Product: Railly Clean
Team: Dalektable

Abstract

Railly Clean is a train sanitising robot that clears and sanitises train tables while out of transit to reduce the workload of cleaning staff and improve customer satisfaction.

The fourth demo focuses on the robot's ability to detect and push the button at the end of the carriage, as well as take user input of custom parameters. The product now comes with a free app allowing for custom carriage parameters to be entered to increase efficiency and accuracy of the robot. The robot also successfully detects and pushes buttons from 1.1m to 1.3m away.

1. Project Plan Update

1.1. Task Progress

- New Arm-Head Model [Achieved]
- Button Detection [Achieved]
- Button Pressing [Achieved]
- User Interaction [Achieved]
- Battery and Charging [Partially Achieved]
  We have successfully carried out a system test recording the amount of battery power used by the robot in a full system run. There was unfortunately not enough time to implement the charging functionality.
- Testing [Achieved]
- Report [Achieved]
- Website [Partially Achieved]
  We have created a custom website design and filled in most of the pages. We are yet to add some page content which will be done in preparation for the deadline.
- User Guide [Achieved]

1.2. Work Done

We have continued to use GitHub for tracking milestone tasks. For details on the project structure and organisation, please refer to the Demo 1 Report. Activities carried out since the last demo are as follows:

Daniel modified the train environment to have more realistic proportions and appearance. Daniel also modified the wings of the arm-head to be at 45°instead of 90°so that the rubbish is more effectively moved towards the bin, reducing how many items fall to the ground. Apurv worked further on visual detection of the door button's position with respect to the camera and Caitlin used this to work on making the arm's pointer successfully click and clean a button. Máté focused on modifying the codebase to take parameters from the customer while Suhas created an app for inputting these parameters. These are stored in a database on a server created by Apurv. Anh investigated how much battery power was used by the robot on a system run and researched appropriate batteries which could be installed on the robot. Sean checked the robot's ability to ascend and descend a ramp as well as fixing movement function in the situation that the robot's wheel becomes stuck. Máté, Sean, and Anh tested the robots functionalities and Máté performed system runs. Austin designed the website which Apurv made work while Daniel filled the site with content. Arnav designed and wrote the User Guide alongside Máté and Suhas. Anh, Caitlin, and Austin wrote the demo 4 report. This took approximately 20+ hours per week per person.

1.3. Budget Summary

The project continues to comprise solely of simulation. The £200 budget has not been used.

1.4. Scope Changes

As discussed in the previous demos, we are continuing with our reduced scope of cleaning without passengers on board and having a single robot per train carriage. Due to space limitation on trains, the robots will be positioned at major stations rather than docked on the trains themselves. This has the advantage of letting operators use the same set of robots to clean multiple trains throughout the day, as well as only requiring one cleaner to supervise and work with the robots on these trains at each station.

1.5. Use Case

A member of staff will have the RaillyClean app installed on a mobile device and will use this to set custom parameters. Once the robot is in front of a ramp, it will ascend into the carriage and automatically turn to face the sticker. The robot will then clear and wipe tables and buttons, which are major touch-points for passengers, before exiting the train via the ramp. The cleaner can then walk down the carriage aisle, quickly removing any missed rubbish or valuables left by passengers, and troubleshooting any technical issues which may have arisen.// If an obstacle is detected in front of the robot, it will stop to allow suitable action to be taken by the cleaner to prevent collisions and ensure the safety

of the cleaner. In the future, we believe we can do further development and testing to have the robot entering trains and navigating into carriages using the app.

1.6. Future Plans

Our aims for the Industry Fair are to finish the Website in preparation for the website deadline, make changes to our gather.town room, and improve the video of our robot using our new robot design along with any changes advised. A future feature we would have liked to implement is recognising suspicious or dangerous material and alerting staff.

2. Technical details

2.1. Hardware

Figure 1 shows our final design of Railly Clean's base, body, and arm. The base is custom made with 4 mecanum wheels. All processing in the robot is carried out on a controller in Webots designed with Raspberry Pi 3 in mind.

