BOT BRAINS BATTLE ROUND – 3

REAL-WORLD RUMBLE – PART A, B, C

Question 2A:

What do you understand by the Swarm Drones.

Solution:

Swarm drones are basically a bunch of UAVs that work together towards a common goal. They're like a team of drones that talk to each other and coordinate their actions, just like how swarms of bees or birds do. The cool thing about swarm drones is that they offer a bunch of benefits. First off, they're super efficient at getting tasks done. If one drone stops working, no worries! The others can pick up the slack and keep going. Plus, they're totally scalable, so you can use as many as you need for the job. These drones are pretty smart too, thanks to fancy algorithms and communication protocols. They can do all sorts of complex stuff like scanning huge areas, making maps, and monitoring environments with incredible accuracy and speed.

Question 2B:

In Drone if you want to use ESP8266 with the controller to communicate, how will you do it.

Solution:

To integrate the ESP8266 module with a drone's controller, I would follow these steps:

1. **Hardware Connection:**
    - Connect the ESP8266 module to the controller via UART (Universal Asynchronous Receiver-Transmitter). This typically involves connecting the TX (Transmit) and RX (Receive) pins of the ESP8266 to the corresponding RX and TX pins on the drone's controller.

2. **Power Supply:**
    - Ensure the ESP8266 is powered appropriately. It usually operates on 3.3V, so the power supply from the drone's controller must match this requirement to avoid damaging the module.

3. **Firmware and Libraries:**
    - Flash the ESP8266 with the appropriate firmware that supports Wi-Fi communication. Libraries such as the ESP8266WiFi library for Arduino can be used to simplify the development process.
4. **Coding and Configuration:**
    - Write the code to initialize the ESP8266 module and establish a Wi-Fi connection. This code will include setting up the SSID (network name) and password for the Wi-Fi network, as well as the communication protocol (TCP/UDP) for data exchange.
    - Implement the communication protocol in the drone's firmware to send and receive data via the ESP8266 module.
5. **Testing and Debugging:**
    - Conduct thorough testing to ensure reliable communication between the drone and the ESP8266 module. Debug any issues related to connectivity, data transmission, or integration with the drone's control systems.

Question 2C:

If you are allowed to make changes in the design of regular drone what will you change and justify your answer. Also attach the references for the suggested changes in the design.

Solution:

If I were to make changes to the design of a regular drone to enhance its capabilities for the specified task, I would focus on the following areas:

1. **Enhanced Autonomy:**
    - Implement advanced AI and machine learning algorithms to improve the drone's decision-making capabilities and autonomy. This would enable the drone to better navigate complex environments and adapt to dynamic conditions.
2. **Improved Sensors:**
    - Integrate high-resolution LiDAR and color sensors for more accurate object detection and identification. High-precision sensors would enhance the drone's ability to distinguish targets from other objects based on size, shape, and color.
3. **Robust Communication System:**
    - Upgrade the communication system to use a more reliable and high-bandwidth protocol, such as a mesh network, to ensure seamless data exchange between drones in the swarm. This would enhance coordination and reduce latency in transmitting critical information.

4. **Battery Efficiency:**
    - Incorporate more efficient power management systems and lightweight, high-capacity batteries to extend the drone's flight time. Longer operational periods would allow for more extensive searches without frequent recharging.
5. **Modular Design:**
    - Adopt a modular design approach, allowing for easy replacement and upgrade of components such as sensors, cameras, and communication modules. This would enable the drone to be quickly adapted for different missions or technological advancements.

6. **Enhanced Propulsion System:**

To increase the flying height and weight carrying capacity of the drone, I would enhance the propulsion system. Here are the specific changes I would implement:

This would be a game-changing feature for food-delivery services as drones can evade road traffic which delivery partners like Swiggy and Zomato cannot do.

- **High-Power Motors:**
    - Upgrade to high-power brushless motors that can provide greater thrust and support higher payloads. Brushless motors are more efficient and reliable, which is essential for carrying heavier loads over longer distances.
- **Larger Propellers:**
    - Use larger propellers to improve lift and thrust efficiency. Larger propellers can move more air and generate more lift, which is necessary for carrying heavier payloads.
- **Stronger Frame:**
    - Reinforce the drone's frame with lightweight, high-strength materials such as carbon fiber. A stronger frame will better support the additional weight and improve overall stability during flight.
- **Altitude Control System:**
    - Implement an advanced altitude control system that can maintain stable flight at higher altitudes. This system should include accurate barometers and GPS modules to provide precise altitude data and ensure stable and reliable flight at elevated heights.

