IOT BASED DRONE FOR CONTACTLESS FOOD DELIVERY

A MINI PROJECT REPORT

***Submitted* by**

ARUNKUMAR K.V. 211420104027

EMMANUEL JOSHUA C. 211420104074

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"IOT BASED DRONE FOR CONTACTLESS FOOD DELIVERY"** is the bonafide work of **ARUNKUMAR K.V. (211420104027) & EMMANUEL JOSHUA C. (211420104074)** who carried out the project work under my supervision.

SIGNATURE

Dr.L.JABASHEELA M.E .,Ph.D
PROFESSOR,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Dr. HEMLATHADHEVI M.E,Ph.D
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) were examined in the Mini Project Viva-Voce Examination held on.....

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt. C. VIJAYARAJESWARI , Dr. C . SAKTHIKUMAR , M.E. , Ph.D.,** and **Dr. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,Ph.D** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the HOD of the CSE Department, **Dr. L.JABASHEELA, M.E.,Ph.D.,** who facilitated us in completing the project.

We would like to thank our parents, friends, and Project Guide **Dr. HEMALATHA DEVI M.Tech.,Ph.D.,** and coordinator **Dr.P.J.SATHISH KUMAR M.Tech., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

NAME OF THE STUDENTS

ARUNKUMAR K.V. (211420104027)

EMMANUEL JOSHUA C. (211420104074)

ABSTRACT

In recent times, drones are being used to transport goods autonomously from one place to another. Drone Talk technology is the recently proposed system to implement drone-based deliveries. By utilising this system, we can deliver food to customers in a contactless manner. The Drone Talk system has a drone control system, which supports weather awareness and collision control. We propose a system that features a LIDAR sensor, solar charging system and Visual based recognition to enhance collision avoidance, drone flight duration and verified food delivery. The LIDAR sensor is very effective in 3D imaging at very low latency. We can fine tune the sensor to ensure that it never collides with its environment. The Solar Charging System provides charging even during the flight which extends the flight duration by a few minutes. The drone is attached with a box that opens only after it verifies the customer. For verification, each drone is provided with a unique ID with a QR code which would be scanned by the customer on the event of the delivery. Once the drone ID and the Customer ID is matched, the box that contains the food would be opened for the customer to take. When the drone detects that the food is taken by the customer, It signals the delivery of the food and returns to its station. The proposed system is plausible to be implemented in high traffic regions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A drone-based food delivery system is a method of delivering food using unmanned aerial vehicles (UAVs) or drones. It is an innovative technology that aims to make food delivery faster, more efficient, and more convenient for customers.

The drone-based food delivery system works by using drones equipped with GPS and other sensors to locate the delivery address and navigate to the customer's location. The food is loaded into a container attached to the drone, which is then flown to the customer's location. The customer can receive the food directly from the drone, or the drone can land and drop off the package at a designated location.

There are several potential benefits of a drone-based food delivery system. For one, it can be faster and more efficient than traditional delivery methods, as drones can avoid traffic and take direct routes to the delivery address. It can also reduce delivery costs and provide an eco-friendlier delivery option by reducing the need for delivery vehicles.

However, there are also some challenges and concerns with this technology. Safety is a significant concern, as drones can pose a risk to people and property if they malfunction or crash. Additionally, there are regulatory and legal challenges related to the use of drones for commercial purposes, including food delivery.

Overall, a drone-based food delivery system has the potential to revolutionise the food delivery industry, but it will require careful planning and regulation to ensure safety and efficiency.

1.2 PROBLEM DEFINITION

- Aim of this Project is to Develop an Autonomous Drone Delivery System for Contactless Food Delivery with advanced sensors, cameras, and communication systems that allow them to navigate and interact with their surroundings.

- The Internet of Things (IoT) has revolutionized the way we interact with everyday objects, and the use of drones in logistics and delivery has been increasing in recent years. One of the most exciting applications of this technology is contactless food delivery, which has become increasingly important in the wake of the COVID-19 pandemic.

- This project helps the environment by reducing the air pollution created by food delivery valets, reduces traffic jams and the food is delivered in a more secure way.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

- Amazon announced the possibility of drone delivery on the show 60 Minutes in December 2013, many companies, such as DHL, Flytrex, Flirtey, and Google, have been working on its development.

- There has been less investigation on delivering packages at specific or recipient-chosen locations around a house, this will be a critical part of automating delivery by drones.

DISADVANTAGES

- Uses Sonar Sensor which limits range.
- It lacks accuracy in prediction.

2.2 PROPOSED SYSTEM

- The proposed system is supported by a unique Air Traffic Control System which makes drones to handle network issues, traffic problems and provide Prioritized service with time management.

ADVANTAGES

- Can deliver products based on user request location.
- Air Traffic Control is based on priority Service

2.3 DEVELOPMENT ENVIRONMENT

SOFTWARE REQUIREMENT

flight control and navigation systems may require programming in languages such as C or Python, while web-based user interfaces and order management systems may require development in JavaScript, HTML, or CSS. The choice of programming languages and technologies will depend on factors such as system performance, scalability, and maintainability.

