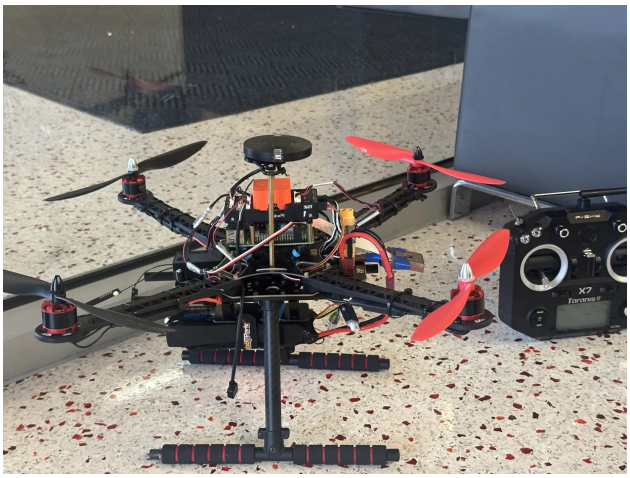
Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Fall 2024



DR Rescue
Aerial Rescue Mapping Drone

Omar Elsaghir
Justin Barrett
David Duong
Anaf Mahbub

Revision History

Revision	Date	Author(s)	Description
0.1	11.12.2024	OE	Introduction (1) rough draft
0.2	11.12.2024	OE	System Overview (2) rough draft
0.3	11.15.2024	JB	2.3 overview
0.4	11.19.2024	OE	Perception Layer (4)
0.5	11.19.2024	DD	Added System Overview diagram (2)
0.6	11.25.2024	AM	Added Subsystems definitions and data flow (3)
0.7	11.25.2024	DD	Proofreading
0.8	11.25.2024	DD	Y Layer Subsystems diagram (5)
1.0	11.15.2024	JB	Decision Making Layer (6)

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

DR-Rescue is a drone designed to map out a route to the object being rescued and send the route to a rover. The rover will then pick up the object. Users of DR-Rescue can designate any object to be picked up by the rover, with the route mapped by the drone.

DR-Rescue consists of two components: a drone and a rover. The system is designed to rescue objects at various locations. The drone maps out a route and sends it to the rover, which then picks up the object.

The goal of DR-Rescue is to retrieve an object from a specific location and map out a route to guide the rover to the object. The ultimate objective is to make the system reliable enough for use in other situations. The current focus is to ensure it functions properly and with high accuracy. Communication between the drone and the rover will be wireless. From the standpoint of the user, a licensed drone operator is required to program and oversee the drone in case of an incident. A licensed operator must be present for the DR-Rescue system to be demonstrated. During the demonstration, users observe the drone working in coordination with the rover. The drone and rover operate together seamlessly to complete the task. The user will observe the drone in action as it navigates to an object placed at a location surrounded by obstacles, designed to test its route-planning capabilities. The rover will then navigate through the obstacles to reach the object. The user will also be able to see the drone's live feed on the drone controller. Additionally, one team member will monitor the system on a computer to ensure everything is working correctly.

Utilizing the Areal Rescue Mapping Drone, should be a very simple process to maintain time criticality. Users of the drone should expect an easy to load console application, which will connect to the drone to the computer. Users will likely be asked to input an identification number specific to the drone, and will connect using an existing protocol, the computer will attempt to establish a connection with the drone once it is given the necessary information. Simplicity will be key in the communication process, and no redundant information should be expected from the user.

2 System Overview

This section provides an overview of DR Rescue. Data from the drone will be transmitted to the Raspberry Pi, which will then send the information back to the drone. The drone will serve as the perception layer. On the rover side, the data from the Raspberry Pi will be passed to the rover, along with commands to determine the path to follow. The figure below presents a high-level architectural view, identifying the primary components and interactions between layers.

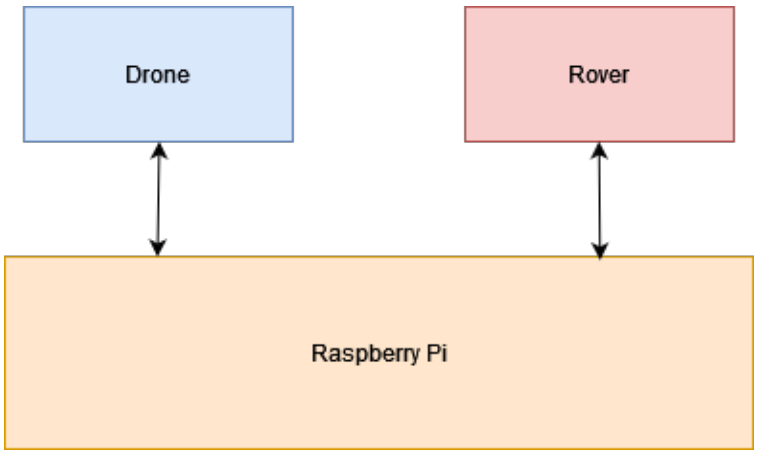


Figure 1: DR Rescue architectural layer diagram

2.1 Perception Layer

The perception layer is responsible for collecting, processing, and interpreting data from the environment. It interacts directly with the hardware sensors, including cameras, LIDAR, GPS, and thermal sensors. The perception layer translates raw data into structured information, such as mapping surroundings, detecting objects, and identifying potential hazards.

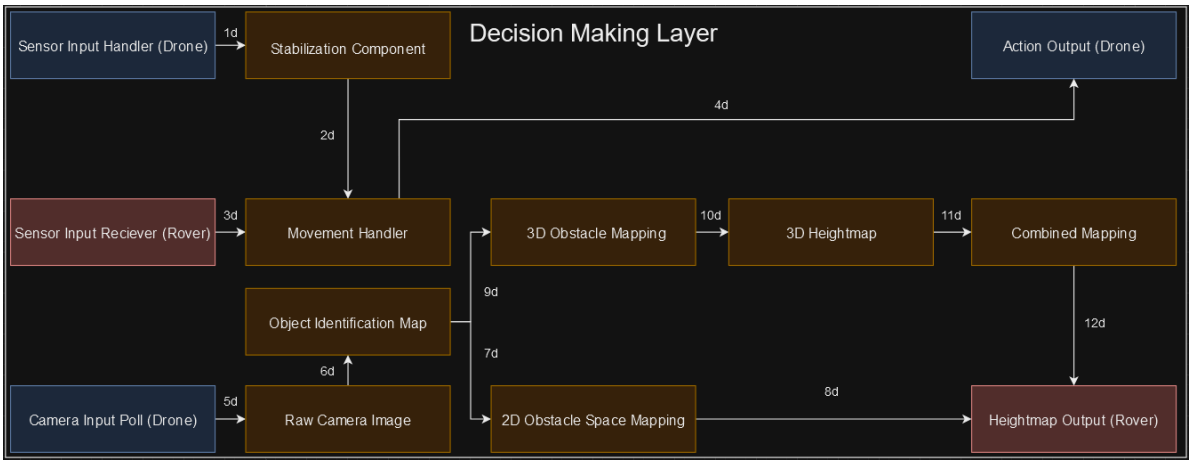
Features and Functions:
Data Collection: Uses sensor modules to gather real-time data from the environment.
Data Processing: Applies filtering and preprocessing techniques to improve data quality.
Critical Interfaces and Interactions:
Input: Directly receives data from sensor hardware
Output: Sends processed data to the Decision-Making Layer for further analysis.
Provided Services:
Structured environmental data for navigation and decision-making.
Naming Conventions:
Sensors are prefixed by type (ex: LidarFront for front LIDAR)
Data flow identifiers are prefixed with Raw or Processed based on the data type.

