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.

The third demo focuses on the robot's ability to center itself automatically and clear rubbish from tables in the carriage.

1. Project Plan Update

1.1. Task Progress

- New Wheel and Base Model [Achieved]
- Three-part Arm Model [Achieved]
- Real and Simulated Valuables detection [Achieved]
- Rubbish Removal [Achieved]
- Robot Centering [Partially Achieved]
  The robot is capable of centring itself in the aisle and facing the correct direction. However, this is subject to failure under certain circumstances and therefore requires further work.
- Button Models [Achieved]
- Button detection [Partially Achieved]
  The button is successfully detected and its position in the image found. A function converting this value to a position relative to the robot to pass to the kinematics has been written but is yet to be tested.
- Button pressing [Partially Achieved]
  The arm has kinematics set up so that it can can successfully move to the button position, however this needs work to become more accurate and robust. It also needs to be integrated with the button detection code.
- Demo Preparation [Achieved]
- Testing 3 [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 arm to include three sections and modified the base to include mecanum wheels. Austin created a model of a train button and added an appendage to the arm to press this button. The kinematics were then modified by Caitlin to adjust for the new three part structure and the button pressing functionality. The machine learning for real-life rubbish vs valuables detection was tested by Apurv while Máté worked with the Webots recognition functions to simulate this behaviour. Máté also summarised areas where our robot was not robust enough and suggested modifications. Apurv then worked on visual detection of the door button's position with respect to the robot which could be fed to Caitlin's kinematics. Suhas focused on using vision to centre the robot, while Sean modified movement functions for the new wheels. Sean and Suhas worked together to fix the robot's rotation and centre in the aisle using vision and side sensors. Anh and Austin worked on creating robust tests which were carried out by Anh, Daniel, and Sean. Finally, Anh, Caitlin, Austin and Arnav prepared the report and video for Demo 3.

1.3. Budget Summary

The project continues to comprise solely of simulation. The £200 budget has not been used. We booked a 20 minute meeting with Thomas Corberes on March 4 to discuss robot movement, and a 15 minute meeting with Julian Habekost on March 10 to discuss vision algorithms.

1.4. Scope Changes

As discussed in the previous demo, we are continuing with our reduced scope of cleaning without passengers on board due to various ethical issues, including using cameras around the public, and the safety risks if the robot blocks the passage during an emergency. Additionally, While our original aim was to have a single more complex robot crossing multiple carriages to clean the entire train, we realized it would be more efficient to have multiple simpler robots, each cleaning a single carriage. These can work in unison to save on wait time, while still keeping the cost of operating 5 robots on a train per day at roughly £15 (based on the price of a single robot calculated in section 4). This is less than the cost of employing a cleaner for two hours a day at minimum wage. Since these robots would not need to spend time crossing carriages, they can focus on cleaning a single carriage more frequently and would be more efficient than a cleaner who has to travel across multiple carriages.

1.5. Schedule Changes

Our aims for Demo 4 are to: complete the Button pressing milestone; simulate charging so we know the size of battery needed in real life; and implement necessary user

interaction for setting required parameters. We will work on making our product more robust and efficient, and work on our product website and User Guide.
Our modified schedule moving forwards is displayed as a Gantt Chart in Figure 3.

2. Technical details

2.1. Hardware

Figure 1 shows our modified design of Railly Clean's base, body, and arm for easier centering of the robot and smoother cleaning action. The current model uses a custom-made base with mecanum wheels instead of TIAGo Base. 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

In the past, our robot was built upon TIAGo Base with 2 motorized rotational wheels. However, the wheels only allowed moving forwards while centering. Therefore, we decided to create a *0.5m*^{2} base along with 4 mecanum wheels and accelerometer. The mecanum wheels allow the robot to move directly left and right while centering itself before moving further down the aisle.

2.1.2. The Body

We have redesigned our custom-built body after the change of our base. Firstly, our robot can now open the bin and store rubbish swept off the table in a bin bag placed inside by cleaners. Secondly, we now have 2 side distance sensors for centering the robot, a front sensor for detecting the wall, and a left one for detecting table and distance to wall. One distance sensor is attached to the roof of the bin (inside the robot) to check how high rubbish has piled up. For the cameras, we have one on top of the body for detecting valuables on the table and one in the centre of the body for detecting door button and the sticker used for centering.