References:

- Sanchez-Lopez, J.L., et al. "A Reliable Open-Source System Architecture for the Fast Designing and Prototyping of Autonomous Multi-UAV Systems: Simulation and Experimentation." *Journal of Intelligent & Robotic Systems*, vol. 84, 2016.
- Shi, W., et al. "Research on Optimization of UAV Delivery Path Based on Improved Ant Colony Algorithm." *Journal of Advanced Transportation*, vol. 2020, 2020.
- Strohmeier, M., et al. "Drone Delivery Systems: Integration of UAS in Urban and Suburban Environments." *Proceedings of the IEEE*, vol. 104, no. 3, 2016.

BOT BRAINS BATTLE – REAL WORLD RUMBLE

Solution 2D

To make sure our drones can operate independently and effectively search for targets, each one will have specific features and follow a structured plan:

I. Flight Path and Collision Avoidance:

We program each drone with a set search pattern that covers different parts of the target area. The LiDAR sensors are always scanning the surroundings to spot any obstacles and avoid crashes by adjusting the flight path in real-time.

II. Object Detection and Measurement:

When a drone comes across an object, it uses its LiDAR sensor to measure its size. If the object is 15 cm tall, the drone goes on to measure its width and depth to make sure it matches the required dimensions.

III. Color Detection:

If the dimensions are correct, the drone activates its color sensor to figure out the object's color. If it's green, the drone knows it's found the target.

IV. Swarm Communication:

As soon as a target is identified, the drone immediately communicates the location and confirms the target to the other drones using our established swarm communication protocol. The other drones then stop their search operations.

V. Information Transmission:

The drone that finds the target sends all the details, including precise coordinates, to the central command or the controlling unit.

This method guarantees an efficient and coordinated search operation by using the strengths of autonomous drones and swarm intelligence to achieve the desired outcome.

Algorithm to Implement the Autonomous Drone Search and Communication System:

I. **Initialization:**

- Initialize each drone with its unique ID and predefined search area coordinates.
- Establish communication links between the drones using swarm communication protocols.

II. **Takeoff and Navigation:**

- Each drone autonomously takes off and navigates to its starting position within the specified search area.

III. **Search Operation:**

- Each drone starts scanning its designated area using LiDAR sensors to detect objects.
- When an object is detected, the drone measures its dimensions.

IV. **Object Verification:**

- If the object's height is 15 cm, the drone measures its width and depth.
- If all dimensions are 15 cm, the drone activates its color sensor to check the object's color.

V. **Color Detection:**

- If the object is green, it is identified as the target.

VI. **Communication:**

- The drone that finds the target sends the target location information to the other drones.
- Upon receiving the information, the other drones stop their search operations.

VII. **Return to Base:**

- All drones return to their base positions and land autonomously.

Calculations for Drone Navigation and Search

Detailed Calculations For Drone Navigation and Search:

To develop a comprehensive autonomous drone system, we will focus, on the following key areas:

i) Search Area Division
ii) Flight Path Planning
iii) Collision avoidance
iv) Communication Range.

1) Search Area Division:

Assuming the total search area is a square field with side length S meters.

- Total Search Area (A):

$$A = S^2 \text{ square meters.}$$

- Number of Drones (N):

$$N = 3$$

- Area per Drone (Ad) :

$$A_d = \frac{A}{N} = \frac{S^2}{3} \text{ square meters.}$$

If the drones are assigned equal areas, each drone will cover an area of $\frac{S}{\sqrt{3}} \times \frac{S}{\sqrt{3}}$ meters.

2. Flight Path Planning :

Assuming each drone follows a lawnmower pattern for efficient area coverage.

- Search Grid Size (G') :

$$G = \sqrt{A_d} = \sqrt{\frac{S^2}{3}} = \frac{S}{\sqrt{3}} \text{ meters.}$$

- Flight Path Length (L) :
  - If the search area is divided into strips of with W meters.

$$L = \frac{G}{W} \times 2 \times G = \frac{S}{\sqrt{3} \times W} \times 2 \times \frac{S}{\sqrt{3}}$$

After simplifying :

$$L = \frac{2S^2}{3W} \text{ meters} .$$

3) Collision Avoidance :

To ensure drones do not collide with each other, maintain a minimum separation distance Ds .

