Names: Adam Lizerbram, Jake Linell

Robot ID: 22

1. Project Proposal (50 points)

1.1 Problem Statement (5 points)

Our robot performs automatic scanning of its environment, maze navigation, and trash detection.

1.2 Sensors (5 points)

Sonar: Detects if there is a wall within 20 cm on each side of the robot's current square.
IR sensors (Line sensors): Detects if there is trash underneath the robot's current position.

1.3 Behaviors, Automation, and Control Architecture (30 points)

STATES:

State 0: Start
Faces servo forward, initializes the adjacency list and visited array, and calibrates the line sensors. The robot can start from anywhere facing any direction, but the initial conditions would have to be manually updated.

State 1: Process Square
Turns the servo to the left, right, and forward, and uses sonar to collect a distance measurement from each direction. It only scans the square if it has not been visited yet. This works no matter the maze layout or obstacles in the way.

State 2: Print Error Message
If the robot is out of bounds or facing an invalid direction, it will print out an error message to the serial monitor and stop moving.

State 3: Update Knowledge
The robot uses the sonar measurements to update the adjacency list to indicate whether each direction in its current square has a wall or not (store -1 if there is a wall, or the number of the adjacent square if no wall). The current square is also marked as visited. This step is entirely dependent on the Process Square step.

State 4: Predict Next Move
The robot uses the adjacency matrix, its current position and orientation, and its given turn priority to determine its next move. A turn priority order of left, forward, right, backward means that the robot will turn left if possible, if not it will go forward, etc. This step is entirely dependent on the Update Knowledge step.

State 5: Move

The robot executes the next move given its prediction. This step is entirely dependent on the Predict Next Move step.

State 6: Check for trash
After moving, the robot will check whether or not its current square contains trash (black square is detected). It only checks for trash on squares that it has not yet visited. This is dynamic and will work no matter where the trash is placed.

State 7: Collect trash
If trash is detected in its current square, the robot will turn 360 degrees and increment the counter displayed on the OLED monitor. This step is entirely dependent on the Check For Trash step.

State 8: Check for completion
At the end of each iteration, the robot will check whether its task is completed. The robot is done if it is at the charging port (square 0) and has picked up all of the trash, or has visited all the squares in the maze. This step is dynamic as long as it knows where the starting point is.

State 9: End
When the robot is done, it will beep and display the entire scanned maze as well as trash locations within the maze to the serial monitor.

TRANSITIONS:

State 0 → State 1 if done moving the servo forward, initializing the adjacency list and visited array, and calibrating the line sensors.

State 1 → State 2 if the robot detects that its current square is out of bounds or the direction is invalid.

State 1 → State 3 if the robot's current square has not yet been visited.

State 1 → State 4 if the robot's current square has been visited before.

State 3 → State 4 if the adjacency list has been successfully updated.

State 4 → State 5 if the next move has been successfully predicted.

State 5 → State 6 if the robot has completed its current movement.

State 6 → State 7 if trash is detected in the robot's current square.

State 6 → State 8 if trash is not detected.

State 7 → State 8 if trash has been successfully cleaned.

State 8 → State 1 if the robot is not at the charging station or not all trash has been cleaned.

State 8 → State 9 if the robot is at the charging station and all trash has been cleaned.

1.4 Finite State Automata (FSA) Diagram (5 points)

See at end of document.

1.5 Control Architecture (5 points)

The robot uses a hybrid control architecture. The robot is reactive in which it reacts to the scanned walls of the current square before making a decision, although the choice of direction it makes is based on a pre-set priority level. Additionally, the robot uses its line sensors to react and change its behavior if it detects trash. Our implementation is relatively simple and does not make use of any controllers. The upside to this is ease of implementation and troubleshooting, although the downside is that the robot may introduce small errors in its movement which it struggles with correcting.

2. Integration of Semester Modules (10 points)

Lab 1: Navigation primitives for movement
Used the primitive movement functions to traverse the maze.

Lab 2: Sonar application for obstacle recognition
Used the sonar to detect whether there are walls within 20 cm at each square.

Lab 8: Line following without obstacle for trash detection
Used the line sensors to detect whether there is a black square underneath the robot.

3. Challenges, Error Handling, and Reliability (15 points)

The robot does not move directly straight (it tends to drift to the right), so to compensate we added a tuned value multiplier to the weak right wheel to rotate slightly quicker.

We tried to use two motor objects to control the robot for sonar and normal movement, but we believe that this caused collision errors and/or caused the program to exceed its maximum memory usage. To resolve this, we added tuned left, right and movement multipliers to our robot to move and turn exactly to the best of its ability.

Additionally, if the robot somehow gets lost or an invalid value is passed to a variable, it will stop moving and notify exactly what error happened through the serial monitor.

4. Testing and Validation (10 points)

We constantly needed to test for tuning the left, right and forward tuned value multipliers in order to correctly move in the direction we need to. We found out that the robot will never be perfectly aligned due to non-systematic errors, such as wheel slippage, uneven surfaces and other environmental and mechanical factors. Performance metrics included the difference between expected position and actual position, accounting for both x and y position and orientation. We iteratively tuned the movement multipliers such that the robot's movements were as accurate as possible with minimal interference to be manually fixed. We tested for edge cases by placing the robot in unexpected locations and seeing how it would act. It was often able to determine that it was not in the correct place by thinking it went out of bounds and correctly displaying the error to the serial monitor.

