Alma Mater Studiorum · University of Bologna

Bachelor's Degree in Computer Engineering and Science
A.A. 2025/26

Smart Drone Hangar

Grazia Bochdanovits de Kavna
matr. 0001117082

Alessandro Rebosio
matr. 0001130557

Contents

Chapter 1

Analysis

This chapter analyses the functional behaviour and hardware structure of the Smart Drone Hangar, outlining its operating sequences, safety logic, and implementation requirements. The objective is to define how the system reacts to flight commands and environmental conditions, and how each hardware element contributes to its autonomous operation.

1.1 Description and Requirements

At startup, the system initializes all components and reads the sensors to determine the current position of the drone. Depending on the measurements from the distance sensor (DDD) and the presence sensor (DPD), the Context is set to either `DRONE INSIDE` or `DRONE OUTSIDE`. Once the state is established, LED L1 turns on if the drone is detected inside, while LEDs L2 and L3 remain off, and the LCD displays the corresponding status message.

Take-off Sequence. When a take-off command is received from the Drone Remote Unit (DRU), the hangar gate opens and the LCD displays "`TAKE OFF`". The system then waits for the drone to exit the hangar. Exit is confirmed when the distance measured by the DDD exceeds D_1 for longer than T_1. At that moment, the gate closes automatically and the LCD updates to "`DRONE OUTSIDE`".

Landing Sequence. When the drone requests landing through the DRU, the presence sensor (DPD) detects its approach. The hangar gate opens and landing is confirmed when the DDD measures a distance below D_2 for longer than T_2. Once the drone is safely inside, the gate closes and the display returns to "`DRONE INSIDE`".

Indicators and Lighting. During take-off and landing operations, LED L2 blinks with a 0.5 s period to indicate activity. In all other states it remains off. LED L1 indicates the presence of the drone inside the hangar, while LED L3 signals an alarm condition.

Temperature Monitoring and Safety Conditions. Temperature control is active whenever the drone is inside the hangar. If the internal temperature exceeds `Temp1` for more than `T3`, the system enters a **pre-alarm** condition: new take-offs or landings are suspended, but ongoing operations are allowed to finish. If the temperature rises above `Temp2` (`Temp2 > Temp1`) for longer than `T4`, the system switches to **alarm** mode. In this state, the gate closes (if open), LED L3 turns on, and the LCD displays "`ALARM`". If the drone is outside during an alarm, the same message is sent through the DRU. All operations remain suspended until the **RESET** button is pressed, restoring normal operation.

Configurable Parameters and GUI Requirements. The parameters D_1, D_2, T_1, T_2, T_3, T_4, `Temp1`, and `Temp2` are configurable for testing and calibration. The Drone Remote Unit (DRU) graphical interface allows the user to send take-off and landing commands, monitor the current drone and hangar states (*rest*, *taking off*, *operating*, *landing*), and view real-time data such as temperature and distance from the ground.

1.2 Summary

The Smart Drone Hangar operates as an autonomous system that controls flight access, monitors temperature and safety conditions, and coordinates all flight operations. At startup, the sensors determine whether the drone is inside or outside the hangar, ensuring that the system state always matches the physical configuration. During normal operation, take-off and landing are managed automatically: the gate opens, LED L2 blinks to indicate activity, and sensor readings confirm the drone's movement before closing the gate. Temperature thresholds ensure that flights occur only under safe conditions, while the alarm mode provides full protection in case of overheating. All operational parameters can be configured and monitored through the DRU interface, resulting in a reliable, modular, and self-contained control system.

1.3 Wiring

Table 1.1 lists the essential hardware components required to build and test the Smart Drone Hangar. Quantities, component types, and their function are summarized for procurement and wiring reference.

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–L3
1	LCD display	Displays system messages
1	RESET button	Clears alarms and restores 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.

