Mini Project

on

“AUTONOMOUS DRONE”

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

Kalluri Hansika: 22WH1A0430

Dhatri Sri Singu: 22WH1A0410

K. Harika: 22WH1A0434

S. Sanika: 22WH1A0451

Under the guidance of

Dr. J Naga Vishnu Vardhan

Professor,ECE and Professor In charge Academics

Department of Electronics and Communication Engineering
BVRIT HYDERABAD College of Engineering for Women
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with A Grade
Bachupally, Hyderabad – 500090 2024-25

DECLARATION

We hereby declare that the work described in this report, entitled **"AUTONOMOUS DRONE"** which is being submitted by us in partial fulfilment for the award of the degree of **Bachelor of Technology** in the Department of **Electronics and Communication Engineering** at **BVRIT HYDERABAD College of Engineering for Women,** affiliated to **Jawaharlal Nehru Technological University Hyderabad**, Kukatpally, Hyderabad – 500085 is the result of original work carried out by us under the guidance of **Dr. J Naga Vishnu Vardhan,Professor,ECE and Professor In charge Academics.**

This work has not been submitted for any Degree/Diploma of this or any other institute/university to the best of our knowledge and belief.

Place: Hyderabad

Date:

Kalluri Hansika: 22WH1A0430
Dhatri Sri Singu: 22WH1A0410
K. Harika: 22WH1A0434
S. Sanika: 22WH1A0451

Department of Electronics and Communication Engineering
BVRIT HYDERABAD College of Engineering for Women
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with A Grade
Bachupally, Hyderabad – 500090

Certificate

This is to certify that the major/mini project report, entitled **“AUTONOMOUS DRONE”** is a record of bonafide work carried out by **Kalluri Hansika 22WH1A0430, Dhatri Sri Singu 22WH1A0410, K Harika 22WH1A0434, S Sanika 22WH1A0451** in partial fulfilment for the award of the degree of **Bachelor of Technology** in the department of **Electronics and Communication Engineering** at **BVRIT HYDERABAD College of Engineering for Women,** affiliated to **Jawaharlal Nehru Technological University Hyderabad**, Kukatpally, Hyderabad – 500085.

Supervisor

Head of the Department

External Examiner

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of the task would be incomplete without the mention of the people who made it possible.

We wish to express our deep sense of gratitude to our guide **Dr. J Naga Vishnu Vardhan, Professor, ECE and Professor in charge Academics**, BVRIT HYDERABAD College of Engineering for Women, for his able guidance and suggestions, which helped us in completing this project work on time.

We would like to thank **Dr. Nagesh Deevi,** Associate Professor and Head, Department of Electronics and Communication Engineering for his guidance, support and encouragement.

We express our gratitude towards our honourable Principal, **Dr. K V N Sunitha** and the **management** for providing all the facilities.

We also thank all the **faculty** and **non-teaching staff members** of the Electronics and Communication Engineering department, who supported us directly or indirectly in the successful completion of this project work.

Finally, we thank all our **friends** and **family members** for their continuous support and help.

Kalluri Hansika: 22WH1A0430
Dhatri Sri Singu: 22WH1A0410
K. Harika: 22WH1A0434
S. Sanika: 22WH1A0451

ABSTRACT

This thesis presents the design and implementation of an Indoor Autonomous Drone System that utilizes a self-assembled drone integrated with a companion computer and external sensors to achieve autonomous indoor flight. Unlike outdoor autonomous drones relying on GPS, this project employs Visual-Inertial Odometry (VIO) for accurate indoor localization and mapping. The thesis provides an in-depth guide on hardware assembly, the necessary software setup for autonomous flight, and an evaluation of system performance. Challenges encountered during the development process are also discussed.

CONTENTS

LIST OF FIGURES AND TABLES

1. INTRODUCTION

1.1 Introduction to Drones

A drone, also referred to as an Unmanned Aerial Vehicle (UAV), is an aircraft that can fly without a human pilot on board. These machines can either be remotely operated by a person on the ground or fly autonomously using pre-programmed instructions, GPS, and onboard sensors. The different parts of drone are shown below in figure 1.

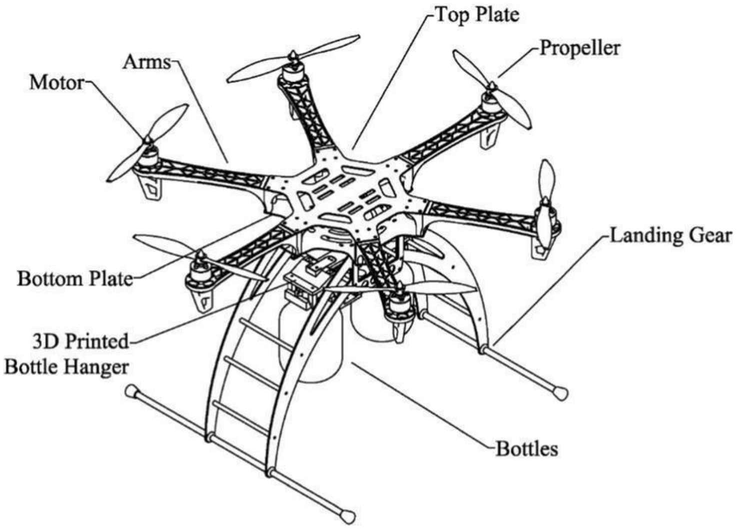


***figure 1**: Diagram of a basic Drone*

The term “drone” originates from the buzzing sound made by male honeybees, which closely resembled the early versions of pilotless aircraft. Over time, the term has become synonymous with a wide range of aerial vehicles used in military, commercial, industrial, and environmental applications.[1] [2] [3]

Different sources define drones in slightly different ways, but they all point to the same core concept. For example:

1. “A drone is an unmanned aircraft or ship that can be guided by remote control or onboard systems.”
2. “A drone is a flying robot capable of autonomous or remote-controlled flight using embedded systems.”
3. “Drones are UAVs that operate without a human crew and can be either automated or remotely piloted.”

1.1.1 Quick Facts about Drones

Abraham Karem is widely recognized as the pioneer of modern UAV technology. He developed early drone prototypes that laid the foundation for the drones used today, especially in defense sectors. Franz von Uchatius, an Austrian engineer, is credited with designing the first drone concept in the mid-19th century, marking the beginning of unmanned aerial systems.

Even Nikola Tesla contributed indirectly through his development of remote control and tele-automation, ideas that later helped shape drone communication and control systems. In World War II, the need for drones became evident. With over 40,000 aircraft and 80,000 crew members lost, militaries saw the value in using pilotless aircraft to reduce casualties and cost. The early motivation behind drone development was both strategic and humanitarian—reducing the risk to human life while maintaining surveillance and attack capabilities.

1.1.2 Applications of drones

Drones (UAVs) are revolutionizing industries by accessing hard-to-reach areas, collecting real-time data, and performing tasks with minimal human effort. They are used in surveillance, inspection, environmental monitoring, disaster response, and agriculture. The seed-scattering drone developed in this project is designed for precision agriculture and reforestation, enabling automated, uniform seed or fertilizer dispersal over large or inaccessible areas. With IoT and telemetry integration, it offers a smart, eco-friendly solution for efficient land management.

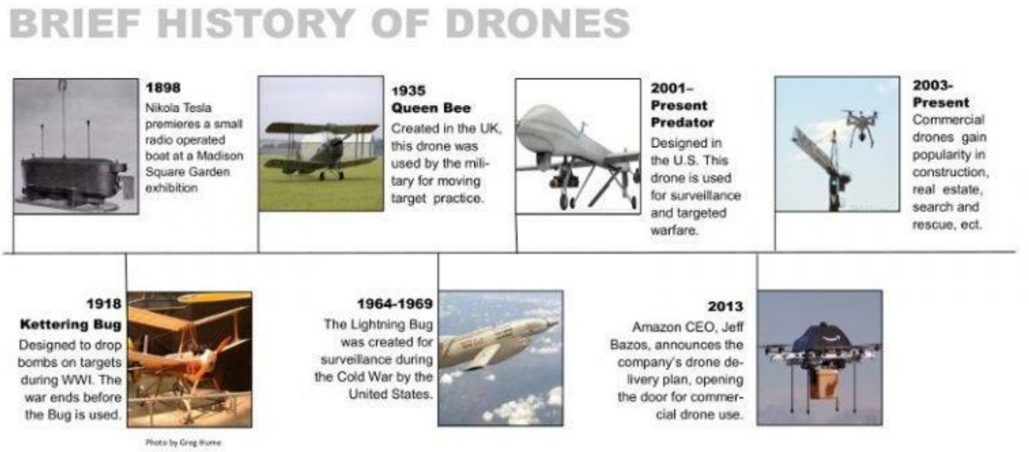
1.1.3 History of Drones

The evolution of drones can be traced back to 1783 in France, when the Montgolfier brothers launched the first unmanned hot air balloon, a primitive example of pilotless flight. Since then, drone technology has steadily advanced through major historical milestones:

1. Early 1900s: Military experiments with unmanned aerial torpedoes.
2. Mid-1900s: Radio-controlled aircraft emerged in warfare.

