WM-3 — Intelligent Robot Arm

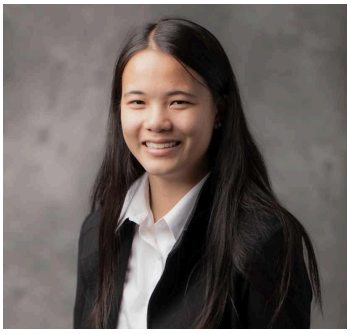
SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

CS 4850 – Section 03

Fall 2024

Professor Perry

Aug, 27, 2024

 Zhiwen Zheng Team Leader	 Ellie Ireland Developer

Team Members:

Name	Role	Cell Phone / Alt Email
Zhiwen Zheng (Team Lead)	Documentation and Test	678-818-5275 zhiwen20010111@gmail.com
Ellie Ireland	Developer	863-221-3999 elirel973@outlook.com
Waqas Majeed	Project Owner and Advisor	470-578-6005 wmajeed@kennesaw.edu

Table of Contents

1.0 Introduction....3
1.1 Purpose....3
1.2 Project Goals....3
1.3 Definitions and Acronyms....3
2.0 Design Constraints....3
2.1 Environment....3
2.2 User Characteristics....3
2.3 System....4
3.0 Functional Requirements....4
4.0 Activities....4
5.0 External Interface Requirements....5
5.1 Hardware Interface Requirements....5
5.2 Software Interface Requirements....5

Revision History

Name	Date	Reason For Changes	Version
Ellie Ireland	08/27/24	First draft	v0

1.0 Introduction

This software requirements specification serves to define the scope and steps that need to be taken in order to deliver the desired outcome for this project. The project goal will be briefly overviewed, design and functional requirements will be detailed, and the predicted tools and interfaces will be mentioned along with how they will be applied in this project.

1.1 Purpose

The product of the initial release (v0) of this project will be a four-joint, physical, functioning, robotic arm sat atop a mobile robot that, with the software we develop, may perform grasping and manipulation tasks. The hardware used in this application will include the OpenMANIPULATOR-X manipulation robot and *TurtleBot3* Waffle Pi mobile robot from the *ROBOTIS* company. This document is meant for developers, project managers, users, testers, and documentation writers.

1.2 Project Goals

The primary features that must be achieved in order for this project to be successful include the implementation of grasping and manipulation functionality in the robot arm's capabilities. This will be incorporated via software written in some variant of the C programming language supported by ROS.

1.3 Definitions and Acronyms

Acronym	**Definition**
ROS	Robot Operating System

2.0 Design Constraints

2.1 Environment

The environment in which testing and production will be done over the course of this project will be dynamic. The introduction and conceptual practice of robotics will be handled in a virtual environment called *ROBOTC*. Due to limitations with the hardware required, initial design and development for the project will be done in a virtual environment that simulates the actions of OpenMANIPULATOR-X called *Gazebo*, which is hosted on the Linux-based ROS. Software will be written in a variant of the C programming language supported by this simulation tool. The goal of this phase is to achieve basic functionality and mobility.

After six to eight weeks, once the mobile robot and manipulator hardware has been received from *ROBOTIS*, the software will be uploaded to the physical system, and the work environment for project progress will be moved to a lab setting. From here, we will continue working on the ROS to develop the software, moving our subsequent robot action from the simulated environment to the physical, real world. This portion of the project will be used for fine tuning the grasping and manipulation functionality of the system.

2.2 User Characteristics

This project serves as a point of research within the technical community. Therefore, while lacking direct interacting users, the product may be seen and referenced by interdisciplinary students, professionals, or enthusiasts looking to learn more about robotic manipulators and the peripheral software and hardware that falls within the scope of this project.

2.3 System

The OpenMANIPULATOR-X hardware supports ROS, which is a Linux-based open-source software, and it is compatible with the *TurtleBot3* Waffle Pi, which is the mobile robot the manipulator will sit atop. Within the ROS platform exists a package called *MoveIt!*, that, when imported, abstracts much of the manipulator's control mechanisms. Prototyping and design will be initialized with the *Gazebo* simulator, which is a virtual environment hosted on ROS.

Additional versions of this project may be created based on the timeframe and progress made over the course of the project's lifespan. These will be specified in section 3.0 of this document.

3.0 Functional Requirements

3.1 Movement

Manipulator can change locations with its mobile robot attachment

3.2 Vision

Manipulator unit may see via camera input

Manipulator unit may detect objects using its vision

3.3 Grasp

Manipulator unit relocates to be near the object it intends to pick up

Manipulator is positioned over the object in the correct orientation

Manipulator tightens grip to grasp and pick up the object

3.4 Smart (Optional)

Manipulator unit may detect objects and move towards them autonomously without human influence

4.0 Activities

Watch videos and become educated on topics pertaining to Raspberry PI, C, Linux, robotics, machine vision, and machine learning and artificial intelligence

Research the backgrounds for ROS, TurtleBot3 Waffle Pi, OpenMANIPULATOR-X, Gazebo, ROBOTC, and Vex Robotics

Install *ROBOTC* simulation program on PC to get oriented with the software of robotics

Develop and document a project plan, listing objectives and goals

Install ROS on a Linux system

Install the *Gazebo* simulator on the ROS

Get familiar with using the OpenMANIPULATOR-X in *Gazebo*

Research C implementations with the *MoveIt!* package and how it may be used with OpenMANIPULATOR-X

Develop a script to have the robotic manipulator grasp different objects

Get familiar with using the TurtleBot3 Waffle Pi in *Gazebo*

Research C implementations for the TurtleBot3 in simulation

Combine knowledge of OpenMANIPULATOR-X and Waffle Pi in simulation

Design and test a machine vision model to have the robot unit identify objects present

Train the robot unit to decide how to pick up an object based on the shape characteristics it sees

Test according to the specifications initially listed in the project plan

Apply software to the hardware in lab once it is received

Test hardware in combination with software according to the project plan specifications

5.0 External Interface Requirements

5.1 Hardware Interface Requirements

The hardware required for this project includes the TurtleBot3 Waffle Pi and the OpenMANIPULATOR-X. Additionally, a PC will be needed.

TurtleBot3 WafflePi

- Supports a maximum of a 30kg payload
- Equipped with a Raspberry Pi microcomputer
- 32-bit ARM Cortex®-M7 with FPU
- RC-100B + BT-410 Set Remote Controller
- XM430-W210 Actuator
- 360 Laser Distance Sensor LDS-01 Laser Distance Sensor
- Raspberry Pi Camera Module v2.1
- Gyroscope 3 Axis and Accelerometer 3 Axis
- GPIO 18 pins and Arduino 32 pin
- Peripheral: UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
- Inputs: Push buttons x 2, Reset button x 1, Dip switch x 2

OpenMANIPULATOR-X

Supports open-source software (ROS, MoveIt! package)

DYNAMIXEL-X Series actuators (quadruple jointed)

Open-source hardware allows for simple modification via 3D printed parts

Aluminum frames

PC

Capable of running Linux

5.2 Software Interface Requirements

The software IDEs and environments needed for this project include *ROBOTC*, ROS run on Linux, and *Gazebo* simulation platform.

ROBOTC

Simulation platform for different products of *VEX Robotics*

Version 4.56 (EXE)

Windows 10 requirement

ROS

Set of software libraries and tools

MoveIt! package

Open source

ROS 1 LTS for Ubuntu 20.04

Gazebo simulation platform hosted on ROS 1

Linux requirement