

Request for Proposal

MIE444: Mechatronics Principles

Team 3

Primary Lab Session 1

Name	Student Number
Zian Zhuang	1002449870
Siyan He	1001335021
Shing Chak Samuel Wong	1002172123
Xu Deng	1003268126

Submission date: October 11th, 2019

1.0 Project Requirement

The project aims to design a rover that can carry out a set of tasks autonomously in a 4' * 8' walled-maze on a randomized checkered surface. The rover, which will be placed randomly in the maze, needs to find an object in the loading zone, pick it up and drop it in the unloading zone in 5 minutes. While navigating, the rover should avoid any collision with obstacles. It also needs to display signals in the phases of loading zone arrival, object detection, and delivery completion. Up to two trials can be conducted to test the final design. Pick-up point and drop-off point will be informed before trials. The structural, electrical and programming requirements are stated below.

For structural, it must incorporate with 3D printing components, not exceeding 5 lbs in weight and 12" * 12" * 12" in size. Premade rover kit or component must not be used. Gripping mechanism must be not done by adhesive or velcro. For electrical, the design must be powered by battery packs on board solely. Touch sensors must not be used for collision avoidance and object detection. The design must be able to visualize the localization during navigation by electrical and programming approaches.

2.0 Detailed Rover Design

The bottom layer of rover has three Omni-wheels, three 6V DC motors with encoders, an ultrasonic sensor, and a gripper with two servo motors (Figure 1). The top layer has four ultrasonic sensors, Arduino, compass sensor, battery pack, breadboard and LED lights.

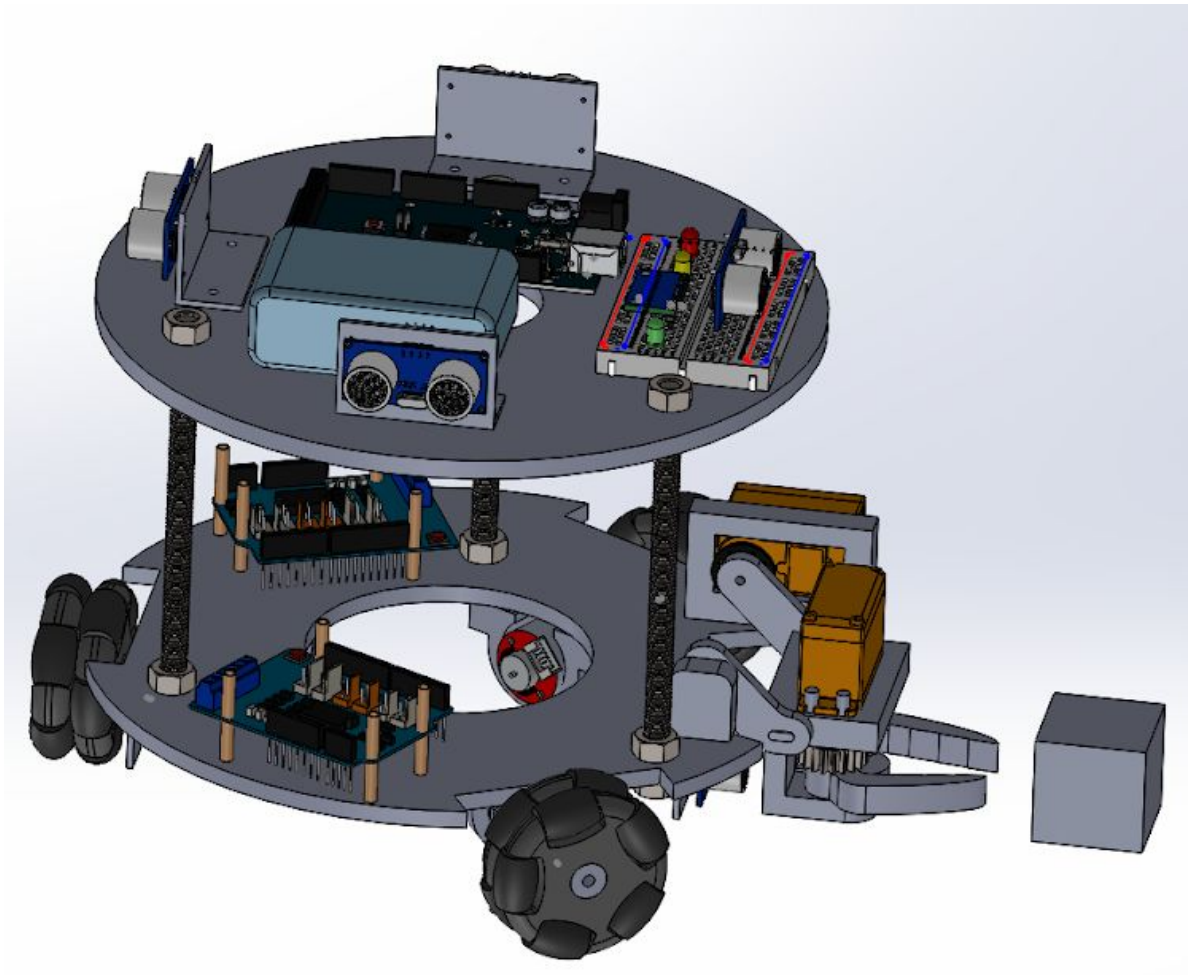


Figure 1. Rover design

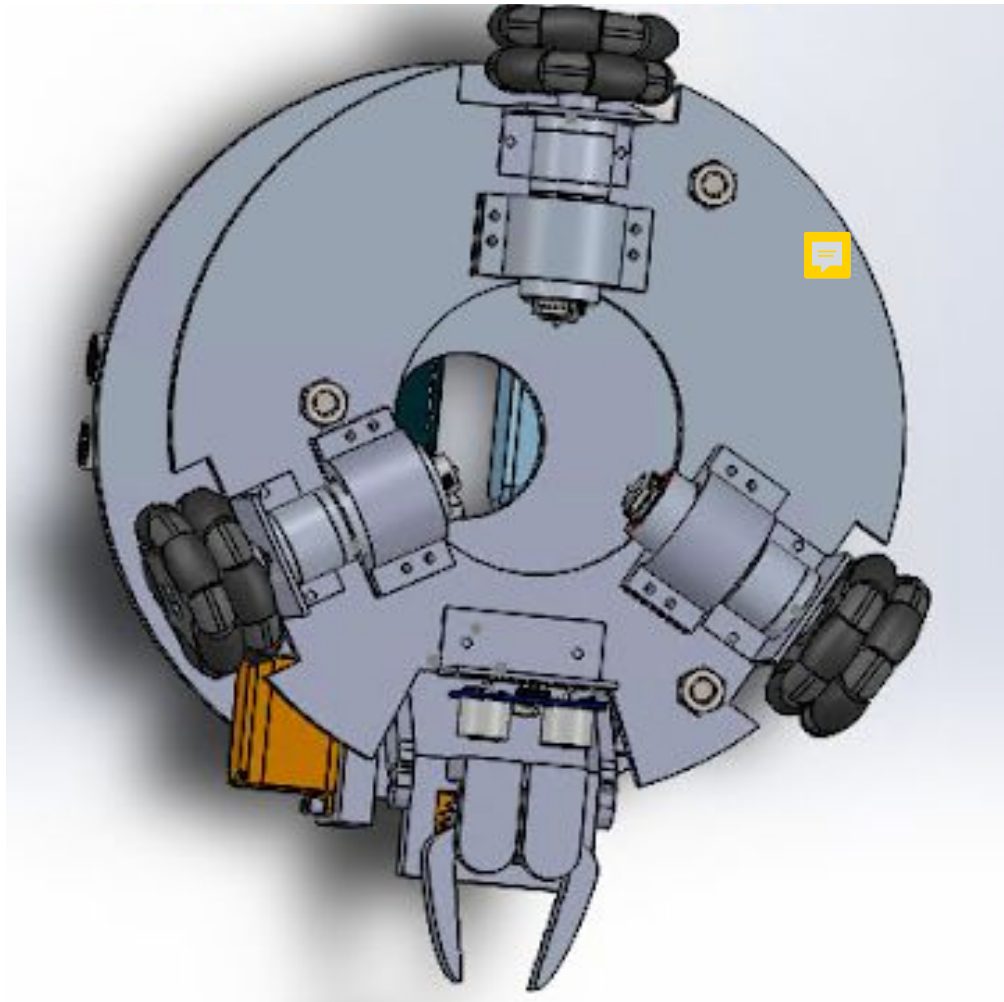


Figure 2. Bottom view of rover

2.1 Movement

Omni-wheels are separated by 120° . Two wheels are driven while driving straight. All wheels are driven when the rover is self-rotating.

2.2 Obstacle avoidance & Object detection

Four top ultrasonic sensors detect distances for collision avoidance. The top and bottom front sensors are aligned vertically for object detection. The rover rotates for object searching with direction obtained by sensing a distance difference of the object and wall. Object detection is conducted twice to ensure the object is picked up.

2.3 Gripping

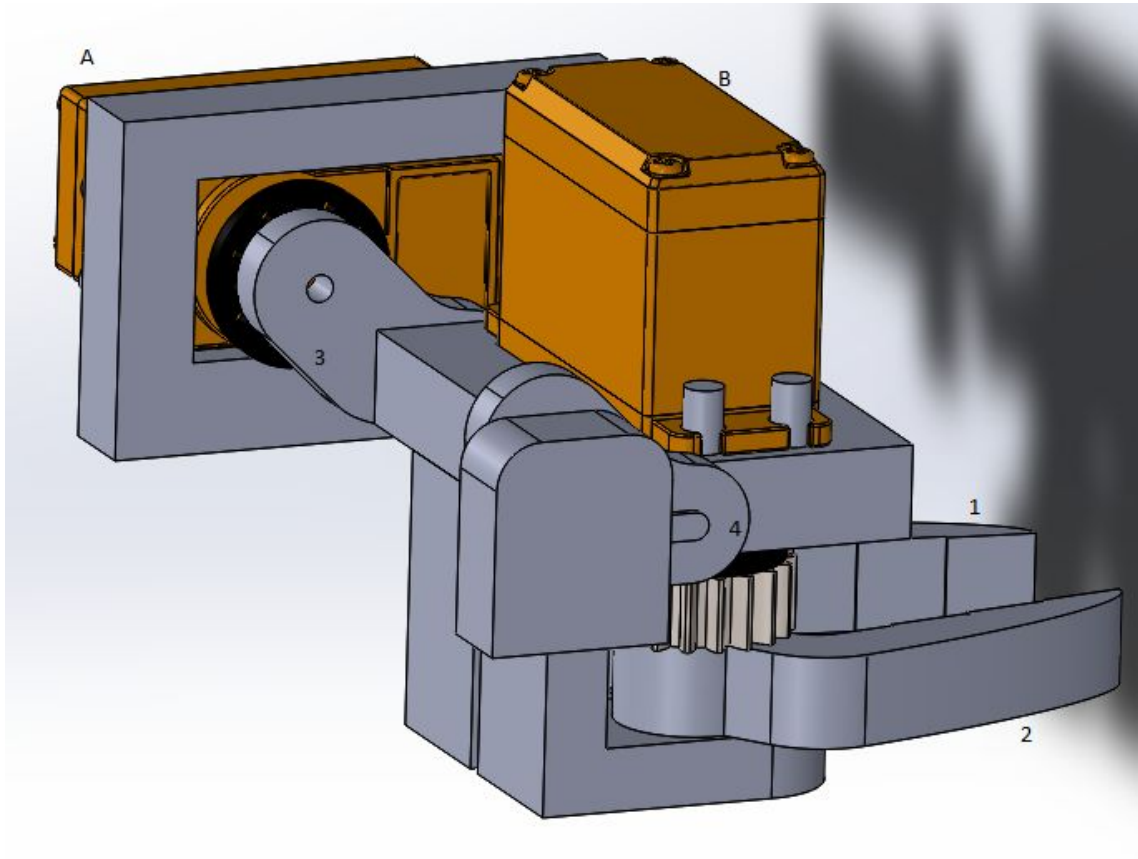


Figure 3. Gripper mechanism

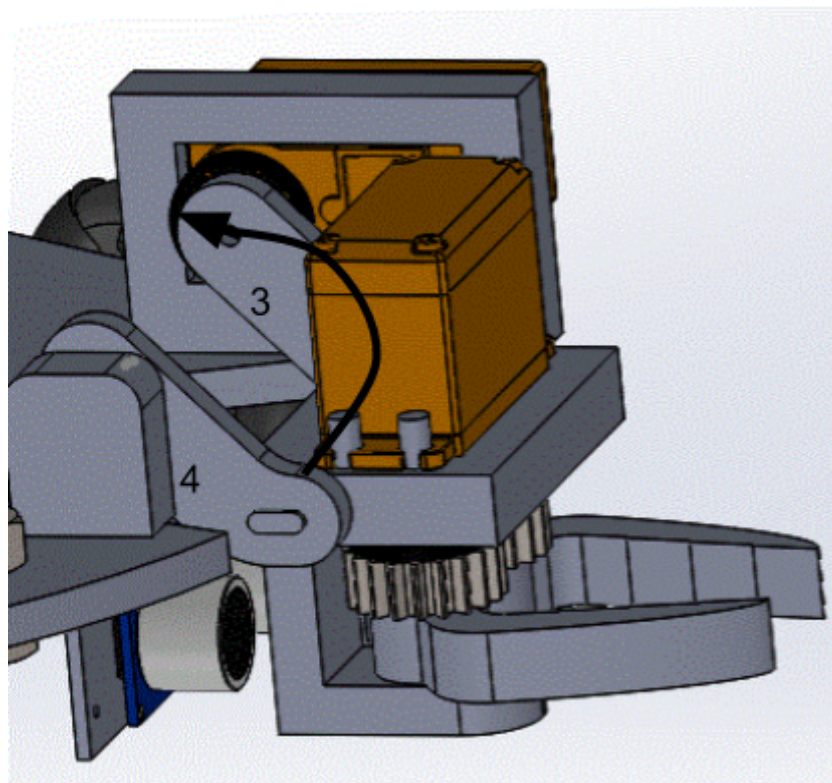



Figure 4. Rotation direction of the gripper

The gripper is lifted up while navigating to ensure clearance required for bottom sensor. When the rover arrives at the pick-up location, Motor A drives linkages 3 and 4 to move down. Motor B drives spur gears to move linkages 1 and 2 towards each other for gripping. Once the block is gripped, the gripper restores to its original position. Same principles apply while unloading the object.

3.0 Bill of Material

No.	Part Number	Quantity	Description	Price of total quantity (\$CAD)
1	SL-NM-05	1	12V 1600mAh rechargeable NiMh battery	39.93
2	WH-01	1	Wiring harness with battery connector	6.60
3	N/A	3	6V DC motor with encoder	25.83
4	MEGA 2560 R3	1	Livraison gratuite Mega 2560 R3 Mega2560 REV3 Conseil ATmega2560-16AU	7.97
5	N/A	1	6ft USB 2.0 A-B Male Cable	1.33
6	N/A	3	58mm plastic omni wheel	9.12
7	L298P	3	Dual channel DC motor driver Shield Expansion Board L298NH Module Driving Module Replace L298P	3.67
8	HC-SR04	5	Ultrasonic ranging module smart car ultrasonic sensor module	5.04
9	MPU6050	1	GY-521 MPU6050 Module 3 Axis analog gyro sensors + 3 Axis Accelerometer Module	0.82
10	HC-05	1	6Pin JY-MCU Anti-reverse RF Transceiver Wireless	4.09

			Bluetooth Serial Module 3.3V for Arduino	
11	N/A	1	Mini breadboard 8.5cm x 5.5cm 400 holes	1.36
12	MG995	2	55g servo motor digital metal gear	3.70 
13	N/A	1	Acrylic Sheets (~12" x ~12")	0
Total:				109.52

4.0 Maze Solving Strategy (Localization)

Figure 5 shows the maze layout. The direction with respect to which is referred to this figure.

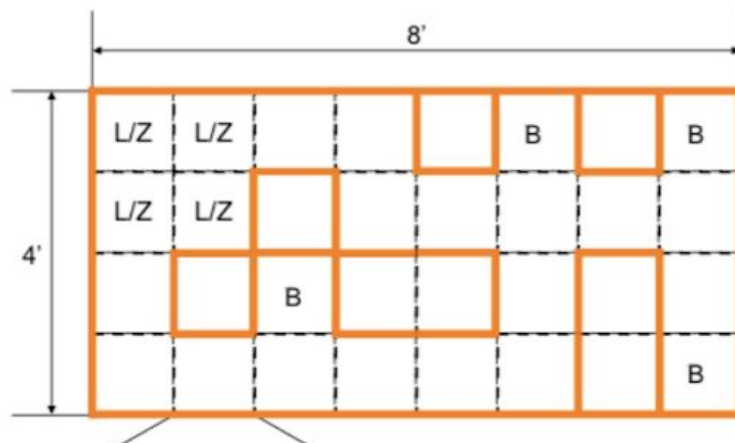


Figure 5: Maze layout

The rover aligns itself to face the left side of the maze using a digital compass at first (direction programmed in before the test). The rover then follows the logic in Figure 6 to navigate to the loading zone.

Rover moves 3" as one step. Omni wheels' speeds are adjusted by motor encoders to compensate movement error. Rover will be adjusted to center while left and right sides are facing walls by matching the distance detected by sensors. Digital compass reading is used to verify heading direction after each rotation. Confirmation of entering the loading zone will be checked by certainty to eliminate false localization. It will be confirmed once reaching the criteria three times in a row (will be adjusted after later testing).

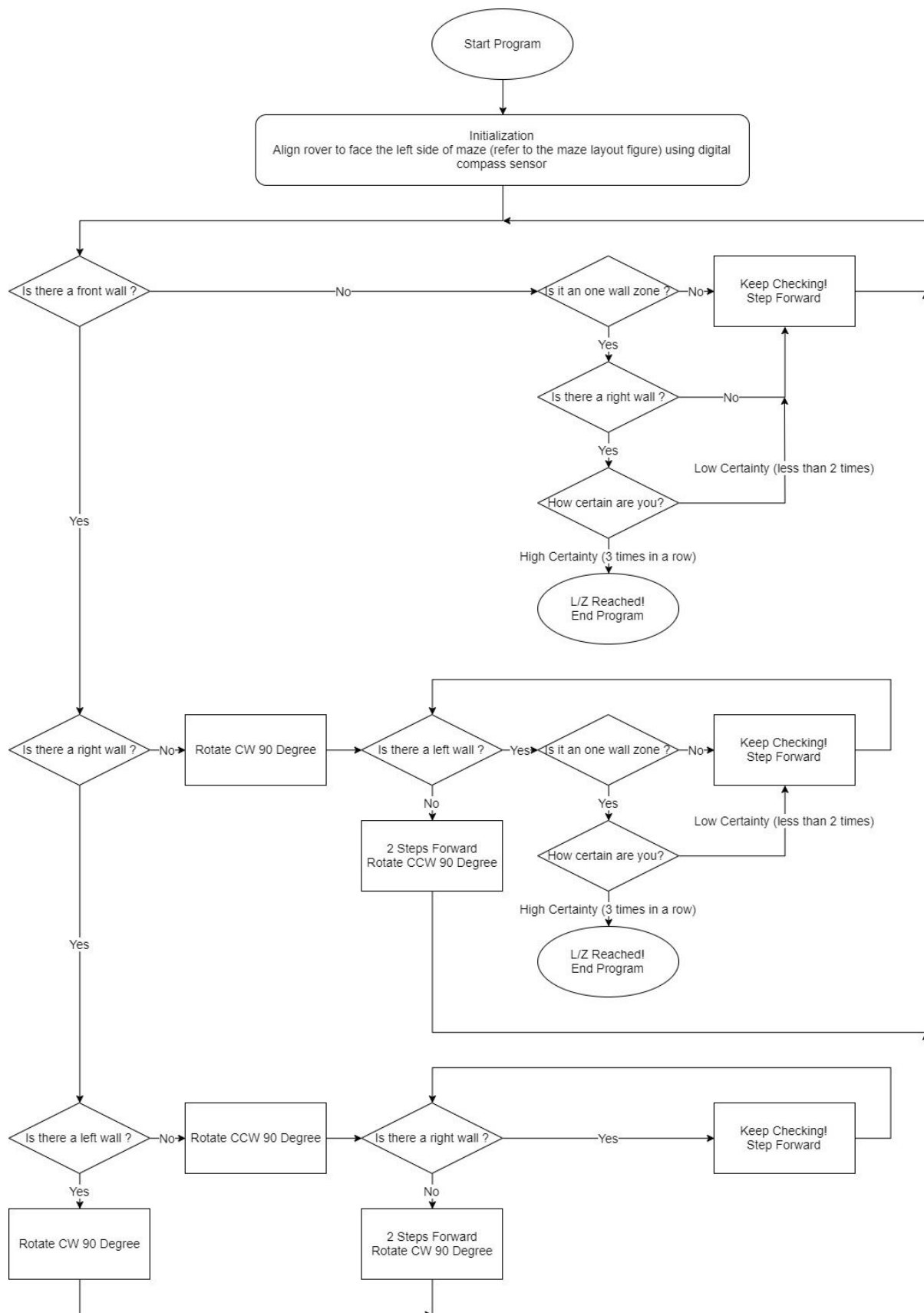


Figure 6. Maze solving (Localization) logic

Once the rover finishes navigating and picking up the object, it follows the following logic to deliver the object.

- Delivery Path (Hard-coded)
 - Step 1: (Initial Compass Calibration): Align the rover front facing towards the right side of the maze.
 - Step 2: Move backward until a wall distance is reached so that the rover is in the center.
 - Step 3: Rotate 90° CCW to face upward of the maze.
 - Step 3: Move forward until a wall distance is reached so that the rover is in the center. The rover should now be localized at the top left corner of the maze.
 - Four Paths to four possible delivery points B will be hard-coded

The rover drops off the object in the designated area after finding its way to the drop-off zone.

5.0 Attribution Table

Section	Student Name			
	Zian Zhuang	Siyan He	Shing Chak Samuel Wong	Xu Deng
Project Requirements	RD	RD ET	ET	ET
Detailed Rover Design	RS CM	RS RD	RS RD	RS RD
CAD Modelling		CM RD		RD MR
Bill of Material	RS		RD	
Maze Solving Strategy	RD	ET	ET	
Localization	RS RD	MR		ET
All		FP		FP

RS – research

RD – wrote first draft

MR – major revision

ET – edited for grammar and spelling

FP – final read through of complete document for flow and consistency

CM – responsible for compiling the elements into the complete document

OR – other