HARDWARE REQUIREMENT

- LIDAR Sensor
- Altimeter
- Camera
- Solar panel
- Drone
- GPS

CHAPTER 3
SYSTEM DESIGN

3.1 UML DIAGRAMS

3.1.1 USE CASE DIAGRAM

Fig 3.1.1 Use case diagram for Drone based Food Delivery

3.1.2 SEQUENCE DIAGRAM

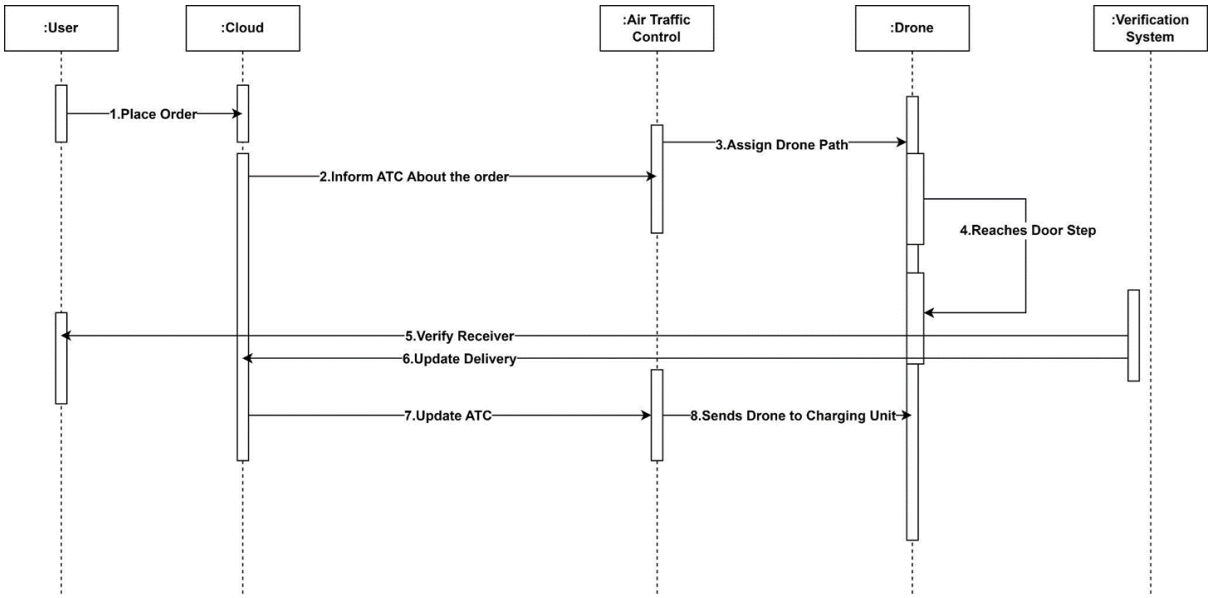

Fig 3.1.2 Sequence diagram for Drone based Food Delivery

3.1.3 INTERACTION DIAGRAM

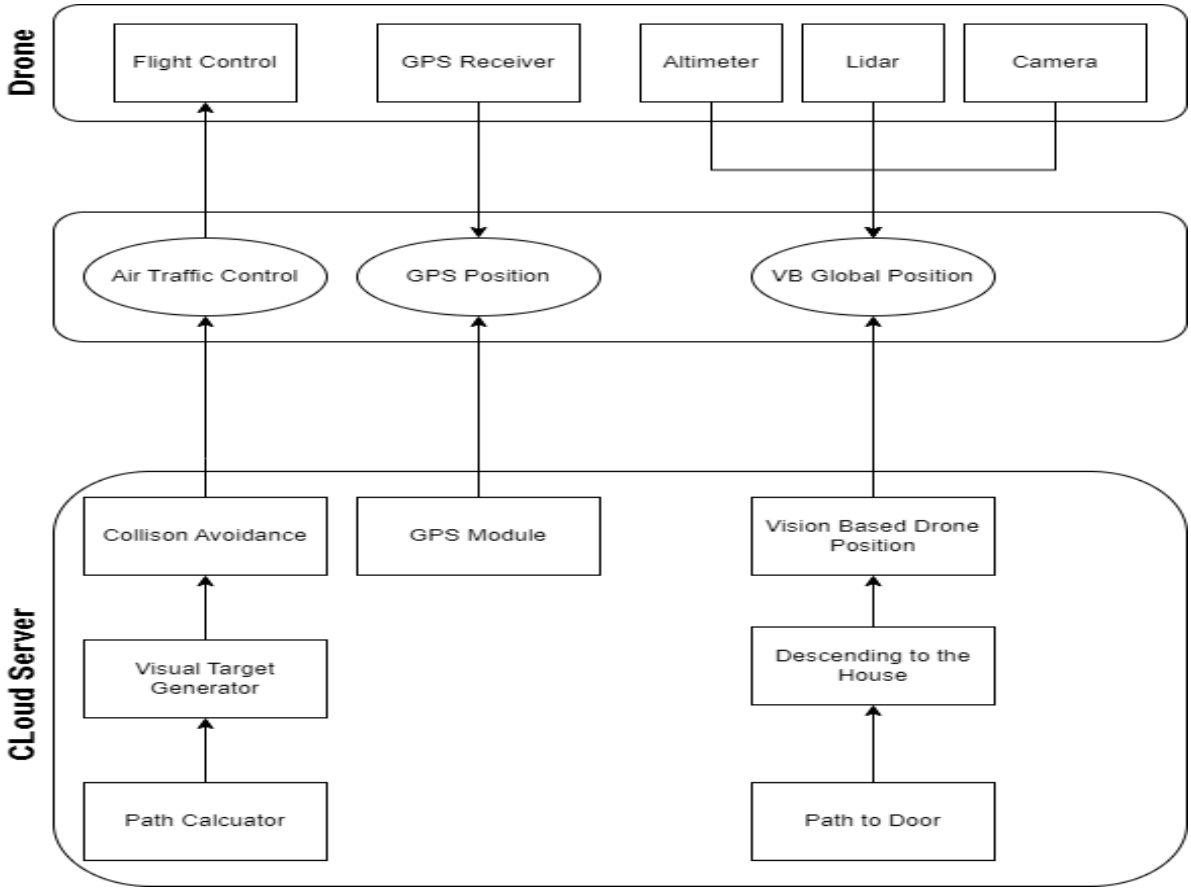

Fig 3.1.3 Module interaction of the system

3.2 DATA DICTIONARY

A data dictionary is a document that provides a detailed description of the data elements used in a database or information system. It typically includes the following information for each data element:

- The name of the data element, such as "customer name" or "order number".
- A brief description of what the data element represents, such as "the name of the customer who placed an order".
- The type of data used in the data element, such as text, numeric, or date/time.
- The maximum number of characters or digits that can be used in the data element.
- The set of values that are allowed for the data element, such as a list of valid states or countries.
- Any rules or constraints that must be followed when entering or updating the data element, such as mandatory fields or unique identifiers.
- The system or application where the data element is created or modified.
- Any relationships or dependencies between the data element and other elements in the system.

Rules for a data dictionary may include:

- Consistency: The data dictionary should be consistent across all elements and documentation should be kept up to date.
- Clarity: All data elements should be clearly defined and explained to avoid any ambiguity or confusion.
- Accuracy: All data elements and their attributes should be accurate and up to date to avoid errors or misinterpretation.

- Accessibility: The data dictionary should be easily accessible to all stakeholders who may need it, such as developers, analysts, and business users.
- Security: The data dictionary should be kept secure to prevent unauthorized access or modifications to the data.
- Standardization: All data elements should follow a standard format or naming convention to ensure consistency and ease of use.
- Cross-referencing: Data elements should be cross-referenced with other related data elements or systems to ensure consistency and accuracy.

CHAPTER 4
SYSTEM ARCHITECTURE

4.1 ARCHITECTURE OVERVIEW

The drone delivery system is modelled based on [1] and is integrated with advancements to make the system more viable and efficient. The improvements involve the use of better sensors, a dynamic charging unit to increase flight duration, more efficient algorithms to reduce transmission and reception latency, a verification system to ensure the right delivery to the customer and a traffic control system

