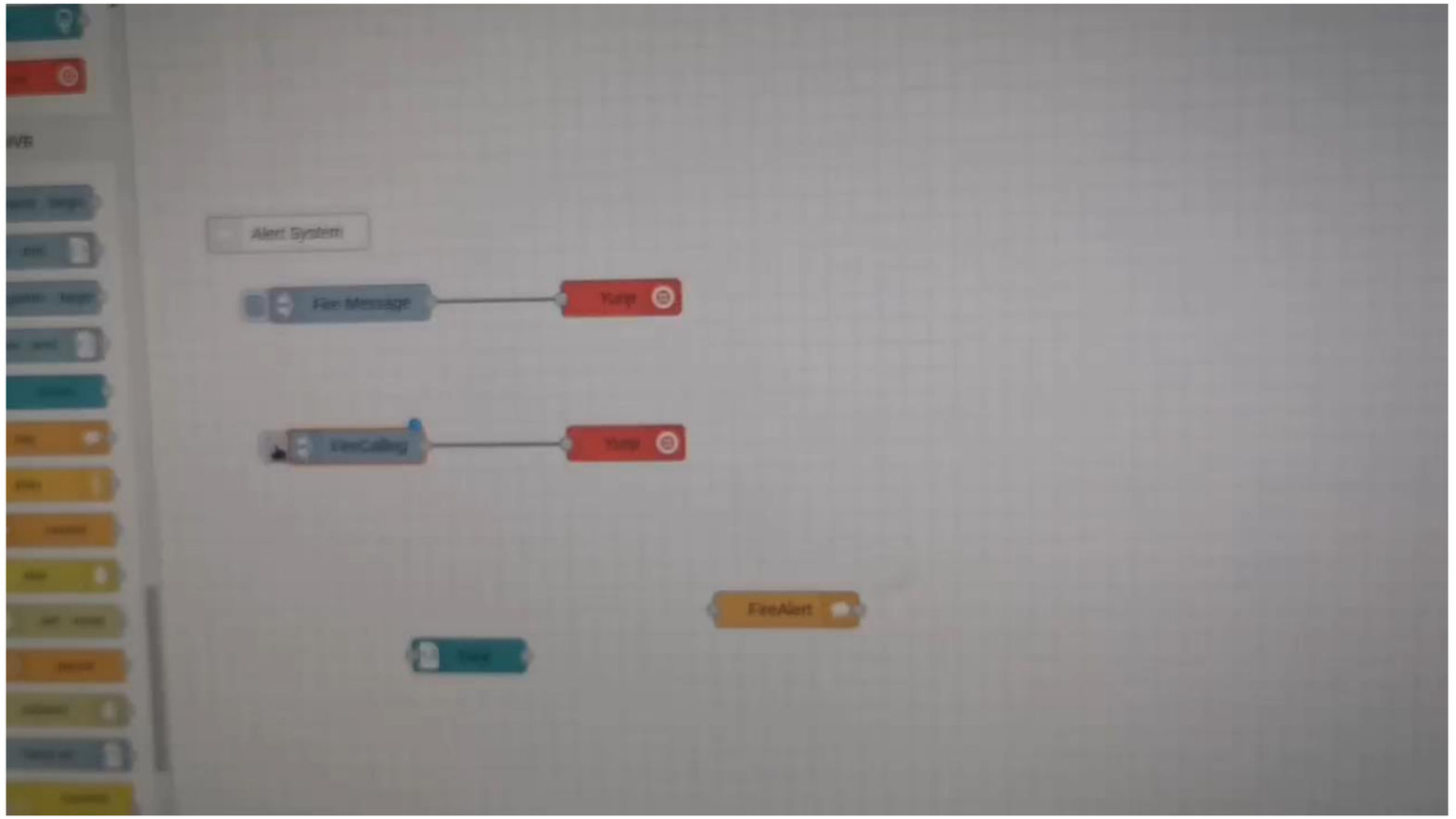
The Twilio node was used to allow phone calls and text messages to be transmitted





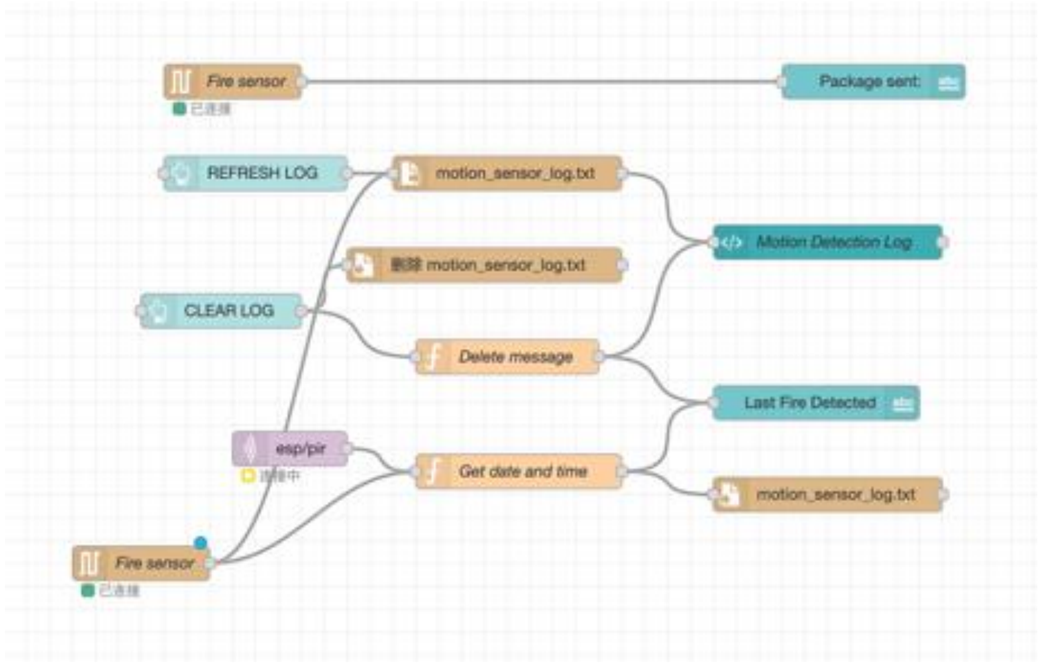
Report

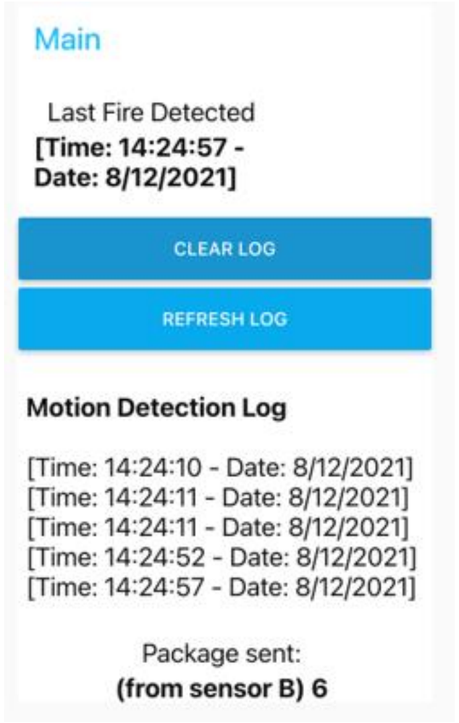
The Twilio node was used to allow phone calls and text messages to be transmitted



Report

Information on the time and location of the fire can be received on the mobile screen





04

UAV

UAV

UAV Communication

UAV CODE

UAV Flying Code

```
31 void loop() {
32
33 //  Serial.println(cnt);
34   if (cnt > 10000) {
35     cnt = 0;
36   }
37
38   cnt++;
39
40   if ((int16_t)(currentTime - rcTime) > 0 ) { // 50Hz
41
42     rcTime = currentTime + 20000; // 20ms
43
44     computeRC(); // rcData[]에 값 받기. 조종기 신호 처리
45
46     failsafeRoutine();
47
48     heightDesign(); // 조종기 값을 통해 고도 조절
49
50     //stickCommand();
51
52     makeFlightmodes(); // anale. horizon 등 비행 모드 설정
```

```
    else {
      if (checksum[port] == c) { // compare calculated and transferred checksum

          evaluateCommand(cmdMSP[port]); // we got a valid packet, evaluate it
        if (cnt > 1000 & cnt < 1100){
          evaluateCommand(MSP_ACC_CALIBRATION);
          evaluateCommand(MSP_MAG_CALIBRATION);
        } else if (cnt > 3000 & cnt < 3100){
          evaluateCommand(MSP_ARM);
        } else {
          evaluateCommand(MSP_SET_RAW_RC_TINY); // we got a valid packet, evaluate it
        }
      }
      state = IDLE;
      cc = 0; // no more than one MSP per port and per cycle
    }
  }
  c_state[port] = state;
} // while
// for
```

