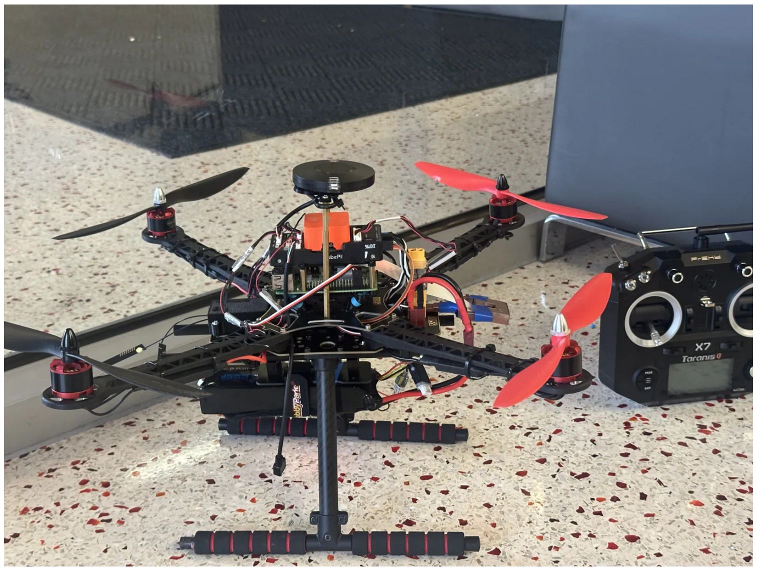
Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2025



DR Rescue
Aerial Rescue Mapping Drone

Omar Elsaghir
Justin Barrett
David Duong
Anaf Mahbub

Revision History

Revision	Date	Author(s)	Description
0.1	2.25.2025	JB	document creation
0.2	2.28.2025	JB	Section 5 draft / Change-log fix
0.3	2.25.2025	DD	Section 1-4 draft

Contents

List of Figures

List of Tables

1 Introduction

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 Raspberry Pi data will be passed to the rover, along with commands to determine the path to follow. The figure below presents a high-level architectural view that identifies the primary components and interactions between layers.

2 System Overview

This section provides a more detailed breakdown of the layer abstraction, illustrating the interactions between the rover, drone, and Raspberry Pi. Each of these three layers consists of multiple subsystems, with arrows indicating the flow of data and interactions between them.

