Product: PAUL

Team: Group 6

Abstract

In our third demonstration, we will present a Hardware prototype implementation that includes the UV disinfection module. We will also present a basic web-based UI that is able to activate the movement of PAUL.

1. Project plan update

1.1. Demonstration Goal:

to have a Hardware prototype implemented that includes the UV disinfection module. We also hope to have a basic web-based UI that is able to activate the movement of PAUL.

- Goal Outcome: **Achieved**

1.2. Milestones

The milestones each sub-team has worked towards for this demonstration were as follows:

- Software and Simulation Team:
  - 1.4 Integrate web UI
    - Success Criteria: PAUL receives instructions from the UI and acts accordingly.
    - Milestone Outcome: **Achieved**
  - 1.5 Simulate Robustness
    - Success Criteria: ensure PAUL is robust and secure during work and for various pole diameters, capable of performing work under conditions with minor vibrations.
    - In Progress - (this is not a change of our plan this goal was always planned for demo 4)
- Web Design Team:
  - 2.3 Web UI for controlling PAUL
    - Success Criteria: a fully-functioning UI that controls PAUL's movements and cleaning actions.
    - Milestone Outcome: **Achieved**
  - 2.3 extension - Accessibility
    - Success Criteria: Adhere to the usability heuristics set by the Nielson Norman Group.
    - Milestone Outcome: **Achieved**
- Hardware Team:
  - 3.2 UV cleaning system
    - Success Criteria: the UV cleaning system has been installed onto PAUL and can be activated remotely.
    - Milestone Outcome: **Achieved**
- Project Management Team:
  - 4.1 Prepare for Demo 3
    - Success Criteria: Prepare content to present live at the Q&A.
    - Milestone Outcome: **Achieved**
  - 4.2 Complete Demo 3 Report
    - Success Criteria: has LaTeX report/video finished and peer reviewed by all members.
    - Milestone Outcome: **Achieved**

See table 1 for a task breakdown. Our group is on track and we have achieved all our goals and milestones for this demonstration and currently, and remain on track for the next demo.

1.3. Group Organisation

For this demo, we had more of a focus on Front-End and Hardware Development and less on Software-and-Simulation. As there is an overlap between many of the team members in the Software-and-Simulation team and the Front-End team we were able to have these members focus on Front-End tasks. Having an overlap between members in sub-teams has been useful in allowing us to prioritise work and allocate work fairly between team members. We have made more usage of GitHub issues and pull requests to conduct code reviews and resolve any issues found in the code. This provides a transparent and safe system of peer approval which increases the quality of our code. In the video presentation for this demo we had more team members present the areas they worked on. We found having members present the work they created ensures that it is explained accurately and adequate justifications for design choices are presented.

2. Technical details

2.1. Hardware

1. **Work with technicians to attach sensors**
   We were issued with six IR and six Bumper sensors. The IR with a range from 4 to 21.5 cm were placed; 3 on top and 3 on the bottom of PAUL's case, to give readings on the go so the position can be calculated as well as detecting possible hazards. The bumper switches had been placed on the sidewalls of PAUL to detect possible interactions with external factors, such as passengers. However, after the test run, we concluded that the Bumper sensors need to be placed in a stronger and more relevant position as they were getting knocked off. We are already working on designing 3D fixings to securely place these.
2. **Phidget Board Sensor Control code**
   We managed to successfully connect to the Phidget board attached to our Raspberry Pi and then get readings from all the Bumper/IR sensors attached to the Phidget. We used the example code given by the technicians as a reference(Phidget).
3. **Discuss safety of UV Cleaning System with technicians**
   UV light can be harmful to humans when they are directly exposed to it (FDA), we decided for safety reason to use a strip of chainable LEDs to simulate the functionality of UVC emitter's cleaning mechanism. These could be swapped by real UVC emitters when its proven that no light escapes from the UV casing.
4. **Design UV Cleaning System**
   After the technicians issued accurate measurements of the prototype, the UV casing was 3D modelled, using CAD software, and printed. The casing is ring-shaped

