Alma Mater Studiorum · University of Bologna

Degree Programme in Engineering and Computer Science
A.A. 2025/26

Smart Drone Hangar

Grazia Bochdanovits de Kavna
matr. 0001117082

Alessandro Rebosio
matr. 0001130557

Contents

Chapter 1

Analysis

1.1 Description and requirements

Initially the system starts with the hangar door HD closed; the DRONE INSIDE state holds, L1 on, L2 and L3 off, and the LCD shows DRONE INSIDE.

Take-off: the drone requests opening via DRU. On command the HD opens, the LCD shows TAKE OFF, and the system waits for exit. Exit is detected by the DDD: when distance > D1 for more than T1 the drone is assumed out, the HD closes and the LCD shows DRONE OUT.

Landing: the drone requests opening via DRU. If DPD detects presence, the HD opens and the LCD shows LANDING. When DDD measures distance < D2 for more than T2 the drone is landed, the HD closes and the LCD shows DRONE INSIDE.

During take-off/landing L2 blinks (0.5 s period); otherwise it is off.

Temperature monitoring runs whenever the drone is inside (rest, take-off, landing). If temperature $\geq$ Temp1 for more than T3 the system enters pre-alarm: new take-offs/landings are suspended until return to normal operation (in-progress operations may complete). If temperature $\geq$ Temp2 (> Temp1) for more than T4 the HD is closed (if open), L3 turns on and the LCD shows ALARM. If the drone is outside, an ALARM message is sent via DRU. All operations stay suspended until the RESET button is pressed; pressing it returns the system to normal operation.

Parameters D1, D2, T1, T2, T3, T4, Temp1, Temp2 are left configurable for testing.

The DRU GUI must allow:

- sending take-off/landing commands (simulate the drone);
- displaying drone state (rest, taking off, operating, landing);
- displaying hangar state (normal, ALARM);
- (during landing) showing current distance to ground.

1.2 Wiring

Below is a concise list of the essential hardware components required to build and test the Smart Drone Hangar. Quantities, components and brief notes on their function are provided to assist with procurement and wiring.

Qty	Component	Notes
1	Microcontroller	Handles sensors, actuators, and serial communication
1	Servo motor	Used as the hangar door actuator
1	Distance sensor (DDD)	Measures the drone's distance inside the hangar
1	Presence sensor (DPD)	Detects the drone approaching the hangar
1	Temperature sensor	Monitors internal hangar temperature
3	LEDs (L1, L2, L3)	System status indicators (normal, activity, alarm)
3	Resistors 220 Ω	Current-limiting resistors for LEDs L1, L2, L3
1	LCD display	Displays system messages
1	RESET button	Used to clear alarms and restore normal operation
1	Resistor 10 kΩ	Pull-up / pull-down resistor for the RESET button

Table 1.1: Essential hardware list

Figure 1.1 shows the wiring diagram of the hangar control unit, highlighting the main connections between the microcontroller, sensors, actuators, and indicators.