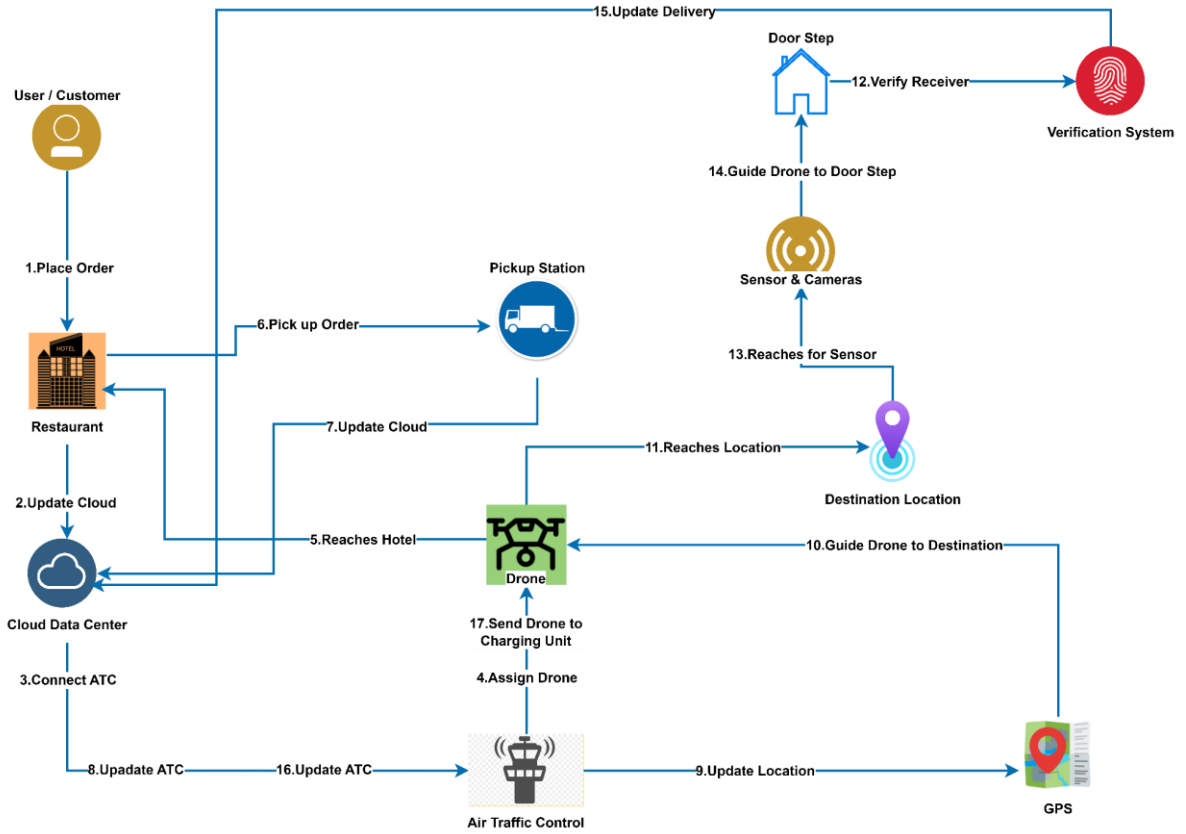

Fig 4.1.1 Architecture diagram for Drone based Food Delivery

4.2 Module Description

4.2.1 Dynamic Charging System:

The charging system integrates a solar unit for recharging the drone during the entirety of its flight, right from the take-off through its delivery and until its return. However, this can be controlled from the launch station whether the charging system needs to be activated or not.

The drone along with its payload can weigh up to 40 kilograms. To provide sufficient power a lithium-ion battery
that produces 25000mAh or more should be used. With this power the drone can have a flight time of approximately 30 minutes. When the charging system is integrated with the drone, the flight time can be significantly increased by 5 to 10 minutes. This can be theoretically calculated using the following equation,

$Q(mAh) = 1000 \times E(Wh) / V(V)$

For instance, let's take a solar panel of 50W and 12V into consideration. The recharging capacity can be calculated as,

$$Q(mAh) = 1000 \times 50/12$$
$$= 4166.6666666667$$

So, from the above calculation we can see that an approximation of 4100mAh can be produced by the solar charging unit. This can be interpreted much more clearly using the visualisation described below.

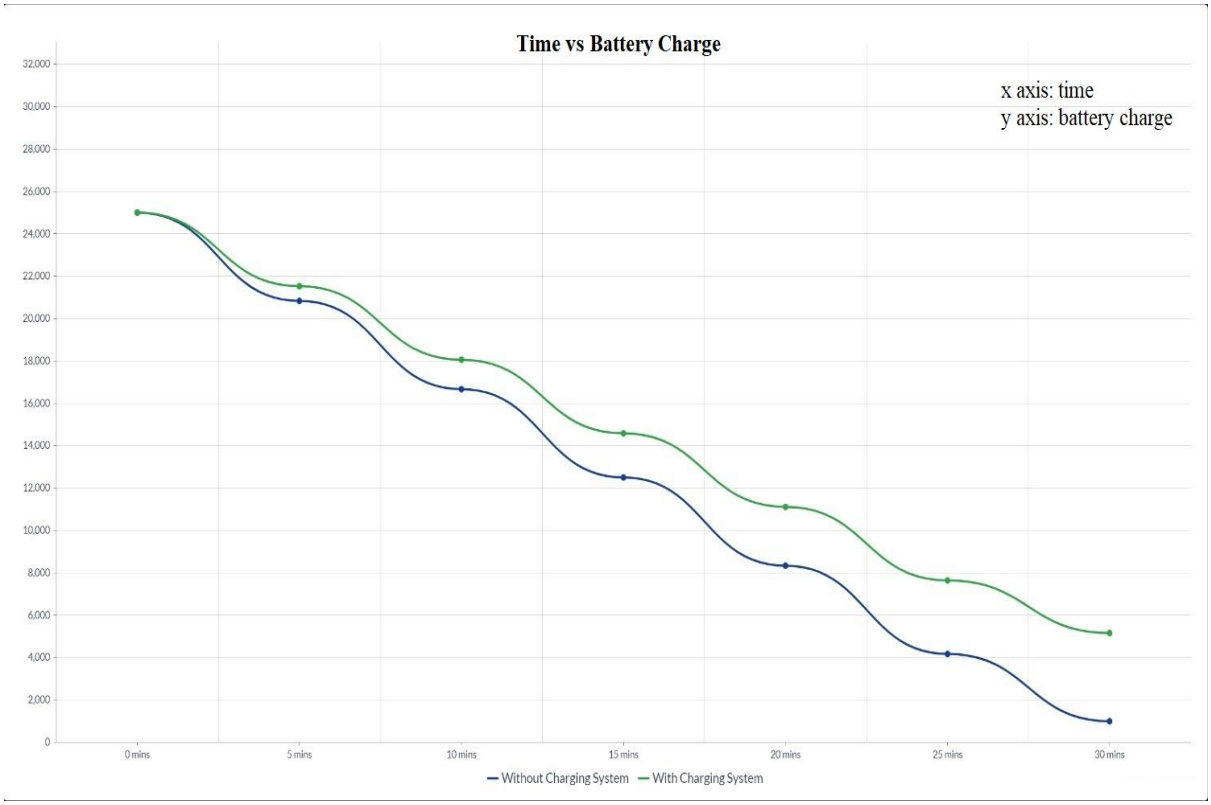

Fig 4.2.1.1 Dynamic Charging System for Drone based Food Delivery

	0 mins	5 mins	10 mins	15 mins	20 mins	25 mins	30 mins
Without Charging System	25000	20833.375	16666.77	12500	8333.4	4166.7	987.2
With Charging System	25000	21527.82	18055.6	14583	11111.25	7638.9	5153.95

The above table helps us to interpret the relation between the charge depreciation as time progresses.