CALIBRATION

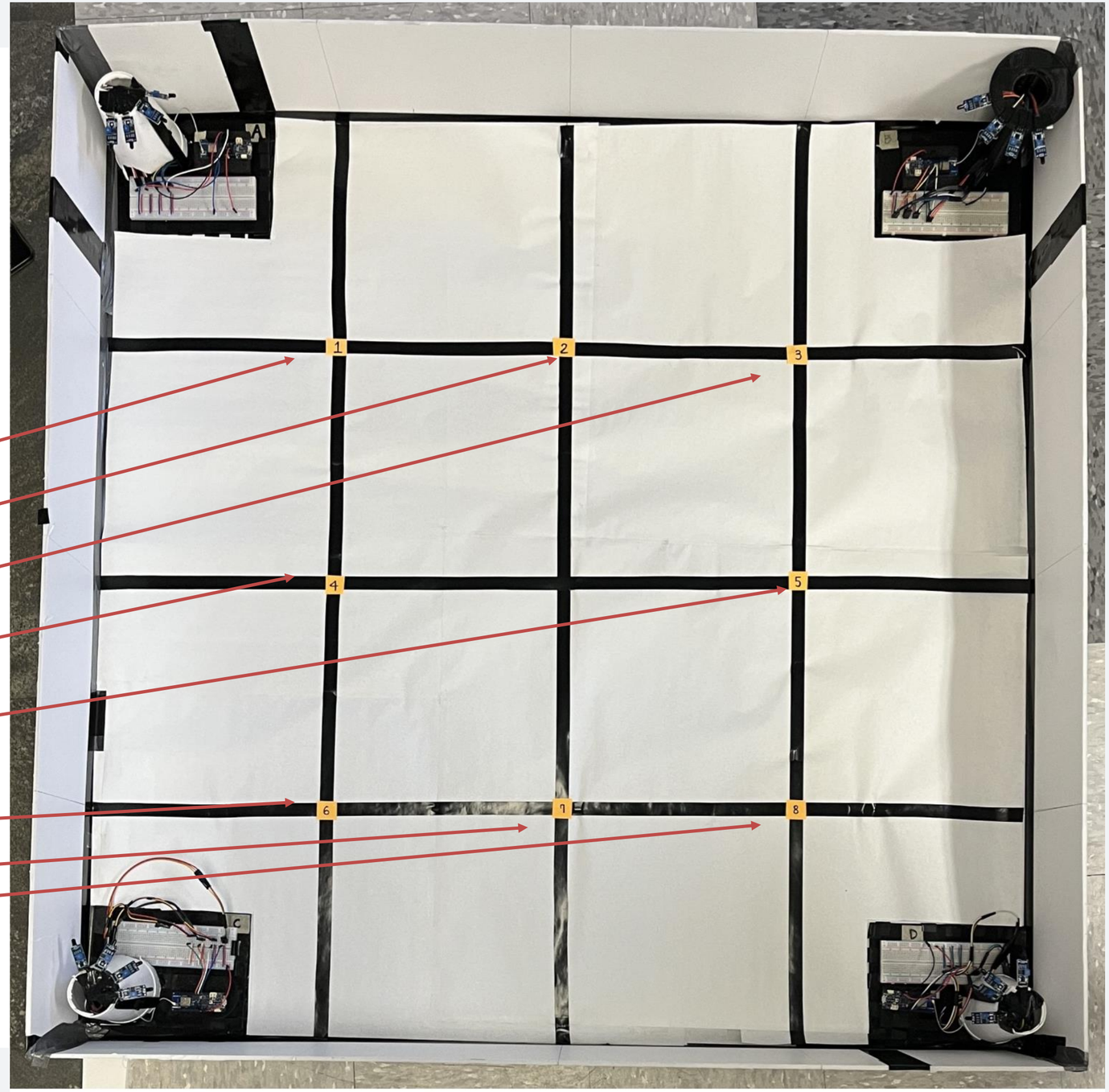
ARM

FLIGHT

```
aux = (auxChannels & 0x0c) >> 2;
if (aux == 0) {
  rcSerial[6] = 1000;
} else if (aux == 1) {
  rcSerial[6] = 1500;
} else {
  rcSerial[6] = 2000;
}

aux = (auxChannels & 0x03);
if (aux == 0) {
  rcSerial[7] = 1000;
} else if (aux == 1) {
  rcSerial[7] = 1500;
} else {
  rcSerial[7] = 2000;
}
for (uint8_t i = 0; i < 8; i++) {    // 지우기
  rcSerial[i] = 1500;
}


switch (coord) {
  case 0:
    node_0();
  case 1:
    node_1();
  case 2:
    node_2();
  case 3:
    node_3();
  case 4:
    node_4();
  case 5:
    node_5();
  case 6:
    node_6();
  case 7:
    node_7();
  case 8:
    break;
}
```