2.1.1. The Base

The base of our robot remains unchanged. The mecanum wheels allow the robot to move directly left and right instead of moving forward for robot centring. The current base is $0.5m^2$ with 4 x *0.05m* radius mecanum wheel.

2.1.2. The Body

We based on the current functionality of our robot and had a small modification of the position of the camera on the robot. The body has a rectangle shape *0.87m x0.3m* and enables the cleaner to place a bin bag inside for storing the trash. Currently, we have 5 distance sensors and 2 cameras attached to the robot. 2 side distance sensors are used for centring the robot, and another for detecting the table and distance to wall. We also have 2 distance sensors for detecting the wall in the front and check the level which the trash has piled up inside the robot. The front camera detects door button and door sticker for centering, while the side camera detects valuable objects on the table.

2.1.3. The Robot Arm

The current arm remains unchanged but there is a small change on the edges of the cleaning head. The current arm contains 4 sections (3 Joints + head). Each joint has dimensions *(length* x *radius) .50m* x *.02m*. The current arm is designed to fit through train doors while still enabling flexible and robust cleaning motion. Each section contains a Rotational Motor and Position Sensor to perform separate movement.
In addition, the current design of the head contains two parts (figure 2). First is the cleaning head has width *.025m*, with two added edges to keep trash in line when sweeping. There is a touch sensor on the cleaning head for feedback on the head's contact with the table. Second is a small pointer next to the edge to tap the train door button. The edges of the cleaning head were perpendicular to the cleaning head. This time we modified the edges open a bit to cover more areas and prevent trash from sliding off.

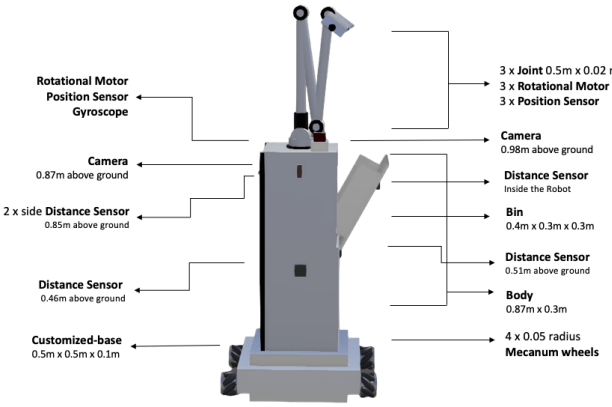

Figure 1. Railly Clean's current design

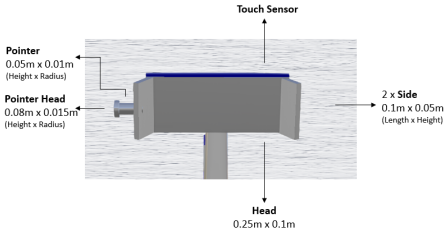

Figure 2. Railly Clean's current cleaning head design

2.2. Software

2.2.1. Button Detection

Once the robot is within 1.3 metres of the end of a carriage, it uses it's front camera to take an image of the carriage end. Using this image and a template image of a train button, the algorithm uses SIFT detection to detect key points in both images. Destination points from one image to the other are then found and a Fast Library for Approximate Nearest Neighbours (FLANN) is used to match destination points in both images within a certain threshold. This gives the position of the button relative to the top left corner of the image and from this, the origin is shifted to find the position relative to the centre of the image.
In WeBots, the 3D position in space relative to the camera is estimated by using the real width of the image and the number of pixels in the image to calculate the width of each pixel. Then, trigonometry is used to find the 3D co-ordinates.
In real life, the camera will be calibrated such that the 2D co-ordinates can be accurately converted to 3D co-ordinates with respect to the robot's position. This calibration will help deal with distortions in the image.

2.2.2. Button Pressing

In order for the pointer on the arm to press the button, the co-ordinates given by the detection function must be shifted to

denote the position relative to the base of the arm instead of the center of the camera. Once this has been calculated, the robot moves forwards to within 0.3m of the end of carriage in order to avoid the arm hitting any chairs. The height and length from the base is passed to inverse kinematics for the arm to move into the correct position. The actual position of the joints is measured by position sensors and the arm continues to move until theses positions are equal to the desired joint positions. The robot then moves forward to allow the pointer to press the button, opening the door so that the robot may exit the carriage. The arm is folded back into its initial state for moving through the door.

