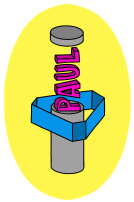
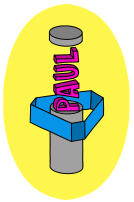


Product: PAUL
Team: Group 6



Abstract

In our second demonstration, we will present a basic hardware prototype that is successfully able to climb a straight pole with a consistent diameter. PAUL and the pole have been built by the technicians in the lab.

1. Project plan update

1.1. Demonstration Goal

Our goal for this demonstration was to have a basic hardware prototype implemented that is successfully able to climb a straight pole with a consistent diameter. We hope to have PAUL and the pole built by the technicians in the lab. We also aim to continue the development of our software in the Webots simulator and hope to demonstrate the UV cleaning process in Webots.

- Goal Outcome: **Achieved**

1.2. Milestones

The milestones each sub-team intended to complete for this demonstration were as follows:

- Software and Simulation Team:
  - 1.2 Simulate case design
    - Success Criteria: the simulated robot should look the same as our design.
    - Milestone Outcome: **Achieved**
  - 1.3 Simulate the concept of the UV module
    - Success Criteria: simulated UV module should be functioning stably when turned on.
    - Milestone Outcome: **Achieved**
- Website Team:
  - 2.2 Website blog system
    - Success Criteria: website has a running blog page.
    - Milestone Outcome: **Achieved**
- Hardware Team:
  - 3.2 Attach the required hardware components
    - Success Criteria: provide technicians with any information they need to attach the hardware components to the case.
    - Milestone Outcome: **Achieved**
  - 3.3 Install and integrate our software onto the prototype
    - Success Criteria: the software to control PAUL has been successfully transferred over to the Raspberry Pi.
    - Milestone Outcome: **Achieved**
  - 3.4 Robot can grip and move up a pole
    - Success Criteria: PAUL is able to move up and down the pole without slipping.
    - Milestone Outcome: **Achieved**
- Project Management Team:
  - 4.1 Prepare for Demo 2
    - Success Criteria: Prepare content to present live at the Q&A.
    - Milestone Outcome: **Achieved**
  - 4.2 Complete Demo 2 Report
    - Success Criteria: has LaTeX report/video finished and peer reviewed by all members.
    - Milestone Outcome: **Achieved**

See table 1 for a task breakdown. Note that some work was also done towards milestones 2.3 and 2.4, these are milestones due for the third and fourth demos respectively. 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. The only change to our plan since the last demo is in regards to the WebUI for Webots. (for more details see the Software section)

1.3. Group Organisation

Our team structure has not changed since the last demo and we have found that dividing work amongst sub-teams is much more manageable than trying to split it between the whole group. Having team leaders has also proved useful as it has ensured that someone is accountable for keeping track of the progress of each sub-team and reporting this progress back to the rest of the group.

We continued to hold weekly meetings to understand wh at progress has been made and what work is soon to be undertaken by each team. This also provides a forum for all team members to voice their ideas or concerns.

Joint team meetings have been facilitated when appropriate. For example, the Software and Hardware team met to discuss the Software integration onto the Hardware and an standard controller (for more details see the Software section). Coordination between sub-teams has proved useful in resolving issues and improving reliability of our system.

2. Technical details

2.1. Hardware

For this demo we focused on setting up the environment and the hardware prototype to climb up and down a metal pole, as well as discussing further hardware development with the use of distance sensors. Below are the tasks as associated with milestones.

1. **Discuss optimal hardware components with technicians**
   On the first test run multiple weak points in the Hardware were detected, the Lego casing was pulling itself apart. We had multiple meetings with the Technicians, regarding improving the chassis's robustness. Such as pegging plastic sheets onto the case.
2. **Purchase hardware**
   There is no need to purchase hardware as the components needed where already available but in the future

Milestone	Task/Aspect	Member
1.2	Adjust dimensions of existing solids	Stirling
	Add new structural components	Stirling
	Add simulated hardware components	Stirling
	Adjust material / colouring	Stirling
1.3	Add Structural Supports	Yasmin
	Add LED	Yasmin
	Integrate LED with Controller	Yasmin
	Conduct UV research to determine wheel speed	Yasmin
2.2	Website Blog System	Gagan
2.3	Research Backup and emergency sequences: State Machine Implementation	Gagan
	Accessibility	Gagan
2.4	System Page and Comments Page	Jim
	Evaluation Page	Naman
	Budget Page	Gagan
	How it works page and Teams page	Zhiqi
3.2	Discuss optimal hardware components with technicians	Pablo, Daniel
	Purchase hardware	Pablo, Daniel
	Attach required hardware	Pablo, Daniel
3.3	Discussion with the software team	Pablo, Daniel
	Set up Remote Access to Raspberry Pi	Pablo, Daniel
	Resolve issue with Raspberry Pi Motors running at the same time	Daniel
	Redesign controller and create a standard controller for both environments	Daniel
	Ensuring software works with hardware	Daniel
3.4	Test run the robot on the environment	Daniel
	Redesign base of pole environment using CAD	Pablo
	Start working on getting readings from sensors	Pablo
	Discuss possible ways to resolve hardware issues with technicians	Pablo, Daniel
4.1	Prepare for the Q&A on Wednesday	Daniel, Stirling
4.2	Report/video for Project Plan Update Section	Daniel
	Report/video for Software and Evaluation Sections	Stirling
	Report/video for User Interface Section	Gagan
	Report/video for Hardware and Budget Sections	Pablo, Daniel
	Edit video	Daniel

Table 1. Contributions of each member towards tasks

we will need to purchase UVC emitters from approved suppliers.

