Sapienza Università di Roma

Cybersecurity

Internet Of Things

LABORATORY REPORT

PROJECT B

Student Name	**Student ID**
Alessio Civica	1916744
Federico Detomaso	1903906

1 Brainstorming and Implementation

For Project B, we initially thought about how to implement drones and what behavior to associate with them. Our goal was to minimize time and data exchange, but always ensure the most up-to-date data and continuous communication between components.
Before developing the optimal algorithm for sensor detection, we first focused on understanding all components of the simulation and how the classes interact.
The two key aspects we focused on are:

- **Range problem**: balloons were unable to reach the sensor values;
- **Minimize drone paths**: initially the drones had the same targets increasing the risk of collision and the time to send data to the balloons or base station.

The first change we implemented involved the positioning of the balloons and drones. We decided to spawn them in a circular formation centered around the base station. This strategy minimizes the time required for sensor detection, as the balloons start in the air and make limited movements, while the drones begin on the ground.
To avoid collisions, we thought of flying the drones at different heights from each other and also from the balloons.
Additionally, we created a distance matrix that records the distances from each drone j to every sensor i. This matrix helps calculate the optimal paths for the drones, aiming to minimize their travel distances. To assign sensors to drones after matrix calculation, the number of sensors is divided by the number of drones and rounded down (For example: 10 sensors / 3 drones = 3). In addition, the sensors are assigned according to the distance to the drone. After assigning sensors, if more sensors remain to be assigned they are added to the path of the drones.
We also adjusted the detection mechanism so that a drone only needs to be near a sensor, rather than directly above it. Specifically, when the distance between a drone and a sensor falls below a defined sensor range, the drone successfully detects the sensor value, which meets our requirements.
To have the sensor data always updated once a drone finishes its path it starts it again.