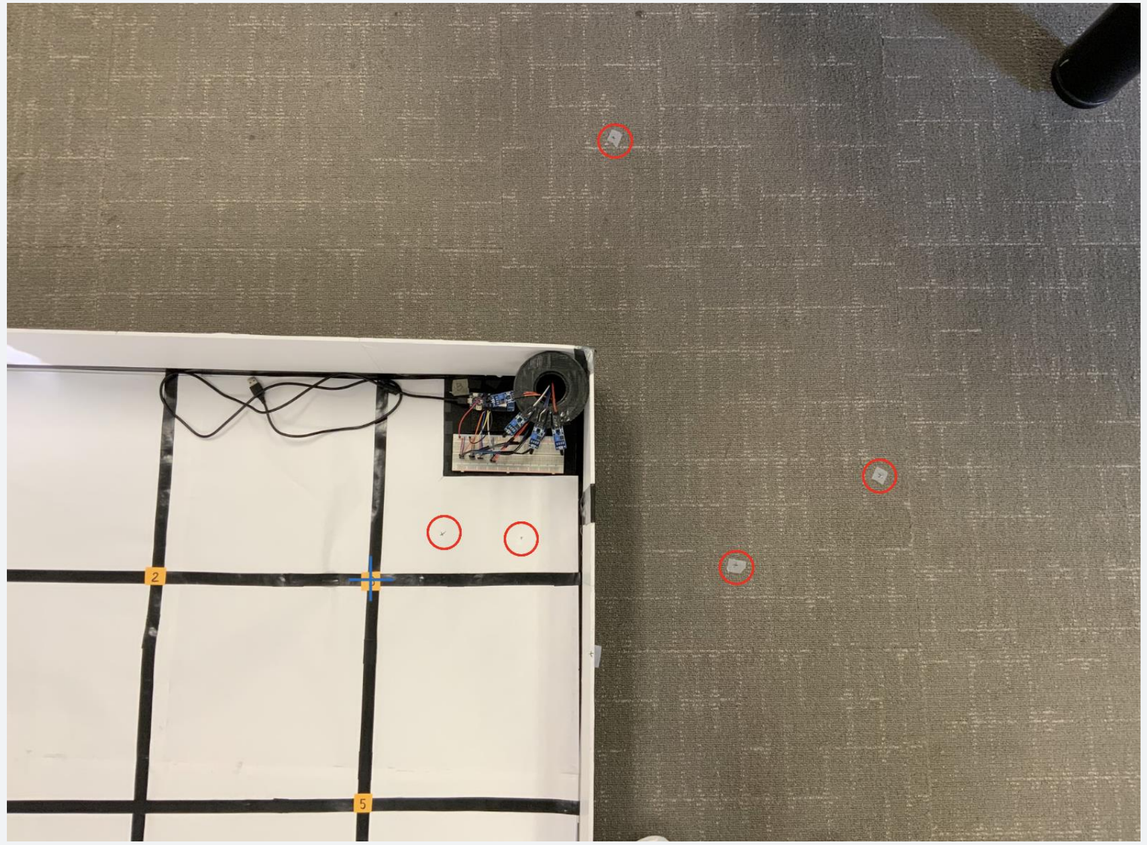


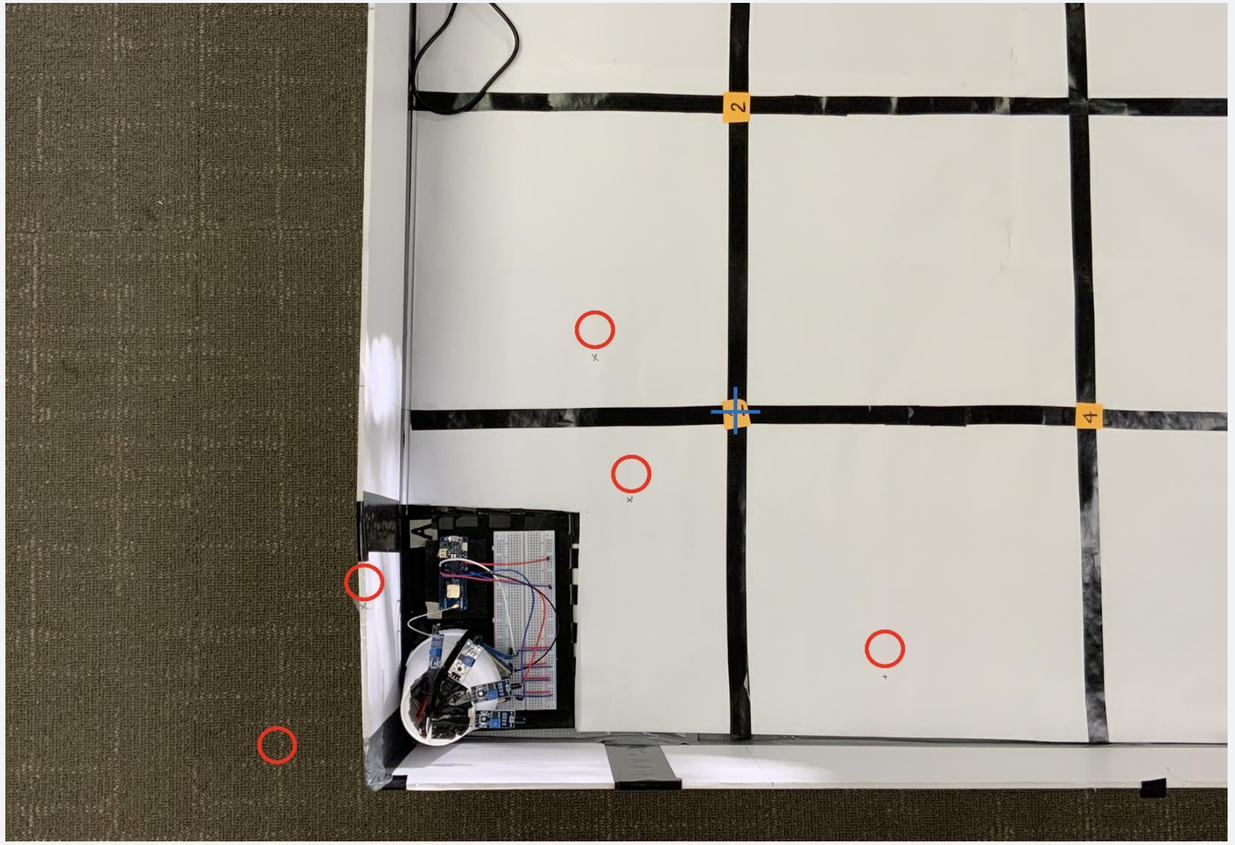
UAV

UAV Demo Video

UAV

Result of UAV Flying





UAV

UAV Fire Suppression Range

Firefighting UAV

Ref : https://ko.aliexpress.com/item/4000218952434.html

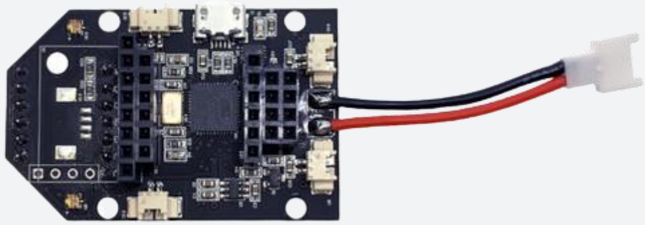
80cm x 80cm : 5 m

Kiwi Drone

20cm x 20cm : 1.25 m

UAV

IR Camera - MLX 90640

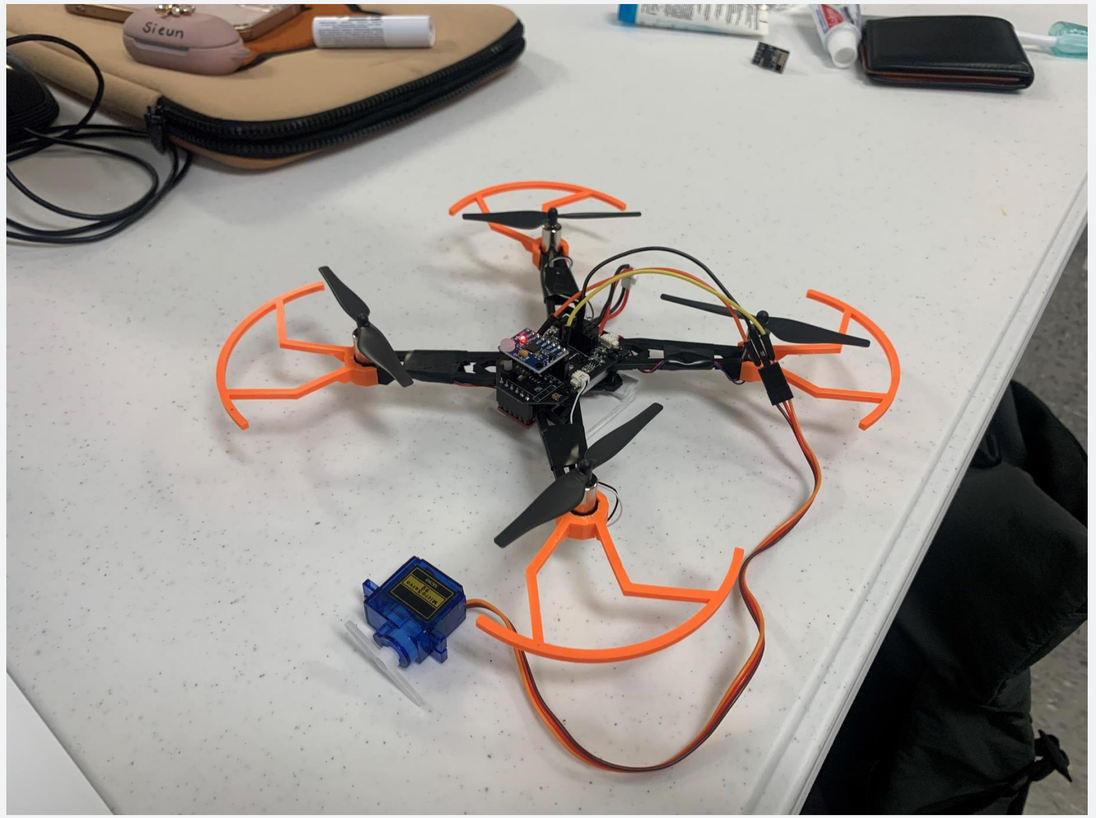


adafruit

Ref : https://www.amazon.com/MLX90640-Interface-Compatible-Raspberry-MLX90640-D55/dp/B07ZMP995T
https://www.arrow.com/en/manufacturers/adafruit-industries
https://blog.naver.com/codingbird/221794916251

