Product: FarmFriend
Team: The FarmFriends

Abstract

FarmFriend empowers farmers by digitising farms, giving them precise and actionable soil health data in real-time for the entirety of the field. In this demonstration, we brought the mechanisms from the previous demos together, along with demonstrating a functional autonomous navigation system of the robot, that is capable of navigating a small replica of a farm layout. In order to achieve this we added ultrasonic sensors, wheel encoders, and an inertial measurement unit (IMU). The connection between the web application and the robot was finished, allowing the web app to show the data collected by the robot. All aspects of the initial project plan have been achieved at this stage of the project, with the system functioning as originally intended. We added a camera to send pictures of the crops to the web app to allow visual monitoring .The reactionary system was finished, with the valves being added, and the overall layout and construction of the robot improved by 3D printing custom parts.

1. Project management update

Due to the previous demonstration's delays in acquiring valves, we had to shift the full sensor and reactive system to demo 3. During our prototyping we encountered issues with the first set of valves, discovering that they required ambient pressure to function as they were solely intended for liquids. After this we switched to a valve that has a minimum pressure of zero, which allowed it to be opened and closed in our demo as we cannot use actual chemicals or salts since this would make the lab unclean.

Throughout the several weeks of prototyping between demo 2 and 3, the reactive subteam encountered several issues with components. Two weeks before this demo both the arduino and the motor board had failures. The arduino did not accept uploads from the Interactive Development Environment (IDE) and when connected via usb, no device could be found. The motorboard also stopped working, although the required voltage for the valves was passed through the Inter-Integrated Circuit (i2c) connection to the power shield did not function. Furthermore, the Thursday before the demo the entire reactive sensor system did not operate due to faults in the new motor board. However, all these issues were resolved with new components.

The navigation team also encountered issues. The Tuesday before the demo the prowler base suffered significant problems, with some motors not working, and the prowler only able to travel a short distance before slowing to a stop. This was resolved by replacing one of the motor boards, and changing the cable in one of the motors. This set the navigation team back by a day, but the navigation was still able to be completed.

The navigation team added ultrasonic sensors to the robot to allow the robot to sense when it was approaching the edge of the arena. If one of the side sensors went below a certain value, it meant the robot was veering off to one side, so a slight steer to the opposite side to straighten out and return to the middle of the lane was implemented. The robot can detect when it is approaching a corner and will turn accordingly. This was implemented along with the wheel encoders that detect the distance travelled. The wheel encoder data is used to determine how far to travel before stopping to take sensor readings and react accordingly. The IMU is used to detect slippage of the robot using the accelerometer, and the gyro feature could also be implemented to indicate if the robot has capsized.

The web app contained UI components to display data to the user, as well as allow for interaction with the robot. The user can play / pause the robot, and select different plots of the farm to see specific sensor data. The grid was shaded depending on the current data view, with a colour gradient for pH and a transparent blue for moisture. The farm grid's layout was also designed in CSS to accommodate for the S-shaped path the robot would take through the field. The most significant work done was on asynchronous communication between the control script on the pi and the web app. This was accomplished using a websocket protocol called Socket.IO which has libraries in python and javascript. The correct version of the library had to be updated on the pi so that the protocols matched. Then, the models representing the robot's internal state and farm data were also implemented on both sides, and the data context was written to hold all the relevant variables inside the webapp.

We split into several sub-teams to effectively distribute the work. The navigation team remained the same as before with Niamh, Alin, and Niveda being the members. The reactionary team was the previous members of the sensor team, Andrew, Charles, and Xingchen. Suryansh continued to be project manager, and additionally worked on creating the wheel encoders for navigation. Gabriel worked on the web application. We used the ClickUp site to effectively track the progress of our various tasks and sub-tasks, and regularly committed and pushed to the group GitHub to integrate our code. In addition, this helped us to review other sub-teams' progress. We conducted regular twice weekly team meetings to discuss and update progress. The team meetings also encouraged collaboration and idea sharing between sub-teams.

This is a rough outline of the work that was completed by each team member throughout the project, while hours differed for each member, we feel that everyone put in an acceptable amount of effort.