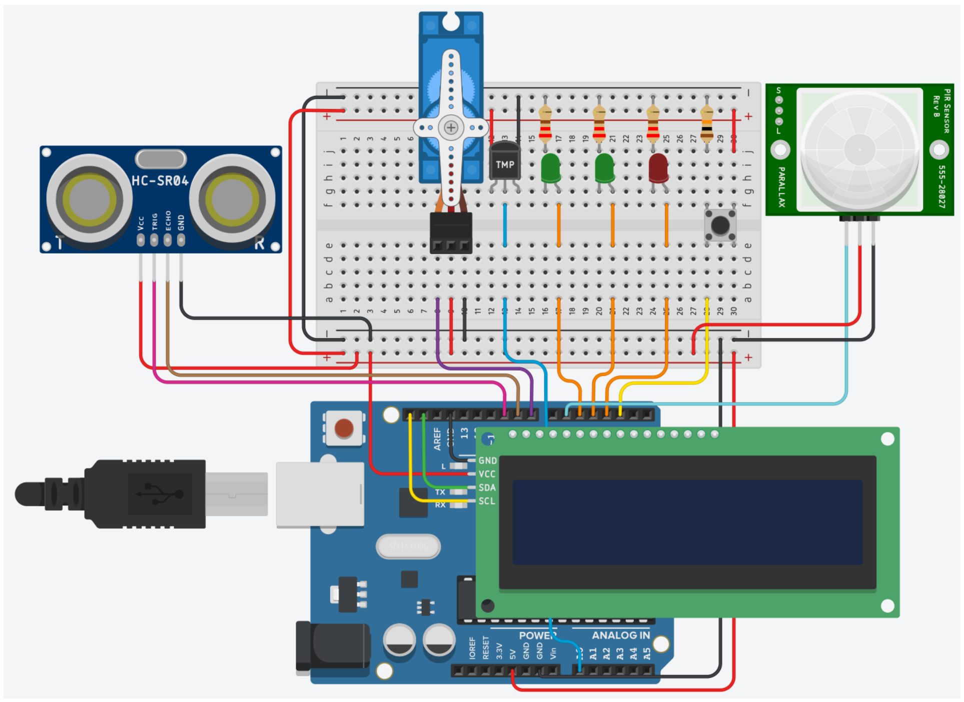


Figure 1.1: Wiring diagram of the hangar control unit.

Chapter 2

Architecture

The system architecture is organized around a set of concurrent tasks, each responsible for a specific function of the Smart Drone Hangar. A lightweight cooperative scheduler periodically activates these tasks in a cyclic order.

All tasks interact through a shared data structure called the *Context*. It acts as a shared memory, enabling coordination between components via Boolean flags and message codes, ensuring modularity and preventing direct coupling between tasks.

2.1 System Task

This task manages the global operating mode of the hangar, switching between three states: `NORMAL`, `PREALARM`, and `ALARM`.

In `NORMAL`, all flight operations are allowed. If the temperature exceeds `TEMP1` for longer than `T1`, the system enters `PREALARM`, suspending new take-offs and landings but allowing ongoing ones to complete. If the temperature remains above `TEMP2` for more than `T2`, the system transitions to `ALARM`: the hangar door is closed, the alarm LED (L3) is activated, and all operations are halted. The system remains in this state until the **RESET** button is pressed, which returns it to `NORMAL`. If the drone is outside during an alarm, an alert message is sent via the DRU.