2.1.3. The Robot Arm

To better control the cleaning action and fit through the train doors, we designed a new arm containing 4 sections (3 joints + head). Compared to previously (2 joints + head), we have shortened the 2 joints from dimensions *(length* x *radius) .78m* x *.02m* and *.70m* x *.02m* to 3 identical *.50m* x *.02m* joints. Based on the same cleaning movement, the current design of the arm makes the cleaning action smoother and prevents the arm from crashing into the table. Each section contains a rotational motor and a position sensor to perform separate movement.
Moreover, we also revised the design of the head to pull the trash towards the robot efficiently and added a touch sensor for feedback on the head's contact with the table. The current design of the head contains two parts (figure 2). First is the cleaning head whose width was modified from *.30m* to *.025m*, with two added edges to keep trash in line when sweeping. Second is a small pointer next to the shape for the upcoming button pressing feature.

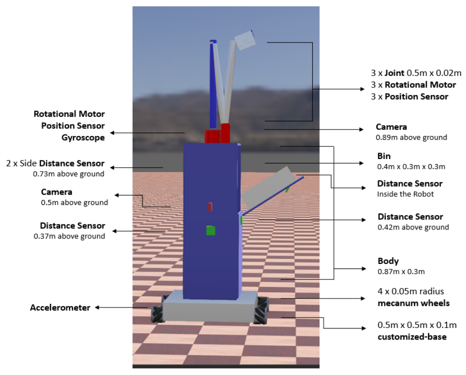


Figure 1. Railly Clean's current design

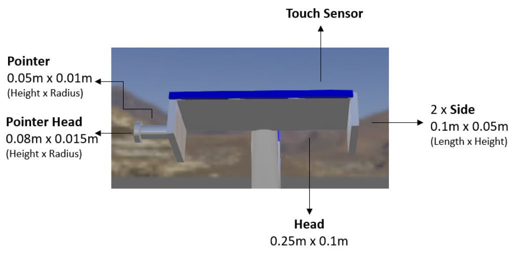


Figure 2. Railly Clean's current cleaning head design

2.2. Software

2.2.1. Wiping Mechanism

The desired position of the arm's head is estimated using the distance to the wall measured by the left sensor and the estimated table height registered by the customer. Inverse kinematics is used to set the joint values on the arm to reach the desired position. The joints are then set to their desired value and at each time-step, the actual value of the joints is measured by position sensors and the arm keeps moving towards its desired position until the joints are within a threshold of the desired joint values. The desired height of the head is then reduced until the touch sensor is activated which implies the head is touching the table or rubbish. The desired length is gradually reduced and height altered to maintain a force between 1.0N and 10.0N while swiping. At table edge, the arm dropping is detected and the robot folds in. At any stage, if the joint values registered by the position sensors were the same for the last four time-steps, the robot is considered to have become stuck and its desired height and length are gradually reduced.

2.2.2. Centering

In order to keep itself in the centre of the aisle facing forwards, the robot uses both upper side sensors, and the front camera. Side sensors are used to find how much sideways movement is needed to center the robot, The robot moves a fraction of the total difference between the distance sensors provided the distance is within a reasonable range. The camera is then accessed and the centroid of the sticker (see demo 1) is located, this is used to find how much the robot needs to rotate. If it is less than a threshold then no rotation happens. If the sticker is not found then the robot rotates in the direction last rotated until it is found. In the physical model, we will have to calibrate the camera using opencv's in-built methods. The calibration is necessary to account for radial and tangential distortion. After this, using the field of view of the camera (given by the manufacturing specifications) we can proceed as in simulation.

2.2.3. Button Detection and Pressing

Once the robot is at 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. 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. The position calculated by the detection function is then passed to an inverse kinematics function to calculate the desired joint values of the arm in order to bring the pointer to the button.

2.2.4. Bin Mechanism

The bin opens on table detection and closes on continuation to the next table. An improvement yet to be implemented is that when the distance sensor at the top of the bin detects objects within 0.03 metres, the bin will no longer sweep rubbish from tables. An alert will go out to tell cleaners that the robot requires emptying.

