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 second demo focuses on the robot's ability to move and wipe empty tables in the carriage. In particular, the robot is able to move through the carriage while automatically detecting the presence of a table on either side and making several wipes to clean the whole table.

1. Project Plan Update

1.1. Task Progress

- Arm model [Achieved]
- Table Recognition [Achieved]
- Table Wiping [Achieved]
- Empty Table Recognition [Not Achieved]
  This is no longer required since we have reduced our scope to cleaning when there are no passengers on board (See section 1.4)
- Associated Movement [Partially Achieved]
  As described in modifications, centering currently does not work in our current environment and therefore we have added a task specifically for centering (See section 1.5)
- Rubbish vs Valuables Detection [Partially Achieved]
  This requires more testing which is now due to occur next week.
- Testing 2 [Achieved]
- Demo Preparation [Achieved]
- Item Models [Achieved]
- Rubbish Removal [Not Achieved]
  This was not achieved as we required more time for associated movement for table wiping and for testing the Table Sanitisation milestone (See section 1.5)
- Bin Robot Modification [Achieved]
  This is an added task for this demo as we decided to modify our model to include a bin for the Rubbish detection milestone.

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 previous report. Activities carried out since the last demo are as follows:

The arm model, modifications to sensor placements, and a draft model of a bin mechanism for the Table Clearing milestone were carried out by Daniel. At the beginning of the Table Sanitisation milestone, Apurv cleaned up the code-base to ensure best Python practices and allow for easier debugging. Apurv and Suhas then worked on a machine learning classifier for rubbish detection, which was later integrated into Webots with help from Máté. Anh focused on creating the main controller for Table Sanitisation and led code integration with help from Caitlin and Sean. Máté wrote a table detection module and helped integrate this into the main controller. Sean created a library of useful movement functions callable by other scripts to ease development. Austin designed arm wiping movement with help from Caitlin, and investigated the arm's ability to remove rubbish in preparation for Table Clearing milestone. Arnav, Daniel, and Sean took charge of forming test cases and testing for Table Sanitisation milestone. Finally, Anh, Caitlin, and Suhas prepared the report and video for Demo 2 with Austin carrying out video editing.

1.3. Budget Summary

The project continues to comprise solely of simulation, and so the £200 budget has not been used. A 30 minute consultation on February 5 during Benedetta Catanzarita's office hour was used to discuss the ethics of having the robot clean while passengers were on board.

1.4. Scope Changes

Our original aim for the project was to design a robot which could clean train tables during train movement with passengers on board. However, after discussing within the group and with Benedetta Catanzarita, we decided that cleaning with passengers on the train would bring up too many technical and ethical issues to realistically expect to solve within the allotted time. Instead, we have decided to focus on the robot's ability to sanitise and clear tables while train is stationary. This still provides value to customers because our robot can work in unison with the cleaners, starting from the opposite side of the carriage clearing the tables. This allows cleaners to focus their attention on other areas, reducing train's cleaning and waiting time. Also, since tables are the main touch point for passengers, this

also reduces the risk of viral exposure to cleaners.

1.5. Schedule Changes

Due to the change of scope, we have removed Avoidance of Public milestone from our schedule, allocating more time to ensuring core functionalities work properly. We added Associated Movement task to Table Sanitisation milestone to work on additional movement needed by the robot to fully clean a table and move between cleaning one side and the other. We also encountered challenges during the Table Sanitisation milestone while trying to make the arm complete sweeping and some problems with the Webots environment we were using. For this reason, we decided to give more time to this milestone to ensure all tasks could be completed. The challenges faced during the Table Sanitisation milestone have also prompted extra time given to Table Clearing milestone which is part of our core functionalities. As robot centering using distance sensors had been shown not to work in a train carriage model fully equipped with tables and chairs, we decided to add Centering task in the last week of Table Clearing milestone, which had been initially allocated to Avoidance of Public.

Our modified schedule moving forwards is displayed as a Gantt Chart in Figure 1, with the outline of Public Avoidance milestone left in to demonstrate where these tasks would have been taking place but are no longer occurring.

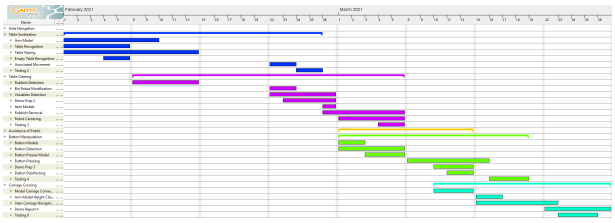

Figure 1. Adapted Schedule Gantt Chart

2. Technical details

2.1. Hardware

Figure 2 shows our current design for Railly Clean. The current model consists of existing Webots nodes such as TIAGo Base as well as our own custom designed body and arm. We plan on using the TIAGo Base throughout our simulation, but our marketed robot would be built manually using similar components. Currently all processing in the robot is carried out on a controller in Webots that we have designed keeping Raspberry Pi 3 in mind. Later milestones may redesigned the robot arm to fit through carriage doors.