Milestone	Task/Aspect	Member
1.4	Create Script interfacing WebUI and Raspberry Pi	Naman
1.5	Research pole vibration	Jim, Naman
	Conduct testing	Stirling, Naman
	Conduct research on distance sensors and noise	Stirling, Naman
	Implement new UV design	Stirling
	Implement controller initialisation checks	Stirling
	Stopping period in controller	Yasmin
2.3	Attend Usability workshop	Jim, Naman, Zhiqi, Gagan
	Build wire-frame: Control Panel	Zhiqi
	Create wire-frame: Login Page	Zhiqi
	Create wire-frame: User Guide / Settings Page	Zhiqi
	Get feedback from the Ryan Bowler	Zhiqi, Gagan
	Convert wire-frames to Code (3 web pages)	Zhiqi, Gagan
	Build Flask base app and set up virtual machine	Gagan
	Accessibility feature - password eye, wrong password	Gagan
	Accessibility feature - narrator:text to speech	Gagan
3.5	Work with technicians to attach sensors	Pablo
	Phidget Board Sensor Control code	Pablo
	Discuss safety of UV Cleaning System with technicians	Pablo,Daniel
	Design UV Cleaning System	Pablo
	Help technicians install UV Cleaning System	Pablo
	Modify Raspberry Pi API to add sensors	Daniel
	Test Sensors	Pablo, Daniel
	Test UV Cleaning System	Pablo, Daniel
4.1	Prepare for the Q&A on Wednesday	Daniel, Stirling, Gagan
4.2	Demo Report & Video	Daniel, Gagan, Pablo, Stirling
	Edit video	Daniel

Table 1. Contributions of each member towards tasks

and it's empty on the inside so that light-emitting devices can be fitted, it also has 3 fixings to attach it onto PAUL. To achieve this, 3 rods with M5 threaded holes on both ends where also 3D designed and printed.

5. **Help technicians install UV Cleaning System**
   The functionality of the UV cleaning system was explained to the technicians for a correct installation of the module. It is to be placed above PAUL's case and fixed through 3 rods. For attaching it, regular M5 screws were used. The strip of LEDs is to be fitted inside of the UV ring, but the strip was too large. Hence, we were not able to attach it properly.
6. **Modify Raspberry Pi API to add sensors**
   To allow PAUL to detect if it was close to the top/bottom of the pole we had to integrate the Phidget sensor code into our robot controller. The code was successfully integrated into the controller and we were able to take readings from the sensors during our test runs. We are using a combination of both bumper and IR sensors to detect when PAUL reaches the top or bottom of the pole. We use the IR sensors to get a distance reading from PAUL to the top/bottom of the pole. Since the IR sensors are not that accurate we decided to take an average of the 3 sensors we have on each side and then compare this value against the threshold reading needed to make PAUL change direction. We also use bumper sensors as a safeguard in case the IR sensors should fail to detect the distance accurately, if any bumper sensor on the side in which PAUL is currently moving is triggered PAUL will change direction.
7. **Test Sensors**
   To test the Phidget Board Sensor Control code before integrating it into PAUL controller we conducted a test run with the help of the technicians. To check the IR sensors we took readings when they were at various distances from a target to determine a threshold reading. We also had the technicians manually trigger each bumper sensor to check that they were working as expected. For more details see the Evaluation section.
8. **Test UV Cleaning System**
   For this test we planned to test if the LED could be turned on and off when specified by the controller however the LED light strip was not installed correctly in time for our test run. However we were still able to test PAUL's climbing ability after adding the UV casing. This test was necessary to ensure the added weight and design of the UV casing did not impact the climbing ability of PAUL. For more details see the Evaluation section.

2.2. Software

1. **Integrate Web UI - Back-end**
   We produced an ssh script which connects to the Raspberry Pi on the hardware model, through the front-end web app. Although, we did experience some problems due to security in the university's network as we had to be inside the network to run the script. This would not be an issue commercially so we focused on other areas of work.
2. **Quantitative Analysis**
   The majority of the work of this team focused on QA. We continued to explore ways in order to ensure the robustness of PAUL. Exploring adding noise to the sensors which PAUL uses. We experimented using different implementations of distance sensors in Webots. This is explored more fully in the Evaluation Section. We started conducting research on how to implement vibrations within the pole for the final demonstration. Ultimately we think this will be extremely technically challenging but we believe it to be possible to use a physics plugin in C++.
3. **Controller Changes**
   Through our QA at demo 2, we discovered a number

of issues and potential features which could be easily implemented within our controller. We added a check on start-up which ensures that operation will not begin if distance sensors are reporting values within their threshold initially. Options were added allowing PAUL to stop for a set period of time after completing a set number of tours (a full climb and descent).

4. **Structure Changes**
   Finally, using the drawings produced by the hardware team, we made changed the UV module to coincide with the physical design of PAUL, by adding an enclosure around the UV emitter.