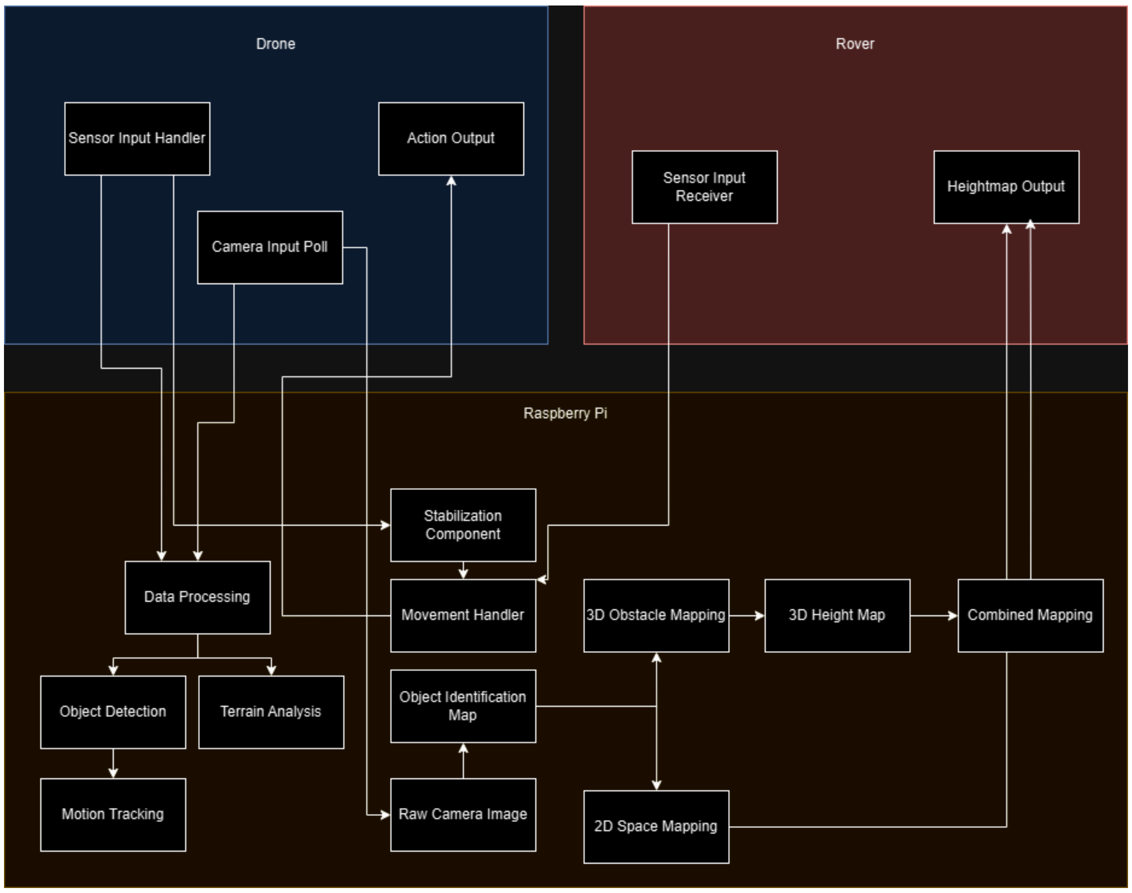


Figure 1: System Architecture

3 Perception Layer Subsystems

In this section, the layer is described in terms of hardware and software design. Specific implementation details, such as hardware components, programming languages, software dependencies, operating systems, etc. should be discussed. Any unnecessary items can be omitted (for example, a pure software module without any specific hardware should not include a hardware subsection). The organization, titles, and content of the following sections can be modified as necessary for the project.

3.1 Layer Hardware

The drone hardware consists of four wings that have batteries attached to them. A GPS sensor is connected to the Cube Orange dock. A Cube Orange chip is used to control the wings and other devices that can be controlled by the Cube Orange chip.

3.2 Layer Operating System

Windows will be used with the Raspberry Pi software to run it.

3.3 Layer Software Dependencies

Coming soon

3.4 Sensor Input Handling

The sensor input handler will hand over all the needed sensors and the drone will also receive data from the sensors. It will move throughout the system and send directions to the Raspberry Pi so that it can be sent to the rover to follow the directions.

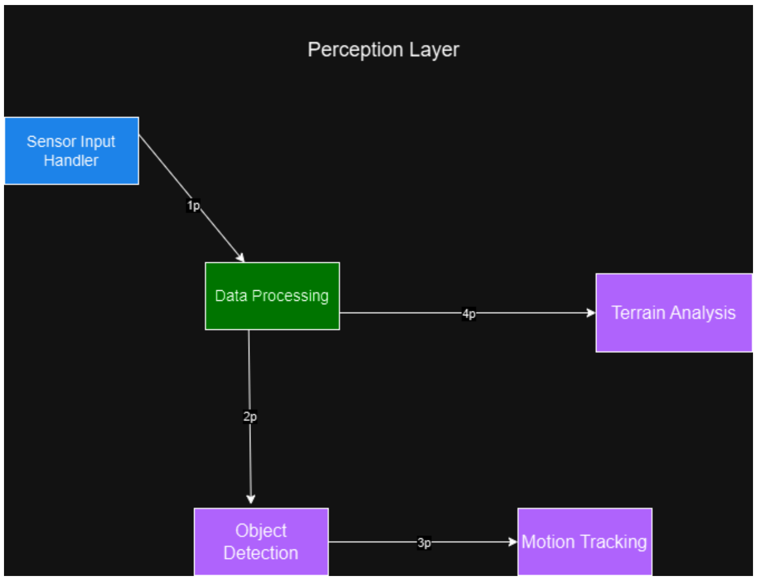


Figure 2: Perception Layer diagram

3.4.1 Subsystem Hardware

Raspberry Pi will be used in conjunction with Cube Orange. The Raspberry Pi will act as a server to receive and send data.

3.4.2 Subsystem Operating System

We used a Taranis QX7 Transmitter controller to control the drone without using the laptop.

3.4.3 Subsystem Software Dependencies

Mission planner will be used on a Windows laptop to control the drone and the Raspberry Pi.

3.4.4 Subsystem Programming Languages

Python

3.4.5 Subsystem Data Structures

A USB will be connected to a Raspberry Pi coming from a laptop to calculate a pathway to send to the rover.

3.4.6 Subsystem Data Processing

Coming soon

4 Rover Layer Subsystems

In this section, the layer is described in terms of the hardware and software design. Specific implementation details, such as hardware components, programming languages, software dependencies, operating systems, etc. should be discussed. Any unnecessary items can be omitted (for example, a pure software module without any specific hardware should not include a hardware subsection). The organization, titles, and content of the sections below can be modified as necessary for the project.

4.1 Layer Hardware

Rover team discussed

4.2 Layer Operating System

Rover team discussed

4.3 Layer Software Dependencies

Raspberry Pi with Windows

4.4 Subsystem 1

Describe at a high level the purpose and basic design of this subsystem. Is it a piece of hardware, a class, a web service, or something else? Note that each of the subsystem items below are meant to be specific to that subsystem and not a repeat of anything discussed above for the overall layer.

