Approach

A tailsitter is an aircraft design where the plane takes off and lands on its tail, often resembling a vertical takeoff and landing (VTOL) vehicle. for short takeoff and landing distances. Tailsitters typically transition from vertical to horizontal flight, using specialized control surfaces

Aircraft Design and Configuration
1. the wingspan does not exceed 120 inches
2.total weight under 3.5 lbs, including all components and payload
3.motors- 3 electric motors, with appropriate propeller sizes for vertical takeoff and horizontal flight
4.Lightweight materials (like foam or carbon fiber) to meet weight requirements
5. A 4-cell (14.8V) Lithium-Polymer battery with a maximum capacity of 3000mAh

Sensors:
Nrf receiver and transmitter,Gps neo 8m,camera module,bno055

A]Autonomous flight logic:

1.takeoff, payload delivery, capture, and return to base.

2.manage the transition between vertical and horizontal flight.

B]Payload handling:mechanisms for payload release and capture. Ensure they can operate within the time limits specified

C]Record and edit videos demonstrating the required flight maneuvers for submission

Will use open cv and will color the payload will detect will returning to base

Payload design for weight balance:

1.Distribute weight evenly across the payload to maintain a balanced CG. This helps in preventing any unwanted pitch or yaw during flight.

2.Position the payload as close to the CG of the aircraft as possible to minimize leverage effects that can alter stability.

3.Design the payload mount to be adjustable. This allows fine-tuning of the CG during assembly.

4.If needed, use small weights or adjust the internal components to shift the CG slightly forward or backward for optimal balance.

5.Use lightweight materials such as foam, balsa wood, or plastic to minimize overall weight while maintaining structural integrity.

Ensure the payload is securely attached to avoid movement during flight.

Payload Configuration

1. Design a flat base to support the payload, ensuring it fits securely within the aircraft's cargo area.
2. Implement a reliable mechanism for payload release. This could be a simple latch or a more complex servo-driven mechanism.
3. If capturing payloads, consider designs that allow easy retrieval, such as hooks or nets.

Payload Dimensions

- Length: 10 inches
- Width: 8 inches
- Height: 4 inches
- Weight: 1.5 lbs (including the delivery mechanism)

```python
import time
import numpy as np
from flight_control import flightcontroller
from sensors import BNO,GPS,CAM
import math

fc=flightcontroller()
bno=BNO()
gps=GPS()
cam=CAM()

def takeoff():
   fc.set_throttle(0.5)
   time.sleep(5)
   fc.set_throttle(0)
   print("takeoff complete,hovering vertical")

def transition_to_horizontal():
   print("transiting from vertical to horizontal flight")
   fc.set_angle(0,0,15)
   time.sleep(3)
   fc.set_throttle(0.7)

def deliver_payload():
   print("delivering payload")
   fc.set_throttle(0)
   time.sleep(1)

def return_to_base():
   print("returning to base")
   fc.navigate(gps_module())
   fc.set_throttle(0.5)
   time.sleep(3)
```

```python
    fc.landing()


def haversine(lat1, lon1, lat2, lon2):
    R=6371
    dlat = math.radians(lat2 - lat1)
    dlon = math.radians(lon2 - lon1)
    a = (math.sin(dlat / 2) ** 2 +
         math.cos(math.radians(lat1)) * math.cos(math.radians(lat2)) *
         math.sin(dlon / 2) ** 2)
    c = 2 * math.asin(math.sqrt(a))
    return R * c
def navigate_to_delivery(lat_target, lon_target):
    current_lat, current_lon = gps.get_current_location()  # Get current GPS
coordinates

    distance = haversine(current_lat, current_lon, lat_target, lon_target)

    while distance > 0.01:  # Continue until within 10 meters
        heading = calculate_heading(current_lat, current_lon, lat_target,
lon_target)
        fc.set_heading(heading)
        fc.set_throttle(0.7)  # Set throttle for forward flight
        time.sleep(1)

        current_lat, current_lon = gps.get_current_location()  # Update
current position
        distance = haversine(current_lat, current_lon, lat_target, lon_target)


    print("Reached delivery point.")
    deliver_payload()
def main():
    takeoff()
    transition_to_horizontal()
    deliver_payload()
    return_to_base()
    haversine()
    navigate_to_delivery()

if name=="main":
    main()
```