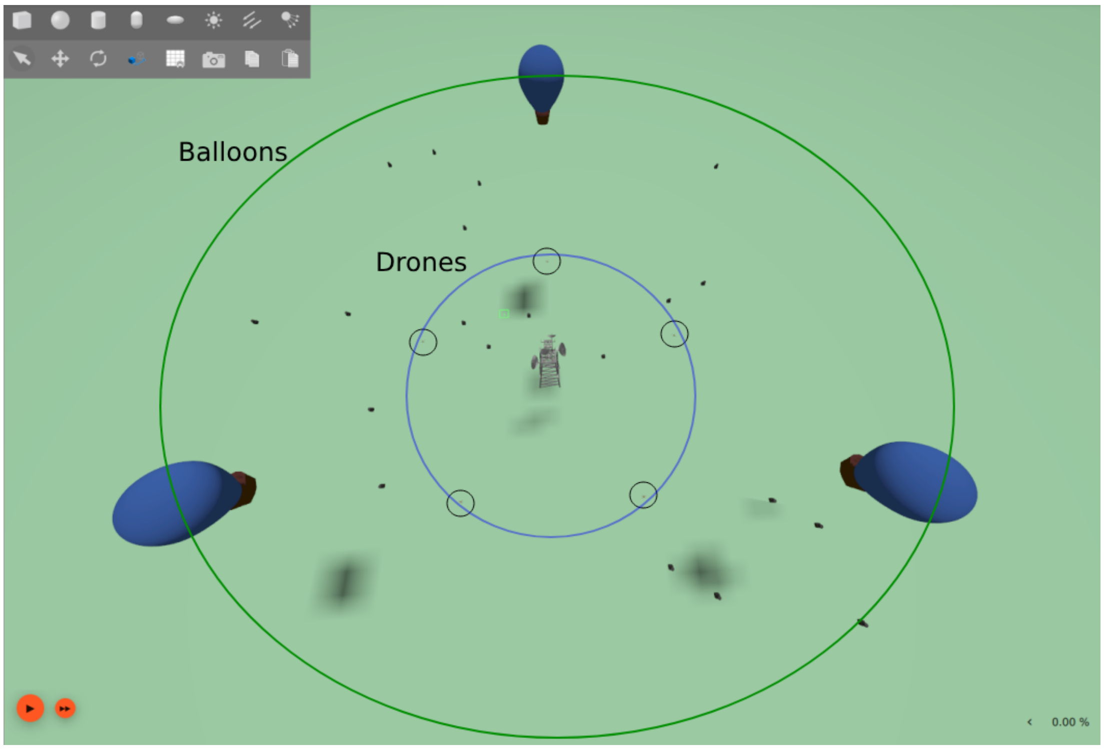


Figure 1: Initial position

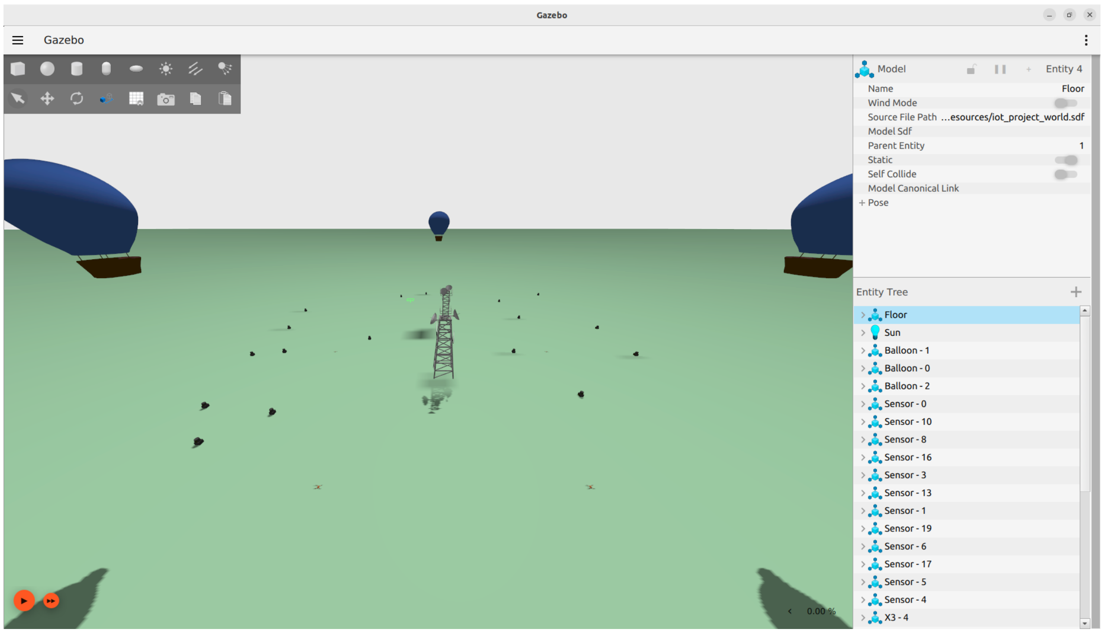
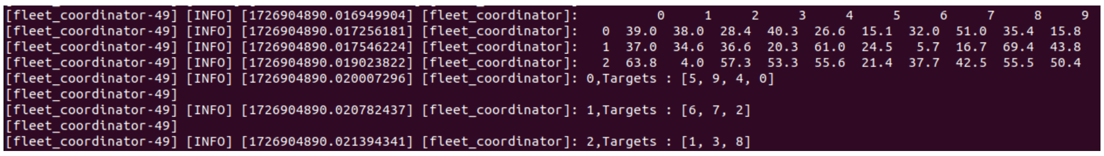
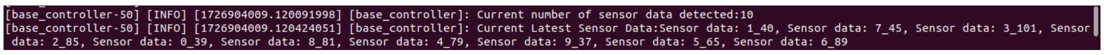


Figure 2: Initial position



```
[fleet_coordinator-49] [INFO] [1726904890.016949904] [fleet_coordinator]:        0     1     2     3     4     5     6     7     8     9
[fleet_coordinator-49] [INFO] [1726904890.017256181] [fleet_coordinator]:   0  39.0  38.0  28.4  40.3  26.6  15.1  32.0  51.0  35.4  15.8
[fleet_coordinator-49] [INFO] [1726904890.017546224] [fleet_coordinator]:   1  37.0  34.6  36.6  20.3  61.0  24.5   5.7  16.7  69.4  43.8
[fleet_coordinator-49] [INFO] [1726904890.019023822] [fleet_coordinator]:   2  63.8   4.0  57.3  53.3  55.6  21.4  37.7  42.5  55.5  50.4
[fleet_coordinator-49] [INFO] [1726904890.020007296] [fleet_coordinator]: 0,Targets : [5, 9, 4, 0]
[fleet_coordinator-49]
[fleet_coordinator-49] [INFO] [1726904890.020782437] [fleet_coordinator]: 1,Targets : [6, 7, 2]
[fleet_coordinator-49]
[fleet_coordinator-49] [INFO] [1726904890.021394341] [fleet_coordinator]: 2,Targets : [1, 3, 8]
```

Figure 3: Matrix used to optimize drone paths



```
[base_controller-50] [INFO] [1726904009.120091998] [base_controller]: Current number of sensor data detected:10
[base_controller-50] [INFO] [1726904009.120424051] [base_controller]: Current Latest Sensor Data:Sensor data: 1_40, Sensor data: 7_45, Sensor data: 3_101, Sensor
 data: 2_85, Sensor data: 0_39, Sensor data: 8_81, Sensor data: 4_79, Sensor data: 9_37, Sensor data: 5_65, Sensor data: 6_89
```