3. Late 20th century: The use of UAVs expanded for surveillance and scientific purposes.

4. 21st century: Drones became commercially available for agriculture, delivery, media, mapping, and environmental monitoring. The History of drone is clearly displayed in figure 2



BRIEF HISTORY OF DRONES

1898
Nikola Tesla premieres a small radio operated boat at a Madison Square Garden exhibition

1935
Queen Bee
Created in the UK, this drone was used by the military for moving target practice.

2001–Present
Predator
Designed in the U.S. This drone is used for surveillance and targeted warfare.

2003-Present
Commercial drones gain popularity in construction, real estate, search and rescue, ect.

1918
Kettering Bug
Designed to drop bombs on targets during WWI. The war ends before the Bug is used.

Photo by Greg Hume

1964-1969
The Lightning Bug was created for surveillance during the Cold War by the United States.

2013
Amazon CEO, Jeff Bazos, announces the company's drone delivery plan, opening the door for commercial drone use.

***figure 2**: History and evolution of Drones*

1.1.4. Drone Regulations in India

Drones are legally permitted in India under the Drone Rules, 2021, released by the Ministry of Civil Aviation (MoCA). These rules define the categorization, registration, operation, and licensing of drones.

1. All drones must be registered on the *Digital Sky platform* and assigned a *Unique Identification Number (UIN*).
2. Pilots operating drones above certain limits are required to have a valid Remote Pilot Certificate (RPC).
3. Drones *weighing under 2 kg* can be flown *for non-commercial purposes* without a license.
4. However, for commercial use or heavier drones, an RPC and regulatory compliance are mandatory.
5. The *Directorate General of Civil Aviation (DGCA)* oversees and enforces these rules to ensure safe and responsible drone use in Indian airspace.

These policies aim to strike a balance between promoting drone innovation and ensuring public safety and airspace management. The different types of zones with examples are shown below figure 3.

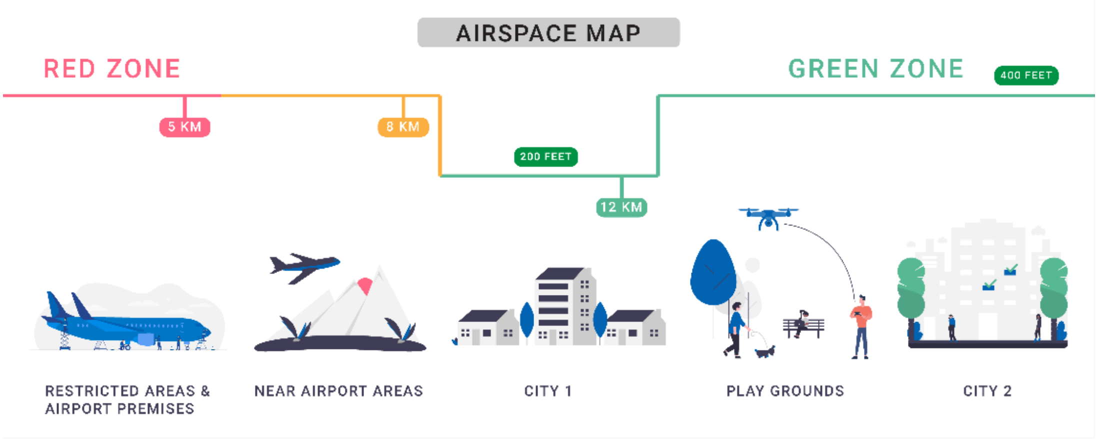


figure 3: *Drone rules and categories*

1.1.5 Types of Drones by Structure

Drones come in various forms depending on their *aerodynamic structure and purpose*:

1. **Multirotor Drones**: These are the most common type, especially in commercial and consumer use. They are lifted and stabilized using multiple propellers. Examples include are shown below in figure 4:
   - Tricopter (3 rotors)
   - Quadcopter (4 rotors)
   - Hexacopter (6 rotors)
   - Octocopter (8 rotors)

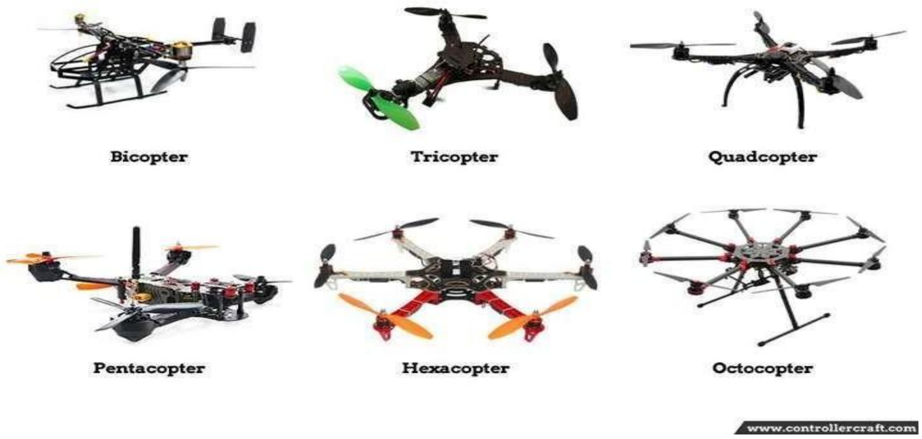


***figure 4**: Quadcopter and Multirotor types*

2. **Fixed-Wing Drones**: These have airplane-like wings and cannot hover. They are ideal for long-range missions and are more energy-efficient during horizontal flight. The fixed wind drone is as below figure 5.

***figure 5**: Fixed wing drone*

3. **Single-Rotor Drones**: These resemble helicopters and use one large rotor and a tail rotor. They can carry heavier payloads but require more complex control.

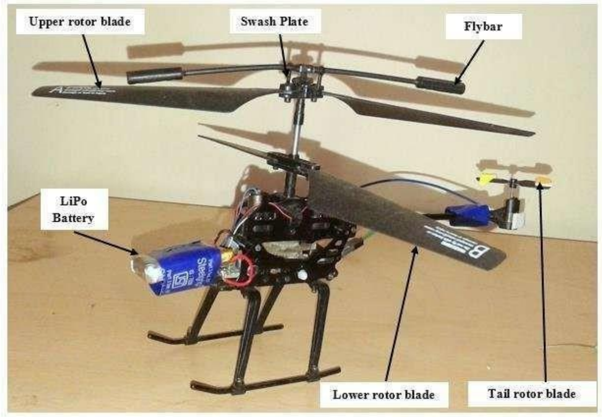


***figure 6**: Single-Rotor drone*

4. **Hybrid VTOL (Vertical Takeoff and Landing)** Drones: These combine the features of both multirotor and fixed-wing drones. They take off like a helicopter and then transition to forward flight like an airplane. The Hybrid VTOL drone is shown below in figure 6.

***figure** 7: Hybrid VTOL drones*

1.1.6 Classification by Weight and Control

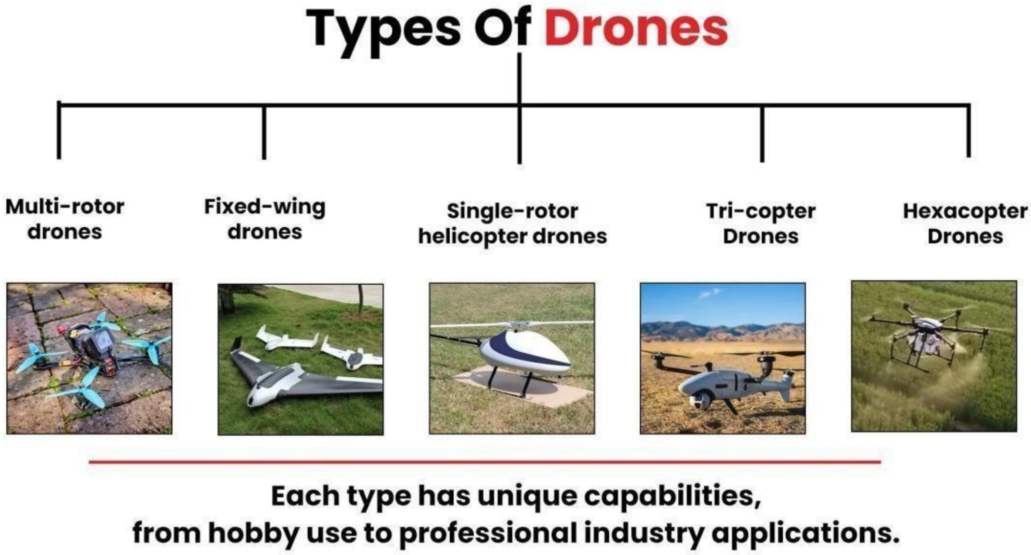

Types Of Drones
Multi-rotor drones
Fixed-wing drones
Single-rotor helicopter drones
Tri-copter Drones
Hexacopter Drones
Each type has unique capabilities, from hobby use to professional industry applications.