2.2 Rover Layer

The rover will contain the sensor input receiver and the heightmap output. These components will be configured on the drone and will provide the output to the rover, enabling it to follow the correct path. The rover will also be equipped with sensors to ensure it does not collide with any objects. If a collision risk is detected, the rover will notify the drone, indicating that an alternate path is required. Additionally, the rover will require a system to control the motors, ensuring smooth operation.

2.3 Decision-Making Layer

The decision layer is responsible for using the output of the perception layer to generate inputs for the action layer and heightmap output. It accomplishes this by processing various types of data, and by using either existing algorithms or those specifically designed for the drone, it utilizes the inputs to perform certain tasks (highlighted in section 6).



3 Subsystem Definitions & Data Flow

This section breaks down the layer abstraction to another level of detail. It shows the interaction between the rover, drone and the Raspberry Pi. All three layers have multiple subsystems. The arrows represent the data flow and interactions between those layers.

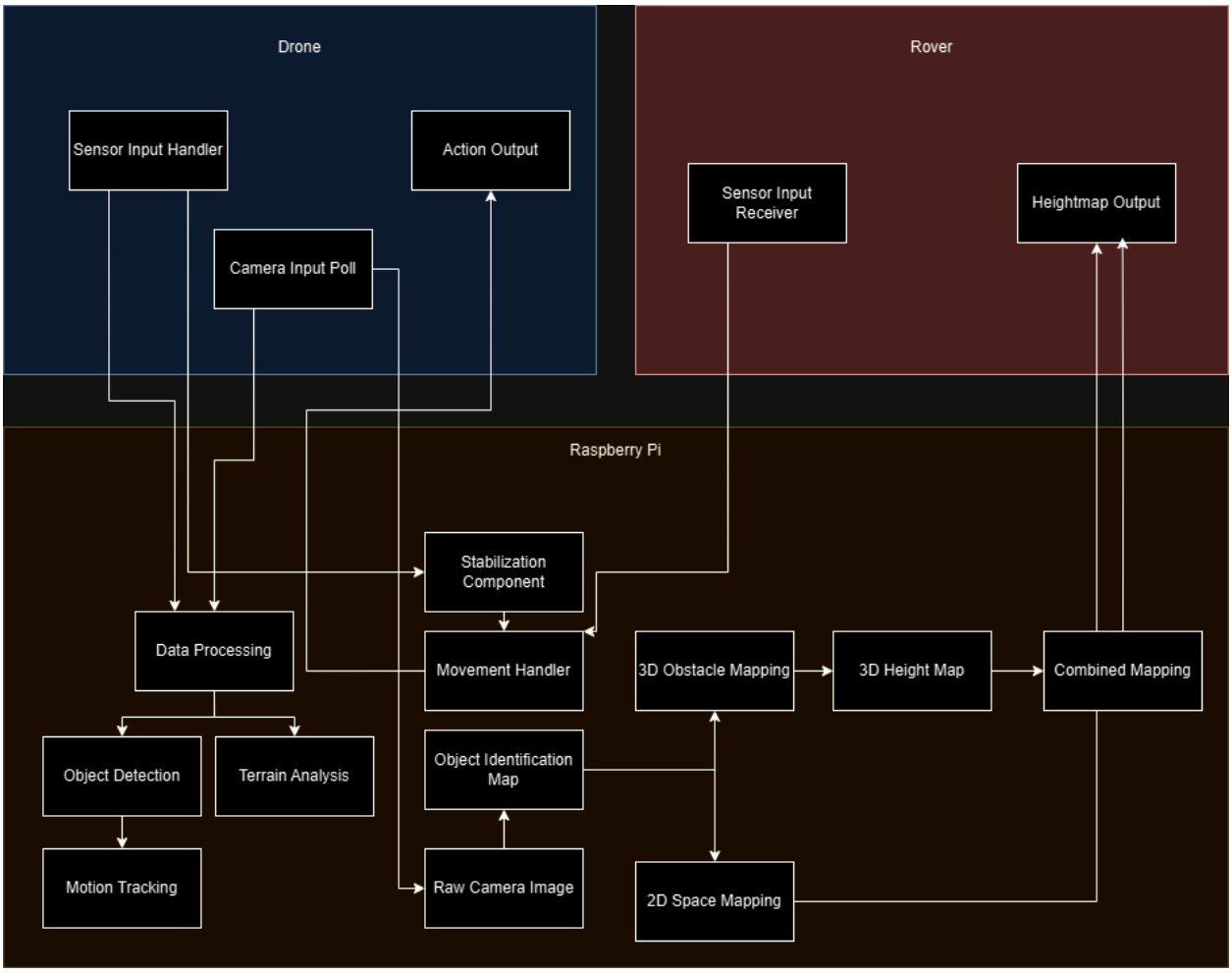


Figure 2: A diagram that breaks down the data flow between layers

4 Perception Layer Subsystems

This section explains the drone side of the perception layer and the requirements for it to function correctly.

4.1 Sensor Input Handling

The Sensor Input Handling Subsystem serves as the sensory organ of the drone, responsible for collecting raw, unprocessed data from the environment.

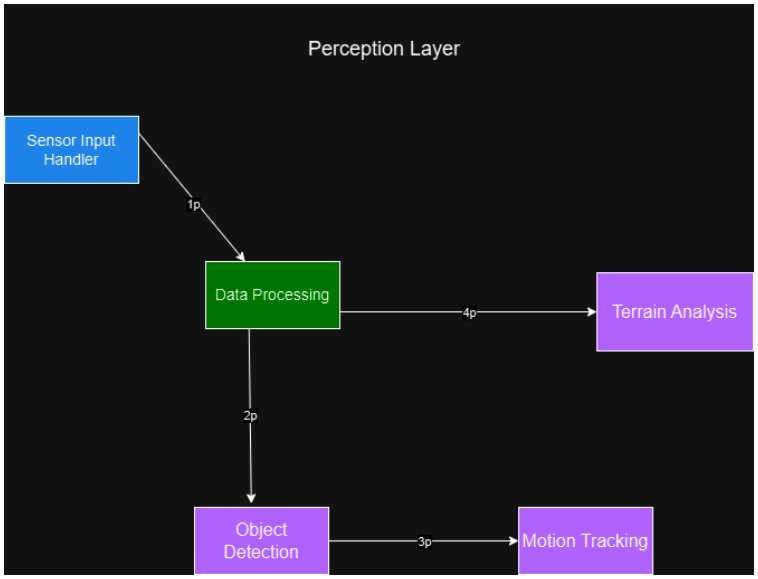


Figure 3: Sensor Input Handler subsystem highlighted in blue

4.1.1 Assumptions

Sensors are configured as inputs for collecting, processing, and interpreting data from the environment. They communicate directly with the onboard processing unit using standardized communication protocols.

4.1.2 Responsibilities

All sensors (LIDAR, Thermal Sensors, GPS Module, Cameras) are used to capture raw data from the environment. Cameras capture high-resolution images or video streams and support optical zoom or infrared modes for enhanced visibility in diverse conditions. LIDAR sensors create 3D point clouds of the environment, assisting in obstacle detection and navigation. Thermal sensors detect heat signatures to locate humans, animals, or machinery. The GPS Module acquires precise latitude, longitude, and altitude data. The data gathered from the sensors is transmitted to the Data Processing subsystem for analysis.

4.1.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#1p	Camera feed interface	Visual data	Raw camera feed
#1p	LIDAR interface	Distance measurements	Raw point cloud data
#1p	Thermal sensor interface	Heat signatures	Raw thermal image
#1p	GPS interface	Satellite signals	Raw GPS coordinates

4.2 Data Processing

This subsystem refines the raw data collected by the Sensor Subsystem, converting it into clean, structured, and usable forms.