- Minimum Separation Distance (Ds) :

  Ds = Separation Distance (typically 5 - 10 meters)

Ensure each drone's path does not overlap and is atleast Ds meters apart .

4) Communication Range :

Ensure the communication module (ESP 8266) covers the necessary range .

- Communication Range (Rc) :

  Rc = Typical range of ESP 8266 , approx 100 meters .

5) Power Consumption and Flight Time :

Estimate the power required to cover the search area.

a) Battery Capacity (C) :

- Given in mAh (milliampere / hour).

b) Drone Power Consumption (P) :

- Given in watts (W).

c) Flight Time (T) :

$$T = \frac{C \times V}{P} \text{ hours}$$

where : V is the voltage of the battery (Typically 11.5 V is applied for a 3S LiPo battery).

Example Calculations :

Assume the following,

S = 300 meters (total search area side length)

W = 10 meters (width of each strip in lawnmower pattern)

D_S = 10 meters (minimum separation distance)

C = 5000 mAh (battery capacity)

P = 50 W (power capacity).

1. Search Area Division :

$$A = S^2 = 300^2 = 90,000 \text{ square meters}$$

$$Ad = \frac{90000}{3} = 30,000 \text{ square meters.}$$

$$G = \sqrt{30,000} \approx 173.2 \text{ meters.}$$

2. Flight Path Length :

$$L = \frac{2 \times 300^2}{3 \times 10} = \frac{1,80,000}{30} = 6000 \text{ meters}$$

3. Communication Range :

$$R_C = 100 \text{ meters}$$

4. Battery Flight Time :

$$T = \frac{5000 \times 11.1}{50,000} = 1.11 \text{ hours} \approx 66 \text{ minutes.}$$

ALGORITHMS AND PSEUDOCODES

BOT BRAINS BATTLE

Round - 3

Real - World Rumble

Algorithms for Drone Navigation and Search :

Step 1 → Initialization :

Each drone is initialized with its unique ID, search area co-ordinates, and communication parameters.

Algorithm for Initialization :

1) Initialize drone with IDs
2) Assign search area co ordinates to each drone.
3) Establish communication links between drones.

Pseudo Code for Initialization :

```
void initializeDrones (vector < Drone > & drones) {
    for (int i=0; i< drones.size(); ++i) {
        drones[i]. setID (i+1);
        drones [i] . setSearchArea (/* coords here */);
        drones [i] . takeOff();
    }
}
```

Step 2 → Search Operation :

Each drone scans its designated area using a lawnmower pattern and checks objects using LiDAR and color sensors .

Algorithm for Search Operation :

1) Drone starts at the designated starting position .

2) Scan Area using LiDAR .

3) If an object is detected , measure its dimensions .

4) If dimensions match the target , use the color sensor to detect color .

5) If the color matches the target , communicate with other drones to inform them .

6) Continue search until the target is found or the area is completely scanned .

Pseudocode for Search Operation:

```
void searchArea (Drone &drone, vector <Drone> &drones) {
    while (true) {
        auto object = drone . scanObject()
        if ( verifyDimensions (object.height , object.width , object.depth
                                                                  )){
            string color = drone .detectColor (object);
            if ( verifyColor (color)) {
                communicateTargetLocation (drone , drones) ;
                return ;
            }
        }
    }
        drone . continueSearch () ;
    }
}
```

3) Object Verification :

Verify if the detected object matches the target Dimensions and Colors.

Algorithm for Object Verification :

1. Measure the height, width and depth of object.
2. Check if the dimensions match the target dimensions.
3. Use the color sensor to detect the color.
4. Check if the color matches the target color.

Pseudocode for Object verification :

```
bool verifyDimensions (double height, double width, double
                                                    depth) {
        return (height == TARGET_HEIGHT &&
                width == TARGET_WIDTH &&
                depth == TARGET_DEPTH);
}
bool verifyColor (const string & color) {
        return (color == TARGET_COLOR);
}
```

Algorithm for Communication :