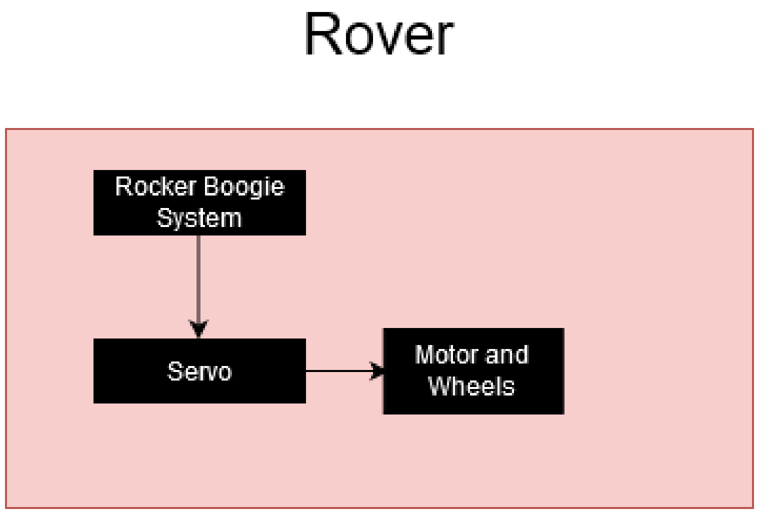


Figure 3: Rover subsystem diagram

4.4.1 Subsystem Hardware

Rover team discussed

4.4.2 Subsystem Operating System

Rover team discussed

4.4.3 Subsystem Software Dependencies

Raspberry Pi with Windows

4.4.4 Subsystem Programming Languages

Written in C

4.4.5 Subsystem Data Structures

Raspberry Pi with Windows

4.4.6 Subsystem Data Processing

Rover team discussed

5 Decision Making Layer Subsystems

Below is a detailed explanation of each component as well as its dependencies

5.1 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.

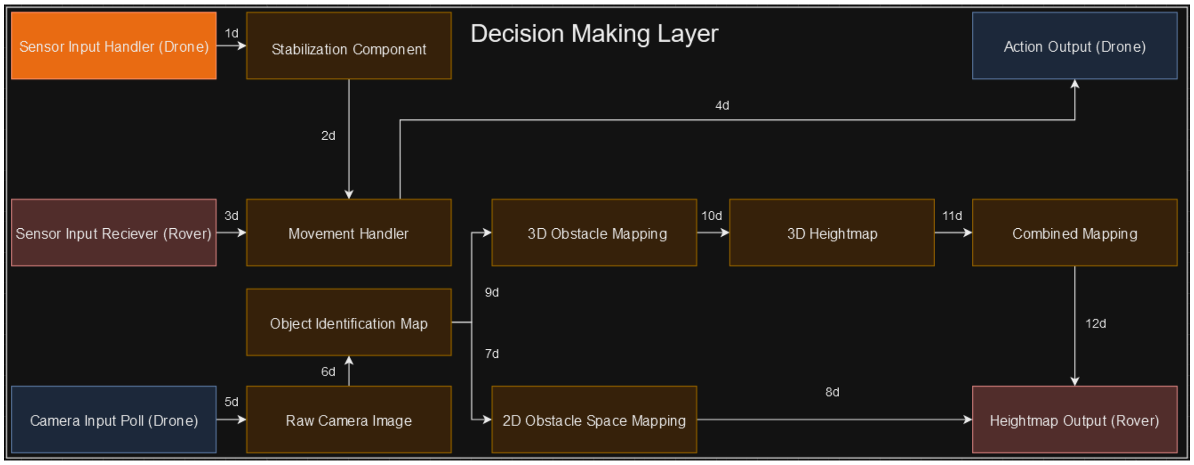


Figure 4: Sensor Input Handler highlighted in bright orange

5.1.1 Subsystem Hardware

We used a Taranis QX7 Transmitter to control the drone.

5.1.2 Subsystem Operating System

Mission planner will be used on a Windows laptop.

5.1.3 Subsystem Software Dependencies

Cube Orange API

5.1.4 Subsystem Programming Languages

Python

5.1.5 Subsystem Data Structures

Cube Orange

5.1.6 Subsystem Data Processing

Raspberry Pi

5.2 Stabilization Component

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.