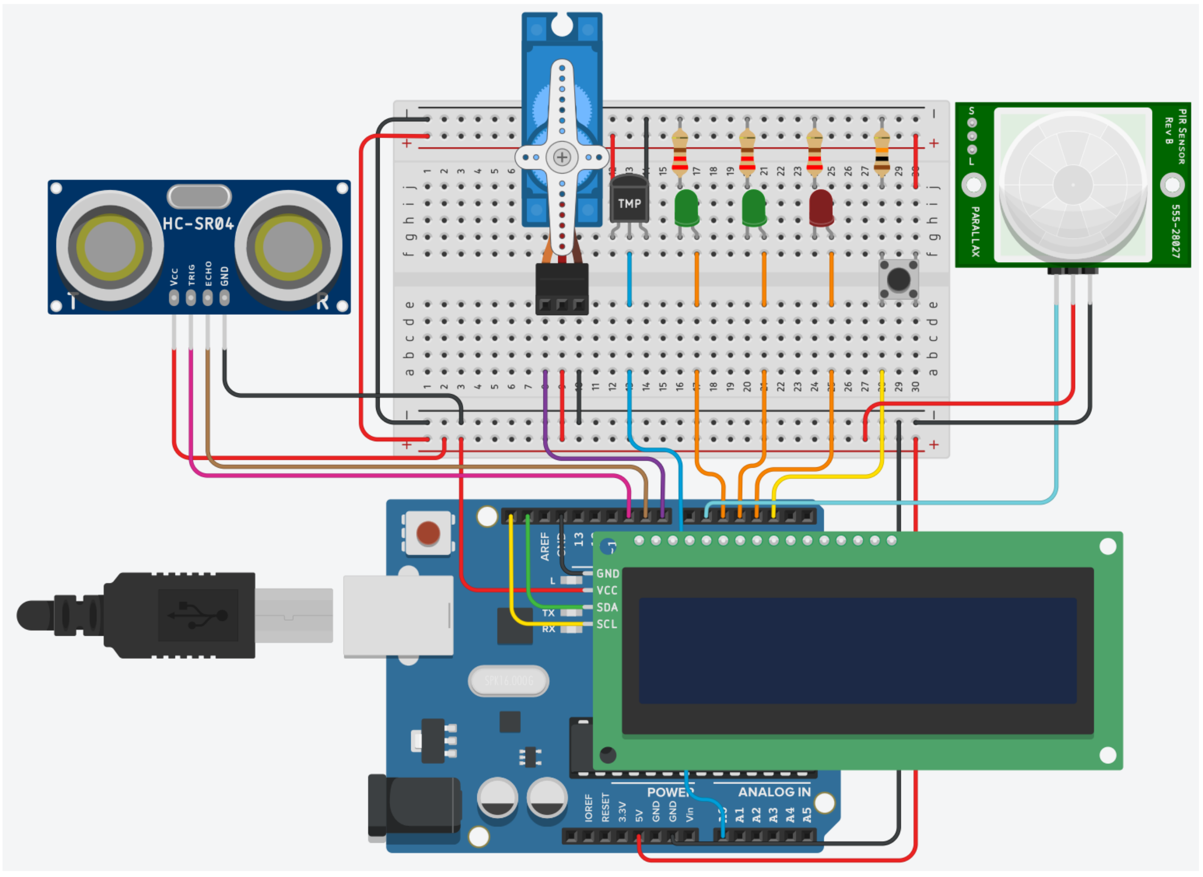


Figure 1.1: Wiring diagram of the hangar control unit.

Chapter 2

Architecture

The implementation is organized into multiple tasks, each responsible for a specific function of the project. A simple scheduler was implemented to run these tasks concurrently. The main tasks are described below.

2.1 System Task

The system has three states: `NORMAL`, `PREALARM` and `ALARM`.

In `NORMAL` take-offs and landings are allowed. If temperature stays at or above `TEMP1` for more than `T1`, the system goes to `PREALARM`; the timer resets when temperature drops below `TEMP1`.

In `PREALARM` new flights are suspended but ongoing operations may complete. If temperature stays at or above `TEMP2` for more than `T2`, the system enters `ALARM`. If temperature falls below `TEMP1` it returns to `NORMAL`.

In `ALARM` the door is closed and the alarm is activated; if the drone is outside an alarm message is sent via `DRU`. The system stays in `ALARM` until `RESET` is pressed, which returns it to `NORMAL`.

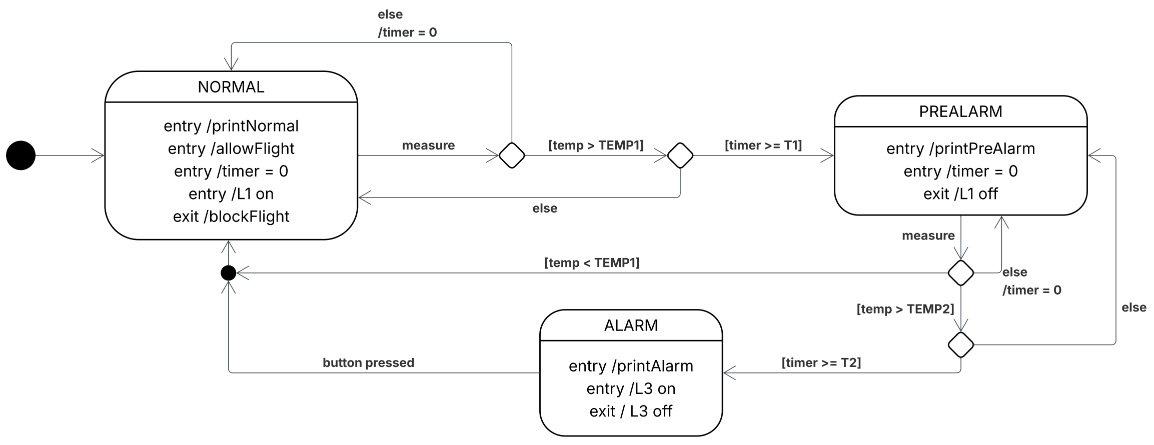


Figure 2.1: System Task state diagram.