5. Innovation and Creativity (5 points)

Instead of the traditional wall following method, we represented the maze as an undirected graph, where each square represents a node, and there is an edge between two nodes *u* and *v* if the robot is able to directly travel from *u* to *v* in 1 movement. Each node can have at most 4 edges, one for each cardinal direction. An adjacency list is used to represent the graph. As the robot traverses the maze, if it has not visited the current square, it scans its surroundings and updates the adjacency list; if it has previously visited the current square, it skips scanning and immediately predicts its next movement. Additionally, it prints the maze's layout to the serial monitor upon completion, including the locations of trash marked with "x". If we had more time available, we could have implemented a path generation algorithm to find the shortest path where each square is visited at least once and the robot ends at the charging station. Another possible addition could have been to follow the shortest path from the robot's current location back to the charging station upon collecting all trash. However, these would require hardcoding the maze layout whereas our current solution does not.

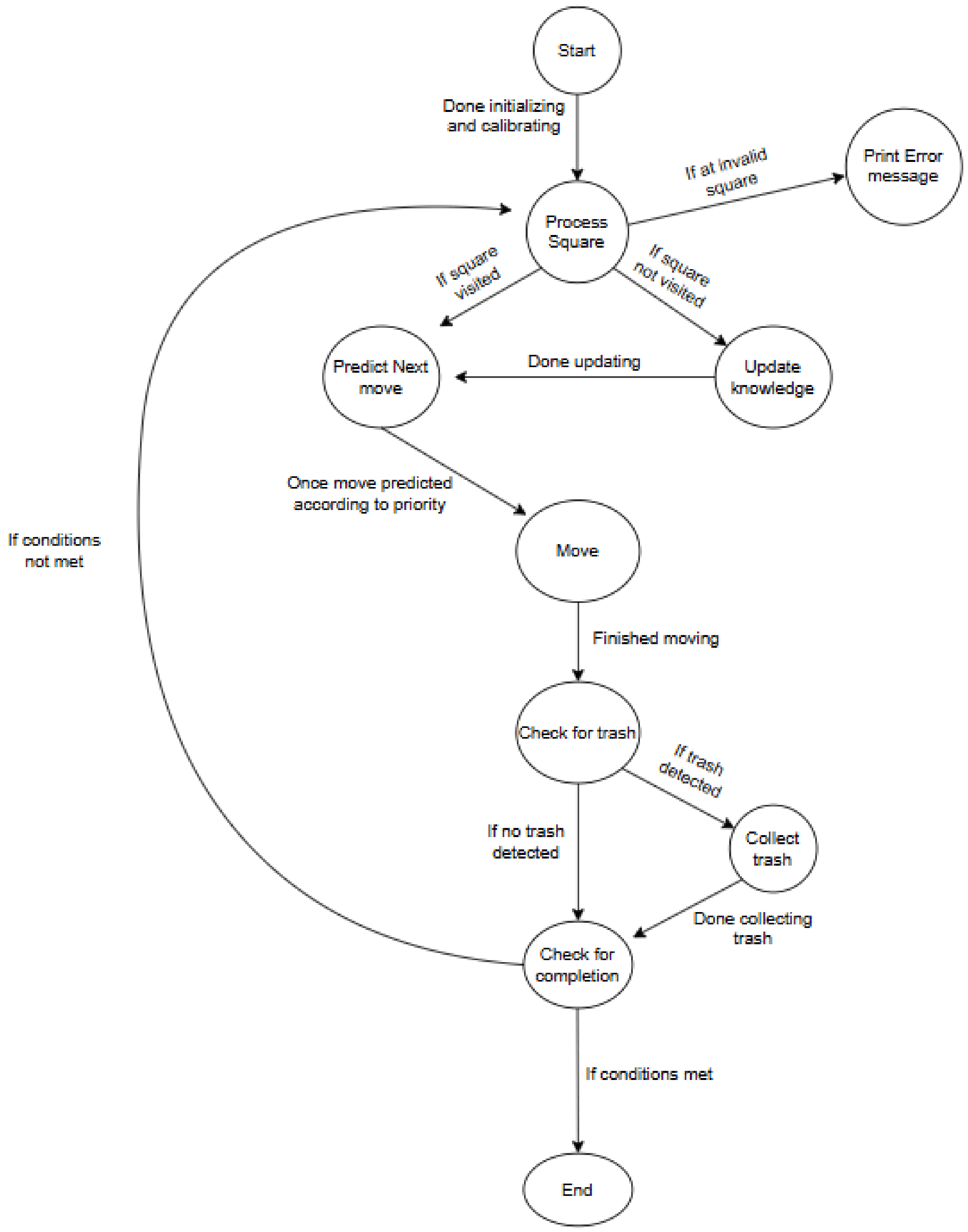
6. Communication and Team Collaboration (5 points)

We both thoroughly worked on this together as an effective team. We didn't split up our tasks too much, but Jake mainly focused on the main movement and tuning code, as well as creating possible outcomes of the robot and project. Adam focused on developing the graphical approach we use to scan the map, as well as the collecting trash method. We were able to get most of our work done during class time and utilized office hours to get even more work done.

7. Use of Resources (5 points)

We often utilized office hours and time before class so that we could complete the majority of our work in order to not procrastinate. We were successfully able to use many office hours sessions to brainstorm, test, and fix issues with our robot. Going back to old labs also helped us to remember how to configure certain sensors, such as Lab 8 where we needed to use the line following values to detect trash.

FSA Architecture Diagram


Start
Done initializing and calibrating
Process Square
If at invalid square
Print Error message
If square visited
If square not visited
Predict Next move
Done updating
Update knowledge
Once move predicted according to priority
Move
If conditions not met
Finished moving
Check for trash
If trash detected
If no trash detected
Collect trash
Done collecting trash
Check for completion
If conditions met
End