2.2.5. Table Recognition

Full details of the table detection algorithm can be found in the demo 2 report.
The distances noted by the sensor are filtered to compensate for sensor noise such that only changes in distance of 0.3m or more are considered a legitimate change in closest object and if the distance noted by the sensor is significantly greater than a reasonable distance to the wall (capped at 2m), the reading is ignored and the previous reading is used.

2.2.6. Trash Classification

While integrating the original classifier with Webots, the cartoonish Webots rendering worked poorly with a classifier trained on real-life images, as the whole environment is interpreted as 'cartoon' and thus classified as 'cleanable'. Since we want the robot to work in a real train, we kept and tested our real-life classifier for real images and opted to use the Webots camera's built in object recognition functionality for simulation purposes. A description of the machine learning approach taken to classify real-life rubbish vs valuable images can be found in the demo 2 report. The Webots approach currently uses a list of predefined objects which are classified as valuable or trash so that the controller knows when not to clean.

3. Evaluation

For simulations in Webots, we manually logged and averaged performance results across 5 repeated runs. For each test, we interchanged variables according to different situations the robot might find itself in.

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. 3-part Arm: Sweep Coverage & Object Removal

Attaching pens to the cleaning head helped visualising sweeping from wall to table's edge. By overlaying a grid over the screenshot taken of the painted table from a bird eye view, we computed the painted over surface area relative to table's surface area.

Test	Sweep coverage (%)
1	95
2	90
3	95
4	98
5	98

Table 1. 3.1: Single sweep from wall to edge on 1.1m long table

Test	Table length from wall (m)	Sweep coverage (%)
1	0.5	×
2	0.6	71
3	0.7	96
4	0.8	96
5	1.0	95
6	1.1	95
7	1.4	87

Table 2. 3.1: Whole table coverage for different lengths

Test	Objects scattered across table	Objects cleared
1	2	2
2	3	3
3	7	4

Table 3. 3.1: Number of objects successfully cleared

Given the arm dimensions mentioned above, we conclude that the arm satisfactorily consistently sweeps 90% or more for table lengths between 0.7m and 1.1m (inclusive), within the expected range for train tables. A single sweep from wall to the edge of a 1.1m long table takes 28.5s, and the robot spends on average 2 minutes at the table, from opening bin at one end to closing bin at another. Assuming LNER's current route from London to Edinburgh which uses Azuma trains with 4 table seats per carriage, the robot can theoretically clear rubbish and clean tables in the 10 minutes the route stops at Newcastle. We have noticed that spherical trash like fruits could be difficult to accurately sweep due to their lack of contact surface with the table. It is worth noting however that half-eaten fruits would lose the spherical shape and behave more like cans of drink.

3.2. Valuable Classifier

We trained and evaluated our TensorFlow Lite classifier on an image set consisting of 1434 cleanable objects (trash) and 1221 valuables, split into training, validation, and test sets using the 80:10:10 ratio. The image set consisted of a variety of objects under different lighting conditions and rotations, with some images showing valuables cluttered with other objects. All the images were resized to 300x300px before being used to train the classifier.

Epoch	Training Accuracy (%)	Validation Accuracy (%)
1	81.43	98.83
2	98.76	99.61
3	99.21	99.61
4	99.11	99.61
5	99.47	99.61
Testing Accuracy (%): 99.62		

Table 4. Results for test case 3.2

We were able to verify that the classifier could run on Raspbery PI 3, which we plan to use on our marketed robot. We have confidence the classifier would perform well in the real world given the diverse nature of images used in training and evaluating, and as it successfully identified valuables on random images not from the data set. The classifier works on 2 labels, 'valuable' and 'cleanable', which are all the robot needs to decide to sweep a section of the table. The inference speed of one image expected of our specific model on Raspberry PI 3 is 639µs (Houston, 2020).

3.3. Rotation Fixing and Aisle Centering

We tested how long the robot took on average to fix its rotation to face straight relative to the sticker placed at end of carriage, then centre itself in the aisle using side distance sensors tracking distances to walls on both sides. The tests were carried out 12.5m away from the end of the carriage, and a negative value represents leftward (rotation or displacement from centre's aisle).