2.2.3. User Interaction

An app allowing for custom carriage paramaters to be set has been developed on android studio using java. In practice we will use bluetooth to ensure more reliable connections, but for simulation purposes we are sending information over the internet. We have made use of the okhttplibrary to send a http POST request with json data to a database. A REST API was created using the FastAPI framework in Python to handle configurations for robots. The API exposes endpoints for creating, requesting, updating, and deleting configuration values for a robot model. The API stores the data in a postgresql database. Both the API and database are deployed on Heroku. A ParameterSetup class is used to read in the necessary parameters from the .json file for use by the code libraries. The class has the factory defaults in place initially and thanks to feedback based control the robot is able to operate with these default values, albeit less efficiently. With this class, there are no longer hard-coded values in our code.

3. Evaluation

For simulations in Webots, we manually logged and averaged performance results across 5 repeated runs, unless stated otherwise. To better model real world settings, all simulations runs were carried out with 10% noise added to all 4 distance sensors (2 on the left, 1 on the right, and 1 in the front) used for centering, table detection, and wall detection. 10% noise was also added to the 128-by-128px front camera currently used to detect sticker at end of carriage.

3.1. Button Manipulation

Test	Distance from wall (m)	Success rate
1	1.0	×
2	1.1	4/6
3	1.2	6/6
4	1.3	5/6
5	1.4	×

Table 1. 3.1: Button pushing from different distances from wall

We note the robot's ability to recognise buttons located 1.25m above ground, and make the necessary adjustments to move forward to push from different distances. This information is useful for knowing when the robot should stop to check for a button, rather than a quantitative analysis about the robot's ability to detect button and push.

Test	Height from floor (m)	Success
1	1.05	×
2	1.15	√
3	1.25	√
4	1.35	√
5	1.45	×

Table 2. 3.1: Button pushing at different locations above ground

Due to the lack of time for development, we were not able to develop button manipulation to the extent that we wanted. In particular, we recognise that its inability to work with buttons placed between 1.05m and 1.15m from the floor disadvantageous to real life application. as buttons designed for accessible use are recommended at 1.2m above floor (Department for Transport, 2015). Seeing as the robot was able to recognise and push button at 1.25m from floor, with further testing, we can improve the arm's mechanism for pushing button to also cover the expected 1.2m height.

We recognise that this is not a hardware limitation, as the arm has been shown to work effectively with tables that are 1-1.1m above ground. Our main source of problem was with precisely computing the location of the button relative to the arm for our kinematics computations. The algorithm will perform better when given a high-resolution template of the button the robot needs to work with.

3.2. Clearing Rubbish from Table

Test	Total	Located side	Successfully cleared average
1	1	Left	1
2	1	Right	1
3	2	Left	2
4	2	Right	2
5	3	Left	3
6	3	Right	2

Table 3. 3.2: Number of items collected into bin

We tested the robot's ability to pull the rubbish into the bin body with the assumption that for each pair of passengers at either side of the table seat, there would be about 1-3 items left behind. We note that the robot is expected to behave with a bin bag inserted, which not only keeps the bin compartment clean but also prevents items from falling out of the bin. We also note that Webots has known bugs with cylindrical objects, related to which we have observed that items would fall straight through the door or the base even though that is not expected to happen in the real world. Therefore, we consider an item successfully cleared into bin if it falls down onto the opened door, since it will slide down the bin bag into the bin.

We observe that the robot's sweeping mechanism performed well not just in terms of table coverage (see last demo report) but also in terms of expected performance in clearing rubbish. This is expected as long as the robot is able to record accurate distances to wall, and the tables are aligned

well with the chairs (as is the case in real train environments), so it can utilise the sideways movement of the mecanum wheels to move close enough to the table to open its bin the right amount.