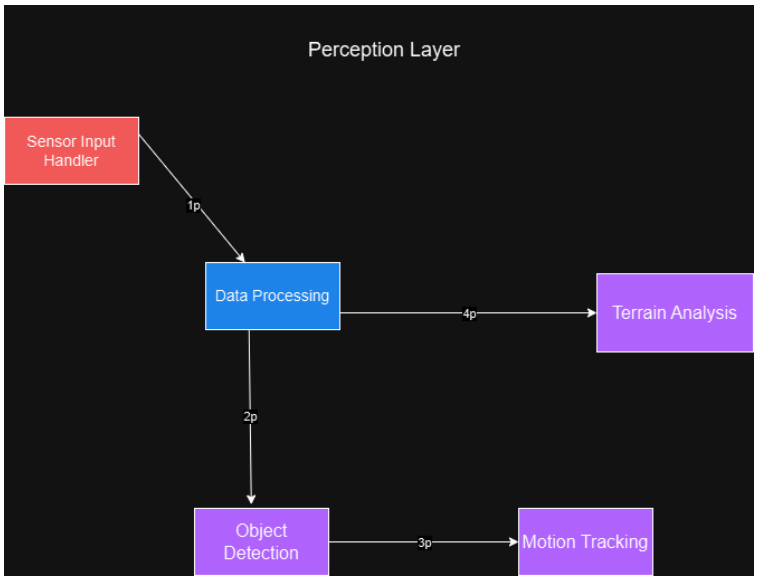


Figure 4: Data Processing Subsystem highlighted in blue

4.2.1 Assumptions

Processing of the data will occur in real-time to support dynamic decision-making. Algorithms are going to be optimized for the drone's onboard computational limitations

4.2.2 Responsibilities

The drone will convert the visual data obtained from the sensors into structured grids or feature sets for analysis. The system will then format the data from the LIDAR and GPS into coordinate-based maps that are usable by higher-level systems.

4.2.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#1p	Camera data processing	Raw camera feed	Filtered visual data
#1p	LIDAR data processing	Raw point cloud data	Structured 3D maps
#1p	Thermal data processing	Raw thermal images	Refined heat signature data
#1p	Sensor fusion interface	Outputs from all sensors	Unified environmental map

4.3 Object Detection

This subsystem is responsible for detecting and classifying objects from the processed data, identifying points of interest for the drone's mission.

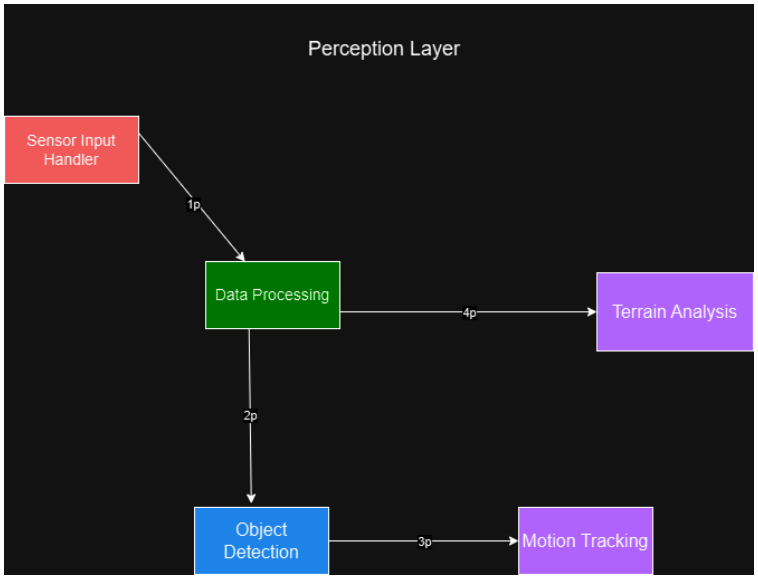


Figure 5: Object Detection subsystem highlighted in blue

4.3.1 Assumptions

The drone will use the data processed that were gathered from the sensors to determine which are objects, and then it will use that data to reroute the movement of the rover. The system uses a combination of deep learning models and rule-based approaches to handle diverse environments.

4.3.2 Responsibilities

The drone will highlight obstacles in the drone's path using LIDAR and visual data. Using the processed data received from the thermal sensors, it will analyze thermal data to identify potential survivors based on body heat patterns and flag heat signatures consistent with human profiles for further validation.

4.3.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#2p	Visual object detection	Filtered visual data	Detected objects (visual)
#2p	Thermal object detection	Refined heat signature data	Detected heat sources
#2p	Obstacle mapping interface	Unified environmental map	Obstacle locations

4.4 Motion Tracking

This subsystem tracks moving objects and provides dynamic updates for obstacle avoidance or target tracking.

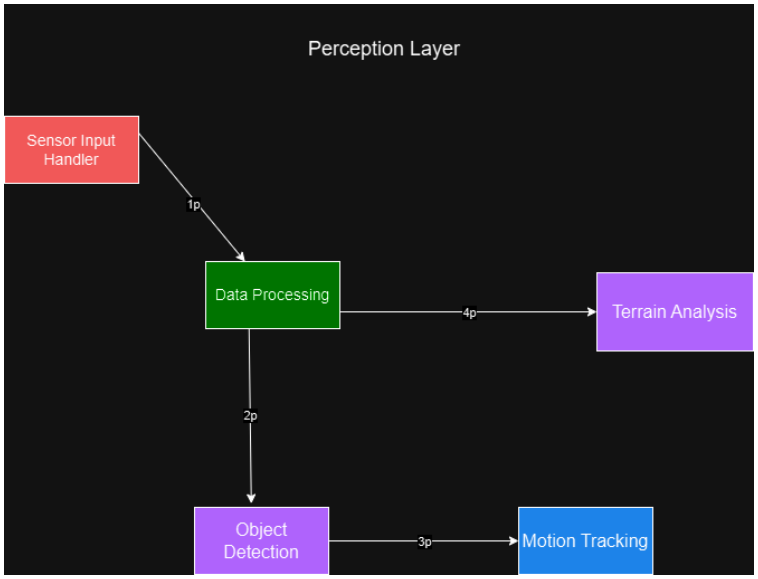


Figure 6: Motion Tracking subsystem highlighted in blue

4.4.1 Assumptions

Through the use of the onboard computer (The Raspberry Pi), the drone will be able to determine if there are any moving objects heading towards it or around the area it is currently located.

4.4.2 Responsibilities

The drone will use the LIDAR sensors to track the position and velocity of moving targets. After the drone processes the data obtained from the LIDAR, it will notify the Decision-Making Layer of rapidly approaching obstacles and send a request to the rover to move in another direction to avoid the obstacles in its path.

4.4.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labeled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

ID	Description	Inputs	Outputs
#3p	Motion detection interface	Visual/LIDAR data	Tracked objects
#3p	Collision alert interface	Object movement data	Proximity alerts

4.5 Terrain Analysis

This subsystem specializes in analyzing the terrain for navigation and mapping purposes.

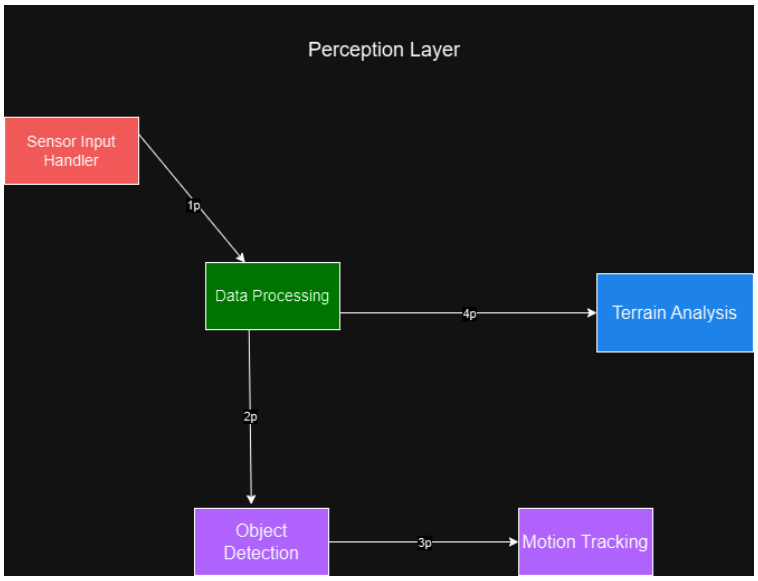


Figure 7: Terrain Analysis subsystem highlighted in blue

4.5.1 Assumptions

It is assumed that the drone processes LIDAR and visual data (from Data Processing Subsystem) to create terrain maps, identify landing zones, and assess navigation paths. Outputs are used by the Decision-Making Layer.