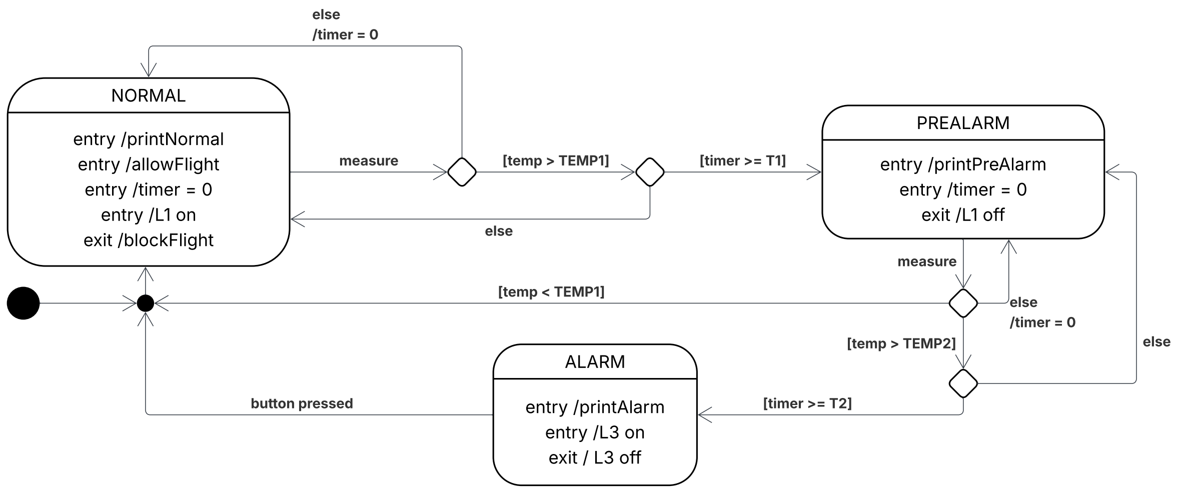


Figure 2.1: System Task state diagram.

2.2 Flight Task

The **FlightTask** manages both take-off and landing operations, acting as the intermediate layer between the system logic and the actuator control. It operates through three main states: `IDLE`, `WAITING`, and `CHECKING`.

At system startup, the task initializes directly in the `CHECKING` state, where it evaluates sensor readings to determine whether a flight operation is in progress or complete. This mechanism ensures that the software remains synchronized with the physical state of the hangar even after a reset or power interruption.

During normal operation, when a take-off message is received from the Drone Remote Unit (DRU) and the drone is detected inside the hangar ($distance < D_1$), the task transitions from `IDLE` to `CHECKING`. In this state, the gate is opened, LED L2 starts blinking, and the system monitors the distance to confirm that the drone has taken off.

If a landing message is received instead, the task enters the `WAITING` state. Here, a timer is started, and the system waits for the presence sensor (PIR) to detect the approaching drone. If the drone is detected before the timeout period T_5, the task transitions to `CHECKING` to monitor the final phase of the landing. If the timer expires without detection, the system returns to `IDLE`.

In the `CHECKING` state, the FlightTask supervises both take-off and landing progress using the distance sensor. A take-off is confirmed when the measured distance remains greater than D_1 for longer than T_3, while a landing is confirmed when the distance stays below D_2 for at least T_4. Upon completion, the gate closes automatically, LED L2 stops blinking, and the system returns to the `IDLE` state.

This design unifies both flight operations within a single task, allowing automatic state recovery at startup and consistent synchronization between software logic, sensor data, and the physical hangar configuration.