2.1.1. The Base

We have continued to use the TIAGo Base for the locomotion of our robot using its 2 motorized wheels to move around, as well as built-in gyroscope and accelerometer.

2.1.2. The Body

We have redesigned our custom built body in line with the changed scope of the project. The body is built to be hollow as it will store trash collected by the robot. It uses 3 sonar distance sensors on the front and sides. We have built 2 cameras in: one is situated at the front for navigation, and one on the left side for trash recognition.

2.1.3. The Robot Arm

Our team chose to revamp the design of the robot arm, designing our own arm. The arm consists of two metal tubes with dimensions *(lengthxradius)* of *.78m*x*.02m* and *.70m*x*.02m*. The dimensions were derived from trial and error assuming a standard aisle width and table length. The longer tube is connected to the base and the shorter tube is connected to the table wiper. Each of the 3 joints in the arm has a rotational motor and a position sensor to allow the arm to clean the table. The arm mimics a digger arm. The base tube connects the robot's body to the other sections of the arm, responsible for opening the arm up to reach the table's further edge and tucking the arm in. The second, middle tube is responsible for the motion that wipes and cleans table. The last section is the cleaning head which directly touches and sanitises tables. We plan to cover the head with a sponge to provide more contact area when wiping and clearing rubbish.

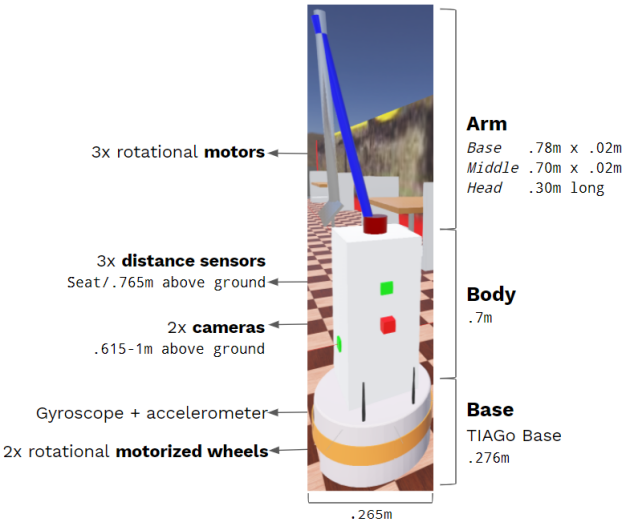

Figure 2. Railly Clean's current design

2.2. Software

2.2.1. Wiping Mechanism

The left-facing arm is designed like a digger arm that uses motors that rotate up and down at three different joints, similar to a human's arm. First step opens up the arm from a tucked in position to reach table's end touching the wall, and lowers it to the table. Second step pulls the arm in with the head slightly angled, so the cleaning sponge will still touch the table while keeping the head tall enough to touch and push rubbish from wall to edge of the table. Last step returns the arm to a tucked in position, where both the base

and middle tubes or levers are almost flushed against each other, and the head angles down at 45°. We use robotic kinematics to compute the joint values required to extend the arm and pull it in.

2.2.2. Aisle Navigation

The robot is assumed to start at one end of the carriage already centred in the aisle, moving along at a constant speed of 2.54 rad/s. Since the arm and side camera are on the left side (from the perspective of the robot), cleaning the left side is prioritized. Upon detecting a table, the robot moves backward to the back edge of the table, wipes from the wall to the table's edge, tucks in, and moves forward by the length of the cleaning head to clean from back to front. The number of attempts needed to wipe a table is computed by dividing the estimated table length by the cleaning head's length. Upon completing the left side and reaching end of carriage, the robot turns its entire body 180° to orient the arm towards right-side tables, then repeats the same series of motion. After completing both sides, the robot turns around once more to head straight to the end of carriage, ready to clean door buttons and cross carriage when such functionalities are implemented.

2.2.3. Table Recognition

When the robot is not cleaning, it scans left and right of the current row for a table that hasn't already been detected. The algorithm requires seat-level side distance sensors to measure distance the robot's distance to a chair.

Table recognition works by recording to a queue, of maximum length 3, the estimated length of chair, of space between chair and table pole, and of table pole. Without cameras to detect these, the algorithm records distance to nearest object and extrapolates from expected train carriage patterns, capping distance from robot to wall at 2m. (a) When distance to nearest object increases by >0.3m from previous, the robot has passed a solid (chair or pole) and is now next to the wall. Algorithm adds to the queue the computed length of last traversed solid given time crossing it and speed. (b) When distance to nearest object decreases by >0.3m because a solid is in between the wall and the robot, algorithm records the length of empty space just crossed.

The algorithm detects a table when the first element of the 3 in the queue is longer than the last, indicating that the robot must have passed a chair, an empty space, and a table pole in succession. This also means that the robot must move past the pole in the middle of the table to recognize it and move back to the edge of the table to start cleaning.