***figure 8**: Types of drones*

Drones are also categorized by:

1. **Weight**: Drones are classified into micro, small, medium, and large categories, which determines how they're regulated in different countries.

Types of Drone	**Weight**	**Remote Pilot Licence**
Nano	= 250 g	Not required
Micro	More than 250 g and up to 2 kg	Not required for non-commercial and required for commercial use
Small	More than 2 kg & up to 25 kg	Required
Medium	More than 25 kg up to 150 kg	Required
Large	More than 150 kg	Required

Table ***1****: Weight classification of drone*

2. **Control Method**:

   - *Autonomous Drones*: Operate without manual input once programmed.

   - *Remotely Piloted Drones*: Controlled by a user in real time using a transmitter or ground station.

   Autonomous drone and remote piloted drone are shown below in figure 8.

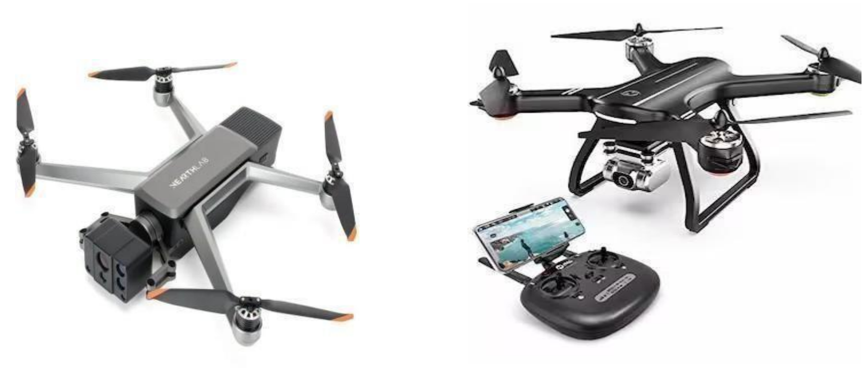


figure 9*: Autonomous and remote piloted drones*

1.1.7 Parts of Drone

A drone is built from multiple electronic and mechanical components, all working in harmony. Each part plays a vital role in flight stability, maneuverability, and performance.

1.1.7.1 Frame

The frame is the skeletal structure that holds all other drone components together. It determines the layout, strength, and weight distribution. Good frame design ensures a well-balanced center of gravity, which is essential for stable flight. Frames vary in shape and size depending on the drone's type and purpose. The frame with two different colors is shown below in figure 9.

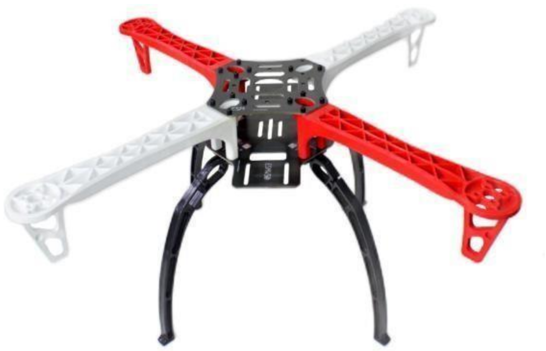


***figure 10**: Diagram of a basic drone*

1.1.7.2 Propellers

Propellers are rotating blades that generate lift by creating a pressure difference between the top and bottom surfaces. This pressure differential helps the drone rise into the air. The 4 propellers used in drone is shown below in figure 10.

***figure 11**: Propellers*

1.1.7.3 Electronic Speed Controller (ESC)

The ESC manages the speed of the motors. It interprets commands from the flight controller and adjusts motor speeds accordingly. This helps in stabilizing the drone and controlling its direction. The ESC is as below figure 11.

***figure 12**: Electronic Speed Controller*

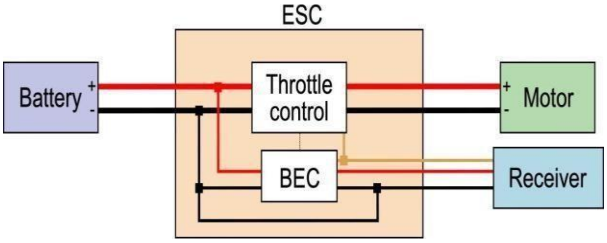


***figure 13**: Block Diagram representing the connections b/w Battery and Motor through ESC*

1.1.7.4 Flight Controller (FC)

The brain of the drone, the flight controller processes data from sensors, GPS, and remote signals. It sends control signals to ESCs, ensuring the drone responds accurately to pilot inputs.

Main roles are Perception, Control, Communication and structure of flight controller is shown below in figure 13

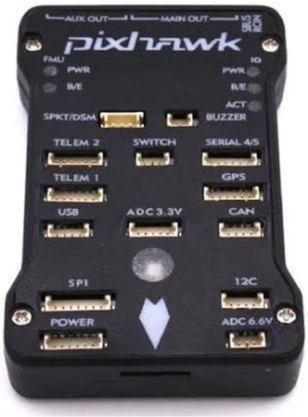


***figure 14**: Flight Controller*

1.1.7.5 Power Distribution Board (PDB)

The PDB distributes power from the battery to various components like ESCs, FC, and peripherals. In many drones, the PDB is integrated into the flight controller for compactness. The PDB structure is shown below in figure 14.

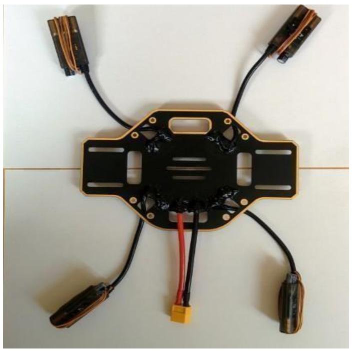


figure 15*: Power Distribution Board*

1.1.7.6 Battery

Drones typically use Li-Po (Lithium Polymer) batteries due to their high energy density and lightweight. These batteries power the motors and all onboard electronics. The battery is as shown below in figure 15.

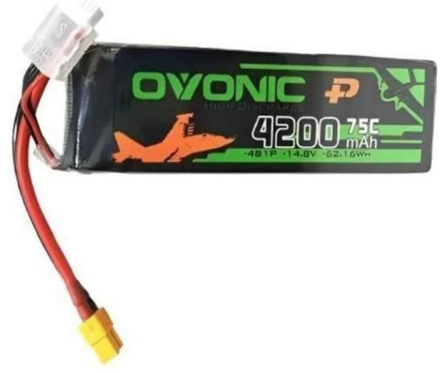


figure 16*: Battery*

1.1.7.7 Radio Controller Receiver (Rx)

The receiver interprets signals sent from the remote controller and forwards them to the flight controller. It translates radio waves into electrical signals to control the drone. The Radio Controller will be as shown in figure 16.

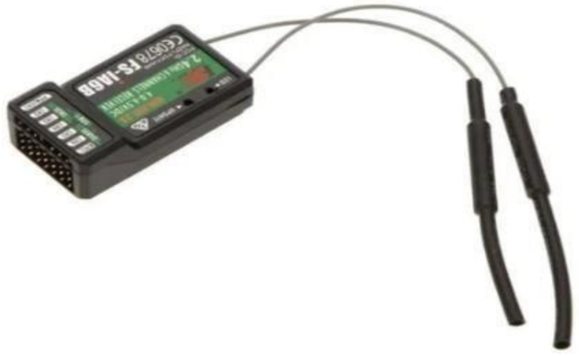


Figure 17: Radio Controller Receiver

1.1.7.8 Radio Controller Transmitter (Tx)

This is the handheld device used by the pilot. It sends control inputs to the drone over radio frequencies via multiple channels and shown in figure 17:

- Roll: Left/Right tilt
- Pitch: Forward/Backward tilt
- Yaw: Rotation
- Throttle: Speed and altitude

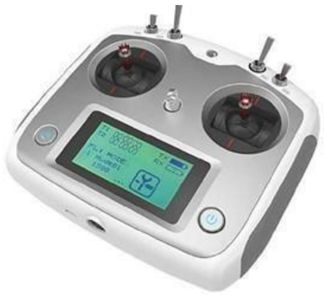


figure 18: Radio Controller Transmitter

1.1.7.9 GPS Module

The GPS module enables the drone to determine its location via satellite signals. It supports autonomous flight features like Return to Home, Waypoint Navigation, and Position Hold and shown in figure 18.

figure 19: *GPS module*

1.1.7.10 Camera

The camera captures video feed, allowing for real-time, FPV flight

figure 20: *Camera*

1.1.7.11 Gimbal

A Gimbal is a support that allows rotation about a single axis. A Set of two or three gimbals mounted at 90 degrees to each other allow a mounted object to remain level and independent of the rotation of its support regardless of the movement of the gimbals. A two axis gimbal will provide stabilization for pitch and roll only, where as three axis gimbal will provide stabilization for roll, pitch and yaw

figure 21: *Gimbal*

2. PHYSICS OF FLIGHT

Understanding how drones fly begins with the physics that governs all aircraft. Flight is not just about technology—it's rooted in natural forces and principles that are also seen in birds, gliders, and even rising smoke. These phenomena are governed by four key physical forces, which must be balanced carefully to achieve and sustain flight. The force of light is as shown below in figure 21.