The Project Manager:

- Suryansh has been responsible for ensuring that each individual sub-component of the robotic system ties in together to create the final working product. This has involved numerous tasks, such as planning, managing, and allocating tasks to each of the sub-teams in order to ensure that each team has goals to work towards throughout the entirety of the project. Since this responsibility encapsulates all aspects of the project, Suryansh has been able to make contributions to each of the sub-teams, helping resolve blocking issues. For example, Suryansh was solely responsible for the concept of the wheel encoders, designing them, and mounting them onto the system; as well as analysing and collecting data to show the improvements that the wheel encoders have brought. Other contributions include: the demonstrable concept of the navigation system, with the use or ultrasonic sensors, and the 'playground' layout for the final demonstration. Since Suryansh had a very good grasp of each component of the system, he also made the slides for each of the first, second, and third demos, as well as presented for all of the demos. He has also made significant contributions to the project reports, with a focus on the quantitative analysis section.
  Total Approx. Hours: 235

Moreover, from the Navigation team:

- Alin helped with setting up Roboting Operating System (ROS), looking into which packages are used by the Turtlebot (as those packages include ones that interface with motors and sensors, which we undoubtedly would have needed), and helped the Sensor team with building the robot prototype. Moreover, Alin helped set up the Raspberry Pi camera, starting with using a simple lane detection program which detects straight lines of a certain colour and publishing the result through ROS. He also helped research ROS navigation modules. He also helped with considering how to remotely send messages to the Raspberry Pi through the web application. Finally, Alin helped with the integration of the IMU within the navigation system, with the integration of the ultrasonic sensors in the ROS system and making the robot use those readings to navigate accordingly, including stopping, turning and stirring in the right direction (with the intention of correcting its course). He has also worked alongside the navigation team and Gabriel with making sure data is properly sent from the web application to the robot and vice versa.
  Total Approx. Hours: 217

- Niamh began with the initial connection of the motors to the pi, and the basic motor control needed for movement. She then wrote the original ROS publishers and subscribers for communication within the ROS framework. Much time was spent with the research and start of implementation of built in ROS navigation systems, such as Simultaneous Localization and Mapping (SLAM), though this was not used going forward. Also, she wrote nodes that allowed the camera to detect moving objects, and those of a certain colour, these were also not used as the core navigational system was altered from the original idea. She assisted with the implementation of the autonomous navigation system. This involved helping with the writing of code for the navigation, and testing of the robot. Some time was spent fixing the construction of the robot itself, and improving the structure of lego components.
  Total Approx. Hours: 183

- Niveda helped design the autonomous navigation of the system. She helped with the integration of ROS with the robot, to help the robot use the data collected from all the sensors to autonomously navigate through the farm. She helped set up the raspberry pi by installing the required ROS packages. She researched various ROS packages that could help with the system navigation. She spent a lot of time trying out various ROS packages and checking whether these packages could be of any use. Initially, we were thinking of using a GPS system to help the system navigate across the field. To try that out, she implemented a fake GPS (IPS) using the pigeon eye cameras. We did not go ahead with using the IPS as we came to know about the fact that using a GPS wouldn't be as feasible and accurate as it should be for a small robot such as FarmFriend. She wrote code to set up proper serial communication between the Raspberry Pi and the Arduinos. She programmed the arduinos to send data to the Raspberry Pi in a usable format. She then set up topics to collect data from the IMU, ultrasonic sensors, camera and wheel encoders using the ROS framework, which was used to implement the autonomous navigation of the system. Finally, helped with testing and fine-tuning the autonomous system navigation.
  Total Approx. Hours: 200

Progress with Web App was:

- Gabriel worked on the web app, which handled user interaction with the robot and displayed data collected from the sensors. Also worked on was the websocket connection between the script on the pi and the web app. This involved getting the right version of the socket.io protocol on both the client and the server, defining a shared schema for the data models transferred, updating the React components based on the app context changing, and writing the UI code to display the data. Also, assisted with the control script running on the pi that handled the sensors, navigation, and socket communication.
  Total Approx. Hours: 105

Finally, from the Reactive team:

- Andrew began with prototyping the Arduino script for the motor to move the gear and also helped make a

basic sequence of the lowering, pausing and lifting of the rack. Helped to build the lego structure and mount it on the prowler. For the second demo, designed the tanks, funnel and spreader for the reactive system. Wrote the code to control the sequence after collecting data, which reports the pH, calculates the amount of chemical needed, turns on an light-emitting diode (LED) to represent a valve opening and controls the direct current (DC) motor which turns the spreader. For the final demo, worked on finalising the reactive system, attaching the valve, tanks and LED strips. Designed the mounts for the tanks and wrote extra code to send data to Pi and open and close valves.
Total Approx. Hours: 144