4.5.2 Responsibilities

The drone will use data obtained from the LIDAR sensor and camera to create detailed 3D maps of the environment, which will be shared with the rover to guide its path. This data will also be used to detect and evaluate flat, obstacle-free areas suitable for landing. Additionally, the drone will assess potential paths for safe navigation, helping the rover avoid steep inclines that could damage critical hardware components or hinder progress toward its destination.

4.5.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#4p	Terrain mapping interface	LIDAR/visual data	3D terrain map
#4p	Landing zone detection interface	Processed terrain data	Landing zone co-ordinates

5 Rover Layer Subsystems

The rover will be communicated through the Raspberry Pi and take the information of what the drone outputs onto the Raspberry Pi and send it to the rover

5.1 Raspberry Pi

The Raspberry Pi will be the main component in connecting the drone and the rover together. It plays a huge part in the success of this project.

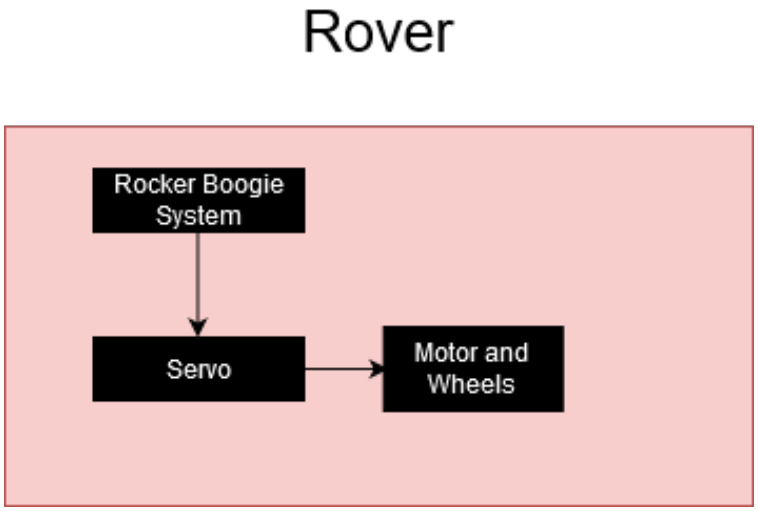


Figure 8: Rover subsystems

5.1.1 Assumptions

The assumption should be that the Raspberry Pi should have no problems connecting with both the drone and the rover.

5.1.2 Responsibilities

According to the diagram, the rover must connect to a servo, which will then control the motor and wheels. The diagram does not include additional components that will be required. The system must first establish a connection to the Raspberry Pi and then synchronize with the drone to ensure all data is accurate. Various safeguards will be in place to stop the drone and the rover if any issues arise.

5.1.3 Subsystem Interfaces

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#	Raspberry Pi to the Rover	Raspberry Pi	Rover
#	Rover to the Raspberry Pi	Rover	Raspberry Pi
#	Rover to servo then to motors and wheels	Servo	Motor and Wheels

5.2 SERVO

The servo requires inputs from the rover to control its operation, which then drives the wheels and motors to ensure proper functionality.

5.3 ROVER SENSORS

The sensors on the rover will detect any objects in the pathway and send an alert signal to the drone to prevent the rover from continuing on its current path. The drone will then generate a new path for the rover to follow.

6 Decision Making Layer Subsystems

In this section, the layer is described in some detail in terms of its specific subsystems. Describe each of the layers and its subsystems in a separate chapter/major subsection of this document. The content of each subsystem description should be similar. Include in this section any special considerations and/or trade-offs considered for the approach you have chosen.

6.1 Sensor Input Handler

This subsystem handles direct communication between the perception layer of the drone and the decisions the drone makes. The sensor input handler will be in charge of collecting the perception layer state and sending it over to components that rely on it.

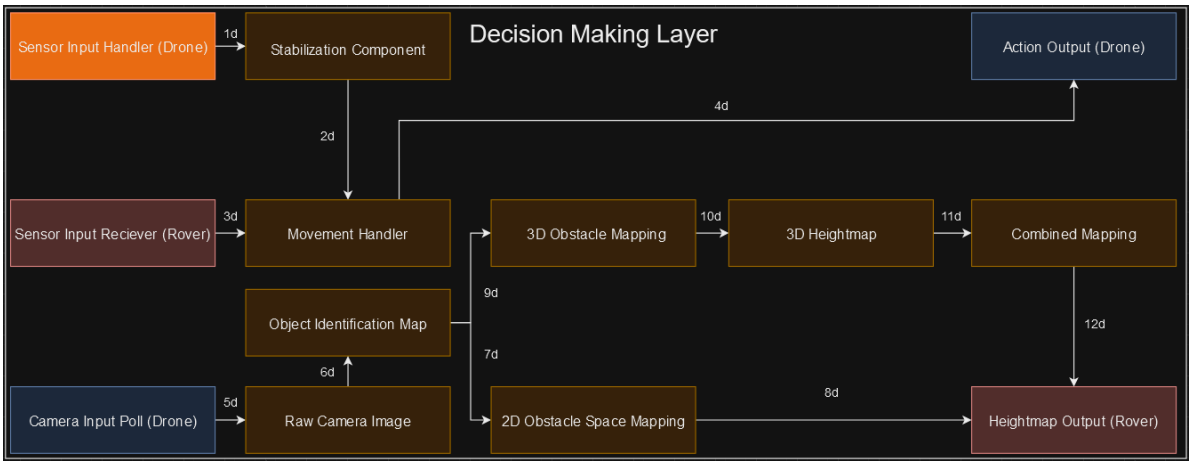


Figure 9: Sensor Input Handler highlighted in bright orange

6.1.1 Assumptions

For now it can be safely assumed that interrupts will be used to take sensor input from the drone. A polling system may be developed if interrupts cause too much system overhead. This layer is assumed to use the same event system as **Action Output**. However, the diagram drawn does not show this event system as an interface that both inherit

6.1.2 Responsibilities

The Sensor Input Handler will listen for an event (ie. a sudden shift in sensor readings). When an event is dispatched to the Sensor Input Handler, it will send the event data to the necessary components that handle adjusting the orientation of the drone.

6.1.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#1d	Unpackages event	Sensor-Event	Sensor-State

6.2 Stabilization Component

The stabilization component is to make smaller adjustments to the drones movement. If the drone has drifted from its current location, it attempts to stabilize, as not doing so may result in an unclear image for the drone. Stabilizing ensures any unintentional motion blur is mitigated and a clear photo is taken.