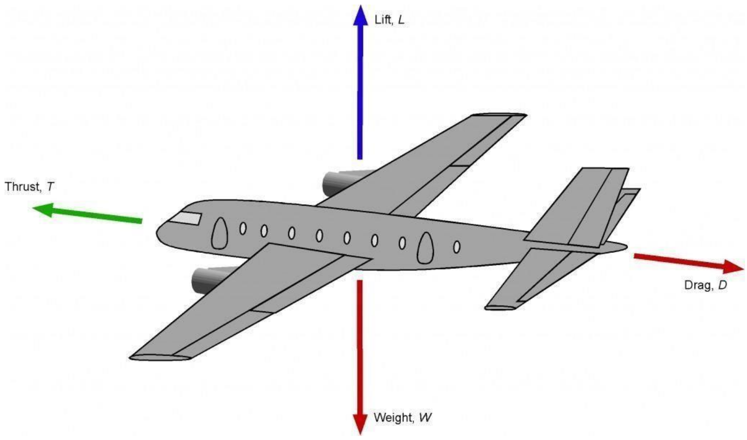


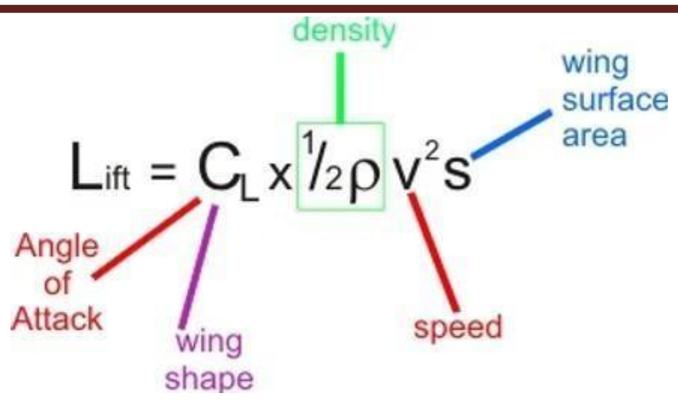
figure 22*: Four forces of flight*

2.1 The Four Physical Forces of Flight

For any heavier-than-air aircraft like drones, stable flight is achieved through a balance of **lift, thrust, drag, and weight**. Each force plays a critical role in the drone's motion and control.

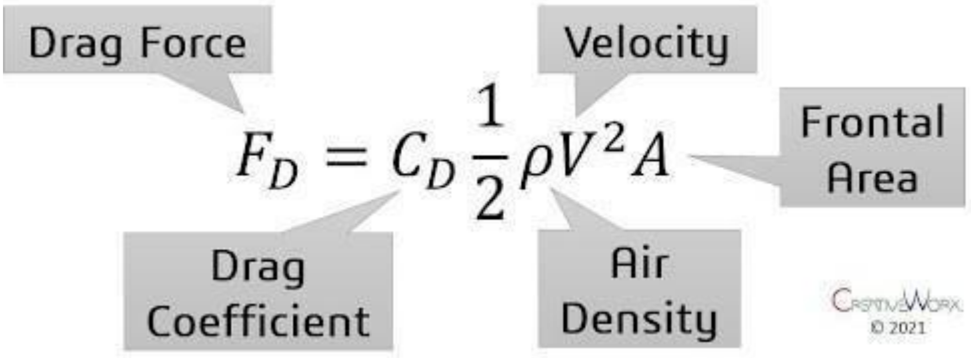
2.1.1 Lift

Lift is the upward force that allows the drone to rise against gravity. In drones, lift is generated by the rapid rotation of the propellers, which accelerates air downward. According to ***Bernoulli's Principle***, faster airflow over the top surface of a propeller or wing creates lower pressure, while the slower air underneath creates higher pressure—resulting in upward lift.



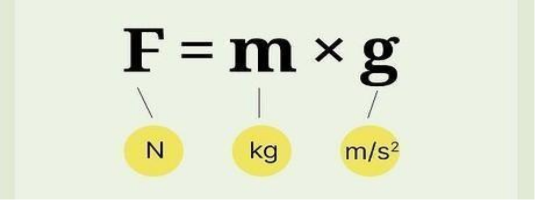
2.1.2 Drag

Drag is the air resistance that acts opposite to the direction of motion. As a drone moves through the air, it pushes against air particles, which in turn resist its motion. Minimizing drag is essential for better energy efficiency and stability. A streamlined body and lightweight materials help in reducing drag



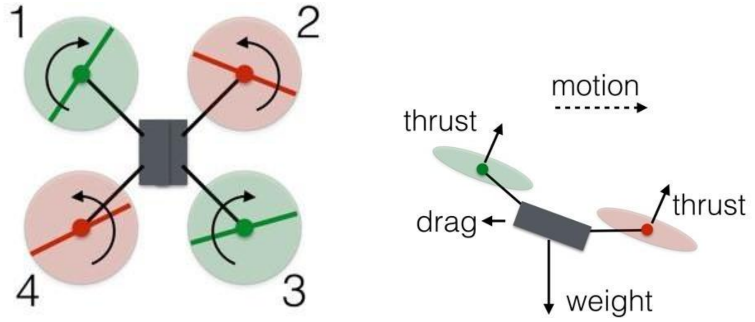
2.1.3 Weight

Weight is the downward force caused by gravity acting on the mass of the drone. It directly affects how much lift must be generated. The total weight includes the frame, battery, electronics, and any payload (such as seeds in your project). Proper distribution of weight also affects the drone's balance and maneuverability.



2.1.4 Thrust

Thrust is the forward (or vertical) force that propels the drone through the air. For multirotor drones, thrust is created by the high-speed spinning of propellers powered by motors. This force must overcome both gravity (to ascend) and drag (to move forward efficiently). Thrust can be calculated using Newton’s second law: ***F=ma.***



.

3. CALIBRATING AND SETTING UP THE FLIGHT CONTROLLER

3.1 Procedure for Setting up Pixhawk

Pixhawk 2.4.8 requires Mission Planner to calibrate and setup, Connect Pixhawk with computer to calibrate.

Step 1. Install Mission Planner. Go to its official website https://ardupilot.org/planner to download and install it on your computer.

Step 2. Connect Pixhawk with your computer by a micro USB data cable.

Step 3. Use Mission Planner to connect and calibrate.

The Calibration and setups of Mission planner are shown below in figures from 20(a)-20(k).

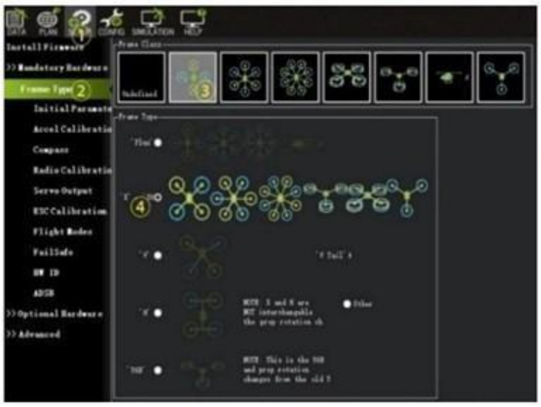
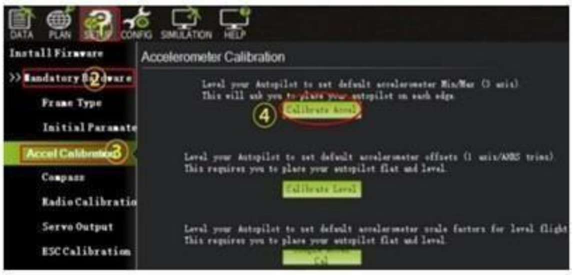


fig. **20** ***(a).*** *Select frame type*



Level your Autopilot to set default accelerometer Min/Max (3 axis)
This will ask you to place your autopilot on each edge.

Level your Autopilot to set default accelerometer offsets (1 axis/AHRS trims)
This requires you to place your autopilot flat and level.