Note: when the entry action `/AllowFlight` is executed, a boolean flag is set to `true`; when the exit action `/BlockFlight` is executed, the flag is set to `false`.

2.2 Flight Task

The FlightTask manages drone take-off and landing operations through three main states. Initially in the IDLE state, the task awaits user commands.

Upon receiving a take-off command, it verifies the drone's distance and, if below threshold D1, transitions to the OPERATING state to execute the departure sequence.

For landing operations, it moves to the WAITING state, where it awaits drone detection within a predefined timeout T5.

In OPERATING, the hangar door opens and LED L2 starts blinking; upon operation completion, the door closes, blinking stops, and the system returns to IDLE.

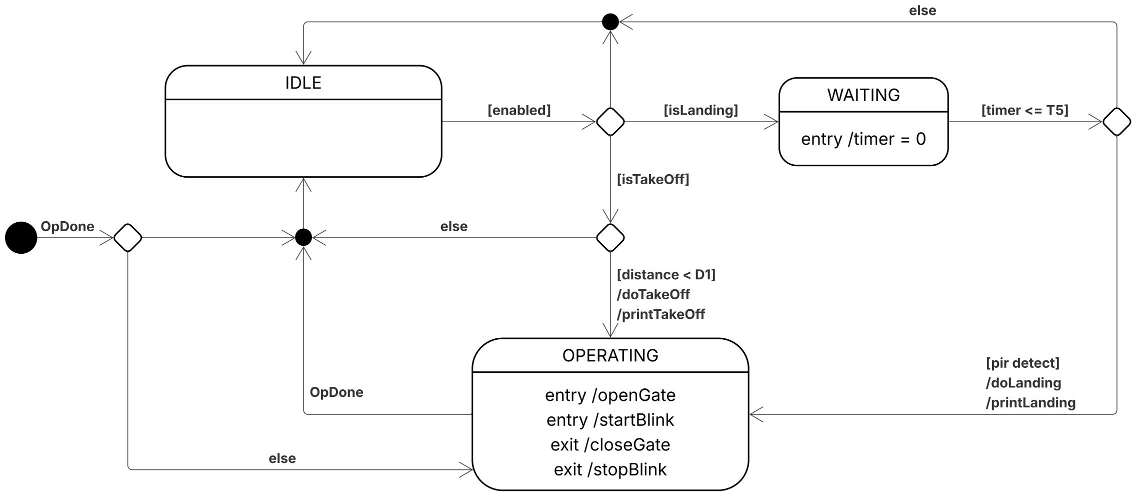


Figure 2.2: Flight Task state diagram.

Note: When the Flight Task enters the OPERATING state, it triggers the /startBlink and /openGate actions to activate the corresponding tasks. Upon completion of the flight operation, it executes /stopBlink and /closeGate to stop these tasks.

2.2.1 Flight Task Operation

Based on sensor checks and conditions, the system then activates either /doLanding or /doTakeoff to initiate the specialized landing or takeoff sequences. These dedicated tasks handle the complete operational workflow and, once finished, signal the Flight Task to transition from OPERATING back to IDLE state.

2.3 Blink Task

It implements two states: `ON` and `OFF`. On each invocation the task toggles its state and updates the LED output accordingly. A guard condition prevents the transition from `OFF` to `ON` when blinking is disabled, so the LED remains off when blinking is not allowed.

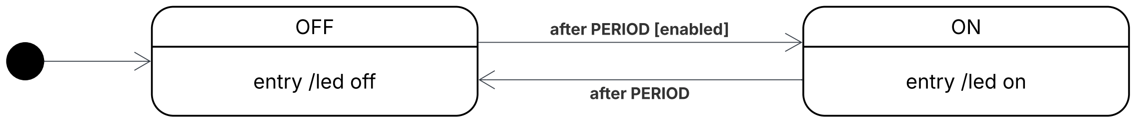


Figure 2.3: Blinking Task state diagram.

