Product: CLYDE

Team: Cloud 9

Abstract

CIYDe is a system that sanitises desk tops and other flat surfaces in high-traffic spaces, like libraries and offices. The team achieved the main goals set out for this demo, modelling a basic design for the robot base and arm using Webots. A physical design was then constructed to copy this. Both the hardware and software teams also conducted a great deal of research into other aspects of the system, a minor deviation from the initial project plan.

1. Project plan update

In our project plan, we specified the following milestones to be achieved for this demo:

Milestone Progress

- Robot arm design - Achieved
- Robot base design - Achieved
- Sensors - Achieved
- Navigation Research - Achieved

Deviations from Project Plan

- The robot scale will be smaller than first intended to accommodate for the available hardware. After speaking to Garry Ellard, he confirmed that this design would be sufficient as "proof of concept" for this robot.
- The research conducted by the software team goes beyond what was specified in the project plan by investigating both the path-finding and image processing aspects of the robot.

The following Technical Details section of this report highlights the changes made. The project plan will be updated to take into account these deviations.

Organisation

As stated in our project plan, we split into hardware and software sub-teams. The tasks for each milestone were assigned to the relevant sub-team. Team members performed the following tasks:

Sean - Investigated ROBOTIS SLAM and navigation2 algorithms, and captured video of the usage of these algorithms for obstacle avoidance and pathfinding.
Janek - Software research into obstacle avoidance, project plan update.
Ivan - Image processing research, especially in the QR code detection part; Evaluation part description writing.
Josh - Integration of ROS libraries with Webots (e.g. for navigation), and demo report writing (software and project plan update sections).
Adel - Researched parts to use for the robot. Configured LIDAR in webots to work with imported WafflePI model. Tested LIDAR in webots. Contributed to Hardware and Evaluation Section in demo report.
Ryan - Set up Webots, researched importing of Blender assets, edited demo report and designed, presented and edited the demo video.
Shining - Research on coping with varying light conditions for image processing, and implementing several relevant algorithms that segment the obstacles on table. Helped with writing evaluation part of Demo 1.
Jeffrey - Research and measurements on different kinds of library desks and helped on the image processing section. Contributed to Hardware, Evaluation and budget sections in the report. Tested the base movement and robot arm control of the wafflePI model provided by Lars.
Lars - Worked with Webots and ROS to get PROTO files for the Wafflebot and Pincher s100. Researched into ROS2 and Webots.

The following organisational tools have been used:

- Discord: This is our primary communication tool. There are separate voice and text channels for both the hardware and software teams, allowing organised communication. It also includes hyperlinks to the other tools mentioned it here. We have also established a text channel for communicating with our mentor, and for planning questions for experts in advance of meetings.
- Trello.com: This is used to track the progress of specific tasks and their deadlines. We can use cards with different priorities and statuses, and can assign target dates as well as deadlines.
- Google Drive: Any group media is stored here. The drive currently consists of note taking and research, and is neatly organised into hierarchies of folders.
- Github.com: A standard git organisation has been set up to assist in code sharing and proper version control usage. This will be our primary tool for sharing code throughout the project.
- Facebook Messenger: This is our backup option for text communication.

Organisational Split

- Full group: We hold regular full group meetings either every or every other day. We update each other here on the work done as separate hardware and software teams, and make sure the group is on track as a whole.
- Hardware: The hardware team is responsible for designing and modelling the physical robot in Webots. Due to Appleton Tower being closed the amount of workload for the hardware team has been reduced significantly. The hardware team will slowly migrate to the software team as they finish their respective tasks, due to the imbalance of the workload.
- Software: The software team is responsible for designing the navigation and image processing aspects of the robot, along with the app.

Budget and Future Modifications
Monetary budget spending so far: £0, as of 01 February 2021.
Technician time budget spending so far: 3h, setting up robot in Appleton.
Presently we do not believe any modifications to the original plan are necessary. Changes to the plan are likely to come in the following demos.

2. Technical details

2.1. Hardware

We have decided to go with the following design for the hardware implementation:

Robot Base

We will be using the TurtleBot3 Waffle Pi as the base of the robot. This is because is it readily available from the University and comes with the sensors we require, most importantly a LIDAR used for localisation and pathfinding. This device is well integrated into Webots and ROS2. This allows us to make use of existing libraries. This greatly simplifies our approach. The Waffle Pi is also modular, allowing us to easily attach additional equipment, such as the robot arm, as required. We considered using the burger version of the turltebot, but found the waffle version ti be more stable.