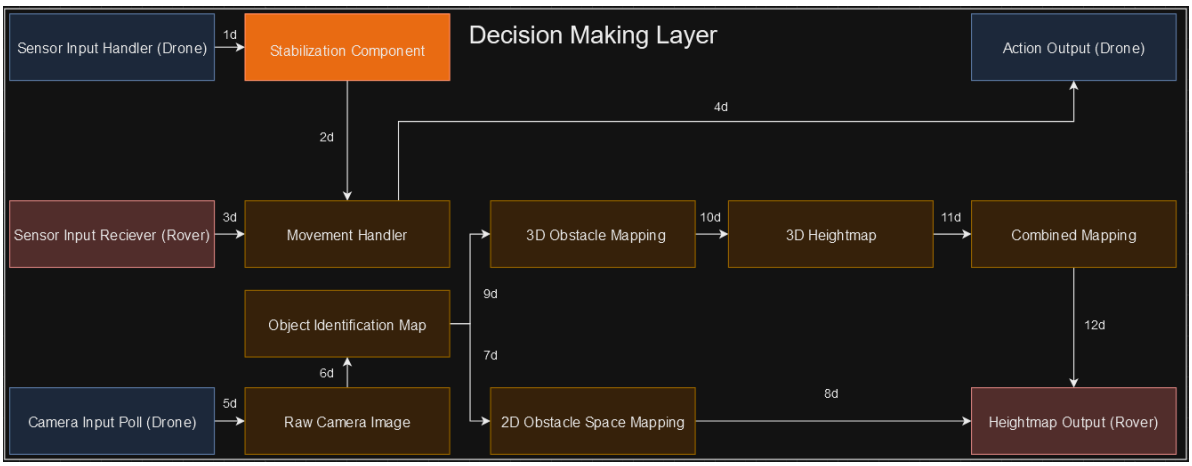


Figure 10: Stabilization Component highlighted in bright orange

6.2.1 Assumptions

This component may not be necessary if the drone has built in mechanisms to prevent drifting in the wind (Stabilization Component will be included if needed). If it isn't, then **interface 1D** may connect directly with **Movement Handler**, where **interface 2D** will be discarded

6.2.2 Responsibilities

This subsystem is responsible for outputting a desirable sensor state. It takes in the current state, and stores a cumulative value for how far the drone has drifted over time. It then sends a "corrective state" which will be what is required to bring the drone back to a neutral position.

6.2.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#1d	Unpackages event	Sensor-Event	Sensor-State
#2d	Sends calculated error	Sensor-State	Corrective-Sensor-State

6.3 Sensor Input Receiver

The Sensor Input Receiver takes in rover communications. This component is used to relay some information that the drone might find valuable. This will mostly handle the TCP connection between the rover and drone. The Rover may also request drone movement, which will be highlighted later

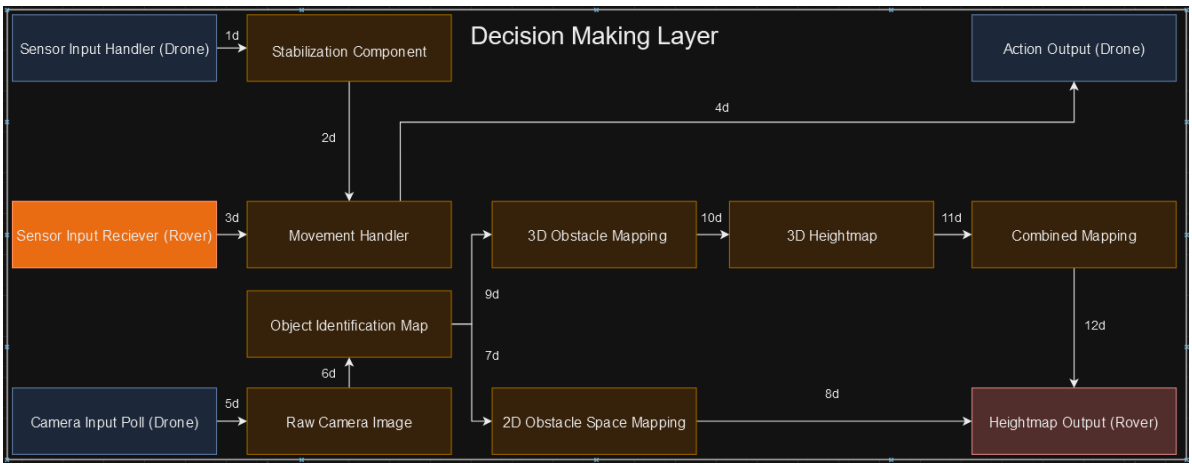


Figure 11: Sensor Input Receiver highlighted in bright orange

6.3.1 Assumptions

It is assumed that the drone requires specific information from the rover; however, this may not be necessary in future iterations. If the rover encounters an obstacle that is not detected by the drone, it may request that the obstacle be added to the obstacle map.

6.3.2 Responsibilities

The Sensor Input Receiver takes rover request packets as an input and outputs the request to the **Movement Handler**. Apon receiving requests from the rover, the drone may change position in order to follow the rovers request, or outright ignore the request entirely.

6.3.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#3d	Sends rover status	Rover-Status-Packet	Rover-Status

6.4 Movement Handler

The movement handler handles the decision of how much power each rotor should receive based on what the rover requests and the drift the **stabilization component** calculates.

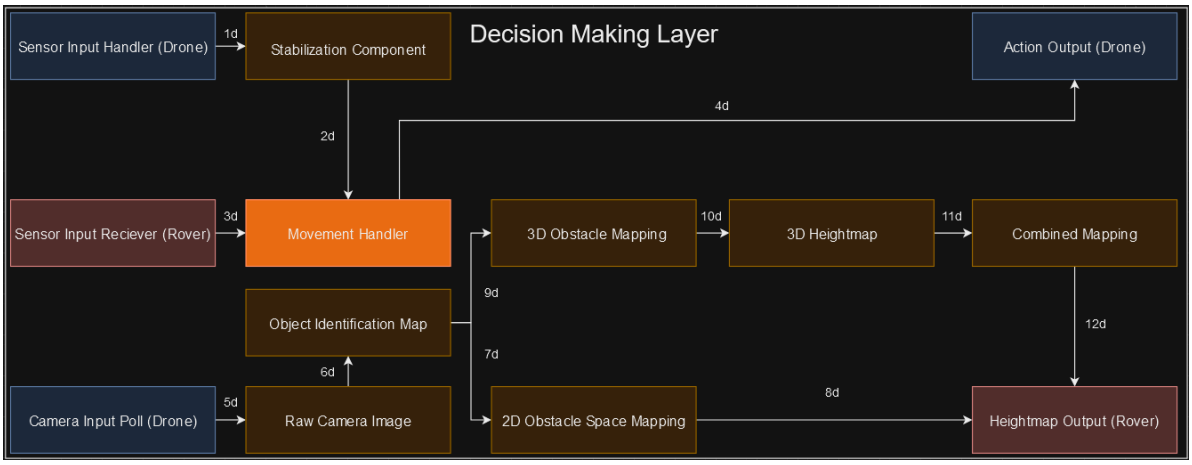


Figure 12: Movement Handler highlighted in bright orange

6.4.1 Assumptions

The assumption is that the **Sensor Input Receiver** for the Rover has the ability to request drone movement, which may not be required in future iterations.

6.4.2 Responsibilities

The movement handler is responsible for calculating the necessary movement speeds and distances to reach a target location. If the **Sensor Input Receiver** requests the drone flies closer to an obstacle to investigate it, then the Movement Handler will reroute the drone as necessary. Otherwise, it will take output from the **Stabilization Component** when necessary and attempt to correct the drone if it flies off course.