Level your Autopilot to set default accelerometer scale factors for level flight
This requires you to place your autopilot flat and level.

fig. **20** (***b***)***.*** *Calibrate Accel*

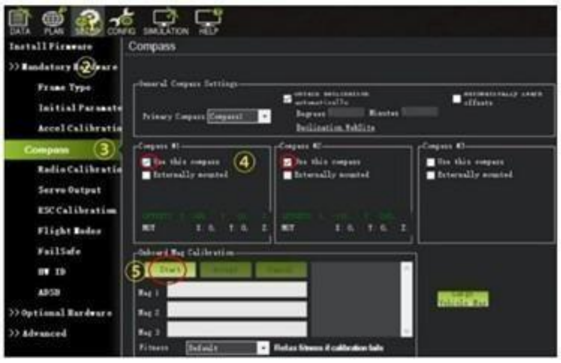


fig **20** ***(c).*** *Calibrate compass*

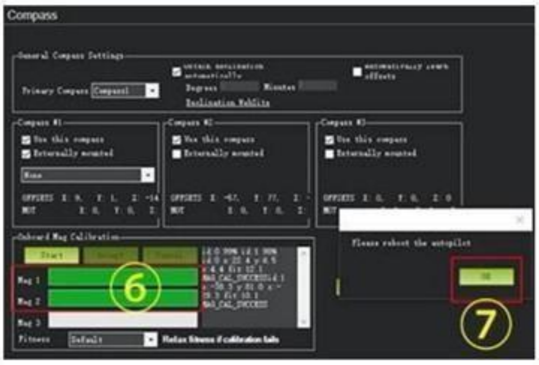


fig. **20** ***(d).*** *Calibrate ESC*

Note: Only Pitch (Ch 2) requires to be reversed.

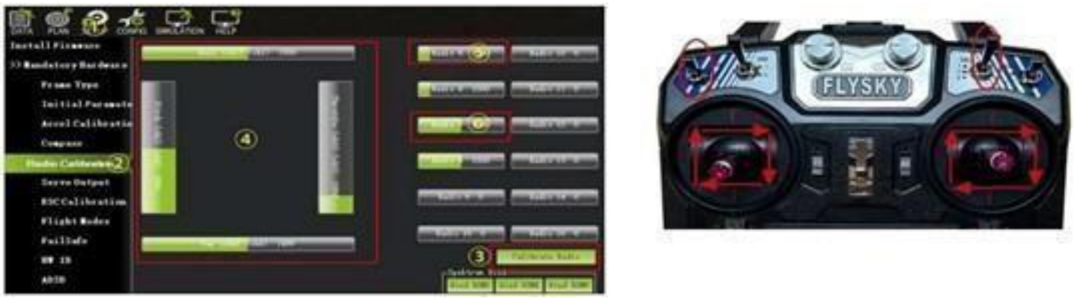
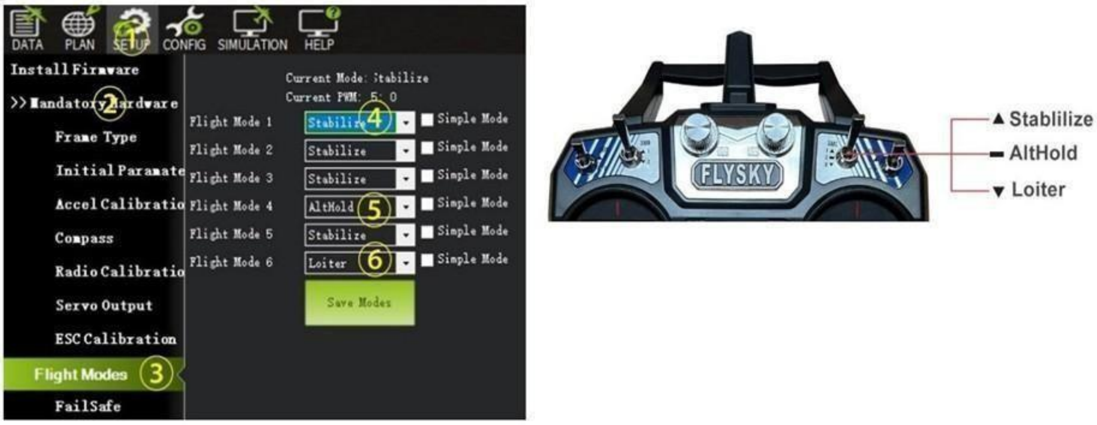


fig. **20** ***(e).*** *Calibrate radio*

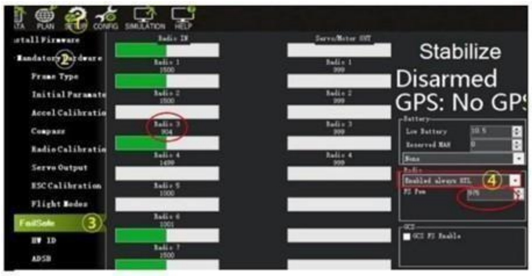


Current Mode: Stabilize
Current PWM: 5: 0

Flight Mode 1 Stabilize □ Simple Mode
Flight Mode 2 Stabilize □ Simple Mode
Flight Mode 3 Stabilize □ Simple Mode
Flight Mode 4 AltHold □ Simple Mode
Flight Mode 5 Stabilize □ Simple Mode
Flight Mode 6 Loiter □ Simple Mode

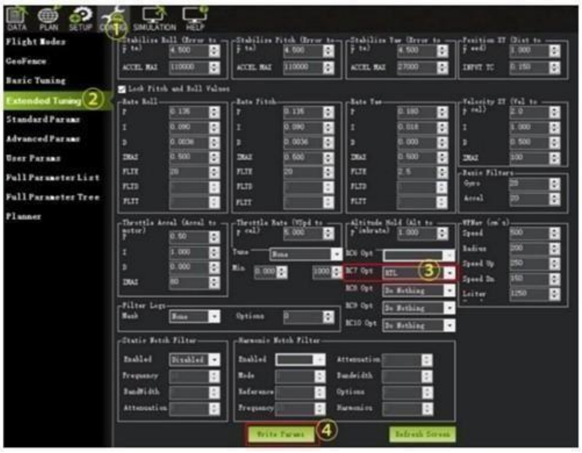
Save Modes

fig. **20** ***(f).*** *Set flight modes*



Stabilize
Disarmed
GPS: No GPS

Enabled always RTL



Ch7 Opt RTL

fig. **20** ***(g).*** *Set fail safe*

fig. **20** ***(h).*** *Set CH 7 RTL*

Step 4. Test motors (Plug in the battery)

1. Check all the signal wires of ESC plugged into Pixhawk right.
2. Check the direction of motor rotation (#1 & 2 CCW, #3 & 4 CW)
3. If the rotation direction is not right, you should exchange any 2 of the 3 wires. Step 5. Calibrate ESCs (Plug and unplug the battery as follows)

Step 5. Calibrate ESCs (Plug and unplug the battery as follows)

1. Turn on the transmitter and push the throttle to the top. Plug in the drone battery and wait for the Pixhawk to reboot.

2. Unplug the battery, plug it back in, and wait for the Pixhawk to reboot again. Press and hold the safety switch for 2 seconds; motors will make one long beep. Pull the throttle to the bottom; motors will make two short beeps and one long beep (or music) — ESC calibration is complete.

3. Plug in the battery and reboot the Pixhawk. Press the safety switch for 2 seconds until the light turns ON.

4. Try arming the drone via Mission Planner; if it fails, check and resolve the error. To arm manually, hold the left joystick at the bottom-right for 3 seconds — motors will start rotating.

fig. **20** ***(i).*** *handling the Controller*

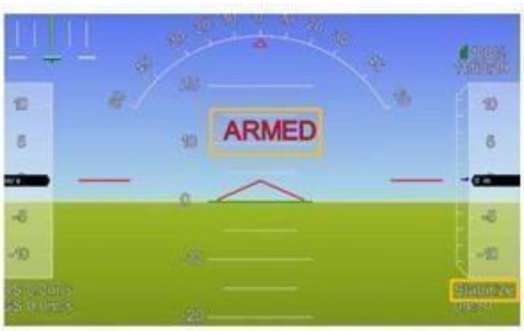


fig. **20** ***(j).*** *Observe the leveling.*

	Left stick	Right stick
Up & down	altitude up & down	movement to forward & backward
Left & right	rotation to left & right	movement to left & right