1. The drone that finds the target sends the target location to other drones.

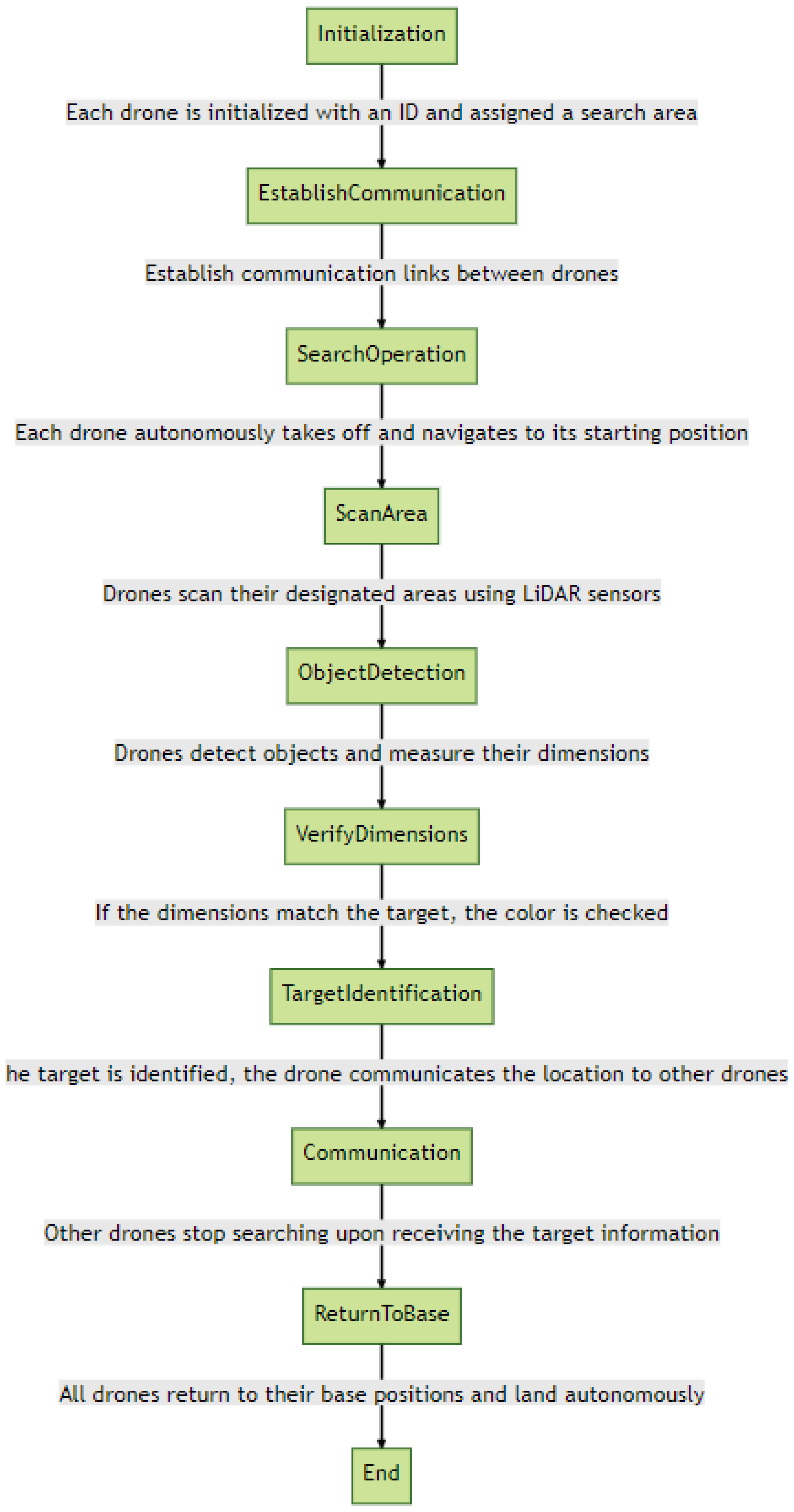
2. Other drones recieve the target location and stop their search .