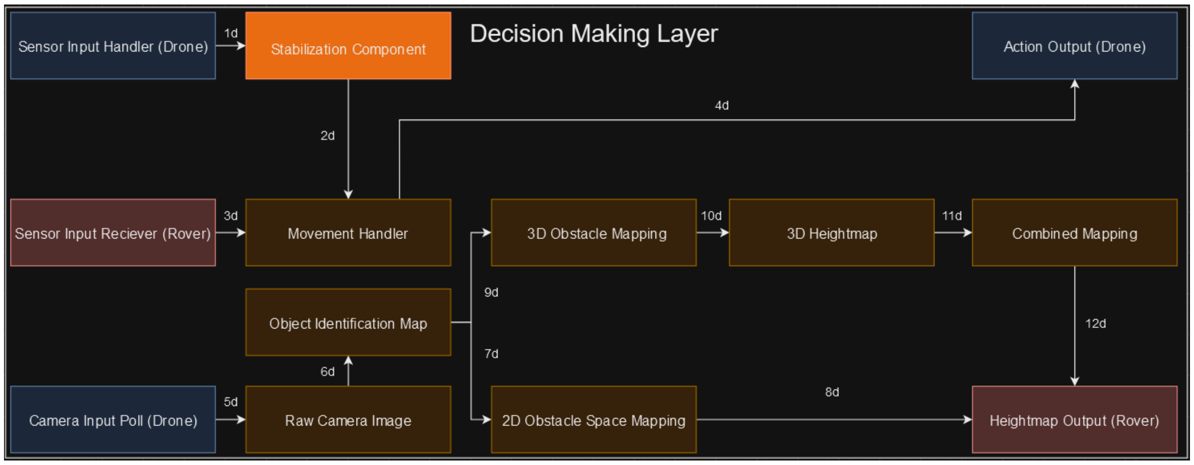


Figure 5: Stabilization Component highlighted in bright orange

5.2.1 Subsystem Hardware

handled entirely by cube orange

5.3 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

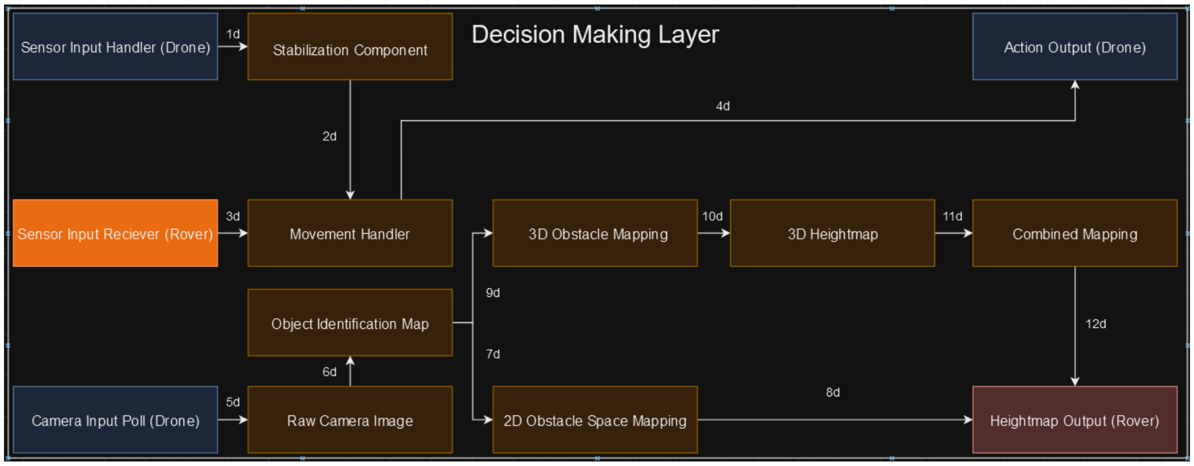


Figure 6: Sensor Input Receiver highlighted in bright orange

5.3.1 Subsystem Hardware

The X8R Receiver will be connected to the Cube Orange to to receive the signal from the controller.

5.3.2 Subsystem Operating System

Python

5.3.3 Subsystem Software Dependencies

TCP / UDP or USB

5.3.4 Subsystem Programming Languages

Python

5.3.5 Subsystem Data Structures

undecided

5.3.6 Subsystem Data Processing

undecided

5.4 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.