- Xingchen implemented the moisture sensor, LCD, motors and ultrasonic sensors with the Arduino, helped implement the valves and the LED bars, and fixed unexpected errors several times, also helped with the Arduino sequence code. He designed and built the lego structure for the gear-rack system, helped with the construction of the robot body, designed some mechanical sub-components of the robotic system, and updated them several times. Xingchen drew the .stl graph for 3d printing a new rack with Tinkerpad and the .dxf file for the box with Maker case and helped with the .stl file of the chemical containers. He also created a poster for industrial day. He has also helped design the ground layout and the navigating sequence, participated in improving the navigation functions and testing, and also found solutions to the friction problem.
  Total Approx. Hours: 172

- Charles helped with the idea of the gear-rack design of the robot, and helped to build this system by using LEGO. Furthermore, helped to write Arduino code for lowering and lifting of the gear-rack system, the setup of moisture sensor and the Arduino code for displaying the moisture level on screen. Moreover, he designed the tanks and spreader for the reactive part. Helped to build the parameters for the gear set of the spreader, roughly 3:1. Helped to build the LED system which represents the valve and the DC motor of the reactive system. Finally, he worked on the final part of the reactive system, helping design the tanks for containing acids and bases.
  Total Approx. Hours: 105

We have used a considerable amount of our £200 budget, mainly in the time between Demo 2 and 3. We used the 3D printer several times over the duration of the course, which in total cost approximately £20. The items which we printed were two identical tanks to contain the reactive chemical compounds, a spreader to distribute the chemicals, four small gears for wheel encoder connection, mounts for the tanks and a mount for the camera. The main cost for this demo has been the building of a custom made demonstration area, which cost £146 to purchase.

Overall, the sensor and reactive sub-teams achieved their goals in the allotted time according to the project plan. In general, the work was distributed throughout the team evenly although for some members of the team worked more or less on demonstration one or two. Communication was also a strong point for the sub-teams, with daily communication and two weekly meetings.

The major problems faced by the team was component failure and delays in acquiring parts. Throughout the course, during prototyping several different elements failed including motor boards and arduinos. However, the teams were quick to identify the problems and source the replacements. When the valves were delayed a demonstrable alternative was needed and the team rapidly built an LED system to represent the valves.

Due to other course demands, team collaboration suffered and much work was done individually. This hindered development as occasional lapses in communication resulted in confusion. As deadlines approached, members had to spend less time on the project which when they coincided with demonstration days presented a challenge in terms of workload for other members.

For the navigation aspect, the integration of ROS into the project went well. The publishing and subscribing framework allowed fast communication between the multiple components of the navigation system. However, there was much about the navigation aspect that was not ideal. The concept of how the robot would navigate changed multiple times, which reduced the amount of time spent on the final script. Much time was spent researching and attempting to implement SLAM, and other robotic vision navigational techniques, all of which was not used in the final product. More planning into how exactly the robot would navigate at the beginning of the project would likely have saved a large amount of time and effort. The prowler chassis failed multiple times, with it being able to turn fully one day, then unable the next. This was a source of great stress and uncertainty, especially as deadlines drew near, though the team was once again able to come up with solutions to these problems.

2. Quantitative analysis and testing

Given the nature of our project, it was quite clear from the start that the navigation system would be one of the greatest risks in this project, which, due to its complex autonomous nature, poses key challenges in terms of reliability. Therefore, in order to test the reliability of the system, it makes sense to predominantly perform tests that validate the functionality of the autonomous navigation system. Other aspects of the project, such as the 'sensor system', which is responsible for measuring and reacting to values measured (if needed) are much more reliable. This is because, unlike the navigation system, the sensor system is limited to the constraints of the system itself; in other words, the navigation system must account for a changing, dynamic environment (such as collision detection), which

adds a layer of uncertainty to it.

Therefore, we broke the tests down into the following:

1. Testing the accuracy with which we can locate the robots position relative to its starting point.
2. Based on the poor results of Test 1, testing the same criterion as Test 1, but with our improved method.
3. Testing the accuracy of the turning motion performed by the system.

The first test is perhaps the most relevant for our system, as it gave us the first glimpse into how the basis of the navigation system could be formed. For example, if this test was to give us satisfying results, then we would use this method throughout in our navigation system, otherwise, we would need to come up with a better solution.

In order to address the first test, we derived a test to measure the distance output (in cm) of the robot with varying levels of power. The aim of this data collection was to understand whether we can reliably estimate the location of our system based on the input that we provide the wheels with. The data gathered from this experiment is shown in 1. For context, at this stage in our development, the system did not have any form of output that we could use to help determine the relative position of the robot; thereby, we tested whether it would be feasible to have such a system that solely relies on the input (power) that we provide the motors with to determine the distance that the robot travels.