Pseudocode for Communication :

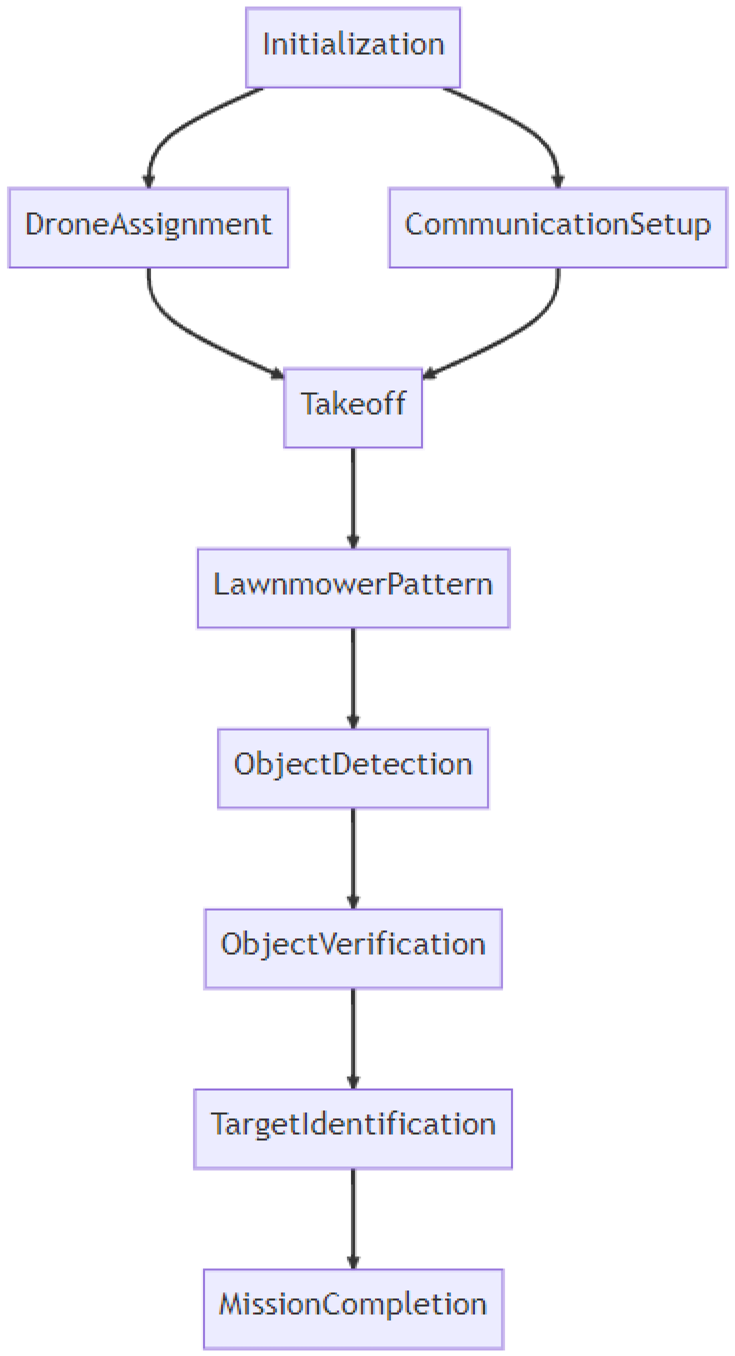
```
void communicateTargetLocation (const Drone &drone ,
                                vector <Drone> & drones {
    for (auto &d : drones) {
        d. recieveTargetLocation (drone'. getCurrentLocation ()));
        d. stopSearch () ;
    }
  }
}
```