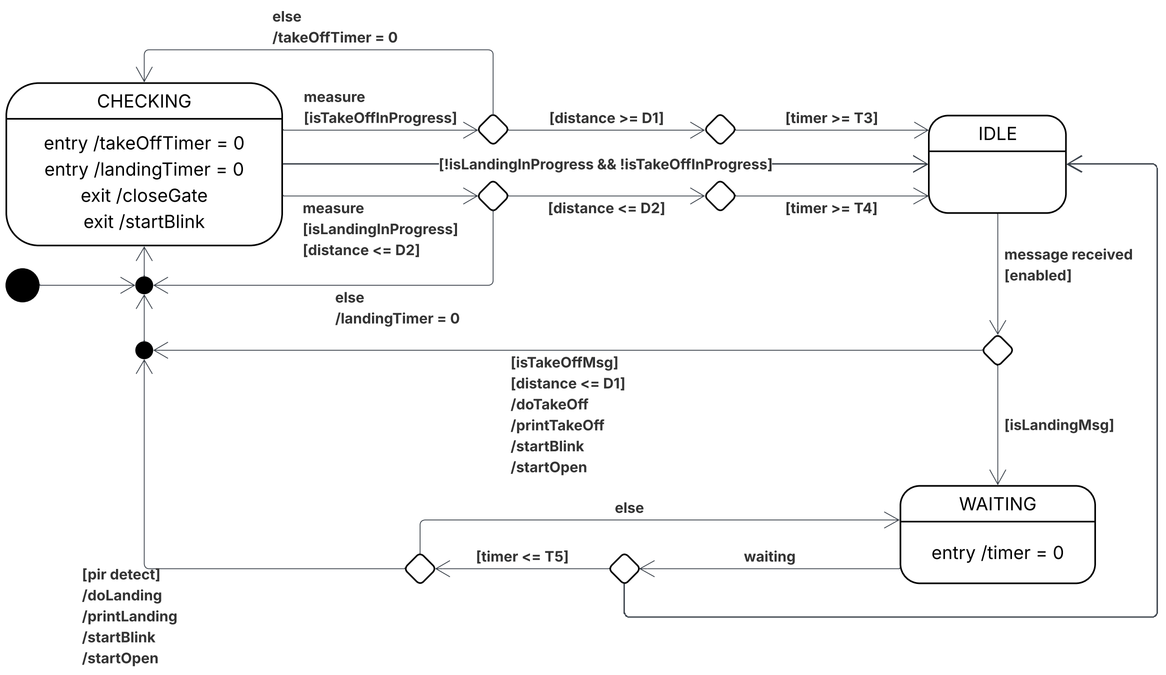


Figure 2.2: Flight Task state diagram.

2.3 Blink Task

The **BlinkTask** handles LED L2 blinking, providing visual feedback during active flight phases. It operates in two alternating states, `ON` and `OFF`. At each activation, the task toggles the LED state; however, a guard condition prevents the transition from `OFF` to `ON` when blinking is disabled, keeping the LED off during idle periods.

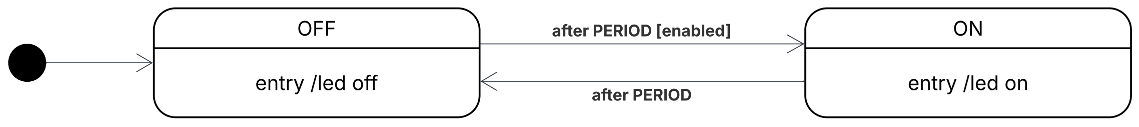


Figure 2.3: Blink Task state diagram.

2.4 Gate Task

The **GateTask** controls the hangar door using four states: `CLOSE`, `OPENING`, `OPEN`, and `CLOSING`. At startup, the gate is in `CLOSE`. When enabled, it transitions to `OPENING` until the door is fully open, then to `OPEN`. If the enable signal becomes false, it moves to `CLOSING` until the door is fully closed. If a new command arrives while the door is moving, the task immediately reverses its direction.

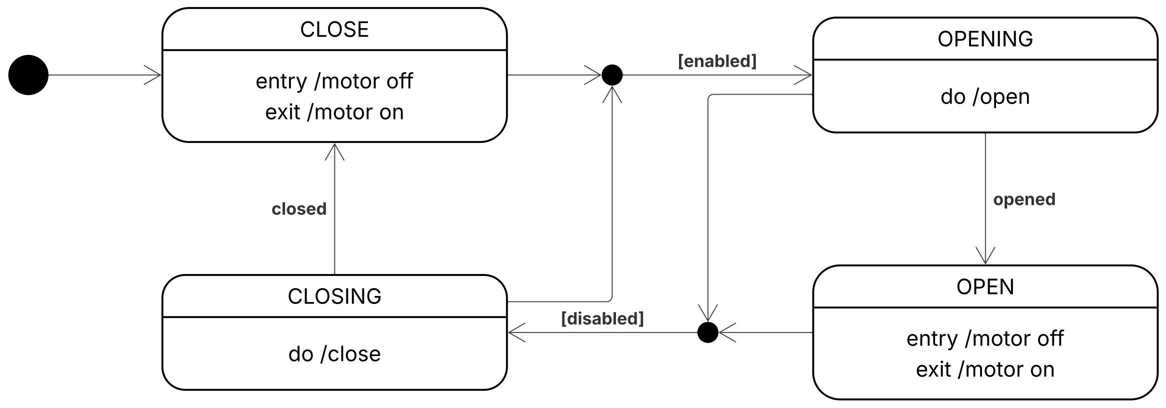


Figure 2.4: Gate Task state diagram.

2.5 Observer Task

The **ObserverTask** periodically monitors sensors and communication channels by evaluating a predicate function at each activation. When the predicate evaluates to true, the associated callback is executed, allowing the system to react immediately to specific events such as sensor updates or incoming commands. This provides a lightweight mechanism for asynchronous observation without interfering with the main control logic.