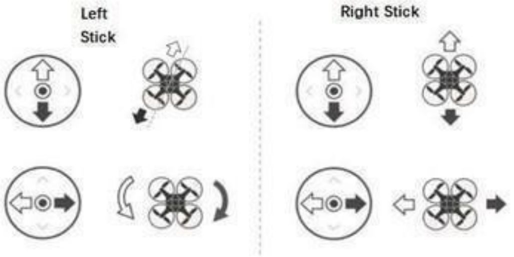


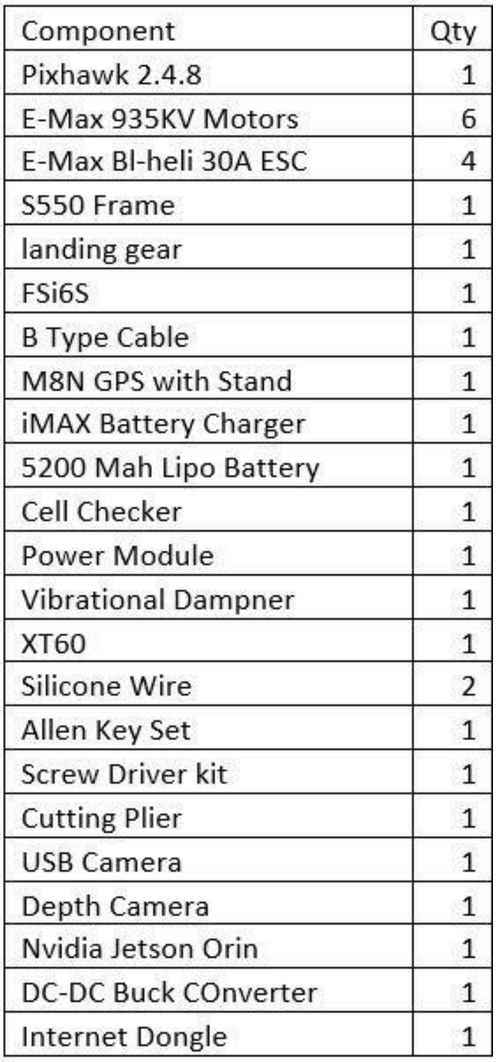
fig. **20** ***(k):*** *Understanding controller behaviour*

To ***disarm*** the drone, turn off the safety switch and disconnect the battery. Hold the left stick at the lowest left corner for about 3 seconds to disarm the drone (motors stop spinning).

4 . METHODOLOGY

4.1 Components Required

Following are the components required to build a Air quality monitoring drone [6] [7]



Component	Qty
Pixhawk 2.4.8	1
E-Max 935KV Motors	6
E-Max Bl-heli 30A ESC	4
S550 Frame	1
landing gear	1
FSi6S	1
B Type Cable	1
M8N GPS with Stand	1
iMAX Battery Charger	1
5200 Mah Lipo Battery	1
Cell Checker	1
Power Module	1
Vibrational Dampner	1
XT60	1
Silicone Wire	2
Allen Key Set	1
Screw Driver kit	1
Cutting Plier	1
USB Camera	1
Depth Camera	1
Nvidia Jetson Orin	1
DC-DC Buck COnverter	1
Internet Dongle	1

Landing gear:

The landing gear is a vital drone component that provides stable support and protects the frame, motors, and sensors during takeoff and landing. Made from lightweight, durable materials like carbon fibre or plastic, it absorbs impact and keeps the drone elevated to avoid damage to propellers and payloads. Landing gear designs range from fixed to retractable, offering shock absorption, stability on various terrains, and easy installation with minimal added weight. For crowd monitoring drones, reliable landing gear ensures safe operations and protects sensitive equipment in different environments. The structure of landing gear is as shown below in figure 21.

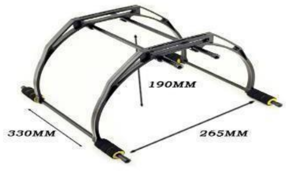


Fig:23 landing gear

Cell Checker:

A LiPo battery cell checker is a compact device used by drone pilots to monitor individual cell voltages and overall battery health. It provides digital readings of each cell's voltage, total voltage, and voltage differences to help prevent over-discharging and ensure balanced charging. Compatible with various battery types, it enhances battery safety and longevity. Advanced models may also offer balancing, discharging, and USB charging features for comprehensive battery maintenance and shown below in figure 22.

Fig: 24 cell checker

Power Module:

The Power Module is a crucial component that supplies stable and regulated power to the flight controller and other onboard electronics for drones. It typically integrates voltage and current sensors, allowing real-time monitoring of battery health and consumption. Designed to handle high current loads, it ensures safe power delivery while protecting against overvoltage, undervoltage, and short circuits. Compact and lightweight, the Power

Module simplifies wiring and enhances drone reliability, making it essential for efficient flight performance and battery management and shown in figure 23.

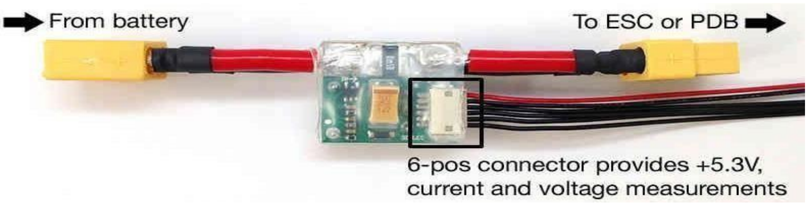

From battery
To ESC or PDB
6-pos connector provides +5.3V, current and voltage measurements

Fig: 25 Power Module

Raspberry Pi 4:

The Raspberry Pi 4 is a powerful, compact single-board computer designed for versatile applications including robotics, IoT, and drone development[4][15]. It features a faster CPU, improved GPU, and increased RAM options compared to its predecessors, delivering enhanced performance for complex tasks and real-time processing. The board includes multiple USB ports, HDMI outputs, and GPIO pins for extensive connectivity and hardware interfacing. Its energy-efficient design and robust community support make the Raspberry Pi 4 an ideal choice for hobbyists and professionals seeking a reliable, customizable computing platform and The structure will be as shown in below figure 24.

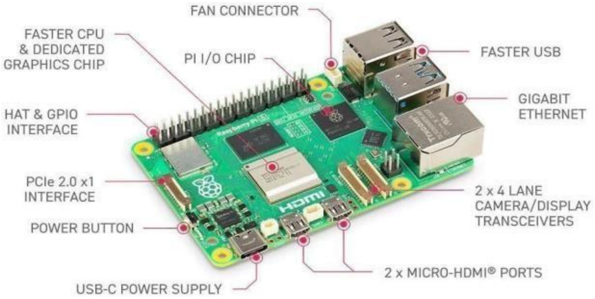


Fig: 26 Raspberry Pi 4

Power bank light weight:

A lightweight power bank for drones is a compact, high-energy backup battery that powers controllers and accessories without affecting flight performance as shown below in figure 29. It offers fast charging, multiple ports, and safety features like overcharge protection, making it ideal for extended field operations.

Fig: 27 Power bank

Jumper Wires:

Jumper wires are flexible connectors used for prototyping electronic circuits as shown below in figure 30. They link components on breadboards or microcontroller pins without soldering. Available in male- tomale, male-to-female, and female-to-female types, they come in various lengths and colors.

Fig:28: Jumper wires

4.2 Software Used:

To successfully implement the **Autonomous Drone** system, several software tools were employed, each playing a vital role in different stages of development, flight control, and real-time data processing.

4.2.1 Mission Planner:

Mission Planner was used as the Ground Control Station (GCS) software to configure, calibrate, and monitor the F450 quadcopter drone. It provided a graphical interface for setting up the drone's flight parameters, uploading autonomous waypoints, and viewing telemetry data during missions. Mission Planner was essential for ensuring accurate communication between the drone and the base station, as it supports the MAVLink protocol and works seamlessly with ArduPilot firmware. Features such as GPS tracking, battery monitoring, live telemetry plotting, and flight data logging were crucial in testing and validating the drone's behaviour during crowd surveillance operations.

4.2.2 Ubuntu:

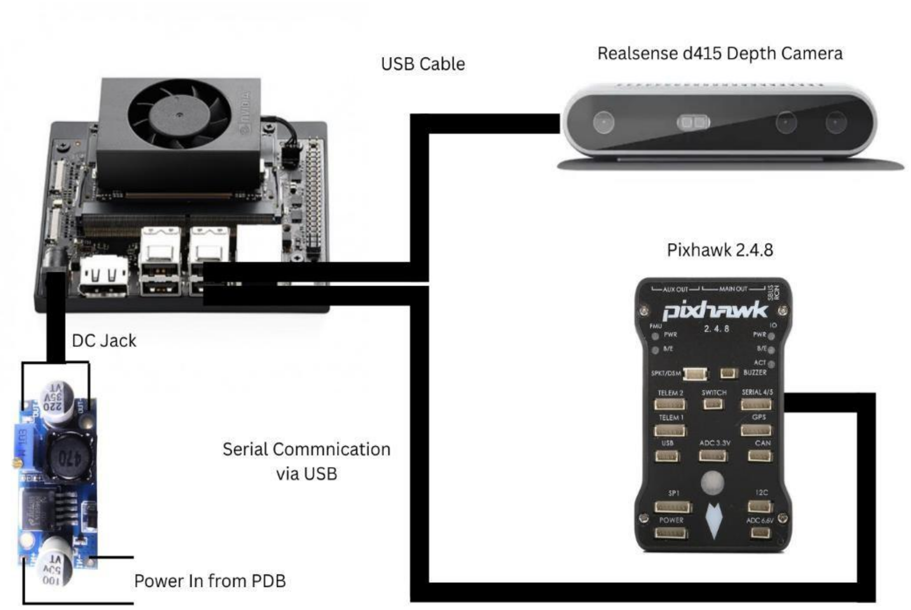
Ubuntu is an open-source Linux operating system, was installed on the Raspberry Pi for running the image processing tasks. Ubuntu offered a stable and lightweight environment suitable for executing Python scripts and integrating machine learning libraries. It allowed smooth handling of real-time video input, camera interfacing, and performance tuning. Being open-source, Ubuntu provided the flexibility and control needed for low-latency, onboard processing without the overhead of unnecessary services found in heavier operating systems.