In order for us to be able to estimate the position of the robot well, we would require that the relationship between the power and the distance travelled is linear. However, it is clear from 1 that this is not the case. It could be argued that the data shows a polynomial trend, however, it is quite clear from 2, which shows the standard deviation in the distance measurements, that as the power increases, our uncertainty in distance also increases linearly.

As a result of these tests, we had to come up with a more reliable solution, that would allow us to estimate more precisely the distance that the robot has travelled. Our new solution made use of custom built wheel encoders that would sit on the shaft of the wheel, in order to measure the output (the number of rotations), from which we could deduce the distance moved. Since the wheel encoders measure the output of the wheels, they would presumably be much more accurate than using the input alone. The results of this test are shown in 3. We recall that our goal for the first test was to have a linear trend between distance, and that is exactly what we accomplish using the wheel encoders. We were therefore able to use linear regression to estimate accurately how many encoder ticks (the units of shaft rotations) relate to which distance. 3 also shows the equation that we used for this.

Moreover, we were also able to reduce the variance in the measured distance (given a target encoder tick) drastically 4. The maximum standard deviation measured decreased by more than 33%, which is a significant improvement. It should be noted however that more data points would need to be collected, for a greater range of encoder ticks, as, in retrospect, it is difficult to deduce whether a similar pattern to that of 2 is occurring.

During the development stage of the navigation system, we noticed that the most challenging part was to make the robot turn corners successfully. Therefore, once we were content with our implementation of the turning algorithm, we decided to make a log of the number of turns that the robot has made during testing, and whether it was able to successfully complete the turn. The goal that we had set out for us to consider the system to be working robustly was 90%. This is because we expect that the turning motion is a critical part of an autonomous navigation system, and that a fault in the turning could lead to the whole robot halting, rendering it useless. However, we were realistic with our goal, and did not aim for 100%, as this was likely not achievable; moreover, given the application of our robot in fields, we do not expect turning to be such a critical component (as compared to autonomous vehicles for example). Out of the 23 turns that we recorded, 16 of these were successful. That is approximately 70%, which means that 3 out of 10 turns are unsuccessful. This of course, did not reach our goal of 90%, which means that further improvements should be made to the system before it is released to market in order to reach the goal. It should be noted however that this of course is not enough data points, and we would need to collect more data to be more confident in the conclusion that we came up with.

Overall, the tests that have been executed by the system are very much exhaustive of all of the critical parts of the navigation system. As discussed before, a majority of the uncertainty in the system comes from the navigation system, and therefore it made sense to focus on this subsystem. The combination of the localisation prediction and the turning also covers edge cases, as when the robot may approach a wall at the wrong angle. This would require the robot to reverse (making use of localisation), and move at a different angle (making use of the turning functionality).

3. Budget

The items that were included in our kit, and that we borrowed, are listed in Table 1:

Item	Quantity	Total Price (£)
Arduino Uno	2	38
Prowler robot	1	≈ 170
Zippy 30C Series 5000	1	≈ 30
Arduino Base Shield	1	3.50
NXT Lego Motor	1	32.39
Raspberry Pi 3	1	30
PowerBank	1	15
Raspberry Pi Camera	1	20
Power Board (Arduino)	1	10
Motor Board	2	20
AA Rechargeable Batteries	8	9.60
Grove LCD RGB Backlight	1	13.10
Grove Capacitive Moisture Sensor	1	6.50
TB3 Waffle Plate	2	25.76
Encoder Board	1	10
Encoders	4	4
Lego mini motor	1	17
LED light strips	2	5
Valves	2	10.62
Grove Ultrasonic sensors	3	12
Plywood Boards	8	146
MPU9250 IMU	1	9.11
3D printed items	-	≈ 13
Lego	-	5
Miscellaneous	-	5
Total		**660.58**

Table 1. The prices of items currently used.