Test	Displacement (m)	Rotation (°)	Time (s)
1	-0.6	60	40.9
2	-0.6	30	8.2
3	-0.3	60	6.5
4	-0.3	30	4.4
5	0.3	-30	4.7
6	0.3	-60	34.8
7	0.6	-30	7.4
8	0.6	-60	7.9

Table 5. Results for test case 3.3

When the sticker was in the field of view of the front camera, the robot would take less than 10s to centre itself before cleaning. If the robot started out unable to see the sticker (facing the wrong direction), it would need to complete up to a rotation to find the sticker, then fix its rotation and position, needing up to a minute before it could start cleaning the carriage. Centering is used to align the robot throughout the carriage, and testing has shown that after cleaning, the robot managed to centre itself before reaching the next table, therefore allowing the robot to more accurately judge the distance to wall when computing joint values of the arm. Assuming LNER's London-Edinburgh route, a minute added to ensure the robot would centre itself would still allow the robot to clean 4 table seats in a carriage in 10 minutes, yet improving the robot's autonomy in the absence of a human to properly position it before cleaning.

4. Budget

Table 6 estimates the total budget for the components in our system accountable at the moment, not including battery.

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
Misc.	£10.00	1		£10.00
			Total estimated cost for system	£492.57

Table 6. Estimated budget for the system up until demo 3

5. Video

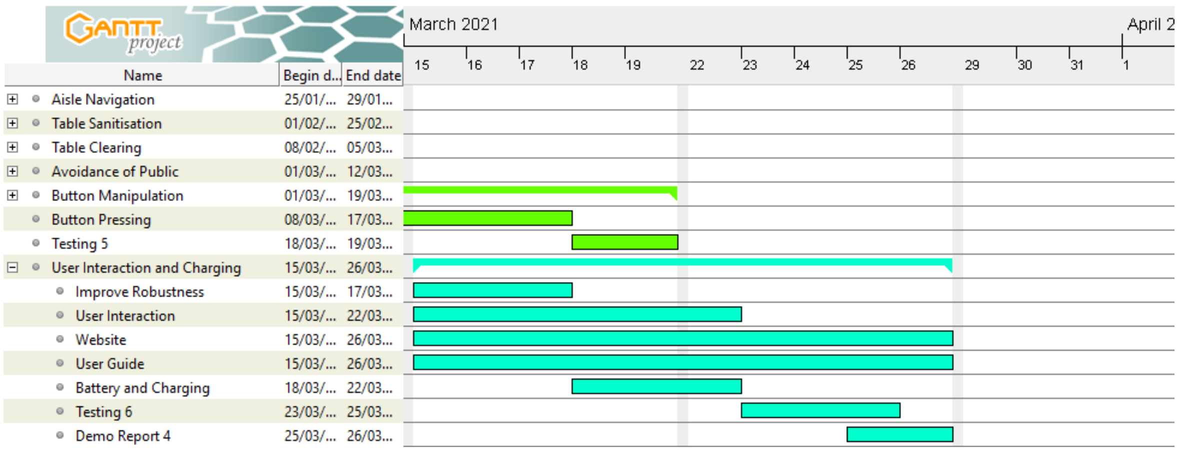
https://uoe.sharepoint.com/:v:/s/SDP2021-Group-10/EX5fwyPj8-JMqaP1FbcQ8lMBWR7Xuq60_yYnD9WsPaaUpw

References

Houston, Monica. Benchmarking TensorFlow Lite on Raspberry Pi, 6 2020. URL https://www.hackster.io/monica/benchmarking-tensorflow-lite-on-raspberry-pi-03963e.

6. Appendix

6.1. Adapted Schedule Gantt Chart



Name	Begin d...	End date
Aisle Navigation	25/01/...	29/01...
Table Sanitisation	01/02/...	25/02...
Table Clearing	08/02/...	05/03...
Avoidance of Public	01/03/...	12/03...
Button Manipulation	01/03/...	19/03...
Button Pressing	08/03/...	17/03...
Testing 5	18/03/...	19/03...
User Interaction and Charging	15/03/...	26/03...
Improve Robustness	15/03/...	17/03...
User Interaction	15/03/...	22/03...
Website	15/03/...	26/03...
User Guide	15/03/...	26/03...
Battery and Charging	18/03/...	22/03...
Testing 6	23/03/...	25/03...
Demo Report 4	25/03/...	26/03...

Figure 3. Adapted Schedule Gantt Chart