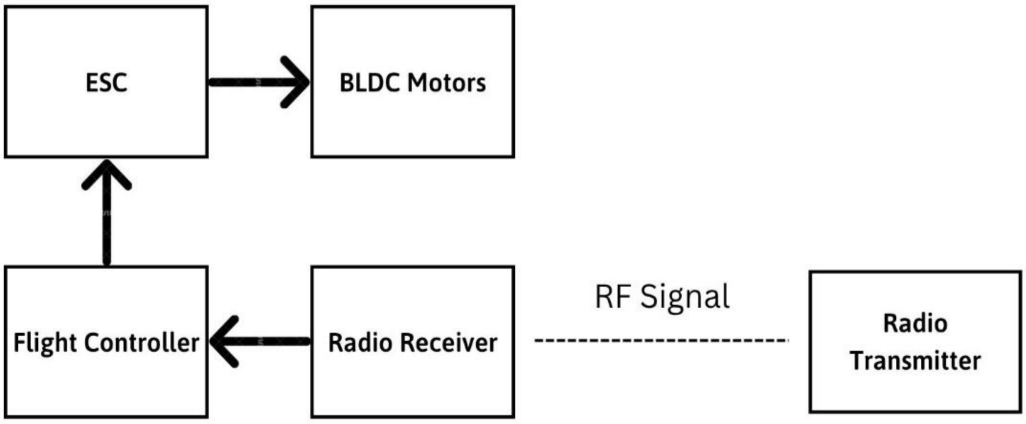
4.2.3 Python:

Python is used to build an environmental monitoring system using a Raspberry Pi. It collects data from a DHT11 sensor for temperature and humidity using the Adafruit_DHT library. For gas sensing, it reads analog values from MQ2 and MQ135 sensors through the MCP3008 ADC, using the spidev library to communicate via SPI.The collected sensor data is interpreted to assess air quality levels. Python then logs this information, along with a timestamp (using the datetime module), into a structured CSV file using the csv module. Additionally, the pandas library is used to read and extract recent entries from the CSV file for analysis or visualization.

Python's simplicity and strong hardware interfacing libraries make it highly suitable for IoT and data logging applications like this, enabling end-to-end data collection, processing, storage, and retrieval.

4.3 Block Diagram:





5. WORKING AND DESCRIPTION

Initialization Phase

Step 1: Initialize Depth Camera (e.g., RealSense)

- Configure and start the depth and RGB streams.
- Retrieve and display device information.
- Obtain the depth scale for accurate distance measurements.

Step 2: Initialize Drone Connection

- Establish a serial connection to the drone flight controller.
- Wait for the drone to be ready for commands.

Takeoff Phase

Step 3: Wait for Manual Arm and Takeoff Command

- Monitor for the drone to be armed and set to a stable mode (e.g., ALT_HOLD).
- Once armed, command the drone to ascend to a predefined target altitude.
- Wait until the drone reaches the desired altitude before proceeding.
- Forward Motion and Obstacle Detection

Step 4: Begin Forward Motion

- Send velocity commands to move the drone forward slowly.

Step 5: Start Frame Processing Loop

- Continuously capture frames from the depth and color camera.
- Retry if frame capture times out (with a maximum retry limit).

Step 6: Detect Large Nearby Object

- Process the depth image to detect any object within a maximum distance.
- Apply image filtering and contour detection.
- Determine the object's location, distance, and estimated area in meters squared.

Step 7: If Target Object is Detected

- If an object is found within close proximity and of significant size:
- Stop the drone's motion.
- Save the current depth, color image, and point cloud data.
- Set a flag indicating that a target sheet/object was detected.

Function	**Purpose**
start_realsense_pipeline()	Initializes and starts the depth camera stream.
wait_for_manual_arm_and_takeoff()	Waits for user to arm and takeoff the drone.
send_body_velocity()	Sends velocity commands to the drone in the body frame.
detect_large_object()	Detects objects in depth image based on size and distance.
roll_right() / pitch_forward()	Commands drone to avoid obstacle via roll/pitch.
save_detection_images()	Saves visual and point cloud data when object is detected.

6. RESULT

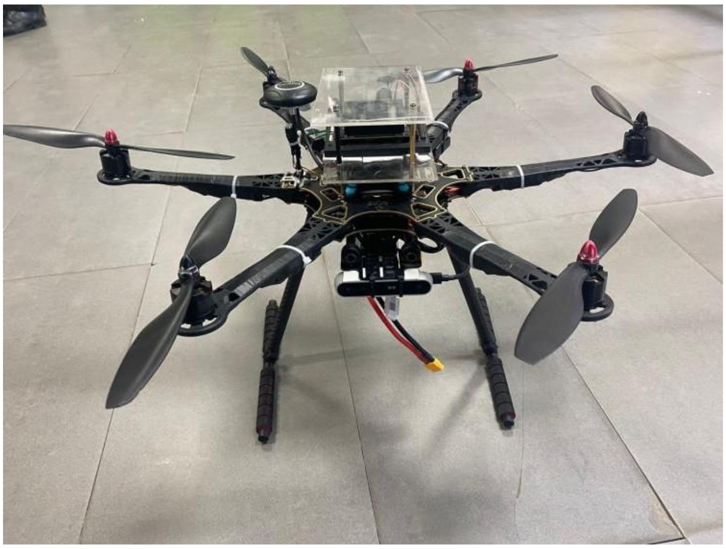


Fig 29: Design of Autonomous Drone

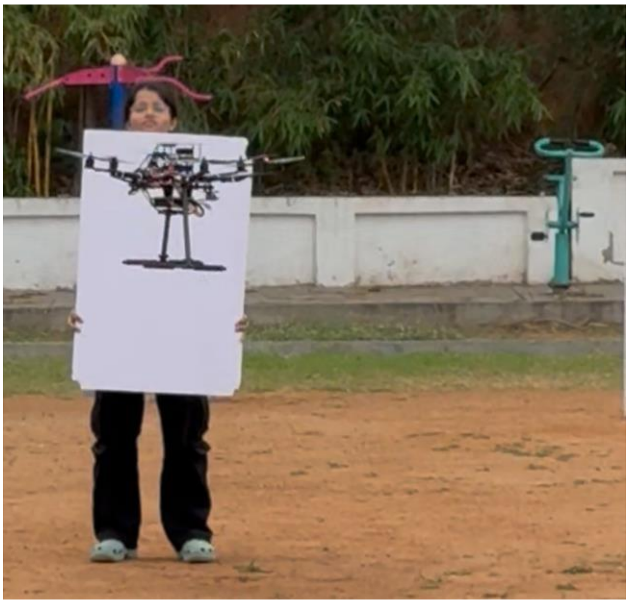


Fig 30 : Illustrates the autonomous drone operating in normal conditions before detecting an obstacle.

Fig :31: Illustrates the autonomous drone operating in normal conditions after detecting an obstacle.