UAV

Ball Drop Device



05

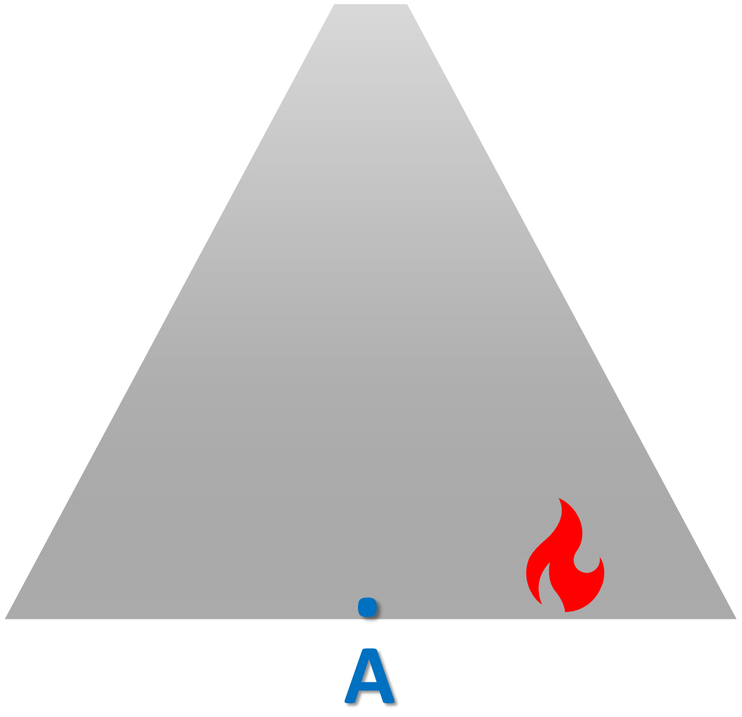
UAV Movement Adjustment

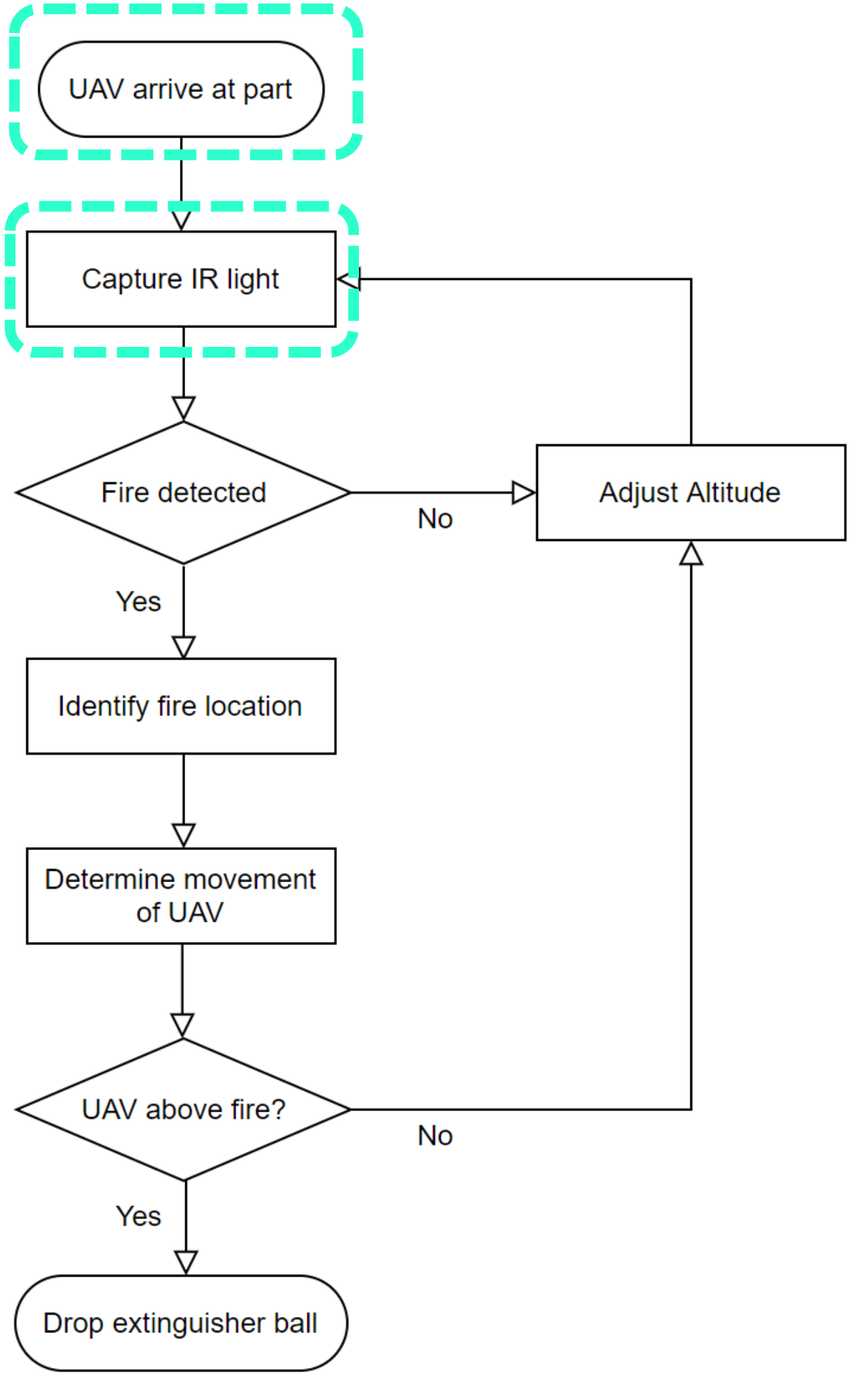
Algorithm

Python Code

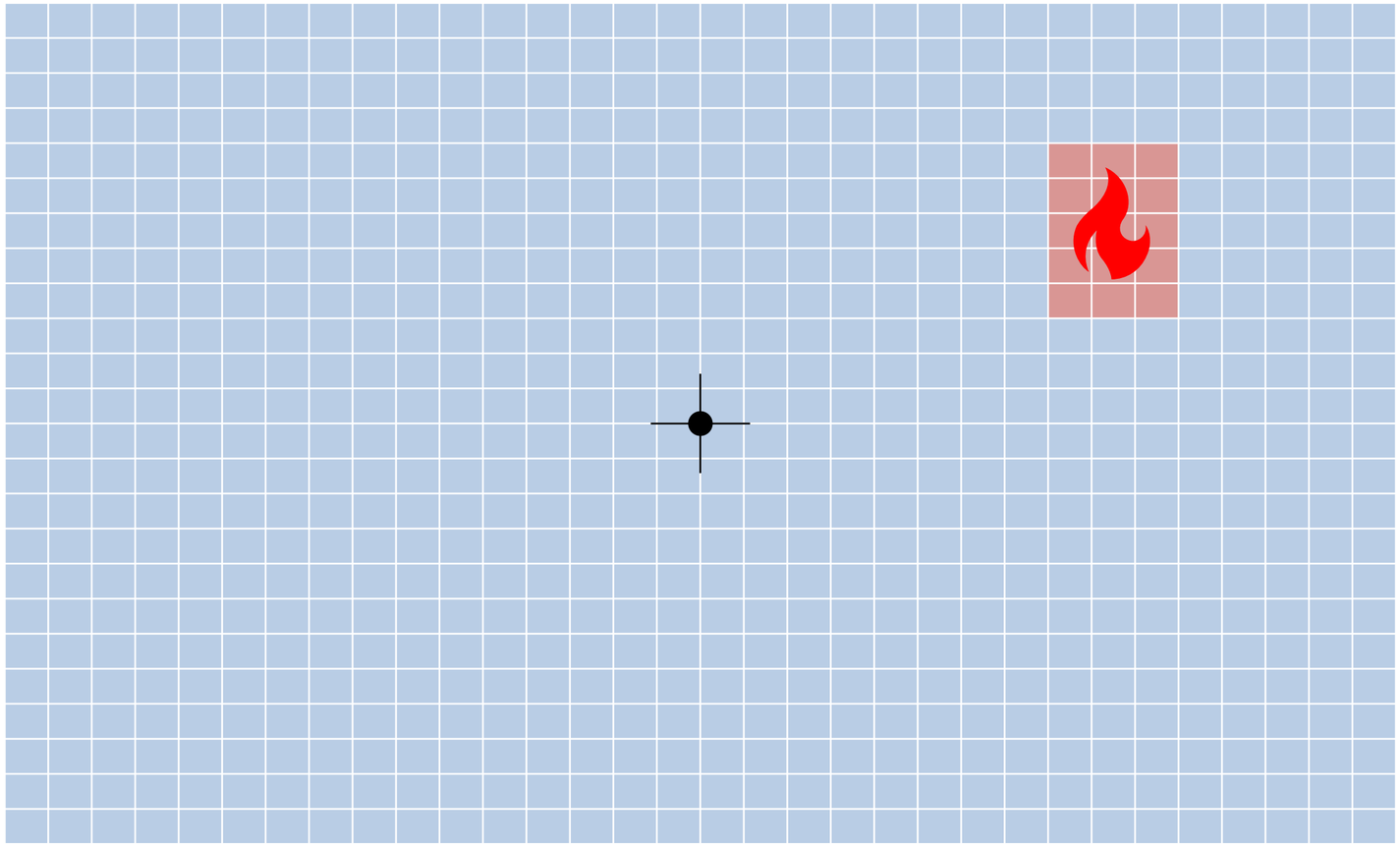
Demo Video

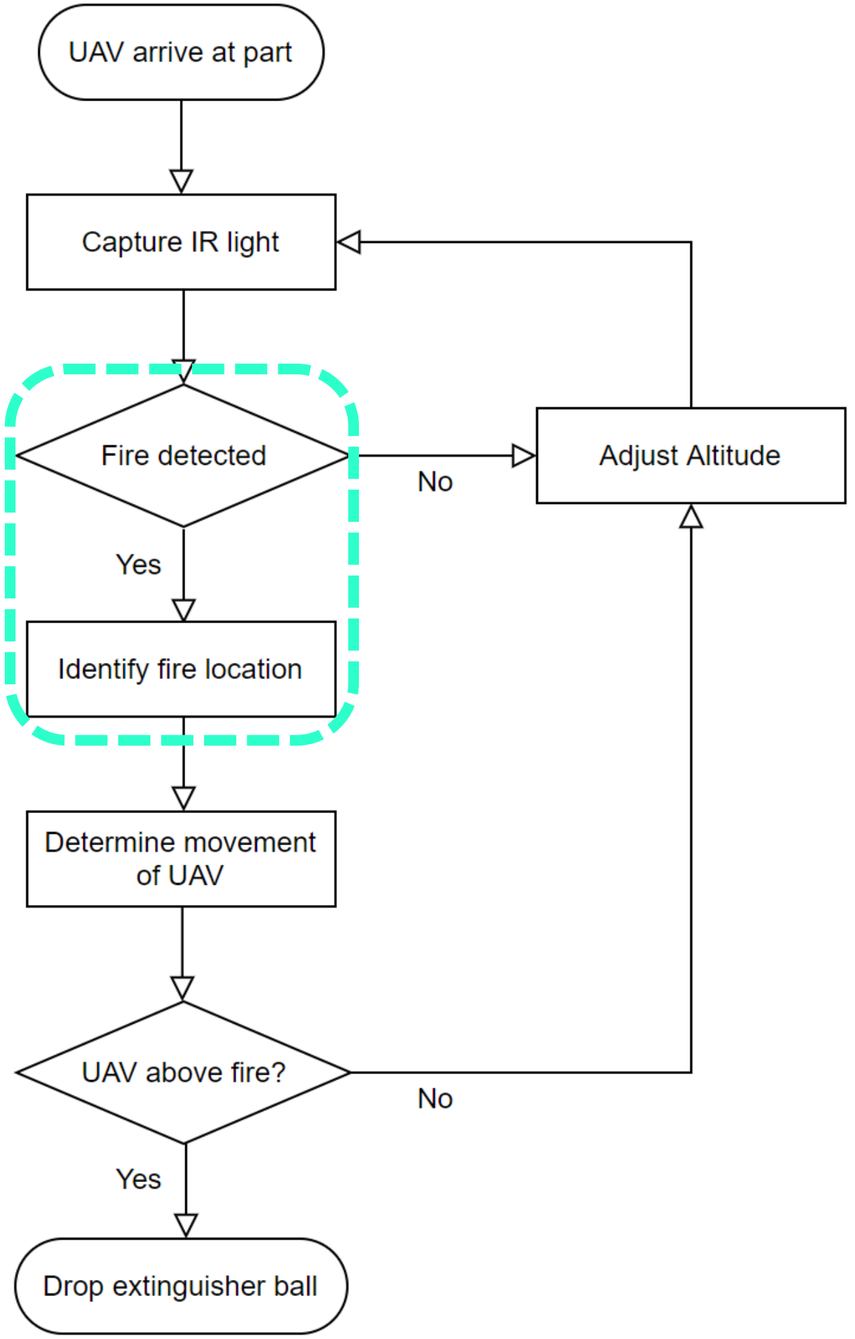
UAV Movement Adjustment Algorithm



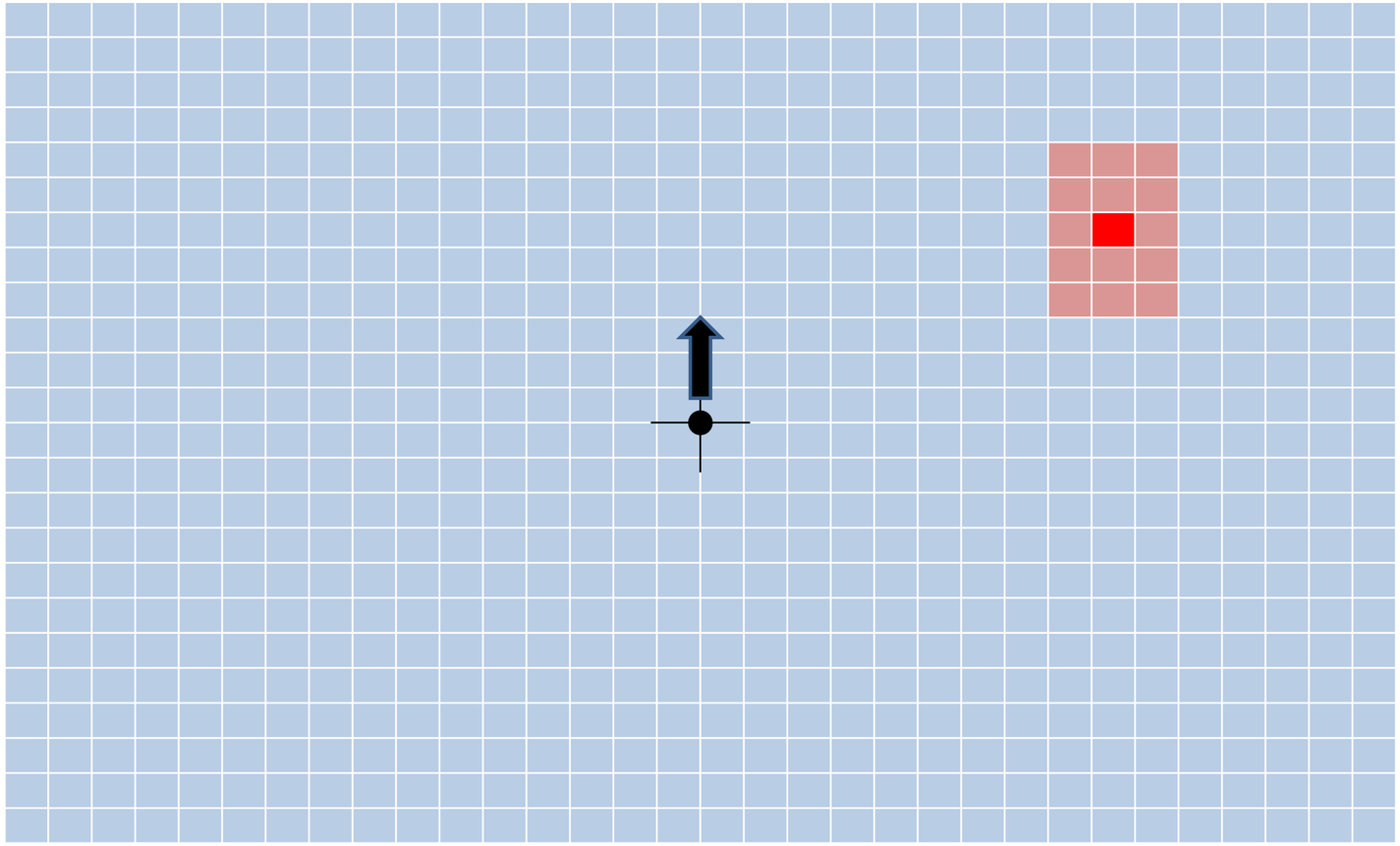


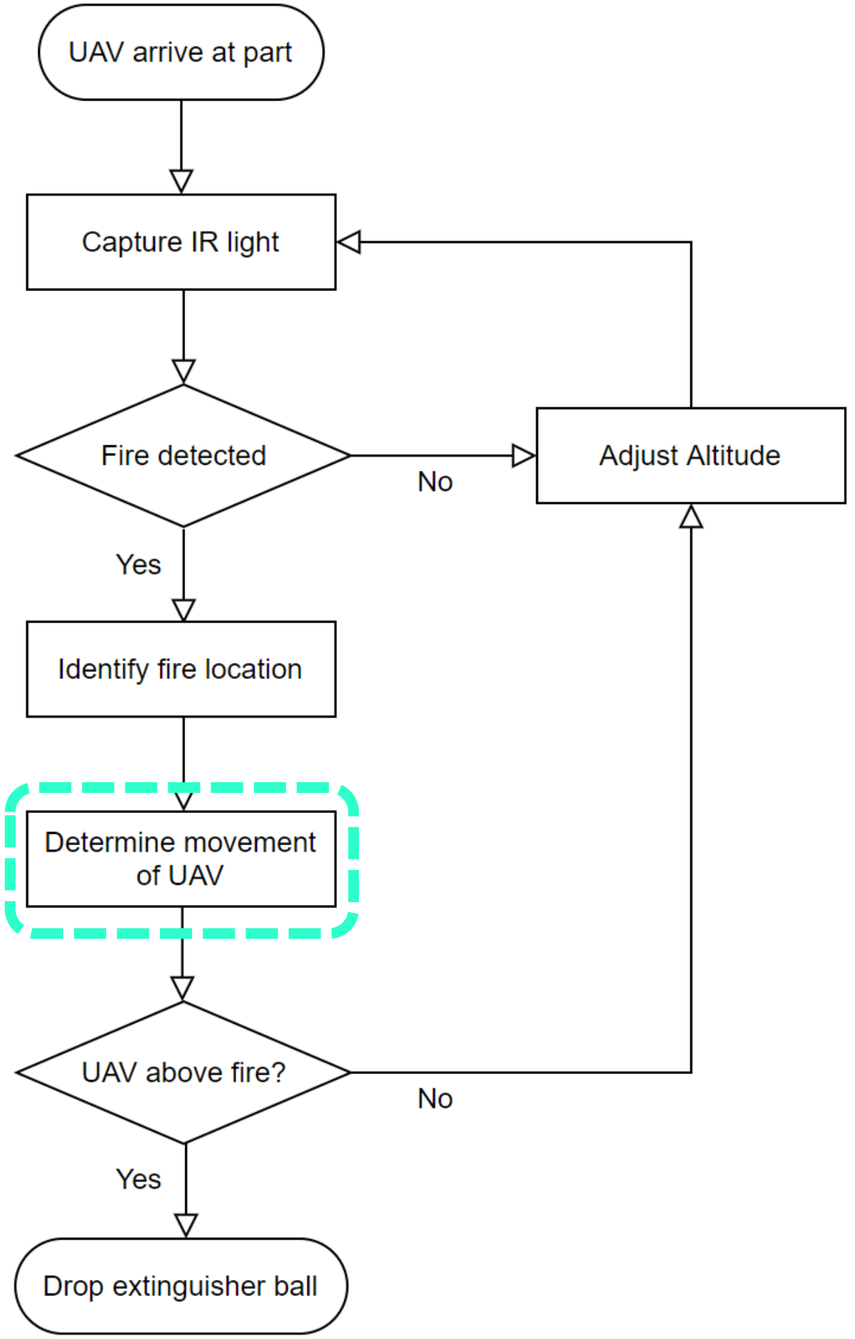
UAV Movement Adjustment Algorithm



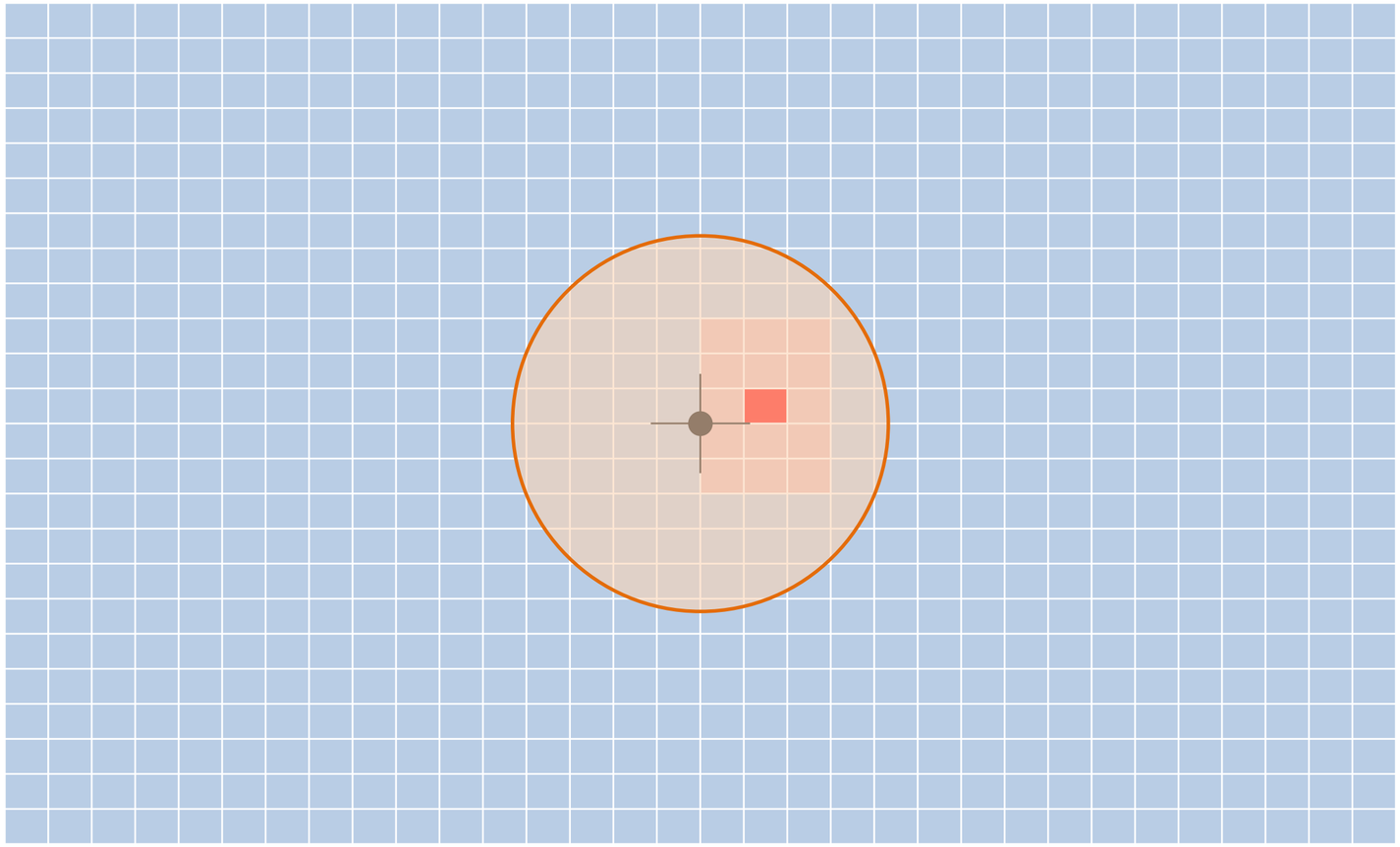


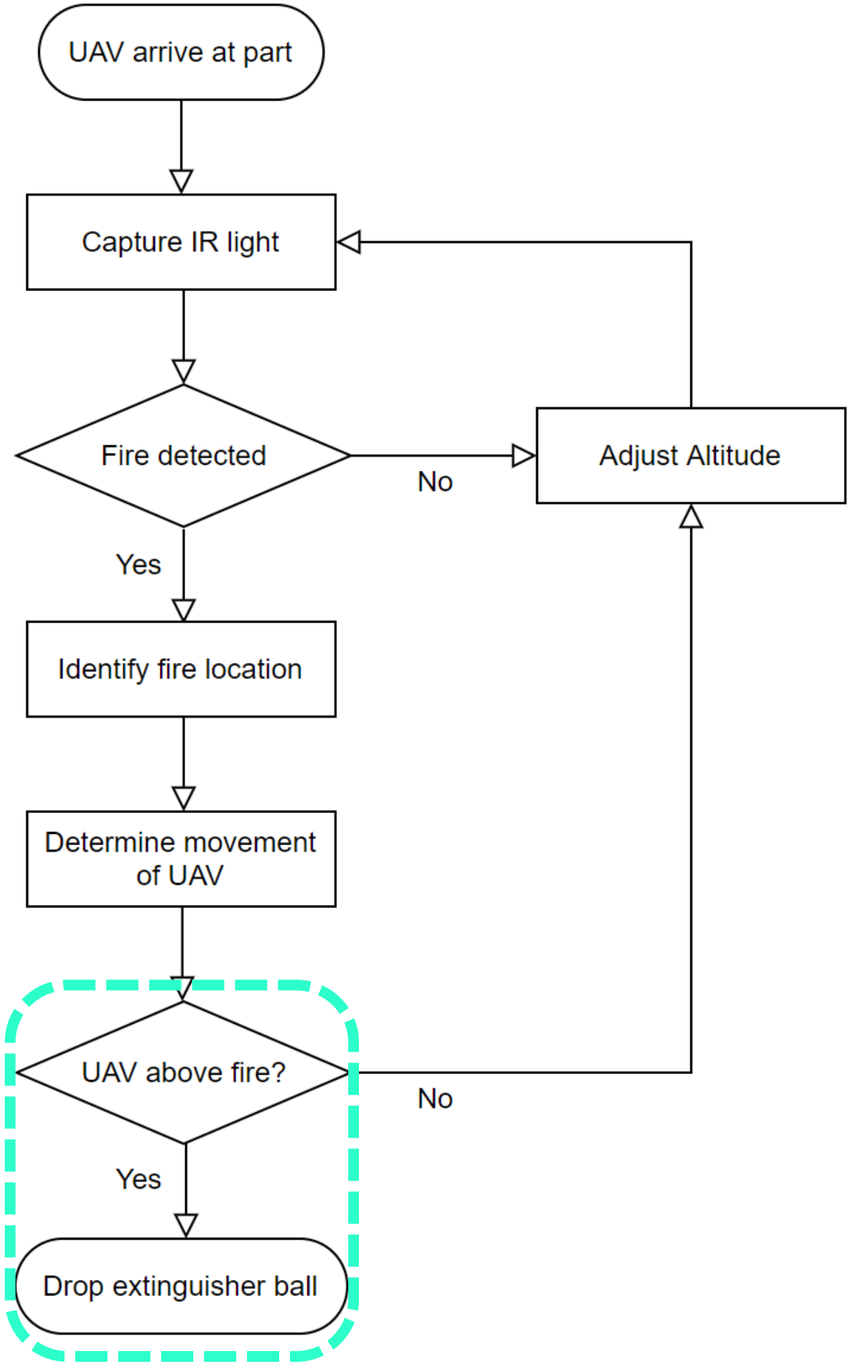
UAV Movement Adjustment Algorithm





UAV Movement Adjustment Algorithm





UAV Movement Adjustment Python Code : 1

Defining the fire area & Printing IR values

IR Graph

- The infrared values of the pixels are printed in the form of a graph.
- The fire area is defined by the pixels with an IR value over 50.

```
for h in range(24):
    for w in range(32):
        t = int(frame[h*32 + w]) # IR value
        if t > 50: # if t>50, fire
            t = '██' # fire symbol
            y_Coor.append(h) # add to fire x coordinate list
            x_Coor.append(31-w) # add to fire y coordinate list