Arm

We are planning to use the Pincher X100 robot arm. We chose this arm due to it being already provided by the University as well as it meeting the basic requirements that a robot arm needs for the task. The 4 DOF provide more than enough mobility to rub the end effector along a 2D plane. The reach of this arm is not adequate for cleaning normal desks but due to our decision to reduce the scale of our project, it will be sufficient for demonstrating the operation of the robot as a proof of concept. Another benefit is that the arm easily integrates with the turtlebot both in terms of hardware and software. We considered constructing a custom robot arm, but after speaking to the technicians we concluded that the X100 was sufficient.

Sensors

There are two primary sensors the robot will need. The first is a LIDAR, which will allow the robot to scan the environment around it and perform SLAM. The Waffle Pi comes with a LIDAR already integrated. The other sensor required is a camera. This is needed for scanning desks, to detect QR codes for positioning the robot and to carry out object recognition to detect any 'clutter'. A good option for this is the C270 Logitech USB Webcam, which is also already provided. This camera can easily be emulated in Webots. We considered ultrasonic sensors for localisation and obstacle avoidance, but the LIDAR to be sufficient.

The camera placement is has not been finalised. It could be placed on the arm, or on a sort of pole to provide more of a 'birds-eye' view. However, this decision is not an urgent matter, since the development of the image processing can be performed independently, and such it can be decided upon later in development.

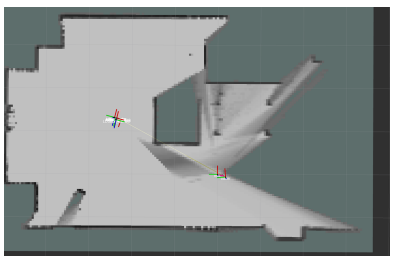
An important note is that this design is not necessarily final, and can be improved on throughout development. Since our design will be produced in a simulation, the software can easily be ported over to any new design given that the foundation remains similar.

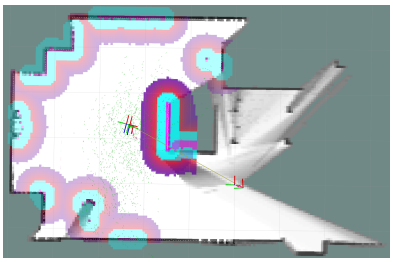
2.2. Software

The main software goal for this demo was to research the navigation method for the robot. No permanent software was due to be produced.

The software team was split into sub groups to instead research each main software milestone for the project: navigation, pathfinding, and image processing. The teams' findings are presented below.

Navigation
Navigation will rely on a pre-computed map of the environment, created either manually or by using SLAM(Simultaneous Localisation and Mapping), with LIDAR for localisation. The SLAM approach currently seems best since it allows us to produce a more general solution, not restricted to a single environment.





The above photos, captured by Sean Strain, demonstrate the usage of the ROBOTIS SLAM algorithm for navigation, showing the algorithms outputs after the robot has moved around an obstacle.(ROBOTIS, 2021) In both photos, we can see where the robot is (the red, green and blue cross) and the robot's origin (the red and green cross). In the top image, the robot has mapped what it can see from its sensors. On the bottom, the SLAM algorithm has generated a cost function which it uses to successfully navigate the robot around obstacles it has identified. We can also see a path (the yellow line) that links the origin and the current position after the move.

Both LIDAR and SLAM are already implemented for TurtleBots, which greatly simplifies this approach.

The team met with navigation expert Christopher McGreavy to discuss this approach, and he confirmed our assumptions that a precomputed map is needed.

Pathfinding

The rospy library will be used to create navigation ROS nodes with Python. This can be easily integrated with our Webots simulation. Two publishers will be required:

- One for setting the initial position of the robot to align the LiDAR scan with the map on boot.
- One for sending goal coordinates for the robot to navigate i.e. to a desk, or charging station.

The ROBOTIS navigation library allows for the implementation of different pathfinding algorithms in Webots. A* search currently seems to be the best suited for our problem (Cox, 2020).

We also discussed obstacle avoidance in our meeting with Christopher McGreavy. The robot will encounter two categories of obstacle.

- Static: these obstacles should be pre-mapped, either manually or using SLAM.
- Non-static: people, trolleys, etc.

The robot should already avoid static obstacles by considering these when calculating the path. For non static obstacles, Christoper suggested to simply wait until the obstacle moves. After a timeout, we could either consider the obstacle as static and try to navigate around, or recalculate the entire path considering the current path as being obstructed.

Image Processing

The team met with Julian Habekost, the vision expert, to discuss the best approach for detecting 'clutter' on desks. Julian suggested simple image thresholding would be sufficient provided the both appearance of each desk, and the camera perspective, are consistent.

Jeffrey Zhang managed to visit the library and obtain images of various desks under different lighting conditions. This allowed us to begin investigating some different thresholding algorithms.