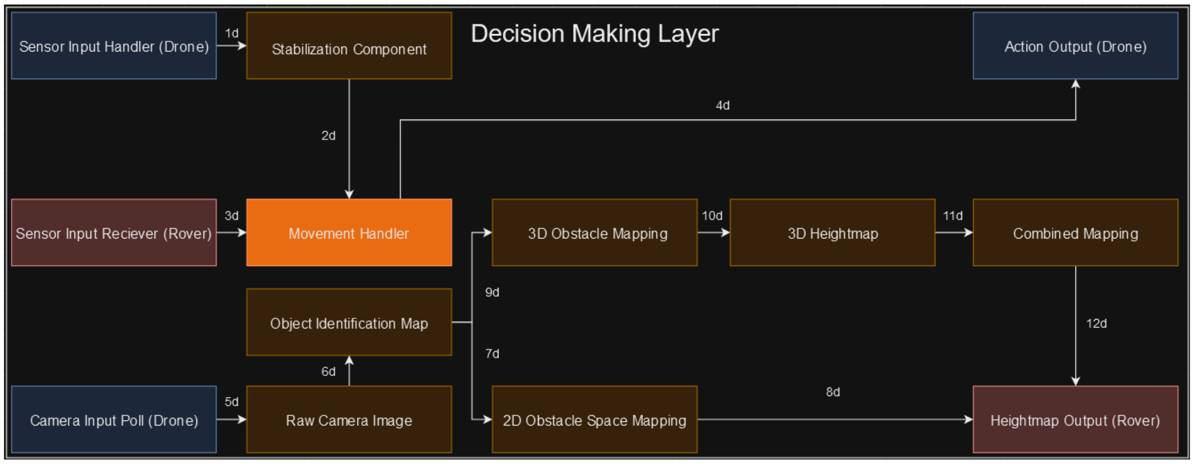


Figure 7: Movement Handler highlighted in bright orange

5.4.1 Subsystem Hardware

handled entirely by cube orange (given through GPS coordinates)