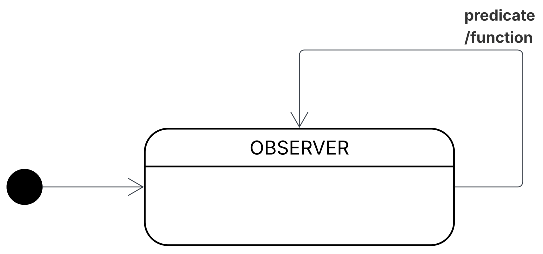


Figure 2.5: Observer Task state diagram.

The Observer acts as a central listener: it periodically reads inputs and messages (e.g., from the DRU via serial communication) and dispatches relevant information to the appropriate tasks — System, Flight, Gate, etc. This approach keeps each functional task lightweight and focused on its own logic. Moreover, the architecture supports multiple SystemTask instances, each with its own Context, enabling a single microcontroller to manage multiple hangars in parallel and simplifying message routing and scalability.

Chapter 3

Arduino Workflow

3.1 Overview

This chapter describes the dynamic behaviour of the Smart Drone Hangar once deployed on the Arduino platform. Each system function is implemented as an independent task, periodically activated by a cooperative scheduler. Although tasks are executed sequentially, their high activation rate makes the system respond as if all processes were concurrent, continuously reacting to inputs and updating outputs in real time.

All components share a common data structure, the **Context**, which represents the global state of the hangar. Through this structure, tasks communicate indirectly by setting and reading Boolean flags that describe the hangar's condition, the drone's position, and the current operating mode.

3.2 System Dynamics

At startup, the system initializes all hardware components and performs a sensor check to determine the drone's position. Depending on the readings from the distance (DDD) and presence (DPD) sensors, the Context is initialized to either `DRONE INSIDE` or `DRONE OUTSIDE`. This ensures that the software state is immediately aligned with the physical situation, even after a reset or power interruption.

The scheduler then maintains continuous control by cyclically activating all tasks to supervise sensors, actuators, and communication interfaces. The **SystemTask** governs the overall behaviour of the hangar. It monitors temperature readings and selects the appropriate operating mode. Under normal conditions, flight operations are permitted. If the temperature exceeds the first threshold (`Temp1`) for longer than `T1`, the system enters a pre-alarm state, suspending new take-offs or landings while allowing any ongoing operation to finish. If the temperature rises above the higher threshold (`Temp2`) for more than `T2`, the system switches to alarm mode: the gate closes, the alarm indicator (LED L3) turns on, and all operations remain halted until the **RESET** button is pressed. This mechanism ensures that all flight activity takes place only under safe environmental conditions.

3.3 Coordination of Operations

The **FlightTask** coordinates both take-off and landing operations, acting as the intermediate layer between the system logic and the actuator control. It operates through three main states: `IDLE`, `WAITING`, and `CHECKING`. At system startup, the task intentionally begins in the `CHECKING` state, where it reads sensor data to determine whether the drone is inside or outside the hangar and restores consistency between the software logic and the physical configuration.

During normal operation, the `IDLE` state represents a standby condition in which the system waits for commands from the Drone Remote Unit (DRU). When a take-off command is received and the measured distance is below the threshold D_1, the task transitions directly to `CHECKING`, activates LED L2 blinking, displays the "`TAKE OFF`" message, and opens the gate. If a landing command is received instead, the task moves to the `WAITING` state, where a timer is started to wait for the presence sensor (PIR) to detect the approaching drone. If the drone is detected before the timeout period T_5, the task transitions to `CHECKING`, starts LED blinking, displays "`LANDING`", and opens the gate. If the timeout expires without detection, the system returns to `IDLE`.

In the `CHECKING` state, the FlightTask monitors distance measurements to verify the progress of the current operation. A take-off is completed when the measured distance remains greater than D_1 for longer than T_3, while a landing is confirmed when the distance stays below D_2 for at least T_4. Once the operation is complete, the gate closes automatically, LED L2 stops blinking, and the task returns to the `IDLE` state.