6.4.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#2d	Sends calculated error	Sensor-State	Corrective-Sensor-State
#3d	Sends rover status	Rover-Status-Packet	Rover-Status
#4d	Sends calculated ideal rotor speed	Corrective-Sensor-State Rover-Status	Rotor-Speeds

6.5 Action Output

The action output is how much power the **Movement-Handler** requests to go to the rotors. The Action output packages this information into an event and sends it to the **Action Layer** to handle the event

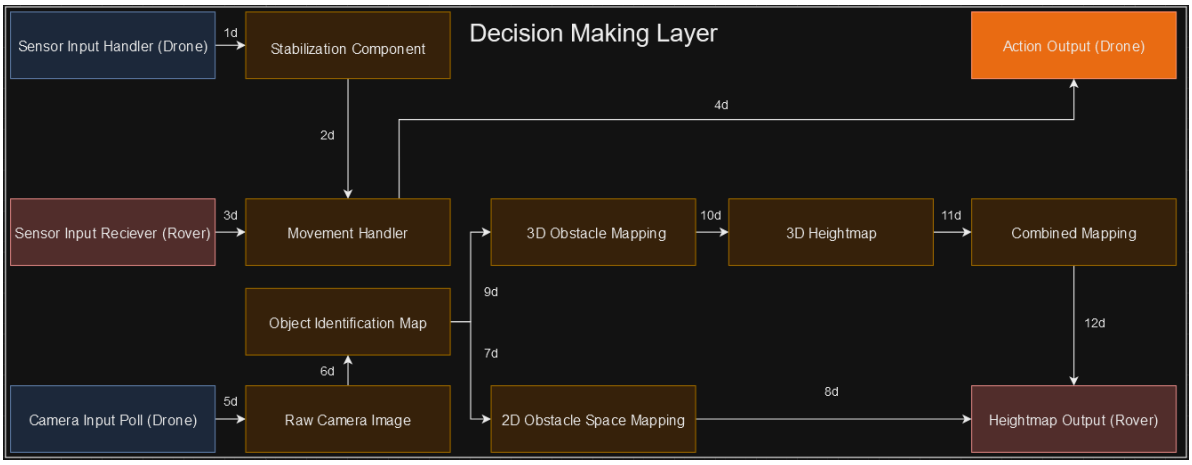


Figure 13: Action Output highlighted in bright orange

6.5.1 Assumptions

This layer is assumed to use the same event system as **Sensor Input Handler**. However, the diagram drawn does not show this event system as an interface that both inherit

6.5.2 Responsibilities

This subsystem dispatches events out to the **Action Layer** for them to be handled by the hardware components.

6.5.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#4d	Sends calculated ideal rotor speed	Corrective-Sensor-State Rover-Status	Rotor-Speeds
#1da	Dispatches rotor event	Rotor-Speeds	Rotor-Event

6.6 Camera Input Poll

An input polling system to get data from the camera on the drone.

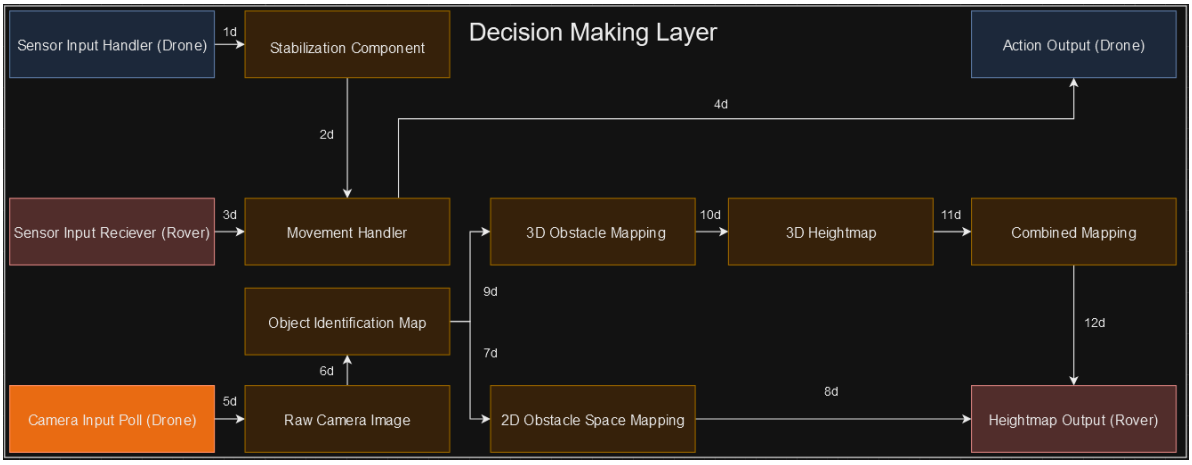


Figure 14: Camera Input Poll highlighted in bright orange

6.6.1 Assumptions

It is assumed that the camera will provide high accuracy and visibility, along with a high resolution. The camera is expected to transmit data in a stream, meaning not all data may be polled simultaneously. If the camera is capable of transmitting all data at once, this subsystem may be eliminated, and the existing connection **interface 5D** could be rerouted to the **Raw Camera Image**.

6.6.2 Responsibilities

Transfer the input stream to the **Raw Camera Image** so it can be used to create a picture of the landscape.

6.6.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#5d	Sends camera input	Input-Stream	Stored-Input

6.7 Raw Camera Image

The raw camera image subsystem takes the input buffer received from **Camera Input Poll** to store the entire image on the Raspberry Pi

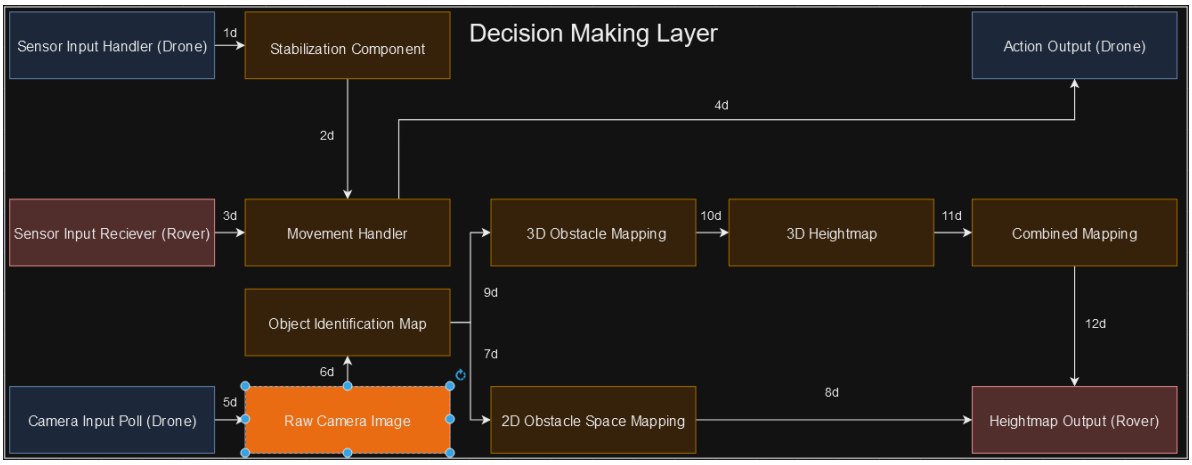


Figure 15: Raw Camera Image highlighted in bright orange

6.7.1 Assumptions

Same assumptions as 6.6.1

6.7.2 Responsibilities

This subsystem is responsible for writing the image to the device for **Object Identification Map** to use later

6.7.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#5d	Sends camera input	Input-Stream	Stored-Input
#6d	Sends image	Stored-Input	Image

