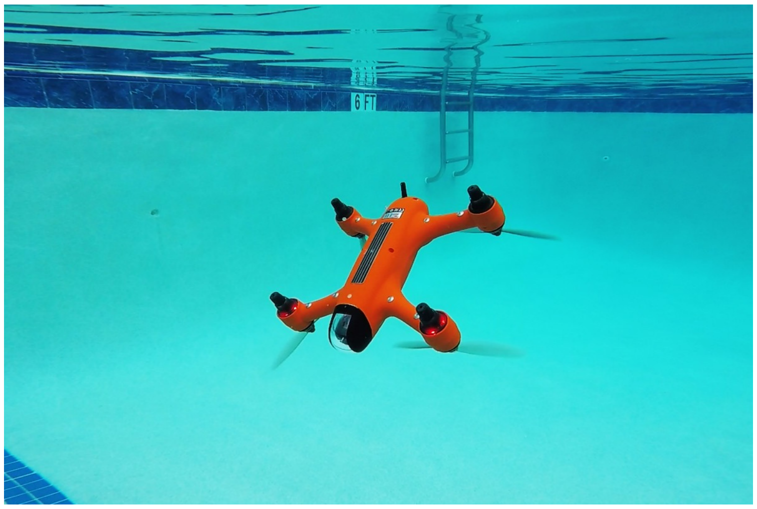
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
THE UNIVERSITY OF TEXAS AT ARLINGTON

DETAILED DESIGN SPECIFICATION
CSE 4317: SENIOR DESIGN II
SPRING 2022



THE DROWNING ROBOTS
OCEAN DEBRIS CLEANUP BOT

HUNTER REDHEAD
APAR POKHREL
JOANNE MATHEW
SEAN WALTER

Revision History

Revision	Date	Author(s)	Description
1.0	3.13.2022	HR, AP, JM, SW	first draft

Contents

List of Figures

List of Tables

1 Introduction

The final outcome of the project is an underwater robot capable of performing a set of required tasks designated in the 2022 IEEE Robotics Competition. This section provides a high-level overview of the desired functionalities, the methods of use, and the targeted audience.

The device shall perform a series of tasks which includes the following. First, the robot shall maneuver through the underwater rings to pick up a block, transport it back through the same rings, and deposit on an underwater shelf. Second, the robot shall push a button to release tennis balls from a box, and then scoop as many tennis balls as it can from the water's surface and drop it in a container. The device will complete these tasks by employing the use of a multi-functional component handler in conjunction with a user-controlled fine-movement system which combines the use of thrusts and ballast tanks.

The design structure involves four main layers: External Component System, HMI/Controller system, Sensors system, and Movement System, each of which are tasked with different functionalities and equipped with a varied level of capabilities.

2 System Overview

The Human Interface/Controller layer allows the user to interact with the mechanical design and the functionalities of the underwater ROV. Similarly, the Sensor System is responsible for collecting data from the environment around the robot. It will stream data out for the pilot or control unit to use in the movement of the robot. The Movement System consists of the parts involved for movement. For us, that will be the thruster and ballast tank subsystems. These subsystems will receive input from the Human Machine Interface/Controller which will control the power levels on the thrusters and pumps regulating the movement of the robot. The External Component System will be used to perform required tasks for the ROV, such as picking up and transporting the block and retrieving the tennis balls from the water surface.