3.4 Supporting Tasks and Context Interaction

Several supporting tasks ensure system responsiveness and modularity. The **GateTask** drives the servo motor that opens or closes the hangar gate, while the **BlinkTask** controls LEDs to indicate activity during flight operations. The **ObserverTask** continuously reads sensor data and serial messages, updating the Context with temperature, distance, and command information.

All tasks interact exclusively through the Context, which acts as shared memory and a coordination channel. For example, the SystemTask may block new flights by setting a flag interpreted by the FlightTask, while the ObserverTask records flight commands that will be processed in the next scheduler cycle.

3.5 Global Workflow

The overall behaviour results from the cooperation of all tasks under the scheduler's control. The ObserverTask updates the Context with real-time sensor data, the SystemTask enforces safety conditions, and the FlightTask coordinates flight sequences by commanding the Gate and Blink tasks according to the current state. When a sequence is completed, the system automatically returns to a safe standby mode, ready for the next command. Through this cooperative and event-driven design, the Smart Drone Hangar achieves robust, modular, and deterministic behaviour, maintaining continuous operation even after temporary faults or power interruptions.

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

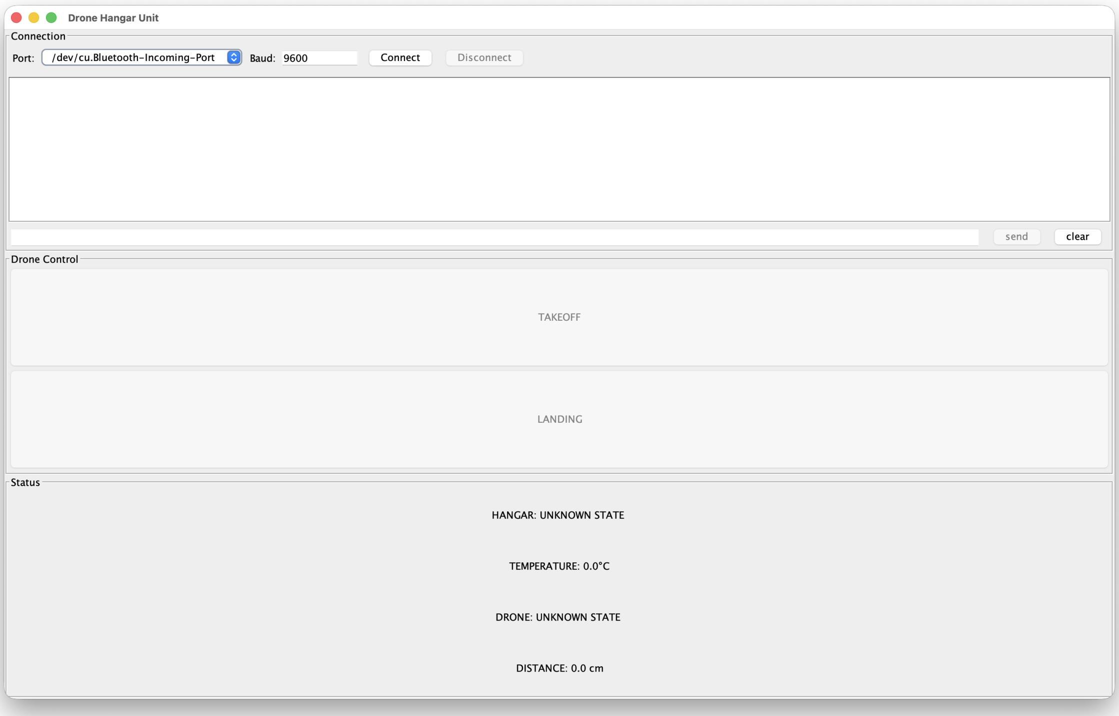
The upper part of the view is dedicated to managing the connection with the serial port. The user can select the desired port, configure the communication parameters, and establish or terminate the connection with the device.

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: 0.0 cm

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 the computer and upload (flash) the firmware to the board using the Arduino IDE or your preferred upload tool. Identify the serial port used by the board and then start the application with:

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

Alternatively you can start the project with:

```
> ./gradlew run
```