2.4 Gate Task

The `Gate Task` controls the hangar door using four states: `CLOSE`, `OPENING`, `OPEN`, and `CLOSING`. At startup the gate is in `CLOSE`. When enabled it goes to `OPENING` until fully open, then to `OPEN`. If enabled becomes false it moves to `CLOSING` until `CLOSE`.

If an opposite command arrives while moving, the task reverses direction immediately.

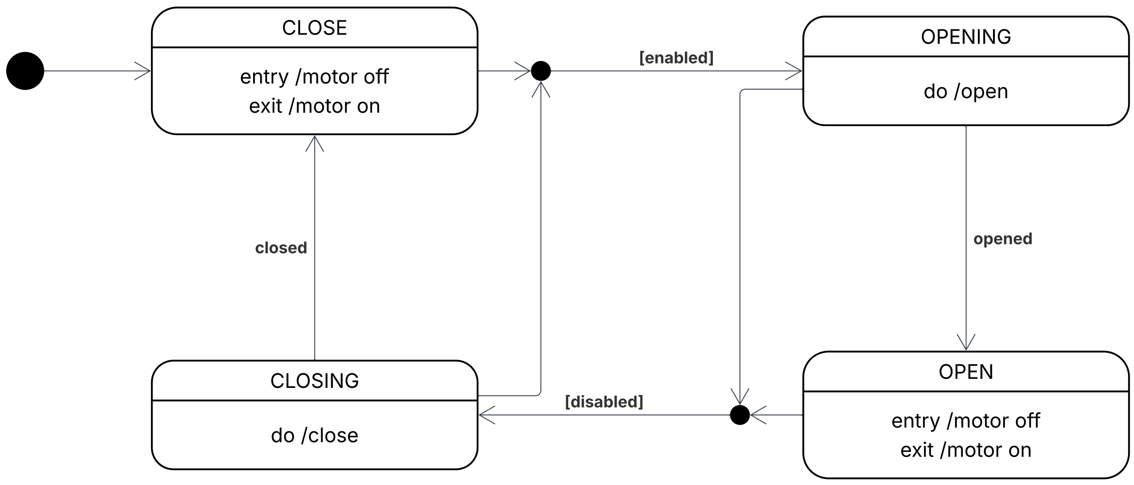


Figure 2.4: Gate Task state diagram.

2.5 Observer Task

The `Observer Task` is responsible for periodically monitoring system conditions through the evaluation of a predicate function. It operates in a single state, `OBSERVER`, where at each activation the associated predicate is checked. If the predicate evaluates to true, the corresponding function or callback is executed, allowing the system to react to specific events or conditions in real time. This task provides a lightweight mechanism for asynchronous observation and event detection without interfering with the main control logic.

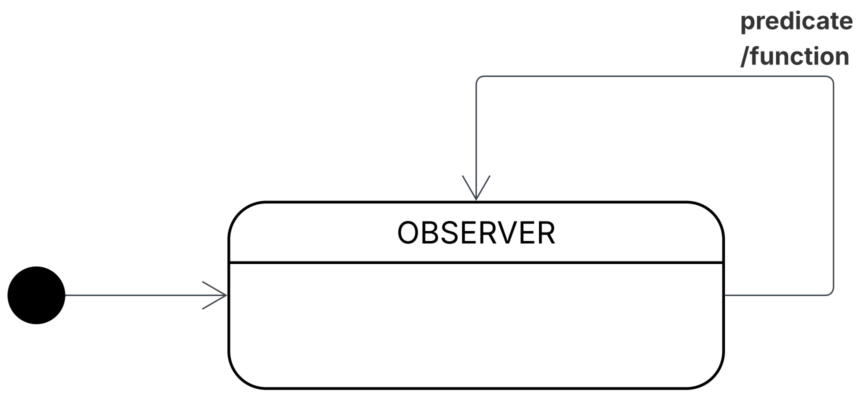


Figure 2.5: Observer Task state diagram.

Note: The Observer Task acts as the system's central listener. It periodically reads inputs and messages (for example from the DRU/serial) and evaluates the associated predicates without consuming the messages inside specific functional tasks. When an event or command is detected, the Observer delegates processing and the relevant values to the appropriate tasks (System, Flight, Gate, etc.), keeping each task lightweight and focused on its own logic.