5.5 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

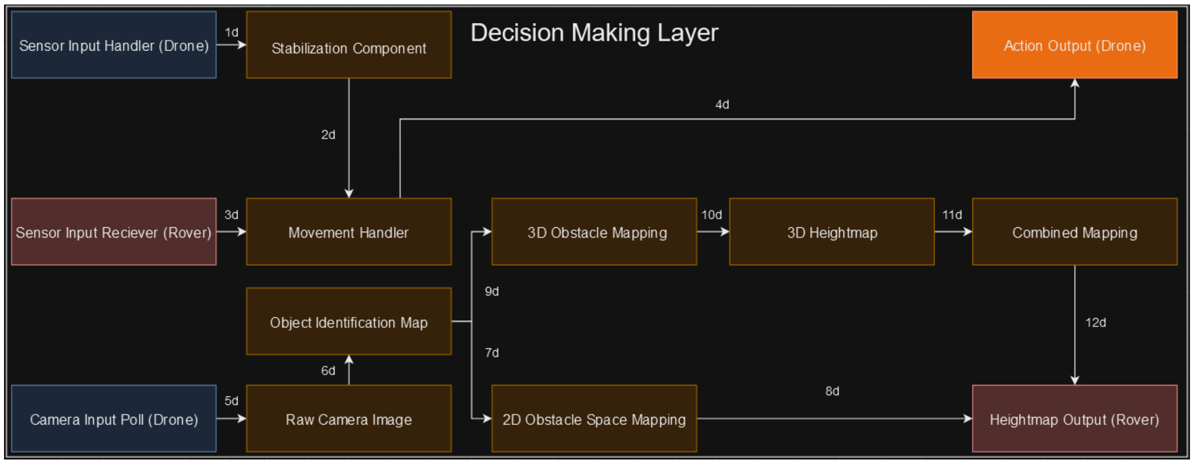


Figure 8: Action Output highlighted in bright orange

5.5.1 Subsystem Hardware

handled entirely by cube orange

5.6 Camera Input Poll

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

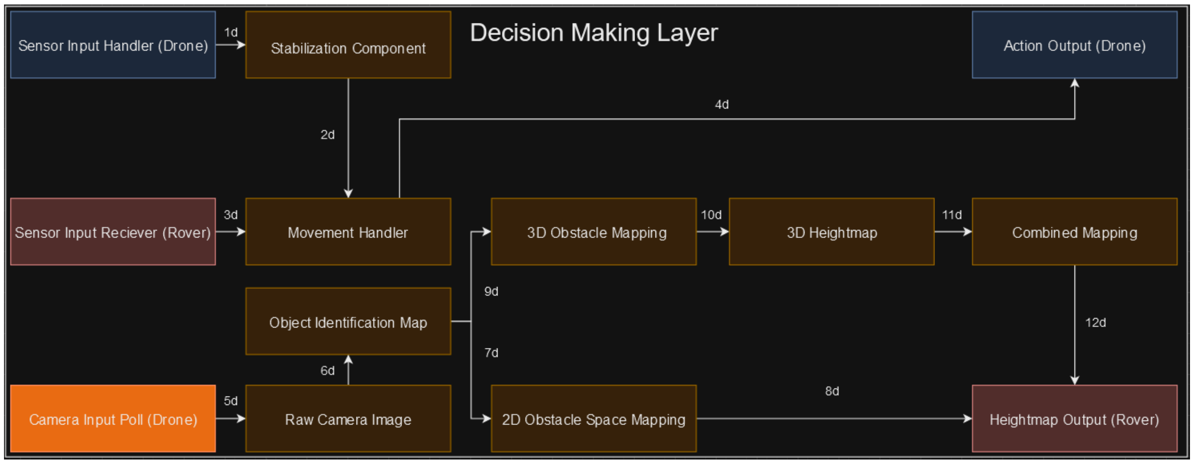


Figure 9: Camera Input Poll highlighted in bright orange

5.6.1 Subsystem Hardware

Raspberry Pi 3 camera

5.6.2 Subsystem Operating System

Raspberry Pi

5.6.3 Subsystem Software Dependencies

Python

5.6.4 Subsystem Programming Languages

Python / C++

5.6.5 Subsystem Data Structures

undecided

5.6.6 Subsystem Data Processing

undecided

5.7 Object Identification Map

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

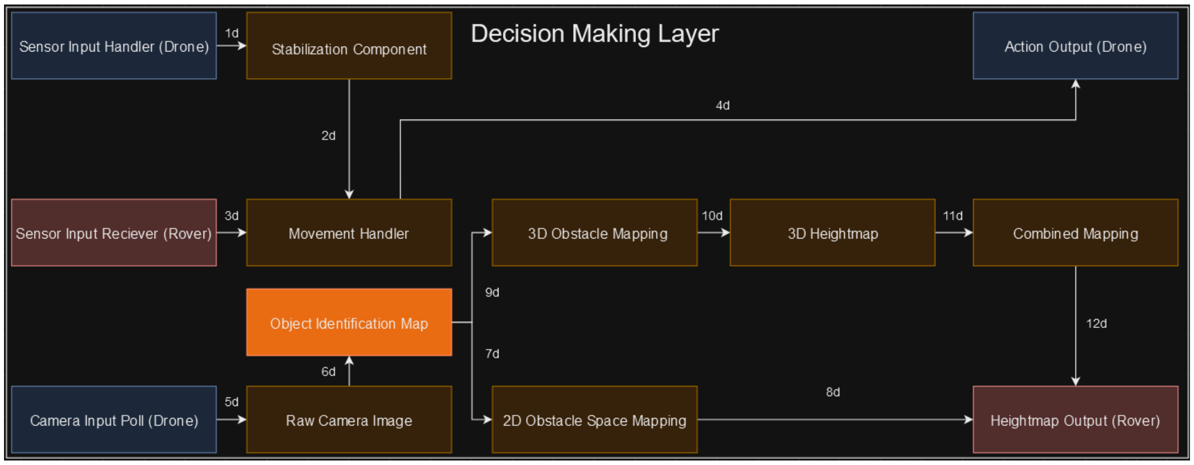


Figure 10: Object Identification Map highlighted in bright orange

5.7.1 Subsystem Hardware

N/A, (not yet known, likely raspberry pi)

5.7.2 Subsystem Operating System

N/A, (not yet known, likely raspberry pi os)

5.7.3 Subsystem Software Dependencies

none

5.7.4 Subsystem Programming Languages

Python / C++

5.7.5 Subsystem Data Structures

rgb map

5.7.6 Subsystem Data Processing

Color difference detection using a kernel

5.8 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.

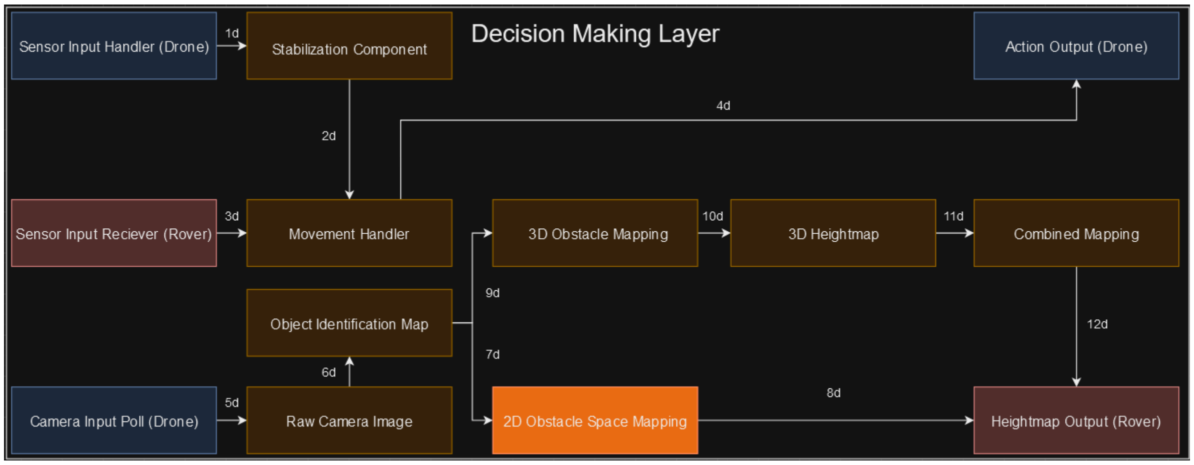


Figure 11: 2D Obstacle Space Mapping highlighted in bright orange

5.8.1 Subsystem Hardware

N/A, (not yet known, likely raspberry pi)

