Software requirement specification (SRS) document



Approval history

Project name: OOP controlled differential drive mechatronic robot

Date: 21/08/25

Version: 1.1.1

By: Michael Liondis



Revision history

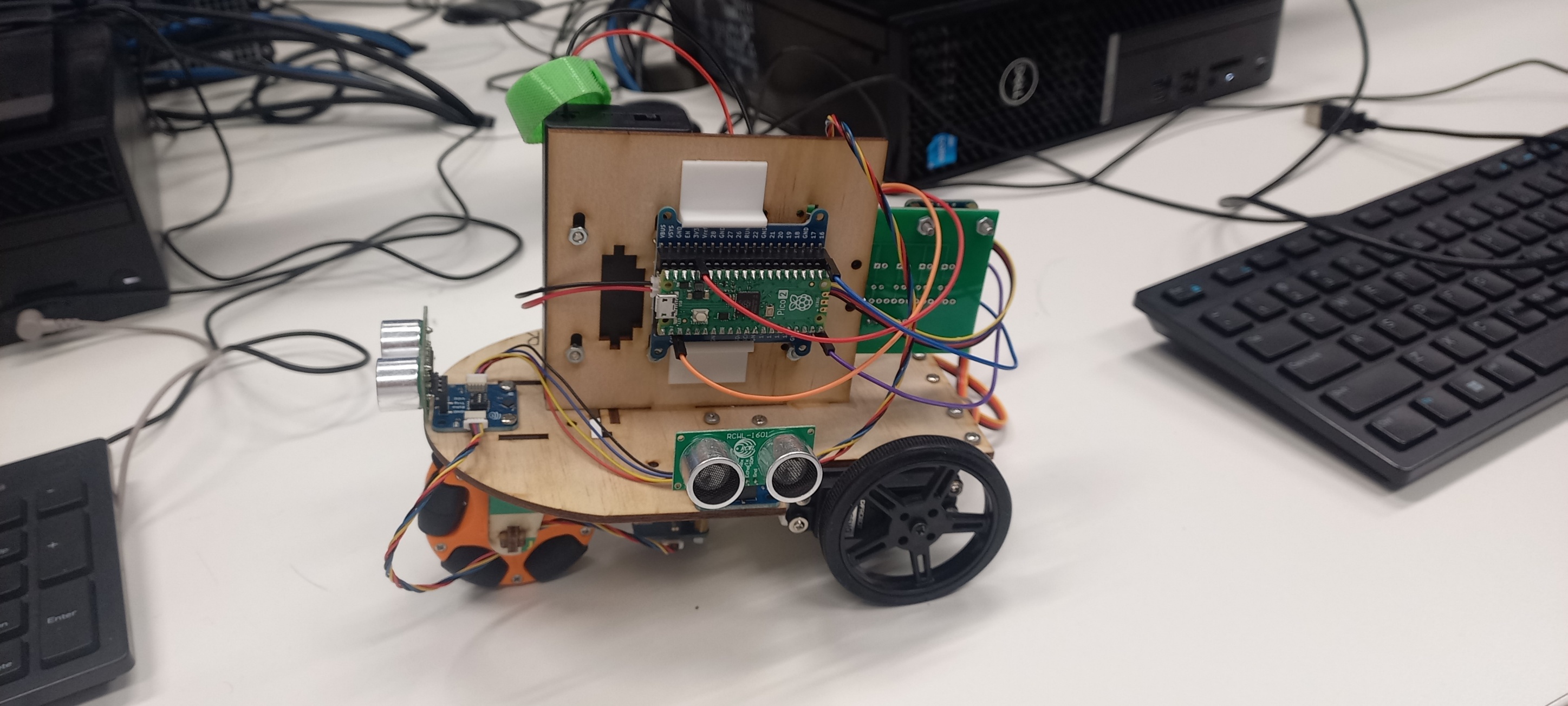
|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Verson description | Date completed |
| 0.1  0.1.1 | Michael Liondis  Michael Liondis | Begin introduction  Finish introduction | 31/07/2025  01/08/2025 |
| 0.2  0.3  0.4 | Michael Liondis  Michael Liondis  Michael Liondis | External interface requirements  Non-functional requirements  Add notif requirements | 02/08/2025  03/08/2025  21/08/2025 |