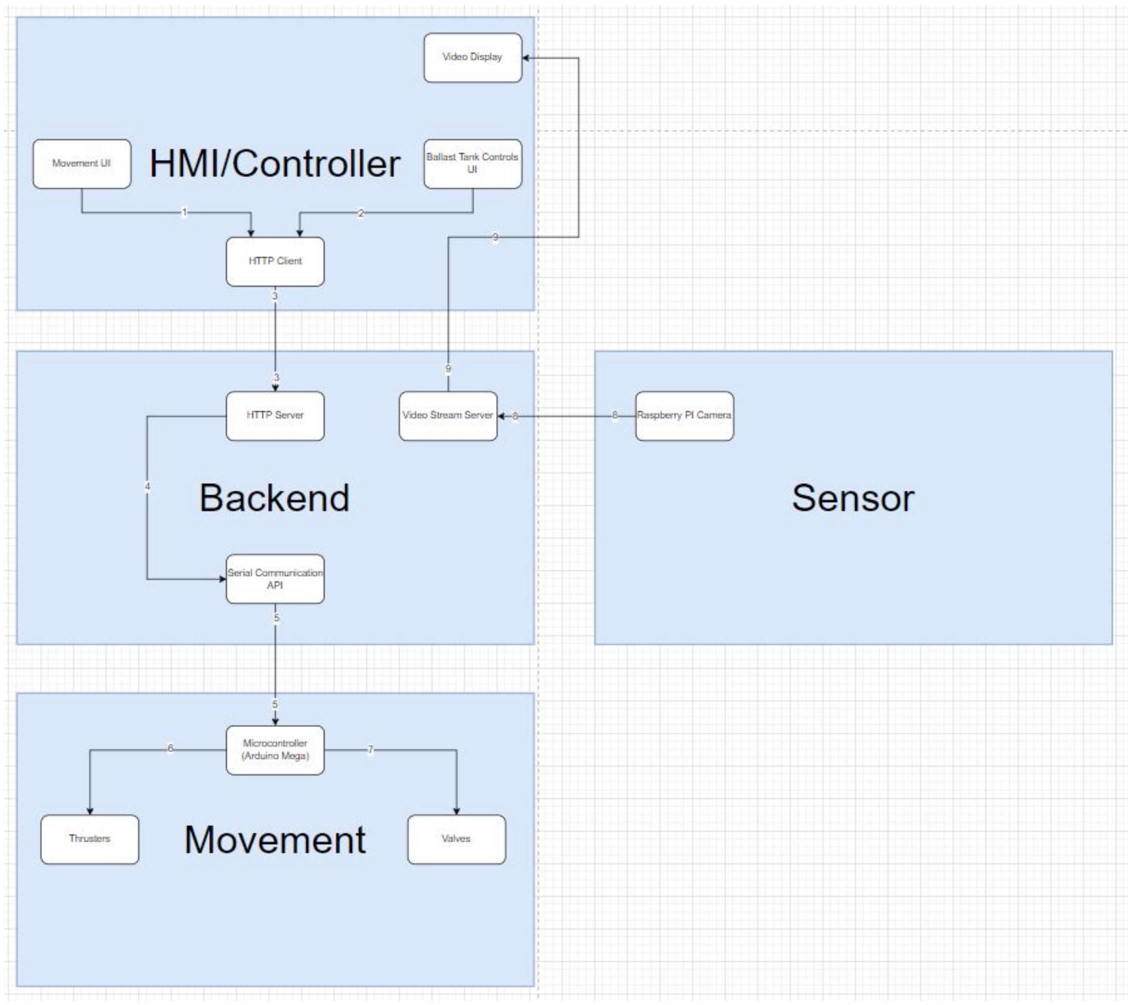


Figure 1: System Architecture

3 Human Interface / Controller

The Human Interface / Controller layer is the layer responsible for taking user input, delivering it to the robot and using that data to interact with the hardware. It is also responsible for feeding data such as video data from the robot to the user. It uses a React JS webpage.

3.1 Layer Operating System

Any OS (Windows, Linux, or Mac OS) can be used for the React JS web app so long as it has access to a internet browser.

3.2 Layer Software Dependencies

- React JS (v17.0.2): Framework for Controller
- Axios (v0.25.0): HTTP API for the Controller to send HTTP messages

3.3 Movement Controller

The UI controls that are responsible for taking the user input for movement commands to the robot.