This design allows multiple SystemTask instances, each with its own Context. The Observer receives serial messages and forwards each one to the correct Context, so a single Arduino can manage multiple hangars. This simplifies message routing and makes the system easier to extend and maintain.

Chapter 3

Arduino Workflow

Appendix A

User Interface

The user interface of the application is organized into three main panels, each serving a distinct purpose within the system.

A.1 Connection Panel

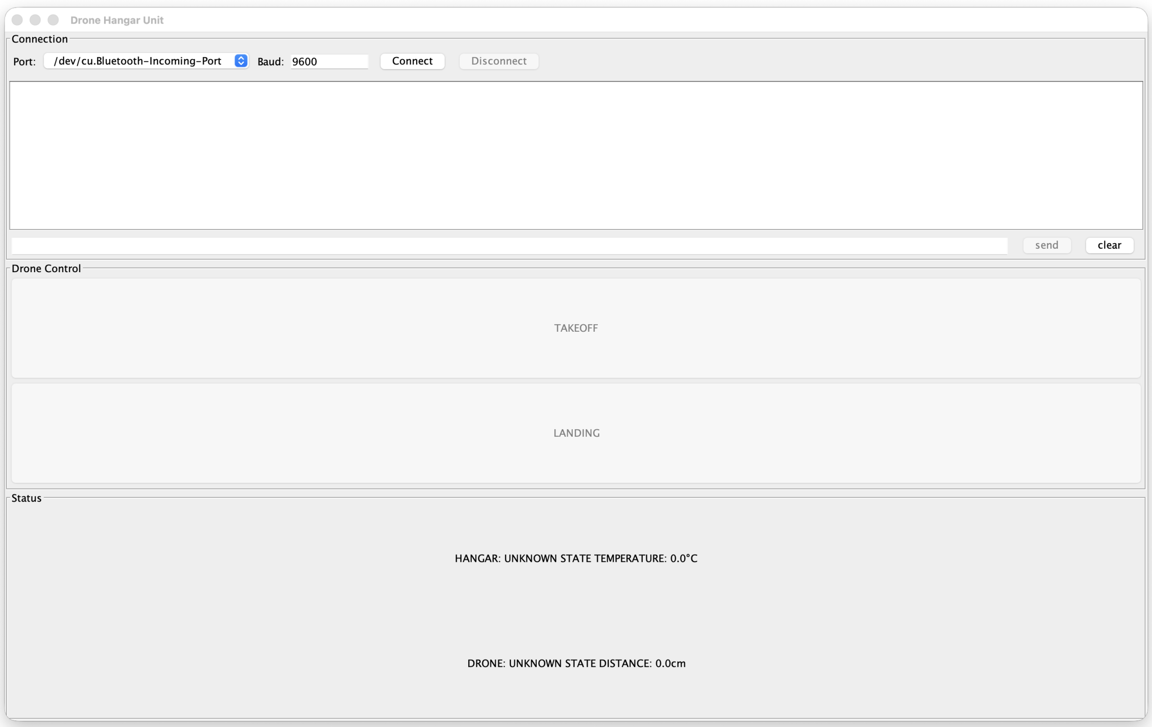
The upper part of the view is dedicated to managing the connection with the serial port. The user can select the desired port, configure the communication parameters, and establish or terminate the connection with the device.

A.2 Control Panel

The central part of the interface provides simple controls for operating the drone. It includes two buttons — *Takeoff* and *Landing* — which send the corresponding commands to the controller. These buttons are automatically enabled or disabled based on the current connection status and system state.

A.3 Status Panel

The lower part of the view displays real-time information about the *Hangar* and *Drone* states, allowing continuous monitoring of the overall system status.



HANGAR: UNKNOWN STATE TEMPERATURE: 0.0°C

DRONE: UNKNOWN STATE DISTANCE: 0.0cm

Figure A.1: App view

Appendix B

User Guide

B.1 Cloning the repository

Clone the project from GitHub and change to the project directory:

```
> git clone https://github.com/alessandrorebosio/ESIOT25.git
> cd ESIOT25/assignment-02
```

B.2 Connecting and starting the application

Connect the Arduino to your computer, identify the serial port, then start the application with:

```
> java -jar drone-hangar-unit-all.jar
```

Alternatively you can start the project with:

```
> ./gradlew run
```