2.2.4. Trash Classification

We took a Machine Learning approach to classifying an item on the table as trash or valuable. The TensorFlow Lite library was optimal for us as it is designed to run on microcontrollers such as the Raspberry Pi 3 we modelled the Webots controller on. Within TensorFlow we have used the Model Maker library, simplifying the process of adapting and converting a TensorFlow neural-network to specific input data. To train the model, we use one set of images for trash and another for valuables (Thung), (Google).

3. Evaluation

For this demo report we chose to do manual logging of repeated runs, where we interchanged the variables according to different situations the robot might find itself in.

3.1. Ability to Move Backward and Forward

3 runs per test.

Test	Direction	Success
1	Forwards	√
2	Backwards	√

Table 1. Results for test case 3.1

3.2. Ability to Rotate by Different Amounts

5 runs per test.

Test	Angle	Success
1	+90°	√
2	-90°	√
3	+180°	√
4	-180°	√

Table 2. Results for test case 3.2

3.3. Upright Stabilisation

This set of tests concerned whether the robot could remain upright at different levels of extension of the robot arm with and without forwards movement. 5 runs per test.

Test	Arm Extension	Forward Movement	Success
1	Minimum	No	√
2	Minimum	Yes	√
3	Maximum	No	√
4	Maximum	Yes	√

Table 3. Results for test case 3.3

3.4. Arm Movement

This set of tests concerned whether robot's arm was able to move from full tuck in to full extension and back. 5 runs per test.

Test	Arm Movement	Success
1	Maximum Extension→Tucked	√
2	Tucked→Maximum Extension	√

Table 4. Results for test case 3.4

3.5. Wiping with Different Table Lengths (Wall-to-Aisle)

This set of tests concerned whether the robot successfully carries out a single wipe on different table lengths with robot centered in aisle. 5 runs per test.

Test	Table Length (m)	Success	Average Time Taken (s)
1	0.2	×	N/A
2	0.6	√	6.78
3	1.0	√	7.58
4	1.4	√	7.59
5	1.8	×	N/A

Table 5. Results for test case 3.5

3.6. Wiping with Different Table Widths (Chair-to-Chair)

This set of tests concerned if whether the robot successfully carries out a complete table wiping with different table widths. 5 runs per test.

Test	Table Width (m)	Success	Average Time Taken (s)
1	0.4	×	N/A
2	0.8	√	18.38
3	1.2	√	28.86
4	1.6	√	28.80

Table 6. Results for test case 3.6

3.7. Detection with Different Pole Radii

This set of tests concerned whether the robot successfully carries out a complete table wiping with different table widths: 5 runs per test.

Test	Pole Radius (m)	Success	Accuracy
1	0.01	√	100%
2	0.03	√	100%
3	0.05	√	100%
4	0.10	√	100%

Table 7. Results for test case 3.7

3.8. Robot Distance from Wall

This set of tests concerned if the robot successfully carries out wiping motions at different distances from the wall. 5 runs per test.

Test	Distance (m)	Success	Average Time Taken (s)
1	0.2	×	N/A
2	0.6	√	5.79
3	1.0	√	6.16
4	1.4	√	7.58
5	1.8	×	N/A

Table 8. Results for test case 3.8

3.9. Full Carriage Cleaning

This set of tests concerned if the robot successfully carries out a complete run cleaning all tables and travelling the full carriage twice. 10 runs per row number.

Table Rows	All Tables	Full Carriage	Avg. Time (s)
2	10/10	10/10	323.0
3	8/10	10/10	466.7
4	10/10	10/10	614.9

Table 9. Results for test case 3.9

4. Budget

Table 10 estimates the total budget for the components in our system we can account for at the moment, including parts to be used in a manually built base replacing the TIAGo Base. This does not include battery as further development is needed to determine whether charging capability can be implemented, and if so, the battery type and capacity needed to complete cleaning one or more carriages.

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	3		£97.17
EV3 Large Motor	£32.39	2	Comes with rotation sensor; for the base	£64.78
EV3 Medium Motor	£25.79	3	Comes with rotation sensor; for the arm	£77.37
GBPro-Squeegee-35cm	£12.95	1	amazon.co.uk	£12.95
Wickes.co.uk 10mm Twin-wall Polycarbonate Sheet 700x2500mm	£29.00	1	Rectangular body chassis: 0.3mx0.7m Cylindrical base: 0.265m(r)x0.276m(h)	£29.00
Plastock.co.uk Acryllic Blue Tube 1830x22mm	£25.01	1	For the arm	£25.01
LSM6DS3TR STMicroelectronics	£4.26	1	uk.rs-online.com; accelerometer + gyroscope	£4.26
Misc.	£10.00	1		£10.00
			Total estimated cost for system	£390.54

Table 10. Estimated budget for the system up until demo 2

5. Video

https://uoe.sharepoint.com/:v:/s/SDP2021-Group-10/EdiEMbylfz9MoMCYNXm5CLABYEqeC9BwPZ7gkB4K3-uF5g

References

Google. Open Images Dataset V6 + Extensions. URL https://storage.googleapis.com/openimages/web/index.html.

Thung, Gary. garythung/trashnet. URL https://github.com/garythung/trashnet.