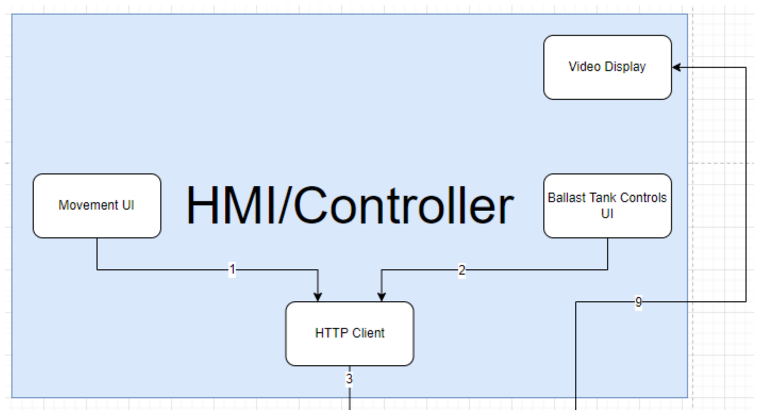


Figure 2: HMI Subsystem Diagram

3.3.1 Subsystem Programming Languages

Javascript to capture the events of key presses from the keyboard.

3.3.2 Subsystem Data Structures

HTTP POST Message packet with two values in the body. An action, a string used for informing what kind of action it is, and a direction, a character that represents the direction the robot is to move in.
Movement Actions:

- "Start" = Turn on thrusters in x direction
- "Stop" = Turn off thrusters in x direction

Movement Directions:

- w = forward
- s = backward
- a = left
- d = right

3.4 Depth Controller

The UI buttons and checkboxes the user uses to control which tanks are being filled and whether they are being emptied or filled.

3.4.1 Subsystem Programming Languages

HTML, CSS, and Javascript to handle the implementation of the buttons for the web app.

3.4.2 Subsystem Data Structures

HTTP POST Message packet with two values in the body. An action, a string used for informing what kind of action it is (in this case action = "Fill" or "StopFill"), a selection of tanks, a string with each tank being targeted, and a boolean value representing whether the tanks are being emptied or filled with water.

Fill/StopFill Actions:

- "Fill" = Fill ballast tanks (either with water or air (emptying))

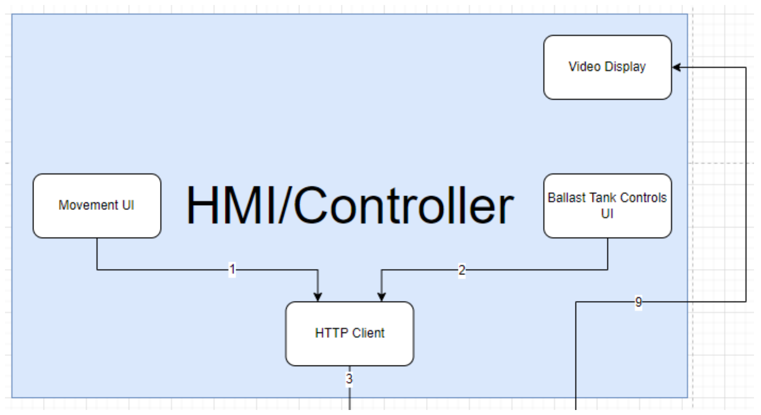


Figure 3: HMI Subsystem Diagram

- "StopFill" = Stop filling ballast tanks (either with water or air (emptying))

Fill/StopFill Data:

- (False, 'a') = Empty ballast tank A
- (True, 'a') = Fill ballast tank A
- (False, 'b') = Empty ballast tank B
- (True, 'b') = Fill ballast tank B
- (False, 'c') = Empty ballast tank C
- (True, 'c') = Fill ballast tank C
- (False, 'd') = Empty ballast tank D
- (True, 'd') = Fill ballast tank D
- (False, 'ab') = Empty ballast tanks A and B
- (True, 'ab') = Fill ballast tanks A and B
- (False, 'cd') = Empty ballast tanks C and D
- (True, 'cd') = Fill ballast tanks C and D
- (False, 'all') = Empty all ballast tanks
- (True, 'all') = Fill all ballast tanks

3.5 Video Feed

Video being streamed from the camera on the robot to the web app for the user to see what the robot is doing.