6.8 Object Identification Map

Identifies objects in the image given by **Raw Camera Image**

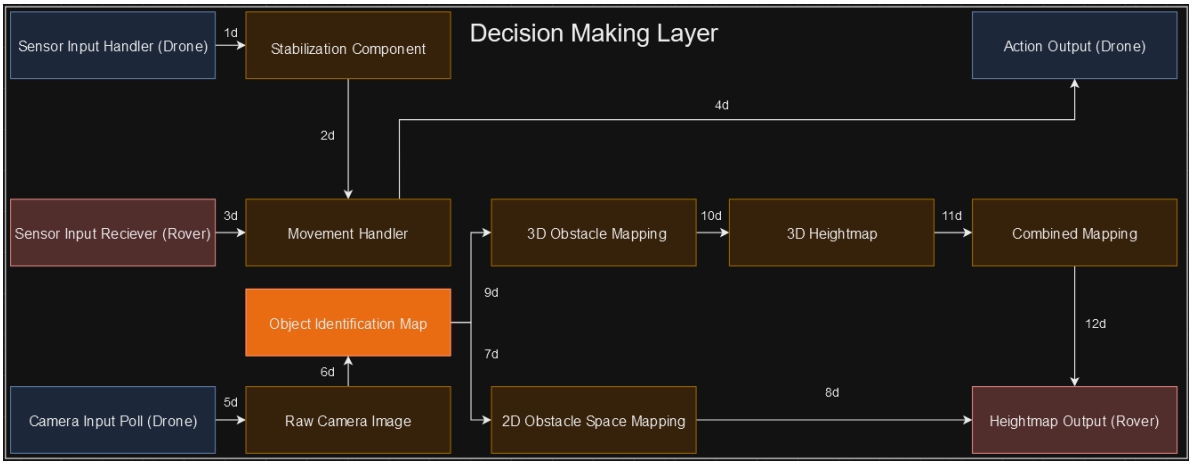


Figure 16: Object Identification Map highlighted in bright orange

6.8.1 Assumptions

- Same assumptions as 6.6.1
- Objects must be a bright red color
- Rover must have a bright blue marker to assist in its position

6.8.2 Responsibilities

The Object Identification Mapping is responsible for the creation of the object map. The object map is the map containing all of the **clearly labeled** objects and representing them in the final 2D/3D map. This is not to be confused with the **Other Obstacle Maps** which have a buffer around the objects equal to half the size of the width of the robot.

6.8.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#6d	Sends image	Stored-Input	Image
#7d	Sends obstacle map	Image	Obstacle-Highlighted-Field Rover-Location
#9d	Sends obstacle map	Image	Image Obstacle-Highlighted-Field Rover-Location

6.9 2D Obstacle Space Mapping

The 2D Obstacle Space Mapping will receive the camera image as well as the **Object Identification Map** to create an Obstacle Space, where the robot can be represented as a single point in space and objects are represented with a buffer around them to avoid the rover running into them.

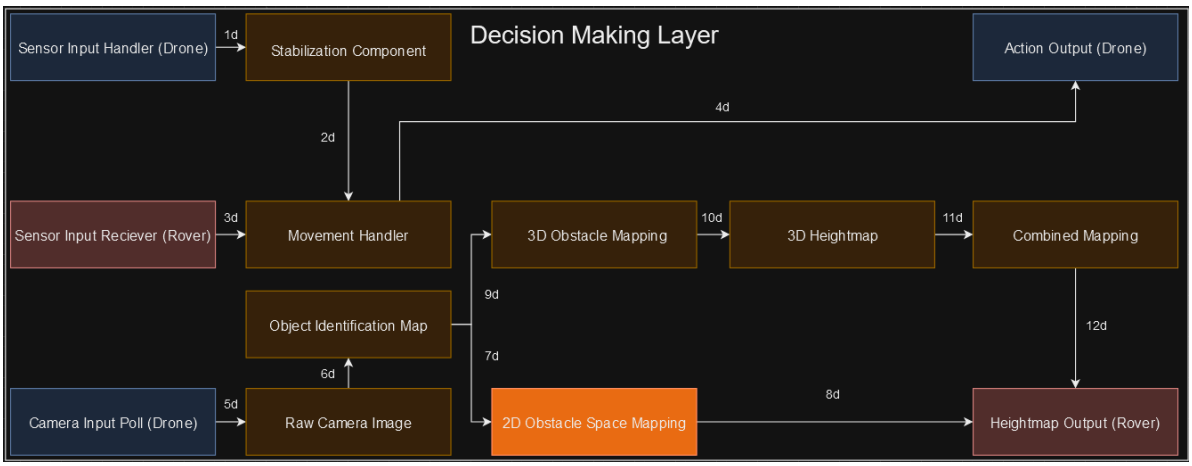


Figure 17: 2D Obstacle Space Mapping highlighted in bright orange

6.9.1 Assumptions

Drones optical capability allows for 2-Dimensional imagery

6.9.2 Responsibilities

The 2D Obstacle Space Mapping is responsible for converting the **Object Identification Map** into a buffered Object Identification Map. A buffered Object Identification Map has one key difference, in that the space the rover takes up is added to every obstacle in the space as a buffer, which helps simplify the algorithms used in path-finding.

6.9.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#7d	Sends obstacle map	Image	Obstacle-Highlighted-Field Rover-Location
#8d	Sends final map	Obstacle-Highlighted-Field Rover-Location	Buffered-Obstacle-Space Rover-Location

6.10 3D Obstacle Mapping

The 3D Obstacle Space Mapping will receive the camera image as well as the **Object Identification Map** to create an Obstacle Space, where the robot can be represented as a single point in space and objects are represented with a buffer around them to avoid the rover running into them.

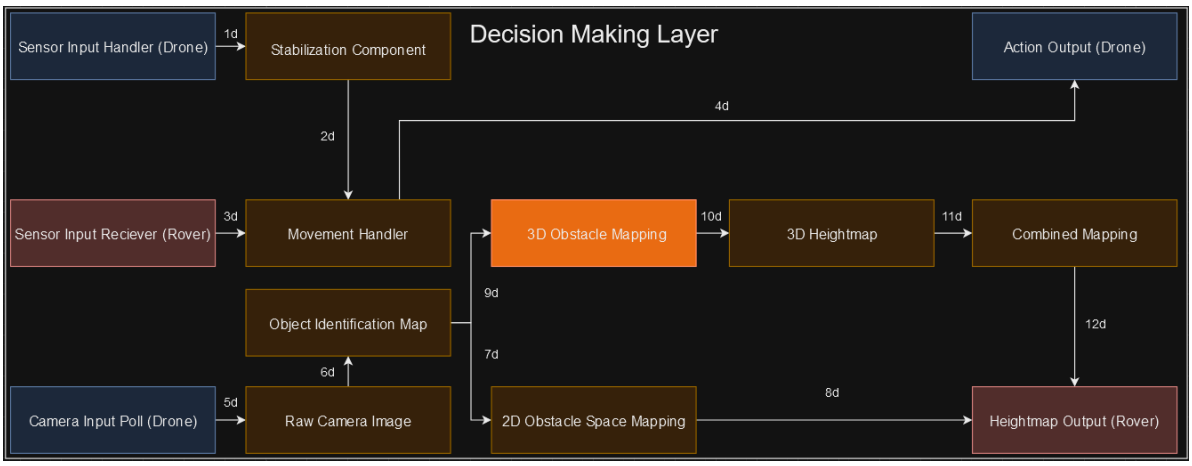


Figure 18: 3D Obstacle Mapping highlighted in bright orange

6.10.1 Assumptions

Drones optical capability allows for 3-Dimensional imagery. Also if the **Camera Input Poll** returns a 3D heightmap, this step will be skipped and **interface 3D** can route directly to **Heightmap Output**

6.10.2 Responsibilities

The 3D Obstacle Space Mapping is responsible for converting the **Object Identification Map** into a buffered Object Identification Map. A buffered Object Identification Map has one key difference, in that the space the rover takes up is added to every obstacle in the space as a buffer, which helps simplify the algorithms used in path-finding. Due to the necessity that data be represented efficiently, there will be a special value applied to objects on the map, wherein any place on the buffered Object Identification Map that is marked will also be marked with a height of 0 in the **3D Heightmap**.

6.10.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#9d	Sends obstacle map	Image	Image Obstacle-Highlighted-Field Rover-Location
#10d	Sends obstacle space	Image Obstacle-Highlighted-Field Rover-Location	Image Buffered-Obstacle-Space RoverLocation

6.11 3D Heightmap

The 3D-Heightmap generates the final 3d heightmap that the drone will use. This heightmap is to be combined with the **3D Obstacle Mapping** in the **Combined Mapping** stage.