2.3. Web User Interface

1. **Gather requirements and build wire-frames**
   The front-end team collaborated with other sub teams to gather all the requirements, this included data that should be displayed on the UI and variables the user can manipulate. After this we use Balsamiq Wireframe (bal) to make three wire frames. We built a first-draft and got it reviewed by SDP's Usability expert, and iterated to build a final product.
2. **Flask Back-end App**
   Before demo two, our team experimented with different web frameworks and decided to go with flask for its easy-to-use capabilities, and considering the small scale of our project. Flask allows us to scale our project through different web services without dealing with much of its own boiler-plates. We built a flask app and set it up in a virtual environment for easy packaging and build capabilities. We used JavaScript for any DOM manipulation on the front-end side and used Python for running any back-end scripts.
3. **Login and Settings Page**
   We built a basic login page and iterated based on other team member's feedback and expert's feedback. Our login page now consists of two input boxes, one for username and one for password followed by a login and "password reset" button. We have made this more inclusive by adding an eye button beside the password input-box. This allows the user to see the password in a text form. Other than this, we provide also both visual and text-based feedback when the user enters a wrong password.
4. **Control Panel and Settings Page**
   The user will use the Control Panel page to control the deployed PAUL-bots. The Control Panel consists of an interactive object of a train which allows the user to control PAUL interactively. It also has a feature to control all robots at once which can be used for scheduled cleaning. The Control panel has a pop-up settings page where the user can customise both PAUL-bot settings and different accessibility and aesthetic settings. In the PAUL-bot section, the user can manipulate UV light, autopilot mode, time, and speed. The accessibility section, allows the user to customise the Control Panel to their liking. This includes a mixture of essential accessibility features and aesthetics features like High-Contrast Mode, Narrator, Vibration Feedback, Light-Mode, and Magnifier.
5. **Inclusivity in User Interfaces**
   After completing the functional requirements of our web UI, our team focused on non-functional features like inclusivity of our User Interfaces. Our team learnt about it through usability workshop, met the usability workshop, and brainstormed to find different ways to make our application more accessible. We cross-checked the accessibility guidelines mentioned in the w3 - Web Accessibility Guide. (w3-) Other than this, we brainstormed ideas to adhere to the NN Group's usability heuristics. (nn-) We provide text alternatives for every graphic feedback. It is easy-to-understand, has minimal Jargon, and has multiple "i" icons which helps the user understand the functionality of different buttons. We built a predictable and consistent design that is easy to navigate. We go into details about this in the demo 3 video.
6. **Qualitative User Testing**
   Our team conducted different types of testing to ensure our product has the highest standards. At first, we ran some regression tests. We ran and tested our Web App on different screen resolutions, browsers, and operating systems, and evaluated it qualitatively. We fixed bugs quickly for specific browsers and added code to deal with them. We also had peer reviews with other sub-team's members who criticised our design and helped us make our interface more intuitive and remove jargon from it. We plan to have a user survey for demo four. This will be part of our User Acceptance Testing.

2.4. Marketing Website

Throughout the semester we have been working on the marketing website. For this demo we just cleaned up some simple things and mainly focused on the Web UI. We aim to add all contents to our site for demo 4.

3. Evaluation

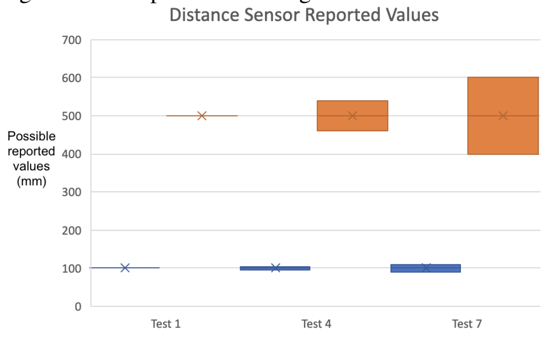
3.1. Webots Testing

Test	Noise <= 100mm (%)	Noise at 500mm (%)	Pass
1	0	0	√
2	1	2	√
3	2	4	√
4	4	8	√
5	6	12	√
6	8	16	√
7	10	20	×

Table 2. Distance Sensor Noise Tests

Values reported by the distance sensor are arbitrary and set by the developer. For this simulation, we set the sensor to report in millimetres. Noise is added, allowing the value reported to be within a percentage range of the actual value. In order to emulate a real-world environment where more noise may be introduced the further from the surface the sensor is, we set the noise at 500mm away to be double that of the values under 100mm, values between 100mm and 500mm are scaled linearly and the sensor does not report values above 500. 100mm is an estimation of how high we expect may be necessary to accommodate for clearance of PAUL and curvature of surfaces near the top of the poll. See Figure 1 for the possible range of values reported by a sensor under different noise settings. We defined success as being able to perform 5 tours of the pole successfully cleaning all areas of the poll outside the 100mm clearance at the top and bottom.

Figure 1. Example of value ranges at 100mm and 500mm



All distance sensors work by emitting rays and performing calculations based on them, so we varied the number of rays, weights on the values returned by rays and the arrangement of those rays. We continue to discuss the economic feasibility of these alternative methods and have not yet decided on the optimal implementation for PAUL. These results suggest that PAUL is robust to some noise, but we believe that through further experimentation with these settings we may be able to improve robustness in environments with substantial noise, e.g. trains. Next we would like to more accurately quantify the relationship between the amount of noise introduced and the distance from the surface.