3.5.1 Subsystem Hardware

Raspberry-Pi camera

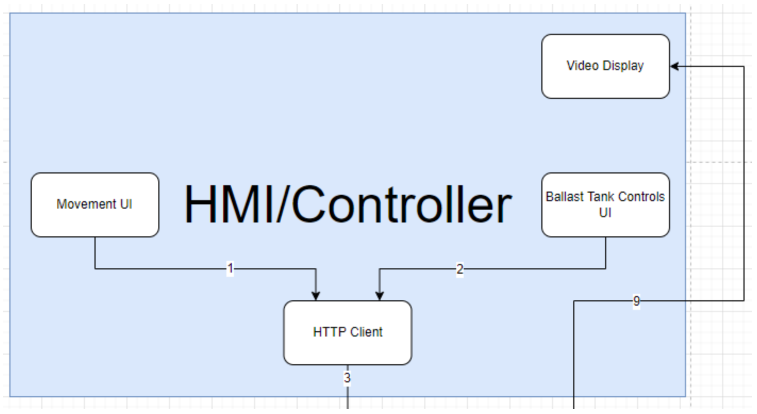


Figure 4: HMI Subsystem Diagram

3.5.2 Subsystem Software Dependencies

Raspberry-Pi legacy camera module support.

3.5.3 Subsystem Programming Languages

HTML and CSS for the displaying the video block on the web app. Python for setting up a server that can stream the images to a web browser.

3.5.4 Subsystem Data Processing

A series of images will be constantly streamed to the video display for video feed.

3.6 HTTP Client

HTTP Requests will be sent from this to the backend to send commands from the user controlling the robot.

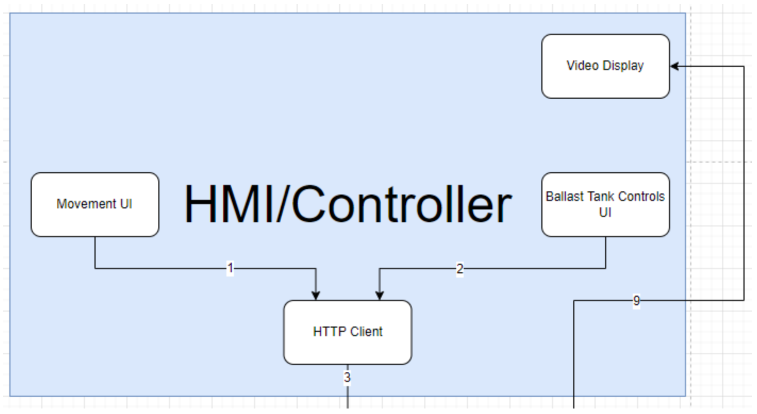


Figure 5: HMI Subsystem Diagram

3.6.1 Subsystem Software Dependencies

Axios (v0.25.0): HTTP API for the Controller to send HTTP messages

3.6.2 Subsystem Programming Languages

HTML and CSS for the displaying the video block on the web app. Javascript for handling the HTTP messages.

3.6.3 Subsystem Data Processing

A series of images will be constantly streamed to the video display for video feed.

4 SENSOR SYSTEM

The sensor system is composed of the camera system. The camera system is responsible for providing the user with live video from both the front and back of the robot.

4.1 CAMERA SYSTEM

The camera system is responsible for providing the user with live video feeds from both the front and the back of the ROV.

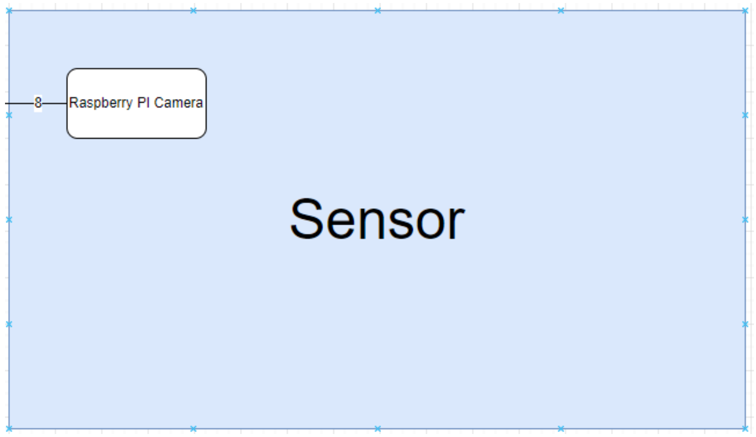


Figure 6: Sensor subsystem diagram

4.1.1 SUBSYSTEM HARDWARE

The system will consist of a camera chip that outputs raw data which can be processed to provide a live video feed.