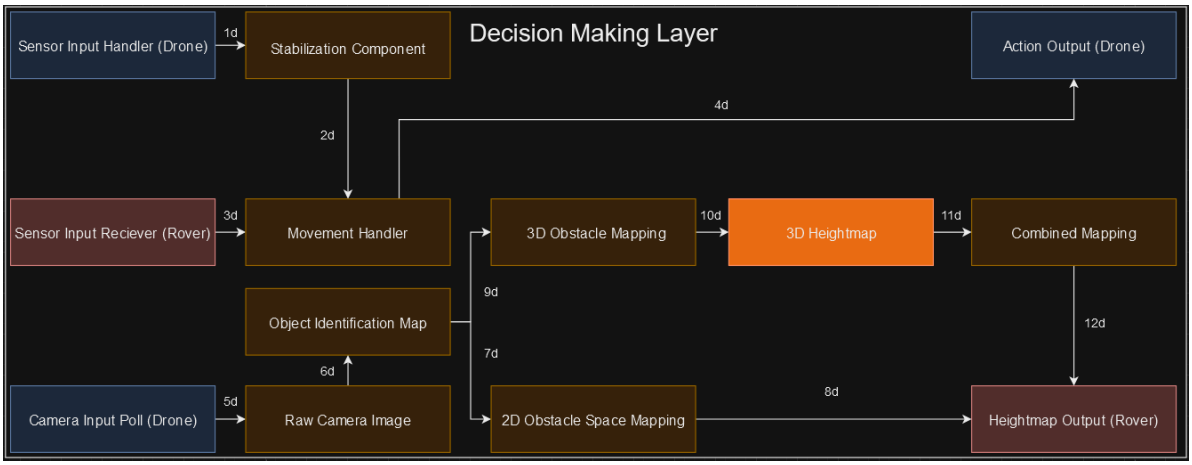


Figure 19: Example subsystem description diagram

6.11.1 Assumptions

Same assumptions as 6.10.1

6.11.2 Responsibilities

The 3D-Heighthmap is responsible for generating a heightmap with respect to the previous steps. This means Obstacles will be represented as heights on the heightmap, to allow more efficient handling of data. The final heightmap will not include color.

6.11.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#10d	Sends obstacle space	Image Obstacle-Highlighted-Field Rover-Location	Image Buffered-Obstacle-Space RoverLocation
#11d	Adds heightmap info	Image Buffered-Obstacle-Space Rover-Location	Heightmap Buffered-Obstacle-Space Rover-Location

6.12 Combined Mapping

The Combined Mapping is the final stage before the **Heightmap 3D** is ready to be sent to the rover. This stage simply represents object as a specific value on the heightmap. This representation will specifically be made to ensure that the rover does not attempt to climb obstacles. Implementation specifics will not be mentioned here

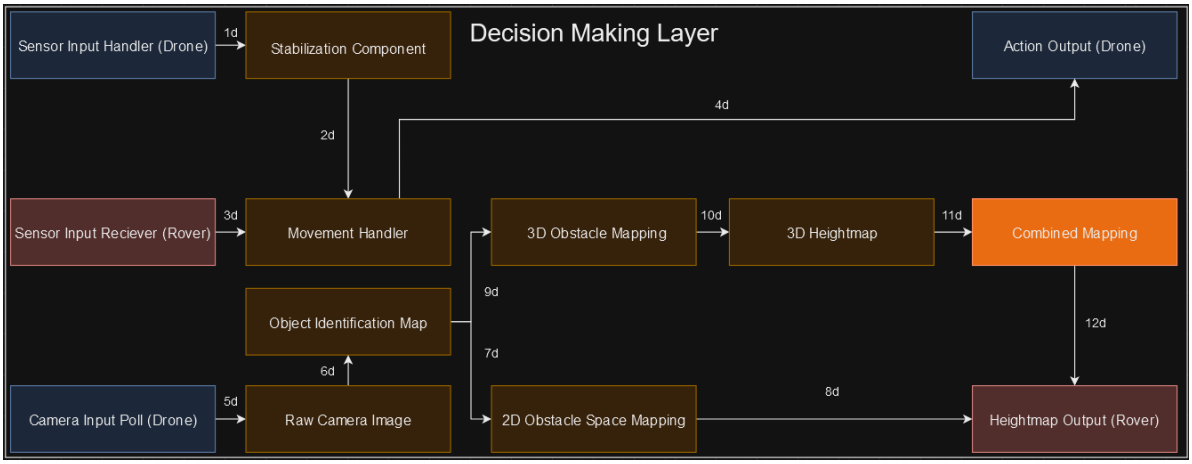


Figure 20: Example subsystem description diagram

6.12.1 Assumptions

Same assumptions as 6.10.1

6.12.2 Responsibilities

The Combined mapping will use either A) a special value in the original value in the **3D Heightmap** or B) a calculated value in the **3D Heightmap** to represent the obstacles given in the **3D Obstacle Mapping** stage. This is to ensure the final mapping is able to efficiently transferred with as little data as possible, and that reliable connection is maintained

6.12.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#11d	Adds heightmap info	Image Buffered-Obstacle-Space Rover-Location	Heightmap Buffered-Obstacle-Space Rover-Location
#12d	Combine map info	Heightmap Buffered-Obstacle-Space Rover-Location	Heightmap Rover-Location

6.13 Heightmap Output

handles sending the map data to the Rover.

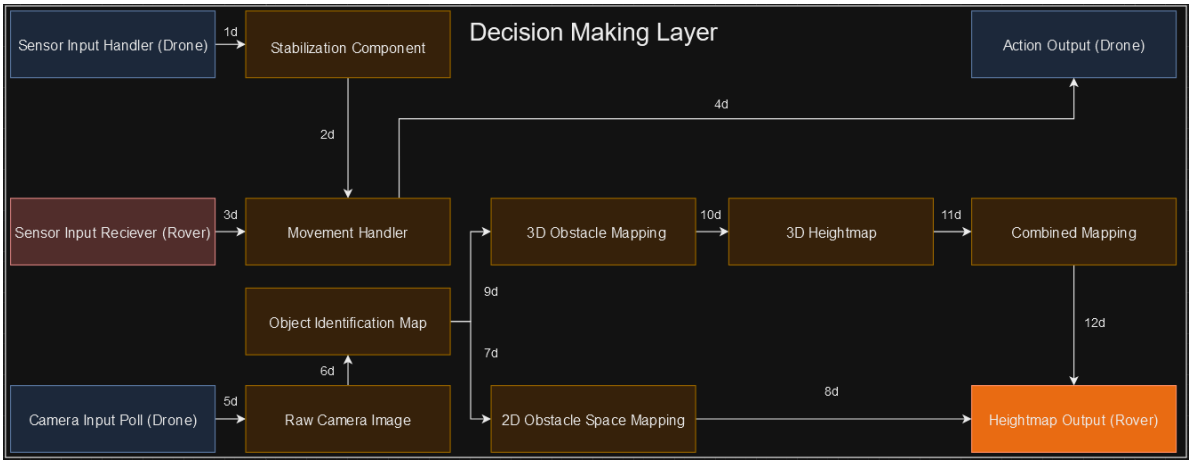


Figure 21: Example subsystem description diagram

6.13.1 Assumptions

Same assumptions as 6.10.1 **OR** Same assumptions as 6.6.1

6.13.2 Responsibilities

The heightmap Output is responsible for sending the final map info to the rover. This may be either a 2D or 3D mapping of the environment.

6.13.3 Subsystem Interfaces

ID	Description	Inputs	Outputs
#8d	Sends final map	Obstacle-Highlighted-Field Rover-Location	Buffered-Obstacle-Space Rover-Location
#12d	Combine map info	Heightmap Buffered-Obstacle-Space Rover-Location	Heightmap Rover-Location

References