5.8.2 Subsystem Operating System

N/A, (not yet known, likely raspberry pi os)

5.8.3 Subsystem Software Dependencies

none

5.8.4 Subsystem Programming Languages

Python / C++

5.8.5 Subsystem Data Structures

binary map

5.8.6 Subsystem Data Processing

binary encoded image with obstacles represented as 0's and free space as 1's or vice versa

5.9 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.

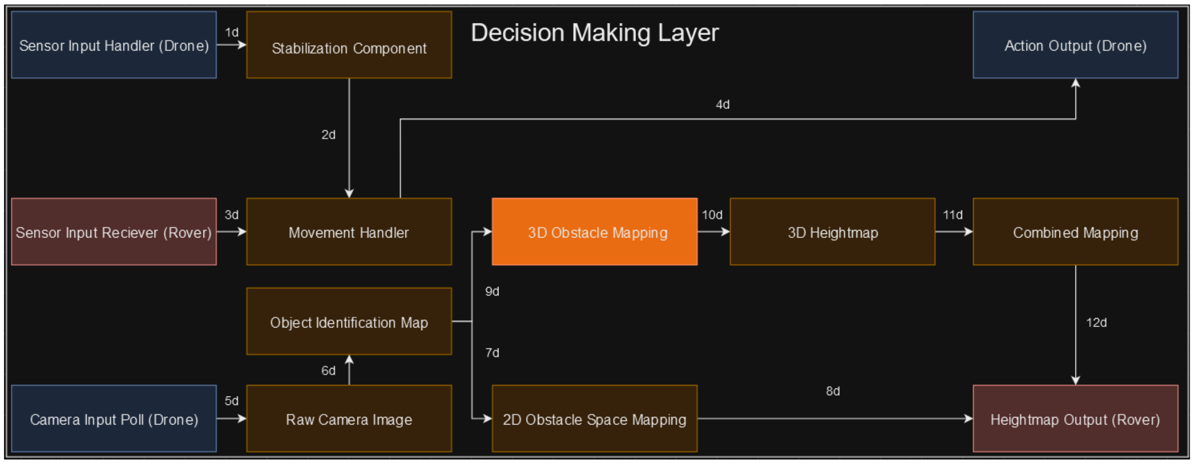


Figure 12: 3D Obstacle Mapping highlighted in bright orange

5.9.1 Other Details

will not be implemented / removed requirement

5.10 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.