4.1.2 SUBSYSTEM SOFTWARE DEPENDENCIES

Raspberry Pi legacy camera module support

4.1.3 SUBSYSTEM PROGRAMMING LANGUAGES

Python

4.1.4 SUBSYSTEM DATA STRUCTURES

Images will be captured as JPEGs.

4.1.5 SUBSYSTEM DATA PROCESSING

The images will be constantly streamed through to the backend video streaming server and then to the HMI/Controller.

5 Movement System

We will be building a ballast tank system with the use of thrusters. The ballast tank system will include four varying valves and a pump with PVC pipes acting as the tanks. For the thrusters we will have one on either side to provide forward movement. We will also be using an Arduino board and the on-board microcontroller.

5.1 Ballast Tank System

The ballast tank system is responsible for submerging, surfacing, and stabilizing the underwater movements of the ROV.

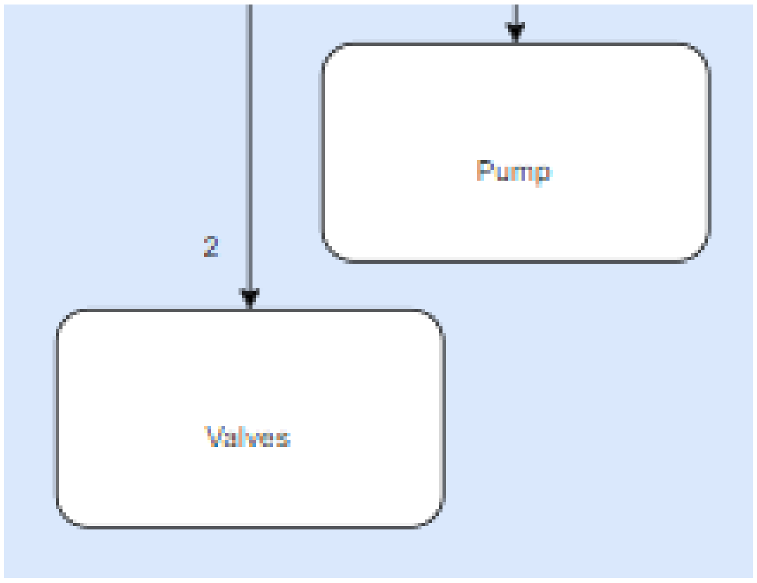


Figure 7: Both pump and valves control the system

5.1.1 Subsystem Hardware

- One pump ()
- Two 1 by 2 valves (uxcell Plastic Water Electric Solenoid Valve)
- Two 1 by 4 valves (uxcell Plastic Water Electric Solenoid Valve)
- Piping ()
- Arduino MEGA 2560

5.1.2 Subsystem Software Dependencies

The software dependencies for this subsystem have not yet been determined. This section will be updated in the future.

5.1.3 Subsystem Programming Languages

Code for this subsystem will either be written in C or in Arduino code.

5.1.4 Subsystem Data Structures

Data structures for this subsystem have not yet been developed. This section will be updated in the future.

5.1.5 Subsystem Data Processing

Data processing methods for this subsystem have not yet been developed. This section will be updated in the future.

5.2 Thrusters System

This system is the four thrusters that control the forwards, backwards and turning movements of our ROV. It is comprised of both clockwise and counter clockwise thrusters.