        graph[h][31-w] = t


# drone location(x_drone, y_drone) = graph[12][16] -> fixed
graph[12][16]='▓▓' # drone location symbol -> fixed

num = len(x_Coor) # = len(y_Coor): num of fire pixels
```

UAV Movement Adjustment Python Code : 2

When a fire is detected

Variable Setting

```
        # fire detected
        try:
            x_avg = sum(x_Coor) // num # average x coordinate of fire pixels
            y_avg = sum(y_Coor) // num # average y coordinate of fire pixels

            graph[y_avg][x_avg] = "▒" # symbol for average coordinate of fire values


            # set drone location(x_drone, y_drone) as (0,0)
            # distance is calculated by subtracting fire average coordinate from drone location
            # moving downwards/backwards = minus
            # moving upwards/forwards = plus

            # if drone location is (1,6) and fire average coordinate is (4,2), next movement is (3,-4)

            x_move = x_avg - x_drone # next x movement of drone
            y_move = -(y_avg - y_drone) # next y movement of drone
            z_move = 0 # altitude of drone. Initial value is 0.
                       # If altitude change needed, value is +k or -k (when k is constant value)

            avrg_coor = [x_avg, y_avg] # average coordinate of fire pixels
            next_move = [x_move, y_move, z_move] # next movement of drone
```

UAV Movement Adjustment Python Code : 3

When a fire is not detected

Altitude Change

- If a fire is not detected, the UAV raises its altitude to capture more pixels.
- If a fire is not detected even after raising the altitude four times, it is regarded as the sensor detection error.

```
# fire not detected - altitude change needed
except:
    print("**************************** fire not detected ****************************")
    no_detect_count+=1
    next_move = [0,0,3] # set k=+3. drone has to go higher to capture more pixels(wider view)

    if no_detect_count > 4: # not problem of altitude. Sensor error
        print("!!!!! Sensor Error !!!!!")
        break # escape loop(quit program)
    else:
        time.sleep(1)
        print(str(no_detect_count)+") Next Movement = " + str(next_move))