Review history

|  |  |  |  |
| --- | --- | --- | --- |
| Approving party | Version approved | Signature | Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Final Physical Mechatronic Product**



# Table of contents



1. Introduction
   1. Product scope
   2. Product value
   3. Intended audience
   4. Intended use
   5. General description

2 Functional requirements

1. External interface requirements
   1. User interface requirements
   2. Hardware interface requirements
   3. Software interface requirements
   4. Communication interface requirements
2. Non-functional requirements
   1. Security
   2. Capacity
   3. Compatibility
   4. Reliability
   5. Scalability
   6. Maintainability
   7. Usability
   8. Other non-functional requirements

5 Definitions and acronyms



1

Introduction

Describe the purpose of the document.

* 1. Product scope

List the benefits, objectives, and goals of the product.

The objective of the OOP-controlled differential drive mechatronic robot is to be delivered as a prototype for a maze solving robot, which will follow walls, avoid collisions, and detect victims along the way with a colour sensor. It will use various components, such as a Pico Pi, servos, and ultrasonic and line following sensors, which are prototypical components of a future and larger system that will have the ability to navigate within a warehouse setting, aiding in transportation and emergency response.

* 1. Product value

Describe how the audience will find value in the product.

The product’s value lies in its ability to automize warehouse tasks. Warehouse managers see a practical tool that can aid in the reduction of labor and the improvement of emergency response, and observers benefit from its responsive UI interface.

* 1. Intended audience

Write who the product is intended to serve.

The intended audience for the differential drive mechatronic robot, when fully completed, are warehouse operations managers, industrial automation engineers, or owners in industrial companies. This is due to the fact that it can heavily assist in efficiency improvements around a warehouse space, providing streamline operations and supporting emergency protocols.

* 1. Intended use

Describe how will the intended audience use this product.

The intended use for the robot is to serve as a prototypical steppingstone that navigates a maze by following the side of a wall, preceding a fully functional warehouse assistance robot that can be used by its audience for various activities like transportation and emergency response.

* 1. General description

Give a summary of the functions the software would perform and the features to be included.

The OOP-controlled differential drive mechatronic robot is built to perform tasks like wall following, colour sensing, and UI display, meaning it can autonomously navigate through a maze-like environment by following walls, identify colored tiles that represent victims using colour sensors, and communicate its, or others, current status through a basic UI interface.

# Functional requirements



2

List the design requirements, graphics requirements, operating system requirements, and constraints of the product.

**Navigation & movement**

* The robot shall autonomously navigate through warehouse environments, including aisles, corridors, and designated storage areas.
* The robot shall plan and follow optimal plans while simultaneously avoiding collisions with static and dynamic structures like shelves, pallets, or forklifts.
* The robot shall detect and follow sensor outputs for route guidance.

**Transportation & Handling**

* The robot shall transport small to medium payloads (e.g. boxes, containers, or emergency supplies) to designated locations.
* The robot shall safely pick up, carry, and deliver payloads using a carrier platform or modular attachment
* The robot shall confirm delivery via its UI interface or through connected warehouse management systems

**Emergency Response**

* The robot shall detect and identify any emergency signals such as alarms, hazardous zones, or victims.
* Upon detection of emergency signals, the robot shall notify the user by displaying audio/visuals on its UI display, or via wireless alerts.
* The robot shall assist in evacuation or emergency supply transport by autonomously re-routing to safe zones.

**User interaction & Feedback**

* The robot shall provide real time status updates via the on board OLED screen. (e.g. “Payload Delivered”, or “Emergency Detected”)
* The robot shall allow users to issue commands through a touchscreen interface or voice activation via tablet.
* The robot shall have a manual overdrive in case of emergency, enabling human operators to take control

**Safety & Compliance**

* The robot shall include emergency stop mechanisms
* The robot shall comply with industrial safety standards for autonomous mobile robots.
* The robot shall use a fail-safe to handle critical errors such as sensor failure, low battery, or motor malfunction, by safely alerting nearby users.

A close up of a machine

AI-generated content may be incorrect.**Material Components list**:

* 2x DFrobot DF15RSMG servo motors
* 2x Ultrasonic sensor
* 1x colour sensor
* 1x battery pack
* 2x 3.7-volt batteries
* 1x OLED screen
* 1x fuse
* 3x diode
* A small robot with wires

  AI-generated content may be incorrect.1x de-amplifier/regulator
* 2x Polarized capacitor
* 2x Capacitor
* 1x Raspberry Pi Pico 2
* 2x circular wheels
* 1x omnidirectional wheel
* 1x wooden chassis
* 4x female to male wires
* 4x male to male JST PH 4-pin wires

**Power Supply Calculations**:

Battery Pack (2x3.7 V Batteries) Capacitors (approximate stepdown of a capacitor ≈ 0.5 V)

Voltage in - 7.4 V (charged) in 7.4V – (0.5 x 2) = out 6.4V



Output to servos (Servo range = 5.0 - 7.2V) Regulator/De-Amplifier (Reduces to 5V)

Voltage in = 5V in 6.4V 🡪 out 5V



Within servo range 👍

External interface requirements



3

* 1. User interface requirements

Describe the logic behind the interactions between

the users and the software (screen layouts, style guides, etc).

The UI panel features a main display panel that can provide real-time navigational outputs such as ‘turning left’ or ‘victim detected’, helping users track the robot’s actions. When it promptly displays an output like ‘victim detected’ it may use LED’s to flash alarmingly, letting the user know that It may require space or interacting by asking the user for assistance.

* 1. Hardware interface requirements

List the supported devices the software is intended

to run on, the network requirements, and the communication protocols to be used.

The OOP-controlled differential drive mechatronic robot is built on a Raspberry Pi Pico controller, which operates without built-in network functions and capabilities. All communication between the Pico and other external devices, such as sensors and motors, is done through standard interfaces including I2C and SPI. Data output and user feedback are handled locally through onboard components such as LED’s, an LCD screen, or via connection to a computer.

* 1. Software interface requirements

Include the connections between your product and other software components, including frontend/backend framework, libraries, etc.

The software interface deals with its range of hardware components through python libraries that handle communication with things like sensors and modules such as the ultrasonic and environmental sensor. Secondly, control algorithms like PID are implemented using modules like PID\_Controller.py, while additional libraries such as servo.py support things like motor control.

The system operates entirely on the Raspberry Pico Pi using micro python, meaning no frontend or backend networks. All software interfaces are handled through direct sensor communication, with output options for data monitoring.

* 1. Communication interface requirements

List any requirements for the communication programs your product will use, like emails or embedded forms

The OOP-controlled differential drive mechatronic robot does not require any external communication programs such as email, messaging services, or embedded forms. All communication occurs locally through physical indicators.

# Non-functional requirements



4

* 1. Security

Include any privacy and data protection regulations that should be adhered to.

The system operates offline with no network access, meaning security requirements are limited to preventing unauthorized physical access or other modifications to the code.

* 1. Capacity

Describe the current and future storage needs of your software.

The current storage requirements for this prototypical robot are minimal, consisting of program code and sensor data, although a future implementation of a fully functional industrial warehouse robot would require expanded storage for things like route tracking, safety logs, and operational data.

* 1. Compatibility

List the minimum hardware requirements for your software.

The prototype OOP-controlled differential drive mechatronic robot’s minimum hardware requirements include a Raspberry Pico-Pi microcontroller, two DC or servo motors to make the robot move, colour and distance sensors to sense ‘victims’ and to follow walls, and a power supply.

* 1. Reliability

Calculate what the critical failure time of your product would be under normal usage.

The OOP-controlled differential drive mechatronic robot is designed to operate reliably over extended periods, with potential failures arising in motor damage and wear and battery degradation.

* 1. Scalability

Calculate the highest workloads under which your software will still perform as expected.

Since this robot is a prototype, the software is expected to run under minimal workloads like motor control, sensor inputs, and status updates. As long as tasks remain lightweight and withing the microcontrollers limitations, then it will perform as expected.

* 1. Maintainability

Describe how continuous integration should be used to deploy features and bug fixes quickly.

Continuous Integration can be used to test and validate new code changes like feature additions and bug fixes, ensuring rapid deployment each time they are committed to the repository without introducing any other errors.

* 1. Usability

Describe how easy it should be for end-users to use your software.

The prototypical OOP-controlled differential drive mechatronic robot does not necessarily need an end user for it to run, as its main goal is to navigate through a maze by following a wall. This means once the user turns it on, all they will need to do is watch, making it incredibly easy and convenient to use.

* 1. Other

List any additional non-functional requirements.

**Energy efficiency**: The robot shall minimize power consumption between idle and active states, using things like sleep states etc.

**Noise levels:** The robot shall operate at a noise level not exceeding a distracting decibel level to avoid disturbing workers

**Battery life**: The robot shall operate continuously for at least 8 hours on a single charge, or 1 full working day

A screenshot of a computer

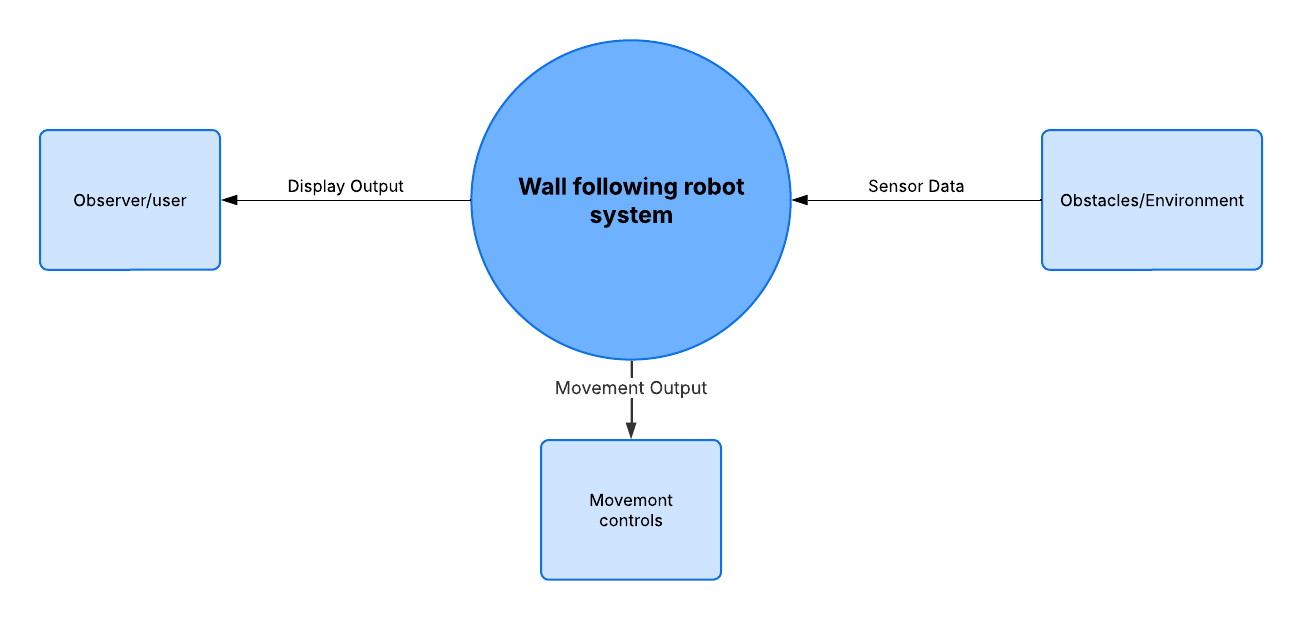
AI-generated content may be incorrect.**Wiring Diagram**

**UML Class Diagram**

A screenshot of a computer

AI-generated content may be incorrect.

**Level 0 Data Flow Diagram**



**Level 1 Data Flow Diagram**

A diagram of a company

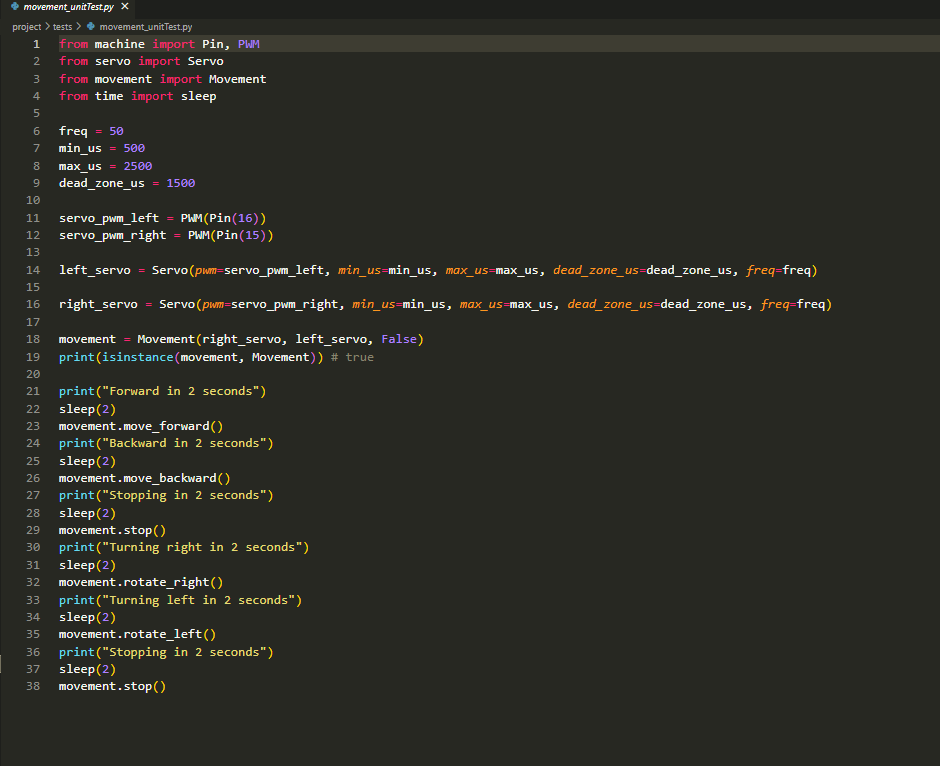
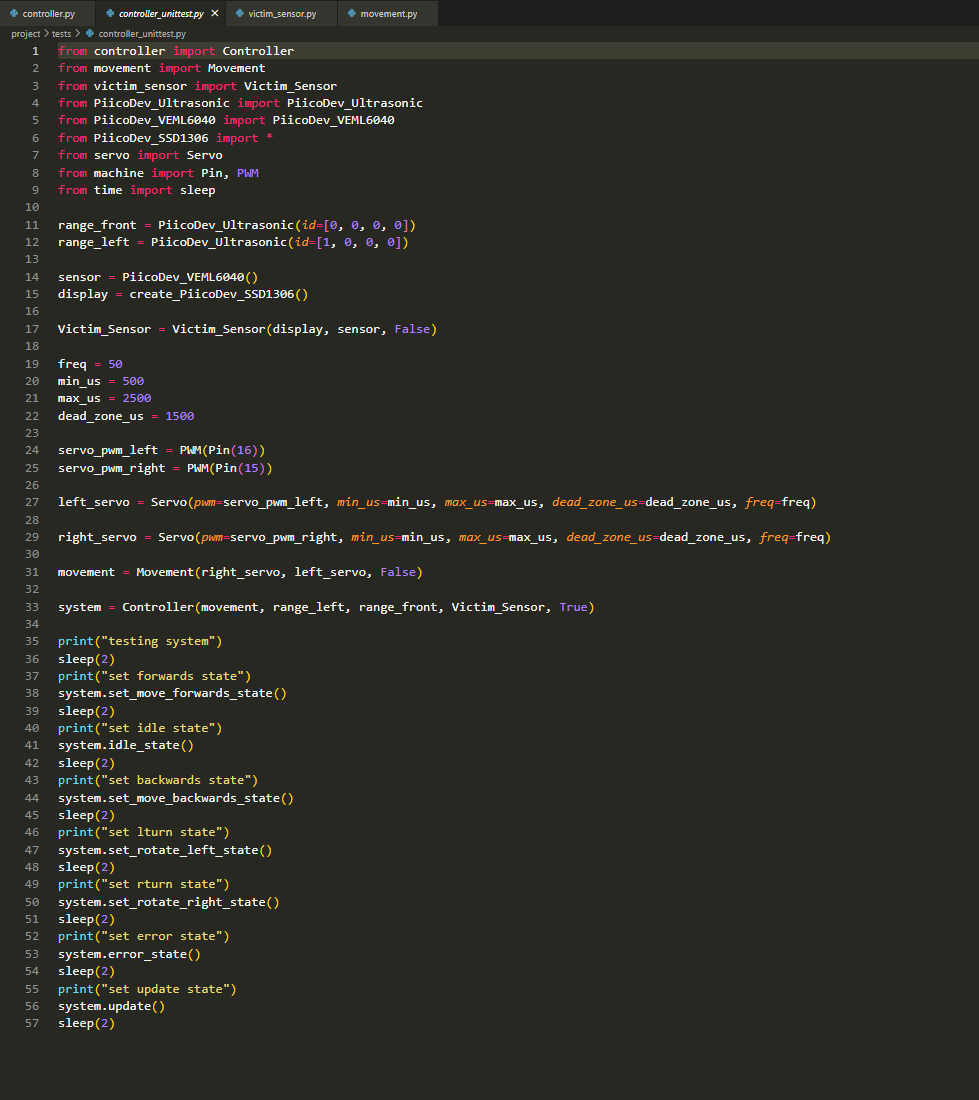
AI-generated content may be incorrect.

**Flow Chart**

**A diagram of a flowchart

AI-generated content may be incorrect.**

**Unit Test Code**



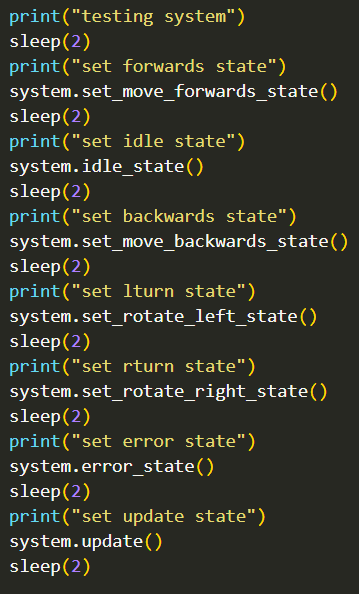
**Controller Unit test ↑ Movement Unit test ↑**

Controller unit test:

A close up of a screen

AI-generated content may be incorrect.

**Abstraction:** controller encapsulates complicated robot behavior behind method calls like set\_move\_forwards\_state()

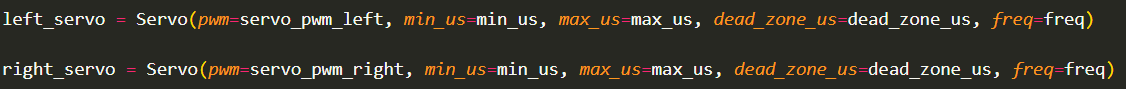


**Abstraction:** The methods such as set\_move\_forwards\_state(), set\_move\_backwards\_sate(), or set\_rotate\_left\_state() call abstract logic for each different state or robot behavior. It interacts with high level commands rather than lower level motor or sensor code.

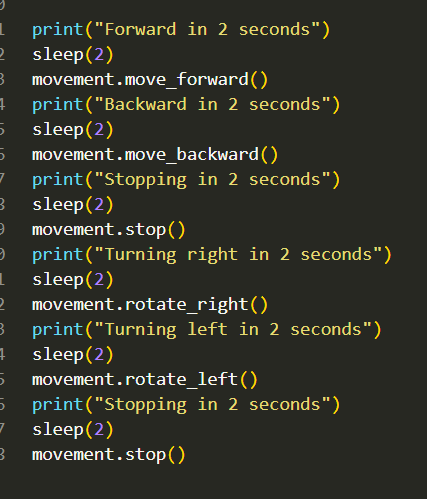


**Encapsulation:** The controller object holds internal state and behavior logic, hiding implementation details from the user.

Movement Unit test:



**Encapsulation:** The servo class encapsulates PWM control logic. You configure it once and then are able to use It repeatedly for high-level movement commands.

****

**Abstraction:** The methods such as move\_forwards(), move\_backwards(), or rotate\_left() call abstract logic for each of the different states similarly to the controller unit test. It interacts with high level commands rather than lower-level motor or sensor code.

Definitions and acronyms



5

|  |  |
| --- | --- |
| OOP | Object-Oriented-Paradigm |
| PWM | Pulse-Width-Modulation |
| LCD | Liquid-Crystal-Display |
| I2C | Iter-Integrated Circuit |
| Ultrasonic Sensor | a device that uses ultrasonic sound waves to detect the presence and distance of objects |
| Colour sensor | detects an object's colour by illuminating it with light |
| Fuse | a safety device used in electrical circuits to protect against damage from excessive current |
| Diode | a two-terminal semiconductor electronic component that functions as a one-way switch |
| Regulator/Amplifier | an electronic device that increases the magnitude of an electrical signal, such as voltage, current, or power |
| Capacitor | an electronic component that stores electrical energy in an electric field |
|  |  |
|  |  |