4.2.2 Algorithms for Flight Control:

Lidar sensors are commonly used in robotics and autonomous vehicles to detect and avoid obstacles in their environment. The YOLO (You Only Look Once) algorithm is a popular object detection algorithm that can be used with Lidar sensors to identify and classify objects in the environment. The Lidar sensor must be installed on the robot or autonomous vehicle, and the necessary software and hardware must be installed to capture data from the sensor. The Lidar sensor can be used to collect data on the environment, including the location and distance of obstacles in the environment. The collected data can be pre-processed to eliminate any missing or irrelevant data points and prepare it for use in the YOLO algorithm. The YOLO algorithm can be trained on the pre-processed data using various machine learning techniques such as supervised, unsupervised, or reinforcement learning. The training process can be repeated until the algorithm achieves the desired level of accuracy. The YOLO algorithm's performance can be evaluated using metrics such as accuracy, precision, recall, and F1 score. Once the YOLO algorithm is trained and evaluated, it can be integrated into the Lidar sensor to identify and classify objects in the environment. By implementing a Lidar sensor using the YOLO algorithm, robots and autonomous vehicles can detect and avoid obstacles in their environment more efficiently and accurately. The YOLO algorithm can help classify objects in real-time and provide timely responses to potential obstacles, leading to safer and more reliable operation of the system.

4.2.3 Verification System:

A customer and product verification system for the drone-based food delivery can help ensure that the right food is delivered to the right customer in a safe and secure manner. The customer must provide their name, address, and contact information when placing an order. This information is then verified against the customer database to ensure that the order is being delivered to the correct person. The food package must be equipped with a unique identifier, such as a barcode or QR code. This identifier can be scanned by the drone before takeoff to confirm that the correct package is being delivered. The drone must

have a camera or other sensor that can capture images or video footage of the delivery location. This can be used to verify that the package was delivered to the correct location and that it was received by the customer. The drone's location and progress can be tracked in real-time using GPS technology. This can be used to ensure that the drone is following the correct delivery route and that it is on schedule. The drone and its payload can be equipped with encryption technology to prevent unauthorised access and tampering. The customer and product information can be securely stored in a database with limited access. The customer can provide feedback on the quality of the food and the delivery experience, which can be used to improve the system over time. By implementing a customer and product verification system for drone-based food delivery, customers can be assured that they will receive the correct food at the right location, and the delivery process can be made more efficient and secure

4.2.4 Air Traffic Control System:

A drone traffic control system that uses an altimeter can help ensure the safety of drones in the airspace. An altimeter is a device that measures altitude, and it can be used to provide information about the altitude of drones in flight. Each drone would be equipped with an altimeter, which would measure the drone's altitude above ground level. A central control system would monitor the altitude of each drone, and would assign specific altitudes to each drone based on its intended flight path. If two or more drones were assigned the same altitude, the control system would adjust the altitudes of one or more of the drones to ensure they remain a safe distance apart. In the event of an altitude violation or other safety issue, the control system could send a warning message to the affected drone's operator or take automatic actions to avoid a collision. The system also incorporates other sensors and technologies, such as GPS, to provide additional information about each drone's position and speed. The control system can also monitor the airspace for any other potential hazards, such as birds or weather conditions, and alert drones as needed. Drone traffic control and navigation using Mobile technology can help improve the safety and efficiency of drone operations in the airspace. All drones that operate

in the airspace are equipped with a SIM card. The SIM card can be used to communicate with the drone traffic control system and other drones in the area. The SIM card can transmit the drone's location and other data, such as altitude and speed, to the drone traffic control system. This data can be used to monitor the drone's movement and ensure that it is following the correct flight path. The drone traffic control system can use Mobile technology to communicate with other drones in the area. This can be used to coordinate the movement of drones and avoid collisions. The SIM card can also be used to provide navigation information to the drone. This information can include the location of obstacles, such as buildings or other drones, and the drone's route to its destination. The drone traffic control system can use Mobile technology to communicate with the drone pilot in case of emergency or to provide updates on weather conditions or other relevant information. This data can be used to improve the efficiency and safety of future flights. By implementing drone traffic control and navigation using Mobile technology, drones can be safely and efficiently managed in the airspace. The use of Mobile technology can help ensure that drones are following the correct flight paths, avoid collisions, and provide the necessary data for drone pilots and traffic control systems to make informed decisions.

In case of connection loss between the drone and its ground control station, an emergency landing system is implemented to ensure the safe landing of the drone. The drone is equipped with sensors, such as GPS and altimeters, to detect the drone's location, altitude, and speed. Emergency landing areas are identified and pre-programmed into the drone's flight plan. These areas are clear of obstacles and safe for the drone to land in. In case of a connection loss between the drone and its ground control station, the emergency landing system is activated. The drone's sensors are used to determine the drone's location, altitude, and speed. This information is then used to calculate the best landing location, considering the emergency landing areas and the drone's flight path. The drone's flight controller initiates the landing sequence, which includes reducing altitude and speed, selecting a landing area, and landing the drone safely. The drone's sensors and cameras are used to monitor the landing process and ensure that the drone lands safely and without causing any damage to the surrounding area. By implementing

this emergency landing system, the risk of crashes or damage to property can be reduced in case of a connection loss. The system ensures the safe landing of the drone and prevents any potential harm or danger to people and property in the vicinity.