Friction Testing: We tried as much as possible to test our robot under different friction settings. Due to the implementation in Webots, friction can only be varied between 1 and infinity, making it extremely difficult to draw conclusions about what the values would represent in the real world. We did not see a suitable way to provide testing data for this. Although, we are generally satisfied that PAUL will be able to climb under different friction settings and believe this would be easier to test with our hardware version.

3.2. Hardware Testing

Hardware Sensor Test: Before attaching the sensors onto the Hardware Prototype casing we tested if they were functioning properly. To check the IR sensors we took readings from the IR sensors when were close (4cm) and far away (21.5cm) from a target. The values 4cm and 21cm were chosen as these are the minimum and maximum ranges of our sensors, so should produce the highest and lowest outputs respectively. We tested the bumper sensors by having the technicians manually trigger each bumper sensor to check that they were working as expected. Success criteria: check that the sensor outputs we received from the Phidget board were within the expected range (as specified by the technicians). We checked that when the bumper sensors were triggered they were equal to 1 and when not triggered they were equal to 0. We checked that the IR sensors close to the target were above the threshold and sensors beyond the maximum range from the target were below the threshold. Note: BS = Bumper Sensors.

Test	Description	Success
1	No BS triggered/21.5cm from target	√
2	Bottom BS Triggered	√
3	Top BS Triggered	√
4	Top & Bottom BS Triggered	×
5	Top & Bottom BS Triggered	√
6	Bottom IR 4cm from target	√
7	Top IR 4cm from target	√
8	Top & Bottom IR 4cm from target	√

Table 3. Sensor Test Runs

Note: On test 4 that the tape holding down one of the bumper sensors during the test fell off and caused it to not be triggered. In test 5 we repeated test 4 with the tape attached properly.

Conclusions: The IR sensors gave a high reading when they were close to the target and a low reading when they were far away as expected. Bumper Sensors were triggered when activated by the technicians as expected.

UV Cleaning System Test: We conducted test runs on our Hardware prototype with the UV cleaning module installed. To start each test we remotely connected to the Raspberry Pi and executed the controller program. A technician then recorded the test run and sent the video back to us. Our success criteria for each test was to check the video and see if PAUL could climbing the pole starting at the bottom without slipping back down and stop before it reaches the top. We also checked that the integrity of the UV casing was maintained throughout the test.

Test	Motor Power (%)	Success
1	100	×
2	90	×
3	80	×
4	70	√
5	60	×

Table 4. UV Cleaning Module Test Runs

Conclusions: At 80,90,100% we faced the same problem as last time with the motors to causing the drive shafts of the motors to become detached which compromises the integrity of PAUL and causes it to fall back down the pole. With its new added weight PAUL was able to climb the pole at 70% without slipping back down the pole. At 60% the motors did not have enough power to climb the pole and PAUL got stuck at around a third of the way up. We encountered no issues with the integrity of the UV casing during these tests.

4. Budget

Table 5 shows our current cost estimate of our Hardware System. All the items except the Metal Pole and Pole Holder were items that are available to all groups and as such our we have only spent £18 of our £200 budget.

5. Video link

SharePoint
Back Up: Youtube

Item	Quantity	Cost per item (£)	Source
Raspberry Pi 3	1	30	(Pricelist)
Lego NXT Motors	3	32.39	(Pricelist)
Encoder board	1	10	(Pricelist)
Motor board	1	10	(Pricelist)
Batteries (AA)	8	1.20	(Pricelist)
Lego Technic Case	1	6	Estimate
Phidget	1	93.31	(Pricelist)
Sharp Sensors	6	6.22	(onecall)
Bump Sensors	6	TBC	-
Metal Pole (60cm)	1	18	(metals4u)
Pole Holder	1	TBC	-
Wires and Misc	1	5	Estimate
Total		**316.40**	

Table 5. Budget Expenses

References

Balsamiq wireframing tool. URL https://balsamiq.com/wireframes/.

Usability heuristics by the nn group. URL https://www.nngroup.com/articles/ten-usability-heuristics/.

Web content accessibility guidelines. URL https://www.w3.org/WAI/WCAG21/quickref/.

FDA. Uv lights and lamps. URL https://www.fda.gov/medical-devices/coronavirus-covid-19-and-medical-devices/uv-lights-and-lamps-ultraviolet-c-radiation-disinfection-and-coronavirus.

metals4u. Approved supplier. URL http://www.inf.ed.ac.uk/teaching/courses/sdp/SDP2020/sdppricelist.pdf.

onecall. Approved supplier. URL https://onecall.farnell.com/sharp/gp2y0a41sk0f/sensor-distance-analogue-output/dp/1618431.

Phidget. Phidget example code. URL https://phidgets.com/docs/Do_Things_with_the_Channel.

Pricelist, SDP. Sdp price list. URL http://www.inf.ed.ac.uk/teaching/courses/sdp/SDP2020/sdppricelist.pdf.