        print()
        continue
```

UAV Movement Adjustment Python Code : 4

When a fire is detected

Ball Dropping

- If a fire is detected within the error range, no location adjustment is needed and the motor is operated to open the box and drop the fire extinguisher ball.

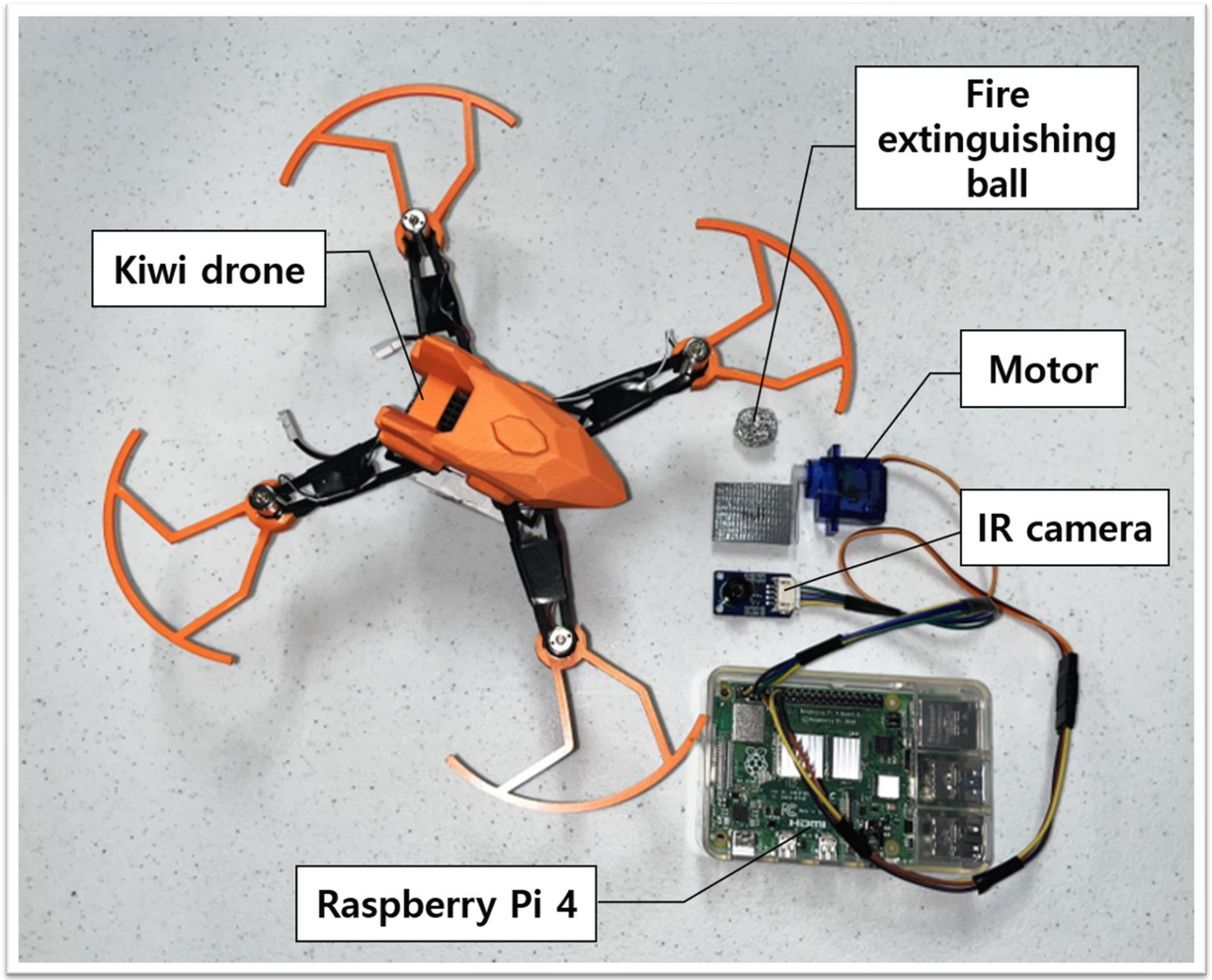
```
if (0 <= abs(x_move) < 6) and (0 <= abs(y_move) < 6):
    detect_count+=1
    print()
    print("------------------------------ Drop the ball ------------------------------")
    print()

    time.sleep(2) # wait until drone go right above fire
    # open box to drop ball
    setServoPos(0) # servo 0 degree
    print("Dropping - box opening")
    time.sleep(2) # wait 2s
    # close box
    setServoPos(180)
    print("Finished - box closing")
    time.sleep(2) # wait 2s
  # stop servo PWM
    servo.stop()
  # reset GPIO mode
    #GPIO.cleanup()
    break
```

```
if detect_count > 1: # if fire not detected at first capture, altitude change needed
    next_move[2] = -3 # lower altitude(drone go downwards)
```