7. APPLICATIONS

1. Utilities and Power
   Wind Turbine Inspection, Tower Inspection, Hydroelectric Inspection, Solar Field Inspection

2. Construction
   Monitoring, Surveying, Roof Inspection, Building Inspection

3. Infrastructure
   Urban Planning, Bridge Inspection, Highway

4. Mining and aggregates
   Project Over site, Volumetric Stockpile Measurement, Slope Monitoring

5. Oil & Gas
   Pipeline Inspection, Refinery Flare Tip Inspection, Tank Level Measuring

6. Public Safety
   Search & Rescue, Cloud Monitoring, Law Enforcement

7. Fire Fighting
   Monitoring, Inspection, Situational Awareness

8. Agriculture
   Crop Inspection, Land Use Surveying, Irrigation Inspection

9. Military
   Intelligent & Research, Defense, Anti Weaponry, Delivery

10. Research
    GIS Studies, Wildlife Management, Environmental Studies

8. APPENDIX

```
 import time
import pyrealsense2 as rs
import numpy as np
import cv2
from dronekit import connect, VehicleMode
from pymavlink import mavutil
import os
from datetime import datetime

def print_device_info(device):
    print("RealSense Device Information:")
    print(f"  Name: {device.get_info(rs.camera_info.name)}")
    print(f"  Serial Number: {device.get_info(rs.camera_info.serial_number)}")
    print(f"  Firmware Version: {device.get_info(rs.camera_info.firmware_version)}")
    print(f"  USB Type Descriptor: {device.get_info(rs.camera_info.usb_type_descriptor)}")

def start_realsense_pipeline():
    pipeline = rs.pipeline()
    config = rs.config()
    config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
    config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
    try:
        profile = pipeline.start(config)
    except Exception as e:
        print(f"Failed to start RealSense pipeline: {e}")
        return None, None, None

    device = profile.get_device()
    print_device_info(device)

    try:
        depth_sensor = device.first_depth_sensor()
        depth_scale = depth_sensor.get_depth_scale()
    except Exception as e:
        print(f"Error getting depth sensor: {e}")
        pipeline.stop()
        return None, None, None

    print(f"Depth scale: {depth_scale}")
    return pipeline, depth_scale, profile

def save_detection_images(depth_frame, color_frame):
    depth_image = np.asanyarray(depth_frame.get_data())
    color_image = np.asanyarray(color_frame.get_data())
    depth_colormap = cv2.applyColorMap(
        cv2.convertScaleAbs(depth_image, alpha=0.03), cv2.COLORMAP_JET)
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    save_dir = os.path.join("detections", timestamp)
    os.makedirs(save_dir, exist_ok=True)

    np.save(os.path.join(save_dir, "detected_depth.npy"), depth_image)
    cv2.imwrite(os.path.join(save_dir, "detected_color.png"), color_image)
    cv2.imwrite(os.path.join(save_dir, "detected_depth_colormap.png"), depth_colormap)

    pc = rs.pointcloud()
    pc.map_to(color_frame)
    points = pc.calculate(depth_frame)
    points.export_to_ply(os.path.join(save_dir,  "detected_pointcloud.ply"),  color_frame)

    print(f"Saved detection data to folder: {save_dir}")

# === Setup RealSense ===
pipeline, depth_scale, profile = start_realsense_pipeline()
if pipeline is None:
    print("Exiting due to RealSense init failure.")
    exit(1)

print("Warming up RealSense camera...")
time.sleep(3)

# === Connect to drone ===
vehicle = connect('/dev/ttyACM0', baud=57600, wait_ready=True)

def wait_for_manual_arm_and_takeoff(target_alt: float = 2.0) -> None:
    print("Waiting for vehicle to be armed and set to ALT_HOLD...")
    while not vehicle.armed or vehicle.mode.name != "ALT_HOLD":
        print(f" Armed: {vehicle.armed}, Mode: {vehicle.mode.name}")
        time.sleep(1)
    print("Vehicle is armed. Taking off...")
    vehicle.simple_takeoff(target_alt)
    while True:
        alt = vehicle.location.global_relative_frame.alt
        print(f" Altitude: {alt:.2f}")
        if alt >= target_alt * 0.95:
            print("Reached target altitude")
            break
        time.sleep(1)

def send_body_velocity(vx, vy, vz):
    msg = vehicle.message_factory.set_position_target_local_ned_encode(
        0, 0, 0, mavutil.mavlink.MAV_FRAME_BODY_NED,
        0b0000111111000111,
        0, 0, 0,
        vx, vy, vz,
        0, 0, 0,
        0, 0)
    vehicle.send_mavlink(msg)
    vehicle.flush()

def stop_motion():
    send_body_velocity(0, 0, 0)

def roll_right(rate_rad_s=0.3):
    thrust = 0.5
    msg = vehicle.message_factory.set_attitude_target_encode(
        0, 0, 0,
        0b00000100,
        [0, 0, 0, 0],
        rate_rad_s, 0, 0,
        thrust)
    vehicle.send_mavlink(msg)
    vehicle.flush()

def pitch_forward(rate_rad_s=0.3):
    thrust = 0.5
    msg = vehicle.message_factory.set_attitude_target_encode(
        0, 0, 0,
        0b00000100,
        [0, 0, 0, 0],
        0, -rate_rad_s, 0,
        thrust)
    vehicle.send_mavlink(msg)
    vehicle.flush()

def detect_large_object(depth_frame: rs.depth_frame,
                        max_distance: float = 0.8,
                        min_area_pixels: int = 5000):
    depth_image = np.asanyarray(depth_frame.get_data())
    max_dist_units = max_distance / depth_scale
    depth_filtered = np.where(depth_image > 0, depth_image, 999999)
    mask = np.array(depth_filtered < max_dist_units, dtype=np.uint8) * 255

    kernel = np.ones((5, 5), np.uint8)
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        largest = max(contours, key=cv2.contourArea)
        area_pixels = cv2.contourArea(largest)
        if area_pixels >= min_area_pixels:
            M = cv2.moments(largest)
            if M["m00"] == 0:
                return False, None, None, None, None
            cx = int(M["m10"] / M["m00"])
            cy = int(M["m01"] / M["m00"])
            distance = depth_frame.get_distance(cx, cy)
            intr = depth_frame.profile.as_video_stream_profile().intrinsics
            fx, fy = intr.fx, intr.fy
            x, y, w, h = cv2.boundingRect(largest)
            width_m = (w * distance) / fx
            height_m = (h * distance) / fy
            approx_area_m2 = width_m * height_m
            return True, cx, cy, approx_area_m2, distance
    return False, None, None, None, None

# === Main logic ===
try:
    wait_for_manual_arm_and_takeoff(target_alt=1.3)
    print("Moving forward at 0.2 m/s")
    send_body_velocity(0.3, 0, 0)

    sheet_detected = False
    roll_done = False
    timeout_retry_count = 0
    MAX_TIMEOUT_RETRIES = 10

    while True:
        try:
            frames = pipeline.wait_for_frames(timeout_ms=2000)
            timeout_retry_count = 0
        except RuntimeError:
            timeout_retry_count += 1
            print("Timeout. Retrying...")
            if timeout_retry_count >= MAX_TIMEOUT_RETRIES:
                print("Too many timeouts. Restarting pipeline.")
                pipeline.stop()
                time.sleep(1)
                pipeline, depth_scale, profile = start_realsense_pipeline()
                continue
            continue

        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()
        if not depth_frame or not color_frame:
            print("Frame missing.")
            continue

        detected, cx, cy, area_m2, dist = detect_large_object(depth_frame)
        if detected:
            print(f"Object at {dist:.2f}m, area ~{area_m2:.3f} m²")
        else:
            print("No object detected.")

        if detected and dist <= 0.5 and area_m2 >= 0.05 and not sheet_detected:
            print("Target object detected. Saving image and stopping.")
            stop_motion()
            save_detection_images(depth_frame, color_frame)
            sheet_detected = True
            time.sleep(1)

        if sheet_detected and not roll_done:
            print("Rolling right to avoid obstacle...")
            while True:
                try:
                    frames = pipeline.wait_for_frames(timeout_ms=2000)
                except RuntimeError:
                    continue

                depth_frame = frames.get_depth_frame()
                if not depth_frame:
                    continue
                detected, _, _, _, dist = detect_large_object(depth_frame)
                if not detected or dist > 1.5:
                    print("Object cleared.")
                    stop_motion()
                    roll_done = True
                    break
                roll_right()
                time.sleep(0.1)

        if roll_done:
            print("Pitching forward to continue...")
            for _ in range(30):
                pitch_forward()
                time.sleep(0.1)
            stop_motion()

            print("Switching to LOITER mode to hover...")
            vehicle.mode = VehicleMode("LOITER")

            print("Drone is now loitering.")
            while True:
                time.sleep(1)

        time.sleep(0.1)

except KeyboardInterrupt:
    print("Interrupted by user.")

finally:
    print("Landing...")
    vehicle.mode = VehicleMode("LAND")
    time.sleep(10)
    vehicle.close()
    if pipeline:
        pipeline.stop()
```

REFERENCES

1. https://www.researchgate.net/publication/381634037_Assembly_and_Programming_of_Hexacopter_Drone_for_Surveillance_and_Medical_Kit_Delivery_Purposes

2. Zhang, Wei, et al. "Mission Planner Software for Autonomous Drone Navigation." IEEE Transactions on Automation Science and Engineering 20.4 (2023): 2390–2402.

3. https://www.researchgate.net/publication/318412678_Understanding_autonomous_drone_maneuverability_for_Internet_of_Things_applications

4. https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle