CS5510 Final Technical Report

Luke Shumway, Mason Francis, Caleb Jeppson, Matthew Larsen

Ball-Moving Robot

1 Breakdown

Job Performed	Name
Object Identification	Luke Shumway
Autonomous Navigation	Luke Shumway
ROS2 Implementation	Luke Shumway
ROS Node Integration	Luke Shumway

2 Executive Summary

This Ball-Moving Robot project aimed to develop a robot capable of autonomously moving towards and latching onto a ball. The project initially included mapping and motion planning components, but a re-evaluation led to a shift in focus towards a reactive approach. Despite encountering challenges and adjusting project scope, significant progress has been achieved.

Contributions to Understanding
The project contributes to the field of robotics by demonstrating the interactions between ROS nodes, the effectiveness of color-based object detection, and reactive autonomous navigation. The project provides insights into the practical challenges and learning opportunities associated with building a ball-moving robot.

Technical Effectiveness and Economic Feasibility
The implemented technology stack successfully detects, locates, and navigates towards a ball autonomously. Although the original mapping and

motion planning goals were revised, the current reactive approach showcases technical effectiveness in achieving core objectives. The feasibility of the methods is evident in the functional ROS2 nodes and their communication, showcasing progress towards the economic viability of autonomous robotics solutions on low-capacity hardware.

Public Benefit
The project's benefits extend to the public through its contributions to autonomous robotics, a field with potential applications in numerous industries. The development of a robot capable of independently interacting with its environment, albeit focused on ball movement, is the base of future advancements in automation and robotics, with implications for sectors such as manufacturing, surveillance, and entertainment.

In summary, the Ball-Moving Robot project demonstrates a commitment to overcoming challenges, adapting to evolving project scopes, and making progress in autonomous robotics. The lessons learned and achievements made contribute not only to the project's objectives but also to the broader understanding of implementing robotic systems in real-world scenarios.

3 Acknowledgements

This endeavor would not have been possible without the guidance and expertise generously provided by **Taylor Anderson**. We extend our thanks to Taylor Anderson for providing valuable insights, particularly in the implementation of the Sonar and Motor Control nodes, which significantly contributed to the functionality of the robot.

4 Accomplishments and Objectives

We had to re-evaluate the scope of this project. As a result, we are no longer using a mapping and motion planning approach. Our current implementation utilizes 5 ROS2 nodes: Camera, Motor-control, Sonar, Locator, and Control-policy, which communicate to move the robot towards a ball, and latch on to it. We have some code that is designed to recognize a goal area after the ball has been acquired, and implements a similar control policy to move the robot towards the goal. However, this section of code is as yet sub-functional.

This award allowed Sargent Pepper's Lonely Hearts Club Band to demonstrate a number of key objectives. The focus of the project was on building a Ball-moving robot with object detection and autonomous navigation.

A number of tasks and milestones were laid out at the beginning of the project. The actual performance against the stated milestones is summarized here:

Tasks	Milestones and Deliverables
1: Object Identification 1.1 Camera Calibration 1.2 Detect Object 1.3 Get Pixel Coordinates	Q1: Object Identification Actual Performance: (10 November) Completed 10 days later due to midterm extension and configuration issues. Can detect an object of a selected color and return the x and y coordinates of the center of the object in the image.
2: Autonomous Navigation 2.1 Motor Control 2.2 Movement Policy 2.3 Obstacle Detection	Q2: Motion Planning & Q4: Complex Pathfinding; Actual Performance: (15 Nov & 1 Dec) Re-evaluation of goal. No longer using a Mapping and Motion Planning approach. Instead we center the ball in the image and move forwards, adjusting course as necessary.
3: ROS2 Implementation 3.1 Setup Pi with ROS2 3.2 Create ROS Nodes	Q3: Combine/Implement on Robot Actual Performance: (22 November) Several Nodes are up and running, which provide limited object recognition and autonomous navigation capabilities.
4: ROS Node Integration 4.1 Function Nodes 4.2 Subscribe/Publish 4.3 Launch File	Q3: Combine/Implement on Robot & Q5: Implement Pathfinding on Robot Actual Performance: (22 Nov & 8 Dec) Re-evaluation of goal. No longer using pathfinding. 5 functioning nodes are live with robust communication. 3 launch files for various levels of execution

5 Project Activities

This project's primary focus was to program a ball-moving robot. The major approach taken was to develop a technology stack comprised of separate, coordinated modules that provide object detection, mapping, motion planning, and autonomous navigation. The object detection relies on color masking and contour detection, the mapping and motion planning were scoped out of the project, and the autonomous navigation utilizes a reactive, centering approach. This project was able to effectively detect, locate, and navigate to a ball autonomously. While the initial deliverables were unmet, substantial progress has been made towards that goal, alongside significant learning opportunities.

As noted above, around the beginning of December there occurred a project scope re-evaluation which resulted in the abandonment of the mapping and motion planning aspects. This was due, in large part, to an overestimate of participation amongst the group. You'll notice one name on the task breakdown. At the time of re-evaluation, it became apparent that the project would have to be completed solo, and there was not enough bandwidth to cover the mapping and motion planning aspects. Instead it was decided that the current reactive approach would be tweaked and improved to accomplish the task (albeit in a different manner).

6 Project Outputs

Description of Code:
The five nodes and their function are outlined below:

- Camera: Though there is a camera node in the code repository, we could not get that to function properly, so we are instead relying on the v4l2_camera_node that is implemented in the ROS2 standard library. This publishes an array form of an image taken by the camera to the /image_raw topic.
- Sonar: Patterned after the implementation provided by Taylor Anderson, this node uses GPIO to publish the distance to the nearest object to the /sonar topic.
- Motor Control: Patterned after the implementation provided by Taylor Anderson, this node subscribes to two topics: /motor_control and /servo_control, and utilizes the Car class also implemented by Taylor

Anderson to convert the messages received into commands that are sent to the motors on the robot.

- Locator: This node subscribes to the /image_raw topic, and uses OpenCV to perform a color mask on the image, then detects contours and publishes the x and y coordinates (alongside a distance estimate) of the largest shape (assumed to be the ball or goal respectively) to the /ball_info (or /goal_info) topic, as well as a boolean /is_ball flag representing the presence of the ball in the image.
- Control Policy: This node subscribes to the /sonar and /ball_info topics, and publishes the calculated motion commands to the /motor_control topic in order to center the ball in the frame of reference, and move towards it, stopping when it has reached it before attempting to find and move to the goal.

There are also nodes for mapping and planning in the repository. Once again, functionality of these nodes could not be achieved before the deadline, resulting in the project scope re-evaluation.

To run the code, use the following commands within the src directory:

```
colcon build
```

this builds the project and prepares all the commands needed to run.

```
. install/setup.zsh
```

this allows the terminal (we used zshell, so if you're using bash, replace the extension) to recognize the commands used to run the project.

```
cp -r attempt1/launch/ install/attempt1/share/attempt1/launch/
```

this copies the launch files into the correct spot. There should be a way to have colcon build do this step, but we couldn't figure it out, and this works.

TO RUN THE WHOLE PROJECT:

```
ros2 launch attempt1 launch.py
```

TO RUN AN INDIVIDUAL NODE:

```
ros2 run attempt1 [node]
```

where [node] is one of the following nodes: { sonar, motors, locate, move }

Final Note: It might be necessary to first create a new ros2 package, before building the code and running it. If that is the case, run the command

```
ros2 pkg create --build-type ament_python attempt1
```

and copy the provided files into the attempt1/attempt1 folder. We have also provided the setup.py file, so replace the auto generated setup.py with ours. Then copy the launch folder provided into the attempt1 folder (the parent, not the sub-folder). After this step you can proceed with the colcon build step.

Expected Outcomes of Running Code:
When running the launch.py launch file, and working under the assumption that there is a red ball in the frame (and no other red object bigger than the ball) the robot will move forward towards the robot, adjusting the path to the left or right as necessary.

Alternatively, we have two other launch files for different executions. The obj_dist.py file is the implementation of Taylor Anderson's demonstration, and keeps the robot 50cm from the nearest object. take_pics.py was used for camera calibration, and saves the images taken by the camera to the img directory. Note that this saves a LOT of images pretty quickly, so be aware of memory concerns.

Results:
We were able to successfully move the robot towards a ball, adjusting course as necessary, and stopping when it reaches it. As mentioned above, this was done through a reactive approach, rather than the mapping and planning approach outlined at the start of the project.
There is an (admittedly less-than-functional) implementation for moving the robot to the goal zone after the ball is acquired. This is unstable, and buggy at best.

More importantly, we learned a lot about raspberry pis, remote ssh, the ROS2 framework, and robotics in general. Substantial learning has taken place, and a non-trivial amount of progress towards even the goals that remain unmet.