DIAGRAMS & FLOWCHARTS

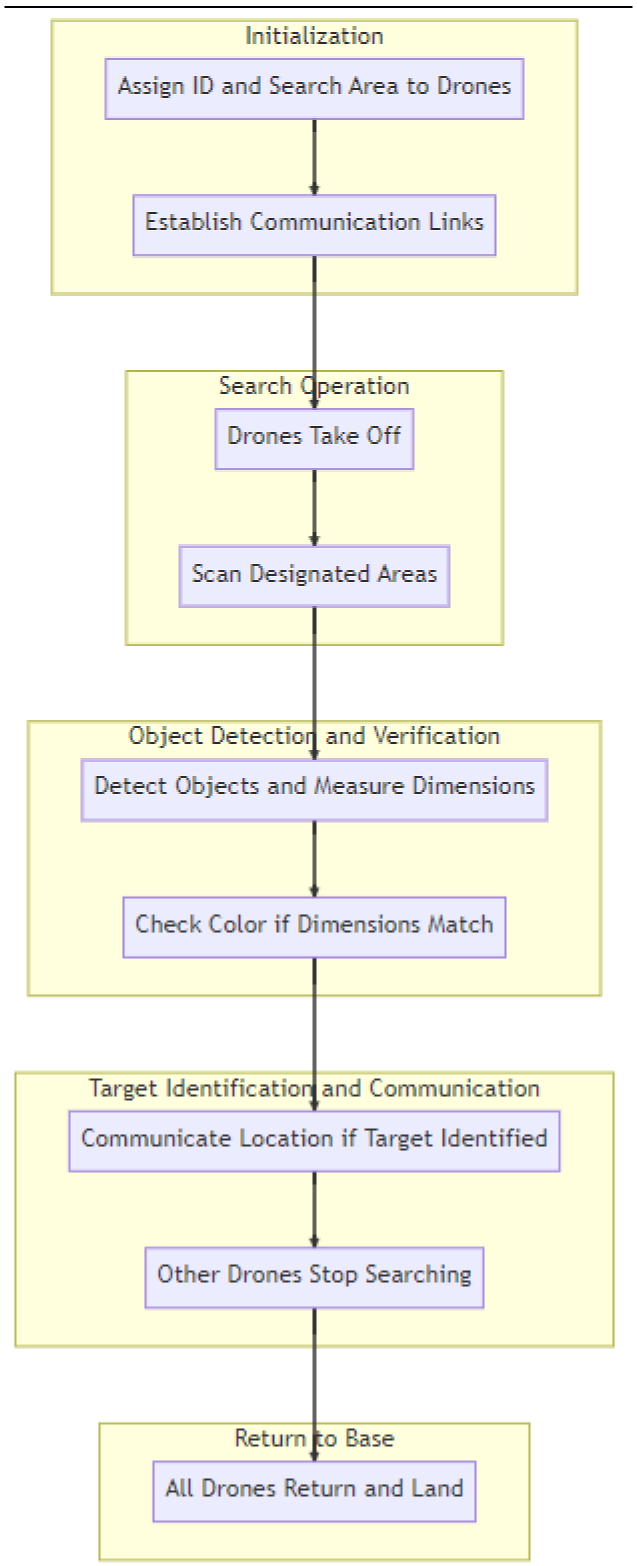
FLOW DIAGRAM:


Initialization
Each drone is initialized with an ID and assigned a search area
EstablishCommunication
Establish communication links between drones
SearchOperation
Each drone autonomously takes off and navigates to its starting position
ScanArea
Drones scan their designated areas using LiDAR sensors
ObjectDetection
Drones detect objects and measure their dimensions
VerifyDimensions
If the dimensions match the target, the color is checked
TargetIdentification
he target is identified, the drone communicates the location to other drones
Communication
Other drones stop searching upon receiving the target information
ReturnToBase
All drones return to their base positions and land autonomously
End

DRONE FLIGH PATH DIAGRAM:

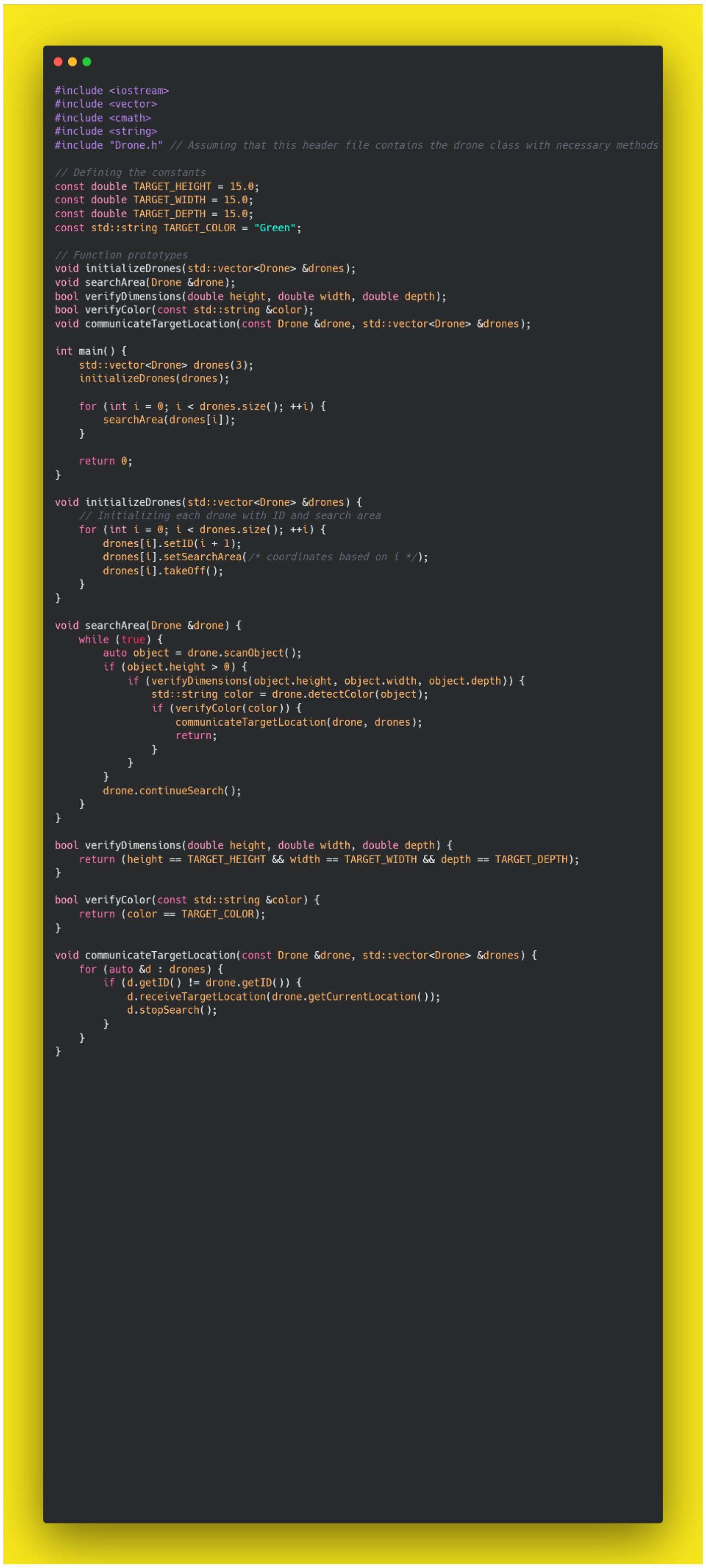


DRONE SEARCH DIAGRAM:


Initialization
Assign ID and Search Area to Drones
Establish Communication Links
Search Operation
Drones Take Off
Scan Designated Areas
Object Detection and Verification
Detect Objects and Measure Dimensions
Check Color if Dimensions Match
Target Identification and Communication
Communicate Location if Target Identified
Other Drones Stop Searching
Return to Base
All Drones Return and Land

C++

PROGRAM (LLD)



```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <string>
#include "Drone.h" // Assuming that this header file contains the drone class with necessary methods

// Defining the constants
const double TARGET_HEIGHT = 15.0;
const double TARGET_WIDTH = 15.0;
const double TARGET_DEPTH = 15.0;
const std::string TARGET_COLOR = "Green";

// Function prototypes
void initializeDrones(std::vector<Drone> &drones);
void searchArea(Drone &drone);
bool verifyDimensions(double height, double width, double depth);
bool verifyColor(const std::string &color);
void communicateTargetLocation(const Drone &drone, std::vector<Drone> &drones);

int main() {
    std::vector<Drone> drones(3);
    initializeDrones(drones);

    for (int i = 0; i < drones.size(); ++i) {
        searchArea(drones[i]);
    }

    return 0;
}

void initializeDrones(std::vector<Drone> &drones) {
    // Initializing each drone with ID and search area
    for (int i = 0; i < drones.size(); ++i) {
        drones[i].setID(i + 1);
        drones[i].setSearchArea(/* coordinates based on i */);
        drones[i].takeOff();
    }
}

void searchArea(Drone &drone) {
    while (true) {
        auto object = drone.scanObject();
        if (object.height > 0) {
            if (verifyDimensions(object.height, object.width, object.depth)) {
                std::string color = drone.detectColor(object);
                if (verifyColor(color)) {
                    communicateTargetLocation(drone, drones);
                    return;
                }
            }
        }
        drone.continueSearch();
    }
}

bool verifyDimensions(double height, double width, double depth) {
    return (height == TARGET_HEIGHT && width == TARGET_WIDTH && depth == TARGET_DEPTH);
}

bool verifyColor(const std::string &color) {
    return (color == TARGET_COLOR);
}

void communicateTargetLocation(const Drone &drone, std::vector<Drone> &drones) {
    for (auto &d : drones) {
        if (d.getID() != drone.getID()) {
            d.receiveTargetLocation(drone.getCurrentLocation());
            d.stopSearch();
        }
    }
}
```