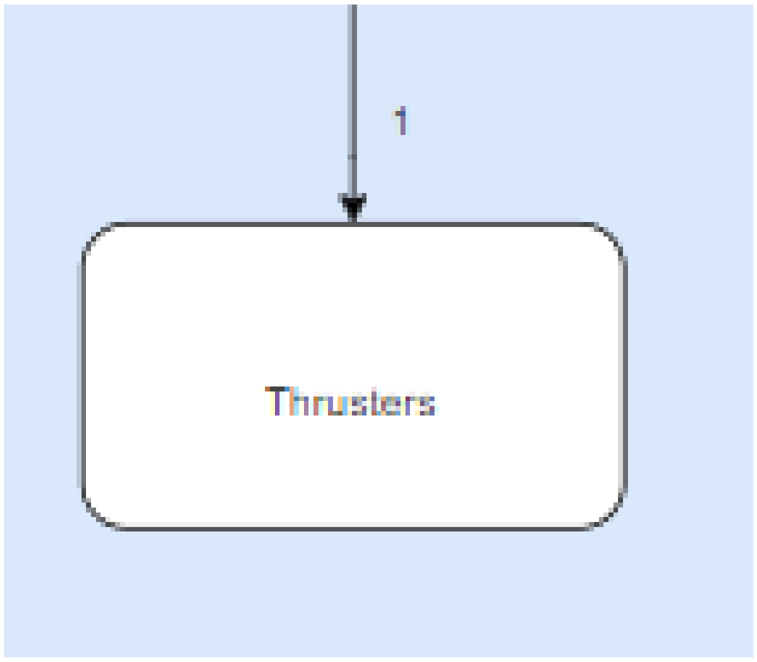


Figure 8: This system mostly just has thrusters

5.2.1 Subsystem Hardware

- One counter clockwise thrusters ()
- One clockwise thrusters ()
- Arduino MEGA 2560

5.2.2 Subsystem Software Dependencies

The software dependencies for this subsystem have not yet been determined. This section will be updated in the future.

5.2.3 Subsystem Programming Languages

Code for this subsystem will either be written in C or in Arduino code.

5.2.4 Subsystem Data Structures

Data structures for this subsystem have not yet been developed. This section will be updated in the future.

5.2.5 Subsystem Data Processing

Data processing methods for this subsystem have not yet been developed. This section will be updated in the future.

6 Backend System

The Backend layer is the layer responsible for taking HTTP requests from the controller and then relaying the commands to the microcontroller. It sends commands to the microcontroller through the serial port. The backend also hosts the server for streaming the video feed from the camera to the controller.

6.1 Layer Operating System

The Raspberry Pi 4 is going to use the Raspbian Linux distribution of Linux.

6.2 HTTP Server

The HTTP Server responsible for handling the HTTP POST requests sent by the controller to the backend for controlling the robot.

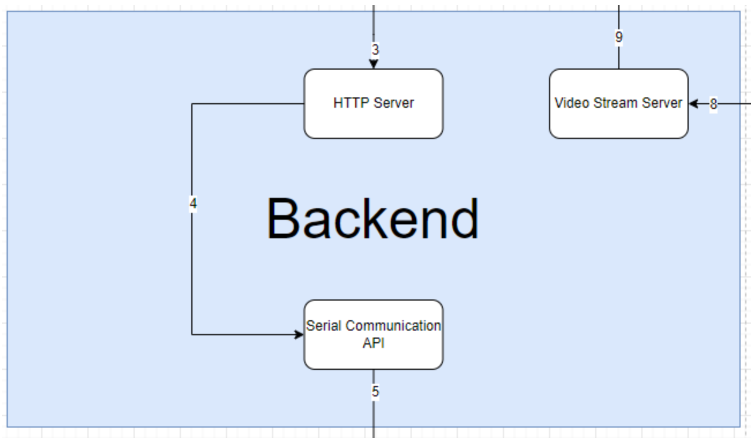


Figure 9: Backend Subsystem Diagram

6.2.1 Subsystem Programming Languages

Python

6.2.2 Subsystem Data Structures

HTTP POST Message packet with two values in the body. An action, a string used for informing what kind of action it is, and a direction, a character that represents the direction the robot is to move in.
Movement Actions:

- "Start" = Turn on thrusters in x direction
- "Stop" = Turn off thrusters in x direction

Movement Directions:

- w = forward
- s = backward
- a = left
- d = right

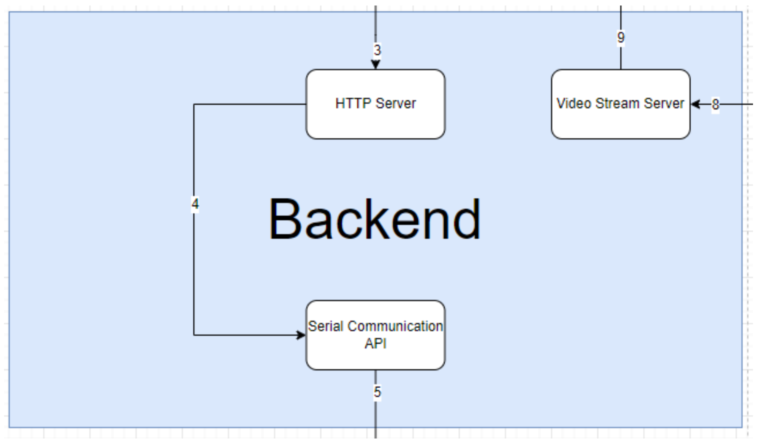


Figure 10: Backend Subsystem Diagram

6.3 Serial Communication API

This is the API that processes the commands sent through the HTTP Server, converts them to a byte-command, and then sends it over to the microcontroller.

6.3.1 Subsystem Programming Languages

Python

6.3.2 Subsystem Data Structures

Converts the POST request data (action, action data) and turns it into a byte command of the following format: 0xAB, where the A digit corresponds to the action and the B digit corresponds to the data for that action.

Actions:

- 0x10 = Start
- 0x20 = Start
- 0x30 = Fill
- 0x40 = StopFill
- 0x50 = Spokes

Start/Stop Data:

- w = forward = 0x00
- s = backward = 0x01
- a = left = 0x02
- d = right = 0x03

Fill/StopFill Data:

- (False, 'a') = Empty ballast tank A = 0x00

- (True, ’a’) = Fill ballast tank A = 0x01
- (False, ’b’) = Empty ballast tank B = 0x02
- (True, ’b’) = Fill ballast tank B = 0x03
- (False, ’c’) = Empty ballast tank C = 0x04
- (True, ’c’) = Fill ballast tank C = 0x05
- (False, ’d’) = Empty ballast tank D = 0x06
- (True, ’d’) = Fill ballast tank D = 0x07
- (False, ’ab’) = Empty ballast tanks A and B = 0x08
- (True, ’ab’) = Fill ballast tanks A and B = 0x09
- (False, ’cd’) = Empty ballast tanks C and D = 0x0A
- (True, ’cd’) = Fill ballast tanks C and D = 0x0B
- (False, ’all’) = Empty all ballast tanks = 0x0C
- (True, ’all’) = Fill all ballast tanks = 0x0D

Spokes Data:

- False = open = 0x00
- True = closed = 0x01

6.4 Video Stream Server

This is the server that hosts the video streaming for the controller. It takes the data from the raspberry pi camera and constantly updates the page with the live footage.

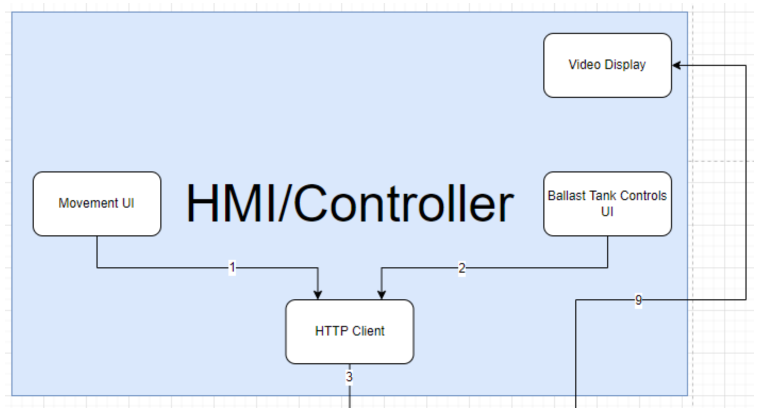


Figure 11: Backend Subsystem Diagram

6.4.1 Subsystem Hardware

Raspberry-Pi camera

6.4.2 Subsystem Software Dependencies

Raspberry-Pi legacy camera module support.

6.4.3 Subsystem Programming Languages

Python for setting up a server that can stream the images to a web browser.

6.4.4 Subsystem Data Processing

A series of images will be constantly streamed to the video display for video feed.

7 Appendix A

This appendix will be updated in the future.

References