Figure 4: Output printed to show all sensor detected by Base_station

2 Definitions

2.1 Explanation of extra Variables

In this subsection, it describes the purpose and usage of each variable in the provided code:

- `SENSOR_RADIUS`: This variable specifies the radius in which the sensors are placed, representing the distance from the center where the sensors will be spawned. It is set to 25 meters.
- `DISTANCE_THRESHOLD`: This defines the distance within which a balloon or drone must be to the base in order for the controller to process and log its sensor data. The threshold is set to 30 units.
- `latest_sensor_data`: This dictionary stores the most recent sensor data received from each balloon and drone, indexed by their respective IDs. It maintains only the latest value for each type of sensor, ensuring that duplicate or outdated data is not retained.
- `assignments`: This variable keeps track of the sensor-to-drone assignments made by the coordinator, ensuring that each drone is assigned to a specific group of sensors to patrol based on proximity.
- `sensor_list`: A list that stores the IDs of sensors once they are detected, ensuring unique sensors are registered for tasks.
- `timer`: A timer that periodically publishes the cache data every second using the `publish_cache_data` method.

  `timer` = `create_timer(1.0, self.publish_cache_data)`
- `sensor_ids = list(self.sensor_positions.keys())`:

  Retrieves the unique identifiers of all sensors currently stored in `self.sensor_positions`. Converts the keys of the dictionary to a list for easier manipulation.
- `distance_matrix=[[0.0]*NUMBER_OF_SENSORS for _ in range(NUMBER_OF_DRONES)]`:

  Initializes a two-dimensional list (matrix) to store distances between each drone and each sensor. The matrix has dimensions `NUMBER_OF_DRONES` *rows* by `NUMBER_OF_SENSORS` *columns*, with all initial values set to 0.0. Each entry at `[i][j]` represents the distance from the i^{th} drone to the j^{th} sensor.

2.2 Explanation of important extra Topics

- `tx_data_publisher`: A publisher created to send `String` messages to the `tx_data_balloon` topic for communication with the base controller. Messages are relayed from the `rx_data` subscription.
- `sensor_data_publisher`: A ROS2 publisher used to transmit aggregated and processed sensor data from the balloons and drones. The data is published as a JSON string and can be accessed by other nodes in the system.

- `self.data_publisher`: A ROS publisher that sends cache data (in `String` format) from the drone to the balloon on the `drone_to_balloon` topic.

  `data_publisher = create_publisher(String, 'drone_to_balloon', 10)`

- `self.data_base_publisher`: Another ROS publisher that sends the same cache data from the drone to the base on the `drone_to_base` topic.

  `data_base_publisher = create_publisher(String, 'drone_to_base', 10)`

2.3 Explanation of extra Functions

- `calculate_and_print_distance_matrix(self)`:
  - Calculates and logs the distance matrix between all sensors and drones.
  - Assigns sensors to drones based on proximity.
- `assign_sensors_to_drones(self, distance_matrix)`:
  - Assigns sensors to drones based on the calculated distance matrix.
  - Ensures a balanced distribution of sensors among drones.
- `get_assigned_sensors_for_drone(self, drone_id)`:
  - Retrieves the list of sensors assigned to a specific drone.
- `process_data(self, balloon_id, msg: String)`:
  - Processes incoming sensor data from the specified balloon.
  - Checks the distance from the balloon to a predefined base position and updates sensor data if within a threshold.
  - Logs the latest sensor data if updates are made.
- `process_data_drone(self, drone_id, msg: String)`:
  - Processes incoming sensor data from the specified drone.
  - Similar to `process_data`, but updates values only if the new data is greater or equal to existing values.
  - Logs the latest sensor data if updates are made.
- `print_latest_sensor_data(self)`:
  - Consolidates and formats all the latest sensor data from balloons and drones.
  - Publishes the aggregated sensor data as a JSON string to the `sensor_data` topic.

3 Test and Performance

in this section are reported different tests:

3.1

10 Sensors and 3 Drones

In this test, we used 10 sensors and 3 drones. The range of the recorded times is as follows:

- The time range for 10 sensors and 3 drones is from 59 seconds to 2 minutes 20 seconds.

3.2 20 Sensors and 5 Drones

In this test, we used 20 sensors and 5 drones. The range of the recorded times is as follows:

- The time range for 20 sensors and 5 drones is from 3 minutes 15 seconds to 3 minutes 24 seconds.

4 Conclusions

In conclusion, we took action on two fronts: improving the positioning of drones and balloons to ensure optimal sensor coverage, reducing the risk of collisions and improve the effecient of drone path with the circular arrangement and paths optimized through a distance matrix, drones can now detect sensors more efficiently, limiting flight time and maximizing data update. The decision to fly drones at different heights, together with dynamic sensor assignment logic, has significantly improved the management of the system.