An obstacle avoidance system for a drone, based on a circular arc trajectory is implemented by equipping the drone with sensors, such as lidar, to detect obstacles in its flight path. Once an obstacle is detected, the drone plans a new trajectory to avoid the obstacle. A circular arc trajectory is planned around the obstacle, which can be computed using the drone's current velocity and the radius of the arc. The angle of the arc can be calculated based on the drone's current velocity and the minimum turn radius. The drone's flight controller can adjust the drone's speed and heading to follow the circular arc trajectory. The drone continues to detect obstacles in its flight path and plans new trajectories as needed to avoid collisions. By implementing an obstacle avoidance algorithm based on a circular arc trajectory, the drone can avoid obstacles and continue its flight path while minimising any deviations from its original trajectory. This can help to prevent collisions and damage to the drone and any surrounding objects.

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 COLLISION AVOIDANCE CODE

```
import math
import numpy as np

# Define the starting and ending points of the trajectory
start_pos = np.array([0, 0, 0])
end_pos = np.array([10, 10, 5])

# Define the radius of the circular arc
radius = 5

# Calculate the center point of the circle
delta_x = end_pos[0] - start_pos[0]
delta_y = end_pos[1] - start_pos[1]
delta_z = end_pos[2] - start_pos[2]
distance_xy = math.sqrt(delta_x**2 + delta_y**2)
distance_xyz = math.sqrt(delta_x**2 + delta_y**2 + delta_z**2)
mid_point = np.array([(start_pos[0] + end_pos[0])/2, (start_pos[1] +
end_pos[1])/2, (start_pos[2] + end_pos[2])/2])
alpha = math.acos(delta_y/distance_xy)
if delta_x < 0:
    alpha = 2*math.pi - alpha
center_point = np.array([mid_point[0] + radius*math.cos(alpha +
math.pi/2), mid_point[1] + radius*math.sin(alpha + math.pi/2),
mid_point[2]])

# Calculate the angle between the start and end positions in the x-y plane
angle_xy = math.atan2(delta_y, delta_x)

# Calculate the angle between the start and end positions in the x-z plane
```

```
angle_xz = math.atan2(delta_z, distance_xy)

# Define the waypoints along the circular arc trajectory
waypoints = []
num_waypoints = 10
for i in range(num_waypoints):
    angle_i = angle_xy + (math.pi/2)*(i/(num_waypoints-1))
    x_i = center_point[0] + radius*math.cos(angle_i)
    y_i = center_point[1] + radius*math.sin(angle_i)
    z_i = center_point[2] + radius*math.sin(angle_xz +
math.pi/2)*(i/(num_waypoints-1))
    waypoints.append([x_i, y_i, z_i])

# Print the waypoints
print("Waypoints:")
for waypoint in waypoints:
    print(waypoint)
```

5.2 OBJECT DETECTION CODE

```
import torch
import numpy as np
import cv2
import open3d as o3d
from yolov5.models.experimental import attempt_load
from yolov5.utils.torch_utils import select_device

# Load the YOLOv5 model
model = attempt_load("yolov5s.pt", map_location=torch.device("cpu"))

# Set the device to use for inference
device = select_device("cpu")

# Define the LiDAR point cloud file path
lidar_file_path = "lidar.ply"

# Read the LiDAR point cloud data into an Open3D point cloud object
lidar_cloud = o3d.io.read_point_cloud(lidar_file_path)

# Convert the Open3D point cloud object to a numpy array
lidar_points = np.asarray(lidar_cloud.points)

# Define the input image size for the YOLOv5 model
img_size = (640, 640)

# Create an empty image with the specified size
img = np.zeros((img_size[0], img_size[1], 3), dtype=np.uint8)

# Convert the LiDAR points to image coordinates
img_coords = lidar_points[:, :2] / 0.05 + 320

# Filter out points that are outside the image bounds
```

```
valid_mask = (img_coords[:, 0] >= 0) & (img_coords[:, 0] < img_size[0])
& (img_coords[:, 1] >= 0) & (img_coords[:, 1] < img_size[1])
img_coords = img_coords[valid_mask]

# Run the YOLOv5 model on the image
results = model(torch.from_numpy(img).to(device).float().permute(2, 0,
1).unsqueeze(0), augment=False)[0]

# Filter out detections with a confidence score below a threshold
confidence_threshold = 0.5
results = results[results[:, 4] >= confidence_threshold]
# Convert the detection boxes from normalized coordinates to image
coordinates
results[:, :4] = results[:, :4] * img_size[0]

# Draw the detection boxes on the image
for result in results:
    x1, y1, x2, y2, confidence, class_index = result
    x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
    class_name = model.module.names[int(class_index)]
    cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(img, f"{class_name} {confidence:.2f}", (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)

# Display the image with the detection boxes
cv2.imshow("LiDAR Object Detection", img)
cv2.waitKey(0)
```