UAV Movement Adjustment Experiment

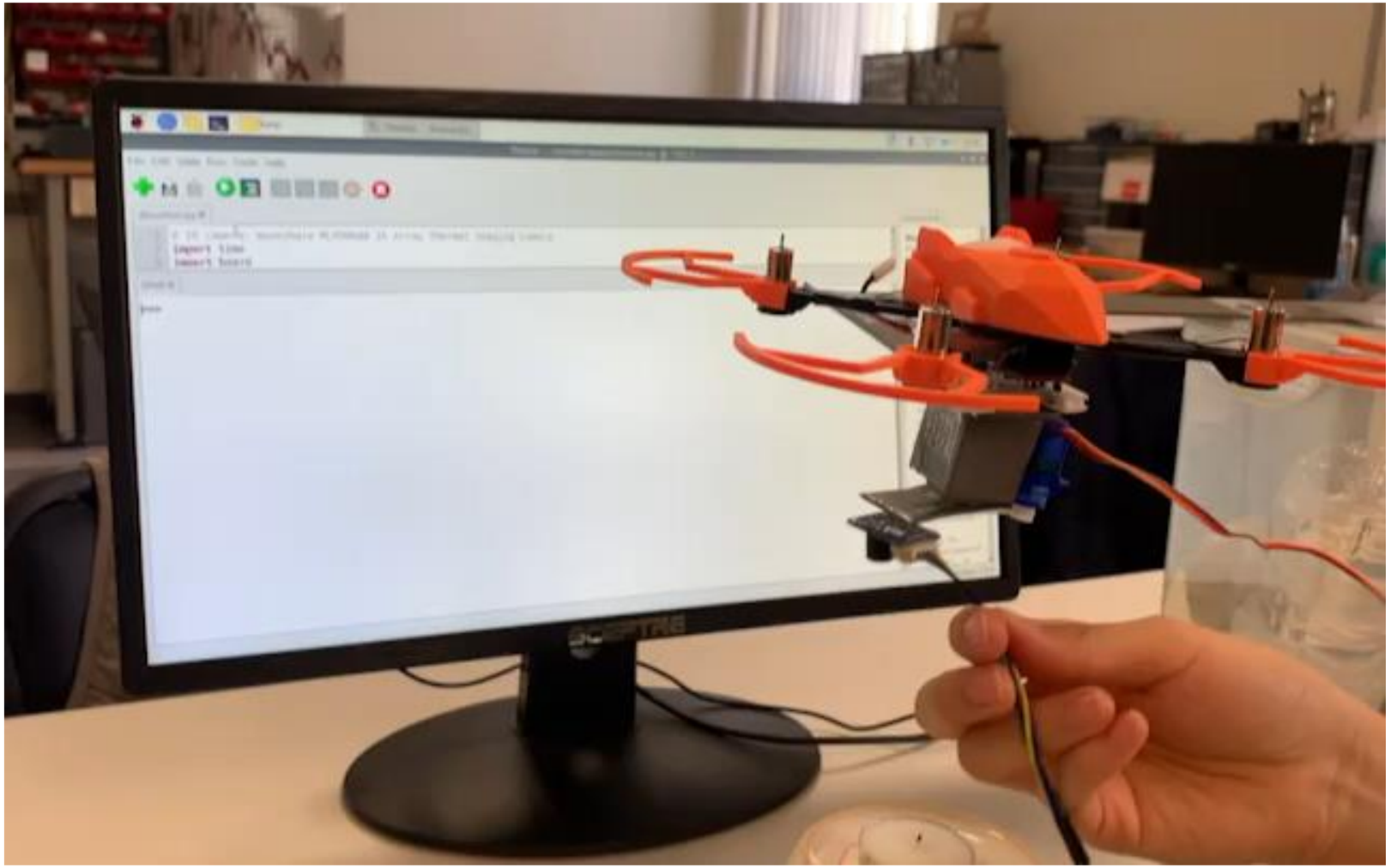
Equipment



- MLX90640-D55 IR array thermal imaging camera and micro servo sg90 were connected to the Raspberry Pi 4.
- A candle was used to set the farm fire for the indoor experiment.

UAV Movement Adjustment Experiment

Demo Video



① Fire not detected
→ raise altitude
→ capture wider range: no need to adjust x, y but adjust z
→ return (0,0,3)
if n>4: sensor error → quit program

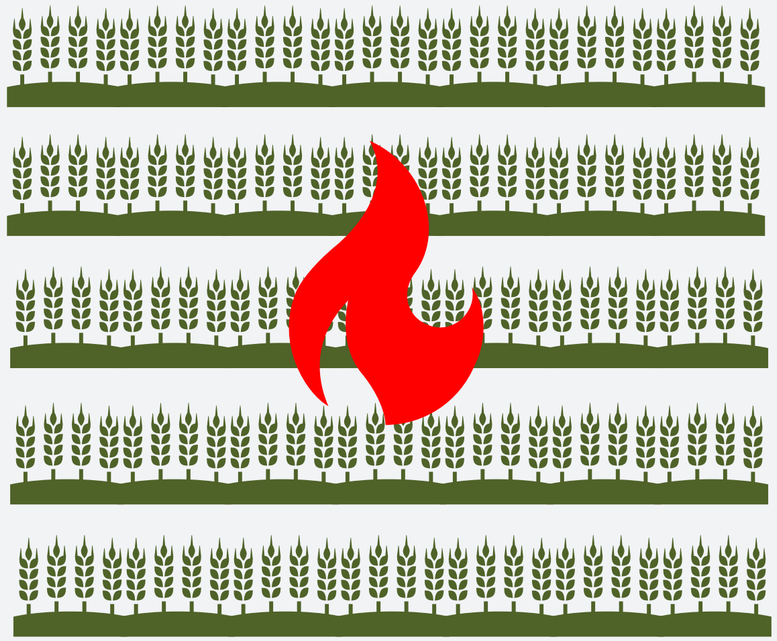
② Fire detected outside error range
→ calculate distance between drone's location & center of fire area
→ return (x_move, y_move, 0)

③ Fire detected within error range
→ no location adjustment, drop fire extinguishing ball
→ return (dropping ball sign)

06

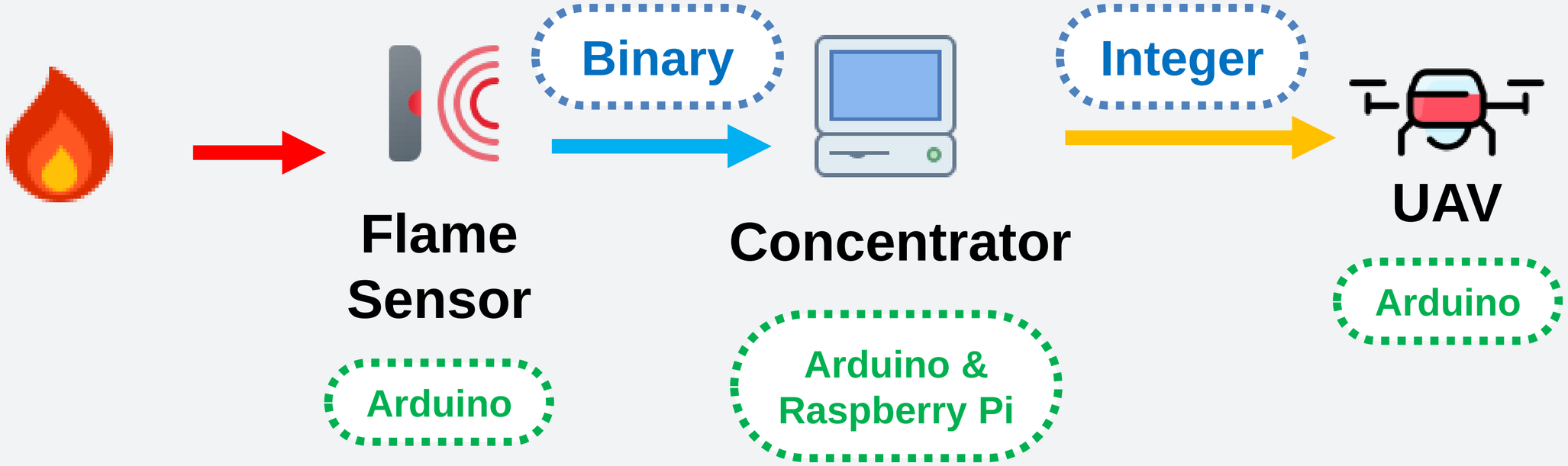
Conclusion

Conclusion

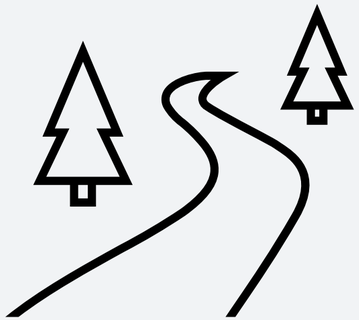


Conclusion

Data Transmission



Conclusion



Thank You

Q&A

Team Beep