3. **Attach required hardware**
   After issuing the 2D plans and measurements of the PAUL prototype, the technicians built it with a lego case, and attached the raspberry Pi, 3 NXT motors, encoder and motor board. All connected to a 8AA battery power-bank. Afterwards, the 8/8/8 Phidget board was added to allow for multiple sensors inputs.
4. **Discussion with the software team**
   We met to discuss how to standardize the hardware prototype with the software simulation team. Pictures and 3D models of PAUL were issued to the software Team. Hence, accomplishing a similar representation of the product on the simulation.
5. **Set up Remote Access to Raspberry Pi**
   Our main reason for choosing a Raspberry Pi over an Ardunio was to allow us to easily remotely connect to our robot. After managed to successfully connect and transfer files over to the Raspberry Pi by following the steps on this tutorial (rpi guide).
6. **Resolve issue with Raspberry Pi Motors running at the same time**
   During our first test run we encountered issues with getting all three motors running at the same time. We discussed this issue with the Technicians who suggested we changed the order in which the motors are activated. This resolved the issue and we ran another test run to confirm that all motors were running.
7. **Redesign controller and create a standard controller for both environments** We have created a controller program in Python which is used to specify what actions PAUL should perform but this program does not specify how each action should be achieved by the separate implementations. Each implementation then has its own API that contains a set of function calls that specify how to perform a function in that implementation. This ensures that both implementations use the same control logic and should allow us to integrate future components from the Webots to the Hardware version more easily.
8. **Ensuring software works with hardware**
   We were able to remotely access our Raspberry Pi to run the motors and receive speed readings back. The technicians then confirmed to us that the all the motors ran and then stopped when we executed the program remotely.
9. **Test run the robot on the environment**
   To ensure our software controller worked with the Hardware Prototype we organised several test runs with the technicians. The results of these tests runs are in the Evaluation section.
10. **Redesign base of pole environment using CAD**
    We discussed with the technicians about a possible

box so PAUL could start directly on the pole surface instead of the 3d printed pole fixing surface, which was wider than the pole (due to 3D printing issues). The team designed the 3d models of the box lid and walls which would be issued to be cut on MDF sheets.

11. **Start working on getting readings from sensors**
To resolve issues with position and possible interaction with external factors, we will use SHARP infrared sensors that produces an analogue output signal. As well as bump sensors with a digital output producing a 1 if is pressed. We are still waiting for the sensors to be attached on to the prototype but hope to demonstrate the sensors in the next demo. We used the Phidget Webpage as a guide (Phidget).
12. **Discuss possible ways to resolve hardware issues with technicians**
Driving the motors at high power will cause PAUL multiple weak points on the Lego casing to disconnect. We had already identified the aforementioned plastic sheets as a potential solution to this. Furthermore, the motors do not turn on at the same time, causing for the base of the chassis to tilt. Using a motion sensor such as a gyroscope would help on stabilization, giving readings on X, Y, and Z positions, allowing for accurate on the run corrections making motors to drive at higher or lower power.

2.2. Software

Our focus for this demonstration was bringing the simulation in line with the hardware, but also designing and implementing a UV module as a proof of concept. Below are the tasks as associated with milestones.

1. **Adjust dimensions of existing solids**
Using the technical drawing provided by the hardware team we changed parameters of solid objects in the simulation to match that of the hardware version.
2. **Add new structural components**
The hardware version differs slightly from our initial prototype we added supports and new solid objects to bring the simulation in line.
3. **Add simulated hardware components**
Added sections which were not necessary for the simulation's function but are present on the hardware board e.g. the Raspberry Pi.
4. **Adjust material / colouring**
We ensured that all sections of PAUL were coloured and textured as they would be in the hardware version.
5. **Add structural supports**
Since the hardware team had not yet designed the UV case, we designed a simple structure to hold the UV LED as it climbed.
6. **Add LED**
Added a small LED which surrounded the entire pole, allowing for the entire circumference to be cleaned with each cycle.
7. **Integrate LED with controller**
Making sure that the UV turned off when errors were experienced and was otherwise activated during a cleaning cycle.
8. **Conduct research to determine wheel speed**
Conducted physics calculations to determine how fast PAUL needed to climb in order to make sure each section of the pole was cleaned fully.