5.3 EMERGENCY LANDING

```
import time
import threading

class Drone:
    def __init__(self):
        self.altitude = 0.0
        self.is_flying = False
        self.is_emergency_landing = False

    def takeoff(self):
        print("Drone taking off...")
        self.is_flying = True
        self.altitude = 10.0

    def land(self):
        print("Drone landing...")
        self.is_flying = False
        self.altitude = 0.0

    def emergency_landing(self):
        if self.is_flying:
            print("Drone initiating emergency landing...")
            self.is_emergency_landing = True
            threading.Timer(5.0, self._emergency_landing_complete).start()

    def _emergency_landing_complete(self):
        if self.is_emergency_landing:
            print("Drone emergency landing complete.")
            self.land()

    def ascend(self, meters):
        if self.is_flying:
            print(f"Drone ascending {meters} meters...")
```

```
            self.altitude += meters

    def descend(self, meters):
        if self.is_flying:
            print(f"Drone descending {meters} meters...")
            self.altitude -= meters
            if self.altitude <= 0.0:
                self.land()

# Example usage
drone = Drone()
drone.takeoff()
drone.ascend(5.0)
time.sleep(2.0)
drone.emergency_landing()
time.sleep(6.0)
drone.ascend(5.0)
time.sleep(2.0)
drone.land()
```

5.4 DRONE FLIGHT BASED ON WEATHER DETECTION

```
import requests
import time
class Drone:
    def __init__(self):
        self.altitude = 0.0
        self.is_flying = False

    def takeoff(self):
        print("Drone taking off...")
        self.is_flying = True
        self.altitude = 10.0

    def land(self):
        print("Drone landing...")
        self.is_flying = False
        self.altitude = 0.0

    def fly_to(self, latitude, longitude):
        print(f"Drone flying to ({latitude}, {longitude})...")
        # Implement code to fly to the specified location

    def check_weather(self, latitude, longitude):
        api_key = "<Your OpenWeatherMap API key>"
        url =
f"http://api.openweathermap.org/data/2.5/weather?lat={latitude}&lon={lo
ngitude}&appid={api_key}"
        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            weather_condition = data["weather"][0]["main"]
            print(f"Current weather condition: {weather_condition}")
            return weather_condition
        else:
```

```
            print("Failed to retrieve weather information.")
            return None

    def fly_autonomously(self, latitude, longitude):
        self.takeoff()
        while self.is_flying:
            weather_condition = self.check_weather(latitude, longitude)
            if weather_condition == "Clear":
                self.fly_to(latitude, longitude)
            elif weather_condition in ["Rain", "Snow"]:
                print("It is currently raining or snowing. Drone returning to
base.")
                self.land()
            else:
                print("Weather condition not supported. Drone hovering in
place.")
            time.sleep(10.0)

# Example usage
drone = Drone()
drone.fly_autonomously(latitude=37.7749, longitude=-122.4194)
```

CHAPTER 6
SYSTEM TESTING

6.1 TESTCASES AND REPORT

Here are some examples of test cases for an autonomous drone delivery system:

Delivery location validation: Test that the delivery location is correctly entered and verified before the drone takes off. This could include checking that the address exists, is within the delivery range, and is accessible for delivery.

Drone take-off and landing: Test that the drone can take off and land autonomously without any issues. This could include testing in different weather conditions and at different times of the day.

Obstacle avoidance: Test that the drone can detect and avoid obstacles in its path, such as trees, buildings, and power lines. This could include testing the drone's sensors and algorithms in different environments and with different obstacles.

Package delivery: Test that the drone can successfully deliver packages to the correct location. This could include testing the accuracy of the drone's delivery system and ensuring that the package is delivered undamaged.

Connection loss: Test what happens if the connection to the drone is lost during the delivery process. This could include testing the drone's emergency landing system and ensuring that the package is not lost or damaged.

Battery life: Test the drone's battery life and ensure that it has enough charge to complete the delivery. This could include testing the drone's flight time and distance limitations.

Return to base: Test that the drone can safely return to base after completing a delivery or if there is an issue with the delivery. This could include testing the drone's landing system and ensuring that it can return to base in different weather conditions.

Security and privacy: Test the security and privacy features of the system, such as authentication and encryption. This could include testing the system's vulnerability to hacking and ensuring that customer data is protected.

These are the test cases that is used for the autonomous drone delivery system.

Fig 6.1 Testcase diagram (1)

Fig 6.2 Testcase diagram (2)

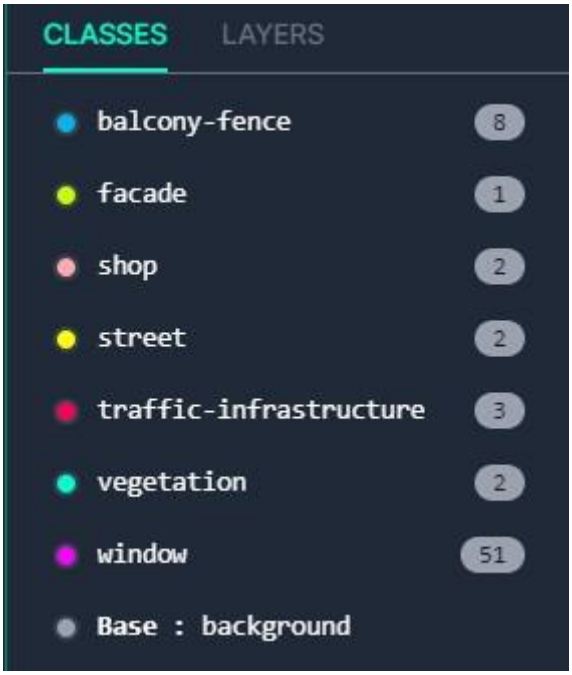

Fig 6.3 Testcase diagram (3)