Shining Liu investigated the results of various algorithms on the images provided. The results above display the application of the OTSU thresholding algorithm, designed to remove any lighting differences in images, which will likely be the basis of our image processing. (OTS, 2020)

For the QR code detection, Ivan Sun researched useful libraries. ZBar seems to be the best suited for our needs, since it is faster and more robust than the alternative OpenCV QR Code Detector. Below shows an example of successfully detecting a QR code using ZBar.(Teja & Kumaar, 2018)

3. Evaluation

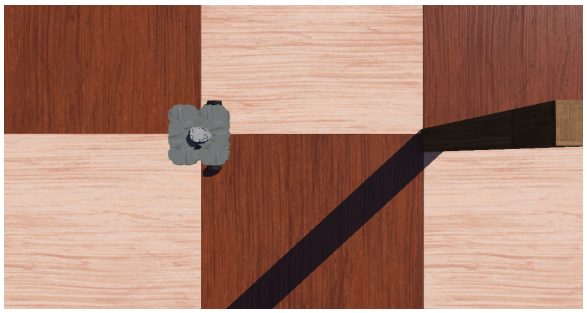
As detailed above, the hardware team have produced a simulation of the TurtleBot 3 Waffle Pi and the Pincher X100 arm in Webots. Tests have been carried out in parallel to the development. These include testing each independent component of the robot as well as its basic movement. Currently, the robot is capable of moving forwards, backwards and making a turn based on certain commands. These test have been performed on both the TurtleBot alone, and with the robot arm integrated. Testing identified problems with

the integrated robot moving forward due to changes in its center of gravity. While it could still move, the movement is inconsistent and not up to the standards that we would like. This is an issue we will face in time for the next demo.

Operation	Base without Arm	Base with Arm
move forward	Success	Failure
move backwards	Success	Success
rotate left	Success	Success
rotate right	Success	Success

Table 1. Testing the robot's ability to move with and without the arm on top of it.

For LIDAR, functionality tests have been carried out in a way such that the robot with a LIDAR sensor on top is placed in the Webots world. The arena object in Webots was used; each square was set to 1mx1m. The robot was placed on one edge of a square and a box on the other side. See the setup below:



In order to measure the readings for each direction, the robot was simply rotated on the spot so that the box would be in a different direction to the sensor. Theoretically, the LIDAR sensor should return 1 as the measuring distance no matter which direction it is facing since the object is 1 metre away. Below is the table which contains the data that we gathered in Webots. North means up in the image.

Direction Robot is Facing	Distance measured (metres)
West	0.99923...
North	0.99886...
East	0.99859...
South	0.99798...

Table 2. LIDAR readings of object 1m away in the 4 cardinal directions.

Even through noise, the sensor returned data extremely close to 1 metre. All the data has shown that the LIDAR is capable of returning an accurate distance against an obstacle and is capable of returning distance measured from any direction. This means that it should be a reliable tool that can be used in the project.

For the arm, it is segmented into three parts. Each part has motors to change the bearing arm and there is a gripper. Here we tested each of the joints individually to verify that they can carry out the range of motion that is required to clean a desk. Note that these ranges have been tested in the simulation, and that when utilised the arm will have it's joint angles adhere to the limits given so to not damage the servos.

Arm Joint	Angle Range (Degrees)	Test Result
waist	360	Success
shoulder	−90 - +90	Success
elbow	−90 - +90	Success
wrist	−90 - +90	Success

Table 3. Test results of manipulating different arm joints within required range of angles.

4. Budget

Due to the pandemic, we cannot build a physical robot so instead must accomplish our project in Webots simulator. Although we haven't spent any of the given budget at this stage, we have investigated the real cost of our robot, in order to ensure our project will be feasible. Most of the monetary budget will spent on the hardware. Currently the robot consists of the following parts, the actual prices of which are listed below:
TurtleBot 3 Waffle Pi — $1,399.00.
C270 Logitech USB Webcam — $25.99.
Pincher X 100 Robotic Arm — $549.95.
Total — $1974.94.

In terms of technical time spent, our group discussed some technical problems with Garry Ellard on 27th January. He helped us test our implementation with a physical TurtleBot via SSH on the 29th and provided us with the video included in the section below. We used all 3 hours of technician time.

5. Video

Follow the link below to watch our demo video:
shorturl.at/hqzDR

References

Otsu's thresholding with opencv. 2020. URL https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html.

Cox, Graham. A* pathfinding algorithm, 2020. URL https://www.baeldung.com/cs/a-star-algorithm.

ROBOTIS. [ros 2] slam, 2021. URL https://emanual.robotis.com/docs/en/platform/turtlebot3/ros2_slam/#cartographer.

Teja, P. R. and Kumaar, A. A. N. *QR Code based Path Planning for Warehouse Management Robot*. IEEE, 2018. URL https://ieeexplore.ieee.org/document/8554760.