Project 8: mRo Quad Zero — "Micro-Maze Explorer"

Topics: micro-UAV, behavioral modeling, event-driven transitions, indoor autonomy

Objective

Create a state-driven control model for a palm-sized autonomous drone (mRo Quad Zero) that navigates through a confined indoor maze using simple onboard sensors, detects obstacles and motion anomalies, and reacts appropriately to sensor faults and environmental disturbances.

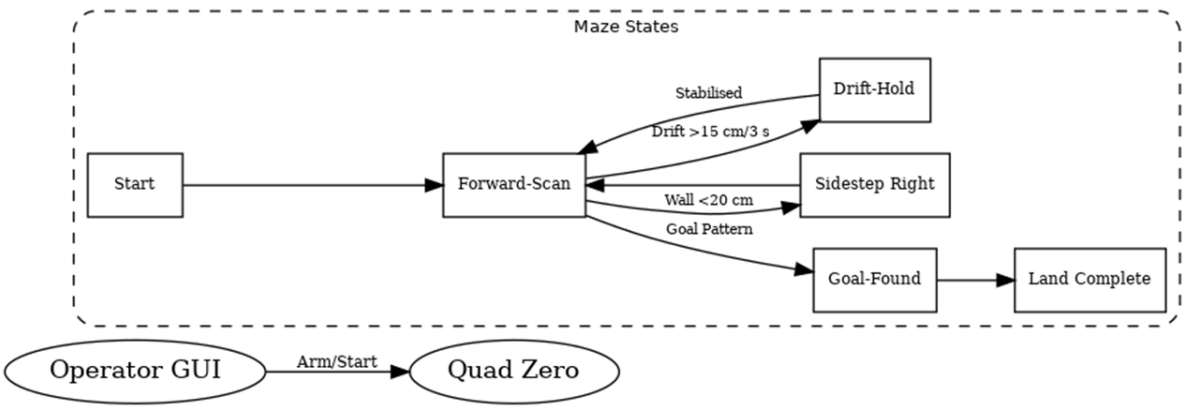
Description

The drone operates within a small-scale foam-wall maze (3×3 meters), navigating autonomously using onboard **optical-flow sensors** and **time-of-flight (ToF) distance sensors**. After take-off, it begins a forward scan until it detects a wall at a distance closer than 20 cm. At this point, it performs a rightward sidestep maneuver to bypass the obstacle, then resumes scanning.

This process is repeated iteratively until the system detects a predefined **"goal pattern"** on the floor, such as a checkerboard or QR-like signal, which indicates mission success. At that point, the drone enters a landing sequence.

The behavior is implemented as a **Finite State Machine (FSM)**, including states for Start, Forward-Scan, Sidestep Right, Goal-Found, and Land Complete. The FSM transitions are triggered by sensor thresholds (e.g., wall proximity) or visual detections.

An **abnormal condition** is defined as **drift greater than 15 cm over 3 seconds** or a **ToF sensor timeout**, both of which indicate potential sensor error or instability. Additionally, fault scenarios such as a sudden spotlight glare (which disrupts optical-flow sensing) require the system to enter a special Drift-Hold state. Here, the drone maintains hover using minimal feedback until stability is restored.



Methods and Tools

- mRo Quad Zero (or equivalent micro-drone) with onboard ToF and optical-flow sensors
- Ardupilot (ChibiOS) with custom parameters for state logic
- FSM control via Python using DroneKit-Python interface
- Simple Tkinter GUI for live state readout and alerts
- Logging of drift metrics, sensor status, and transitions

Expected Outcome

- Successful autonomous navigation through a simple indoor maze
- Robust obstacle avoidance using ToF-based sidestepping
- Drift detection and stabilization behavior under optical disruption
- Event logs with full state transitions and sensor readings
- Brief analysis of reaction time and recovery stability in fault scenarios

---

Project Phases and Workload Distribution

First phase – Industrial IoT course (logic design and sensor emulation):
The FSM and control logic will be developed in simulation using DroneKit-Python and mocked sensor input. Basic obstacle detection and sidestep patterns will be tested, and optical flow anomalies will be emulated by noise injection.

Deliverables:

- Python FSM for Ardupilot integration
- Simulated maze layout and obstacle triggers
- Code walkthrough and functional test video

Second phase – Internship (real-world indoor deployment):
The Quad Zero will be deployed in a physical maze environment with foam walls and printed floor markers. Sensors will be tuned for the environment. The system's resilience to drift and fault scenarios will be tested live.

Deliverables:

- Fully configured micro-UAV with mission parameters
- Test logs of wall detection and sidestepping accuracy

- Live test videos showing goal detection and controlled landing
- Initial deployment report

Third phase – Thesis project (sensor behavior analysis and robustness):
This phase will include a deeper analysis of drift patterns under different light and floor conditions, fault recovery timing, and system performance. Optional extensions include adding multiple path strategies, goal re-identification, or edge-based supervision.

Deliverables:

- Complete thesis manuscript with experiment analysis
- Performance graphs (e.g., wall detection latency, drift variance)
- Full dataset of mission logs
- Optionally: edge GUI or dashboard integration