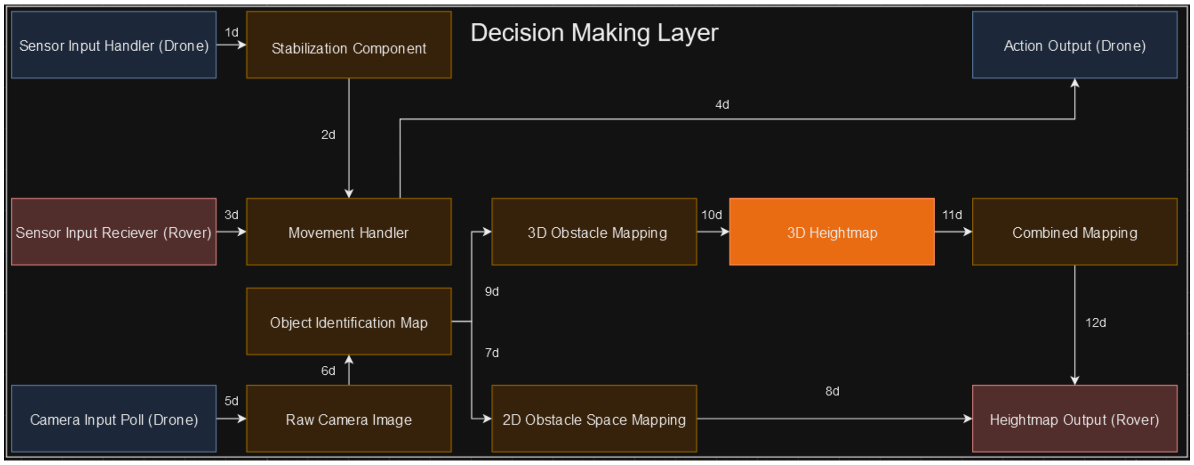


Figure 13: Example subsystem description diagram

5.10.1 Other Details

will not be implemented / removed requirement

5.11 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

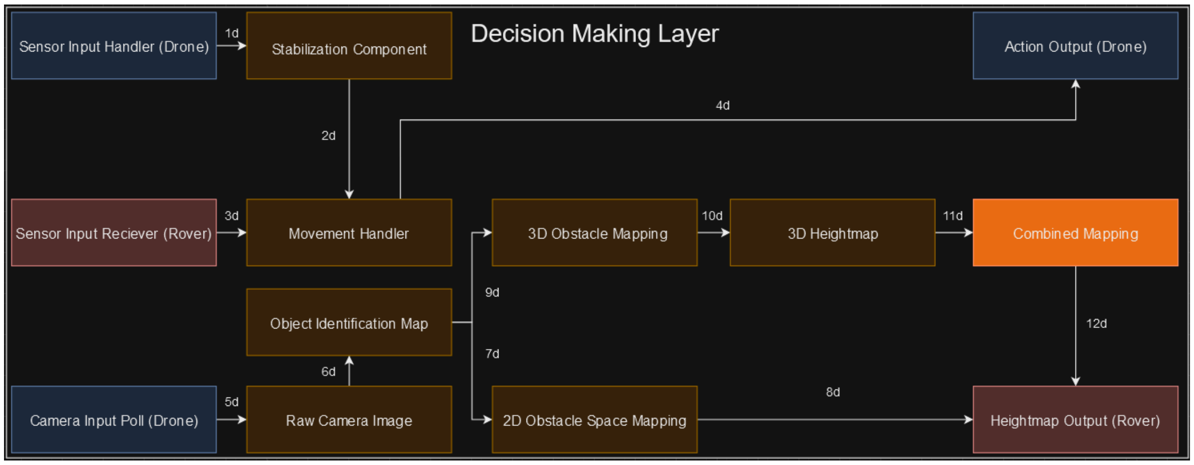


Figure 14: Example subsystem description diagram

5.11.1 Other Details

will not be implemented / removed requirement

5.12 Heightmap Output

handles sending the map data to the Rover.

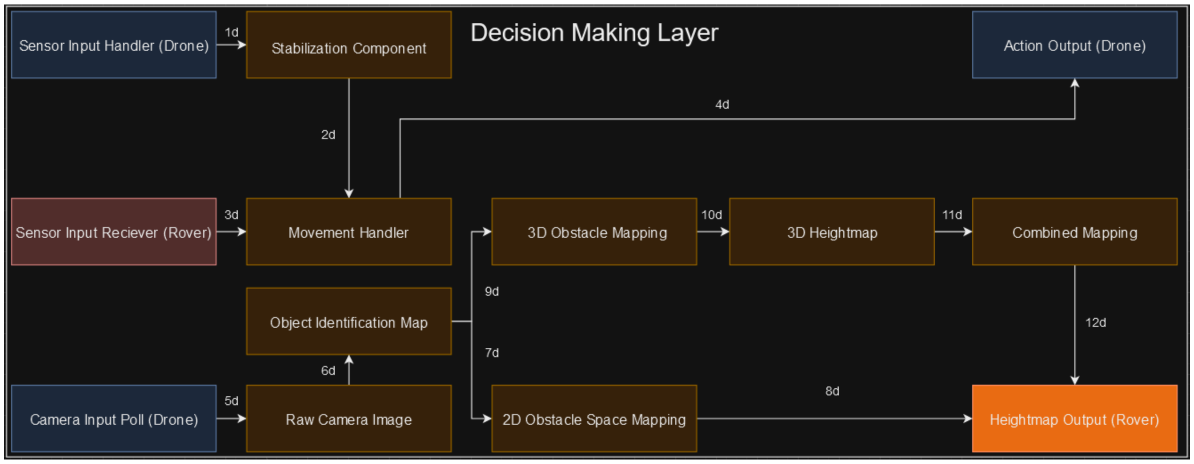


Figure 15: Example subsystem description diagram

5.12.1 Subsystem Hardware

N/A, (not yet known, likely raspberry pi)

5.12.2 Subsystem Operating System

N/A, (not yet known, likely raspberry pi os)

5.12.3 Subsystem Software Dependencies

will require TCP / UDB or other means of data-transfer

5.12.4 Subsystem Programming Languages

Python / C++

5.12.5 Subsystem Data Structures

input buffer

5.12.6 Subsystem Data Processing

input buffer will be sent and processed by the rover team

6 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

References