Additionally, although we had initially planned to integrate the Web UI into both hardware and software versions, our research suggests that this may prove to be extremely difficult due to the number of different servers which are necessary to facilitate these interactions and our lack of computational / hosting powers for these. We made this decision having considered that ultimately, the technical difficulty in the implementation of this would not be present in the hardware version due to easier methods of communicating with the onboard Raspberry Pi in comparison to Webots.
Instead, we focused on implementing the Webots controller API, encapsulating all of the available functionality in simple methods. We did initially experience difficulties due to different expectations for method behaviour between Webots and the Raspberry Pi versions but we have largely resolved this now and continue to work to iron out any remaining issues.

2.3. Web User Interface

Our focus for this demonstration was bringing the simulation in line with the hardware, but also designing and implementing a UV module as a proof of concept. Below are the tasks as associated with milestones.

1. **Flask and React Setup**
The team voted to choose flask as our back-end application due to its light-weight functionality as a framework along with its beginner-friendly guide. For our front-end we chose React for its rich-new features, and flexibility.
2. **Accessible User Interface**
To improve the accessibility in our services, the front-end team decided to work on two new features: Text-to-Speech Services. We tried and used Microsoft's Azure Text-to-Speech services for this. Other than this, we tested out a grayscale version of our interface.
3. **Emergency Back up Procedures**
As discussed in the project plan, we intend to log all actions taken by the operator to understand the cause of any faults, and establish emergency sequences incase our device stops working. We think its better to use SQL as a database as it gives us a table like structure and we could leverage its ACID properties.

2.4. Marketing Website

We achieved all our goals for our marketing website for this demo. They're listed below.

1. **Add content to all our marketing website**
We split up the contents of our marketing website among the team, and everyone individually worked on 1-2 pages.
2. **Use informatics domain** We migrated our website from GitHub pages to the informatics site successfully.

3. **Publish Blogs** Our project management team has been documenting the process of creating new blogs every week. We published all our blogs to our website now.
4. **Adhere to the marketing website template** We added new web pages and changed structure as we thought would be relevant to our website.

3. Evaluation

3.1. Webots Testing

For our webots testing. The following settings are considered to be default and are only changed if specified in the tests below. Pole Height: 2m, Up/Down wheel speed: 0.3 rad/s. Distance Sensors thresholds are 1000 - the measurement this is in is unclear according to the documentation, we are currently trying to work it out. The UV light is engaged by default, and starting positions are maintained due to a saved state of the simulator.

Sensor Threshold	Pass
150	√
300	√
600	√
1200	√
2400	√

Table 2. Distance Sensor Test

Conclusions: we had not adequately measured the size of the modules above the distances sensors, potentially causing collisions and indeed there was no defense mechanism in place preventing startup if both distance sensors reported values within their threshold at initialisation. We will correct both of these with a small patch to the controller.

Up/Down Speed (Rad/s)	Time without error (h:mm:ss)
0.1	> 4:00:00 (no error)
0.3	> 4:00:00 (no error)
1	> 4:00:00 (no error)
2.5	> 4:00:00 (no error)
5	0:06:30
7.5	0:01:13

Table 3. Extended Operation Test

Results show that PAUL runs remarkably well at and far beyond our expected operation speeds. We again implemented a small patch allowing for the simulation to halt for a set period between cleaning cycles to minimise downtime. Taken holistically, these tests amongst others on speed, breaking, friction and height, show that PAUL consistently operates well on parameters near what we estimate to be normal operation. We believe PAUL able to function without major complication in an environment similar to that of a train but continue to add functionality to the controller and fix bugs in order to ensure the simulation justifies a wider range of use cases. (We plan to make all our testing results available on our website)

3.2. Hardware Testing

We conducted test runs on our Hardware prototype with the help of technicians. 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 was climbing (or descending in the final test) the pole and coming to a stop without slipping back down. These videos are included in the our demo video.

We conducted test runs on two different dates. On the first date we were not aware that the high motor speed was causing issues and as such didn't change the motor values, this was resolved on the second date.

Note: The motors use a percentage value to control the speed of the motors, within the range [-100,100], positive represents climbing and negative represents descending.

Conclusions: PAUL is not able to climb the pole with motor power of less than 60% as its wheels appear to not have enough power to lift PAUL up the pole. At 100% the motors appear to cause the drive shafts of the motors to become detached which compromises the integrity of PAUL and causes it to fall back down the pole. PAUL was able to climb the pole at 70% without slipping back down the pole. We intend to carry out further Motor Power testing for the next demo to see if there is an 'optimal' value.

Date	Test	Motor Power (%)	Success
19/03/21	1	100	×
	2	100	×
	3	100	×
	4	100	×
	5	100	×
26/03/21	1	50	×
	2	60	×
	3	70	√
	4	70	√
	5	70	√
	6	-70	√

Table 4. Hardware Test Runs

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.

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

5. Video

Available using this link: SharePoint Link.

References

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 guide. URL https://www.phidgets.com/docs/Main_Page.

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

rpi guide, SDP. Rasperry pi tutorial. URL http://www.inf.ed.ac.uk/teaching/courses/sdp/SDP2020/sdp_rpi_guide.pdf.