3.3. System Run Test

Specification	Value
Energy expended to complete a carriage	60 Wh
Time taken to clean 1mx0.8m wide table	2 minutes
Time taken to detect and push button	25 seconds
Maximum travellable angle of ascent	10°
Maximum time needed to centre	1 minute
Time taken to complete a carriage in Webots	20 minutes

Table 4. 3.3: Whole system specifications

Based on the energy consumption reported by Webots at the end of a whole-carriage system run, a 6000mAh battery (14.4V) would deliver enough power to last a single use even with overhead, and could be easily switched out if short on time or fully recharged in under an hour (especially as the battery will likely not have been depleted).

The coverage remains at over 95% as in the previous demo and as the head has a disinfectant covered sponge attached, which disinfects the tables.

The current cleaning mechanism includes a robust feedback loop that allows the arm to recover itself from any position (see previous report), given the complexity of surfaces and items it will encounter. Our carriage was set up with 6 table seats in total, matching the number expected in a real train carriage. We find the current performance acceptable for use during a deep-clean of the train at night, as the robots can help alleviate the need for many cleaners to deep-clean the trains. For use during the day, more testing and optimisation will be needed to bring the time down to around 10 minutes. The bottleneck was in calculating joint values using kinematics, which can be improved with more accurate calculations of optimal PID values to scale subsequent wipe actions after the first one in the carriage. We also note that time-sensitive performance recorded by Webots may not reflect real world application.

We expect the robot will work on 3 buttons throughout a carriage, one to open the door to get into the carriage, and one at either end of the carriage. With the ability to manipulate the buttons, the robot can travel on and off the carriage by itself without requiring a cleaner's assistance; thus the 75 seconds expected to open and close doors can be considered a significant improvement from the time and risks involved in having the cleaner guide the robot on and off carriage, as well as enabling the cleaner to streamline the process of setting up accessible ramps and powering on the robots. The cleaner can then focus on clearing rubbish and picking up valuables left behind. We also note that if the robot is able to push the button and pass through door to get into the carriage, it will likely to be already close to centred in the aisle and won't require the maximum 1 minute for centering before cleaning.

Given the robot's tested ability to move up and down slopes up to 10° steep, exceeding the standardised maximum gradient of 1:12, the ability to autonomously navigate into the carriage can be developed and tested in the future using the existing implementation for button detection and aisle movement (Department for Transport, 2011).

4. Budget

Table 5 estimates the total system's budget.

Component model	Price	Count	Note	Total
Raspberry PI Camera	£20.00	2		£40.00
Raspberry PI 3	£30.00	1		£30.00
EV3 Ultrasonic Sensor	£32.39	5		£161.95
EV3 Medium Motor	£25.79	4	Comes with rotation sensor; for the arm	£103.16
EV3 Touch Sensor	£19.19	1		£19.19
GBPro-Squeegee-35cm	£12.95	1	amazon.co.uk	£12.95
Wickes.co.uk 10mm Twinwall Polycarbonate Sheet 700x2500mm	£29.00	1	Rectangular body: 0.87m (h) x 0.30m (l) Square base: 0.1m (h) x 0.5m (l)	£29.00
Plastock.co.uk Acryllic Blue Tube 1830x22mm	£25.01	1	For the arm	£25.01
Set of four 100mm Mecanum wheel robot omni wheels	£52.79	1	amazon.co.uk	£52.79
LSM6DS3TR STMicroelectronics	£4.26	2	uk.rs-online.com; accelerometer + gyroscope	£8.52
6000mAh (14.4V) Li-on battery	£43.72	2	electropapa.com; designed for robotic cleaners	£87.44
Makita DC18RC 18V LXT Li-on Fast Battery Charger	£34.95	1	powertoolworld.co.uk	34.95
Misc.	£10.00	1		£10.00
			Total estimated cost for system	£614.95

Table 5. Estimated budget for the system

5. Video

https://uoe.sharepoint.com/:v:/s/SDP2021-Group-10/EVRAOpAkuKhJsztZbrKNnuUBj8_sYg4D57yCaPunUP8_dw?e=01OcYq

References

Department for Transport. Design for Disabled People: A Code of Practice, November 2011. URL https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/3191/accessible-train-station-design-cop.pdf.

Department for Transport. Design Standards for Accessible Railway Stations, March 2015. URL https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/918425/design-standards-accessible-stations.pdf.