RAW DATA


```
{
   "camera": "Generated by Roboflow",
   "classes": [
      "balcony-fence",
      "car",
      "facade",
      "fence",
      "non-building-infrastructure",
      "shop",
      "street",
      "traffic-infrastructure",
      "vegetation",
      "window"
   ],
   "datasets": [
      "ycpyD4UDIk0TdXbikYkr"
   ],
   "destination": "00bf43b6e32c7ce54f0759b09d639485",
   "height": 1024,
   "id": "7dH5ODuDzUmYUONNpo4C",
   "label": [],
   "labels": [],
   "name": "20230329_203307_169_R_scaled_1.png",
   "numSteps": 1,
   "owner": "gINBfG3fXtUHp9MQU1JDomDUO253",
   "preprocessing": [
      "auto-orient"
   ],
   "preprocessingParsed": [
      {
         "name": "Auto-Orient",
         "value": "Applied"
      }
```

```
    ],
    "source": "7dH5ODuDzUmYUONNpo4C",
    "split": "train",
    "split.ycpyD4UDIk0TdXbikYkr.3": "train",
    "status": "generated",
    "transforms": "[\n    \"auto-orient\"\n]",
    "updated": {
        "_seconds": 1681369082,
        "_nanoseconds": 448000000
    },
    "updatedDate": "Apr 13, 2023",
    "updatedTime": "12:28PM",
    "updatedTimezone": "+05:30",
    "versions": [
        "ycpyD4UDIk0TdXbikYkr/3"
    ],
    "width": 1024
}
```

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

- In this paper, we present an integrated system that allows drones to autonomously deliver packages at various locations around a recipient's house.

- Drone delivery system with automatic online weather awareness that supports autonomous flight in mixed indoor–outdoor environments. Moreover, this system simultaneously considers the drone collision problems resulting from communication, processing, and control latencies.

7.2 FUTURE ENHANCEMENT

- Our future work will focus on the operation of the proposed system. We will setup a mail room for drone delivery services. Moreover, we will expand flight paths and establish wireless charging bases.
- We will also add multiple weather stations to the proposed system and develop a real-time wind avoidance algorithm that can change flight trajectories based on real-time wind data. In addition, we will attempt to incorporate delivery trajectory optimization into Drone Delivery System.

CHAPTER 8

REFERENCES

- Kuan-Wen Chen, Ming-Ru Xie, Yu-Min Chen, Ting-Tsan Chu, and Yi-Bing Lin "Drone Talk: An Internet-of-Things-Based Drone - System for Last-Mile Drone Delivery" Ieee Transactions On Intelligent Transportation Systems, Vol. 23, No. 9, September 2022. **10.1109/TITS.2021.3138432**

- Shyam Sundar Kannan and Byung-Cheol Min "Autonomous Drone Delivery to Your Door and Yard" 10 May 2022. **https://doi.org/10.48550/arXiv.2104.05503**

- Weiguo Wang, Luca Mottola, Yuan He, Jinming Li, Yimiao Sun, Shuai Li, Hua Jing, Yulei Wang "MicNest: Long-Range Instant Acoustic Localization of Drones in Precise Landing" November, 2022. **https://doi.org/10.3390/drones5030075**

- Chen-Wei Lee, Wai-Peng Wong "Last-mile drone delivery combinatorial double auction model using multi-objective evolutionary algorithms" 12 April 2022. **https://doi.org/10.1007/s00500-022-07094-9**

- Asif Mahmud Raivi , S. M. Asiful Huda , Muhammad Morshed Alam, Sangman Moh "Drone Routing for Drone-Based Delivery Systems: A Review of Trajectory Planning, Charging, and Security" 28 January 2023. **https://doi.org/10.3390/s23031463**

- Idrees Waris, Rashid Ali, Anand Nayyar, Mohammed Baz, Ran Liu and Irfan Hameed “An Empirical Evaluation of Customers’ Adoption of Drone Food Delivery Services: An Extended Technology Acceptance Model” 2 March 2022.
  https://doi.org/10.3390/su14052922
- Jinsoo Hwang, Insin Kim and Muhammad Awais Gulzar “Understanding the Eco-Friendly Role of Drone Food Delivery Services: Deepening the Theory of Planned Behavior” 15 February 2020.
  https://doi.org/10.3390/su12041440

- Jiandong Guo, Chenyu Liang, Kang Wang, Biao Sang, and Yulin Wu “Three-Dimensional Autonomous Obstacle Avoidance Algorithm for UAV Based on Circular Arc Trajectory” 22 April 2021.
  https://doi.org/10.1155/2021/8819618

- Enrique Aldao, Luis M. González-deSantos, Humberto Michinel and Higinio González-Jorge “UAV Obstacle Avoidance Algorithm to Navigate in DynamicBuilding Environments” 10 January 2022.
  https://doi.org/10.3390/drones6010016