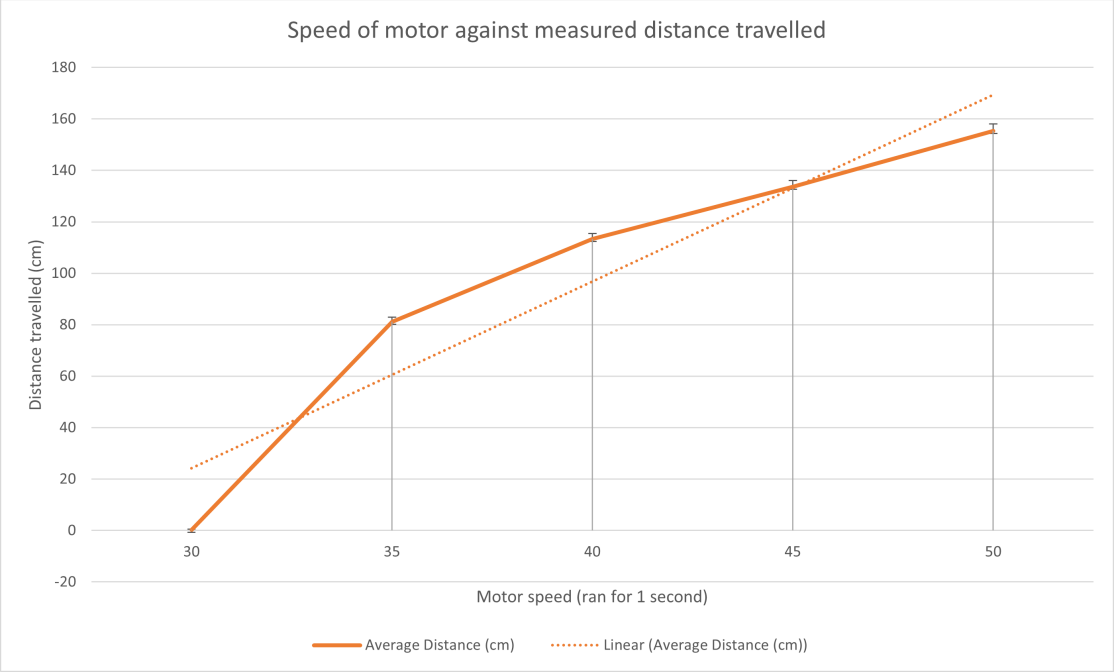

Figure 1. The distance travelled (cm) by the motor with varying levels of motor speeds for 1 second

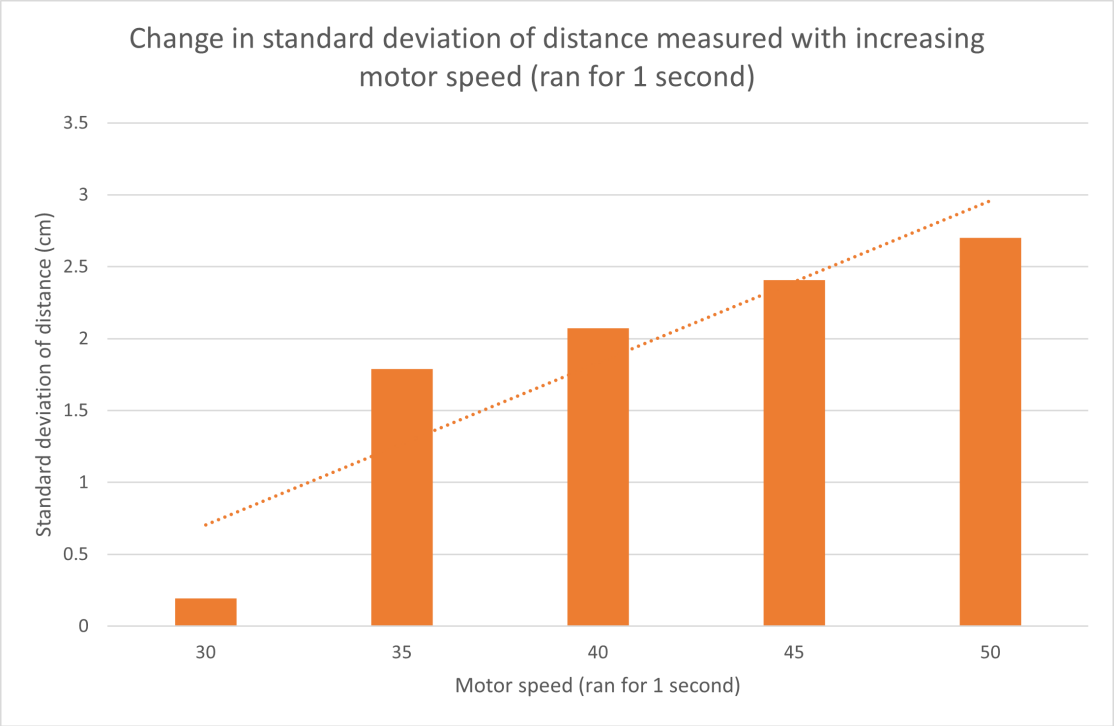

Figure 2. The standard deviation of the distance travelled (cm) by the motor with varying levels of motor speeds for 1 second

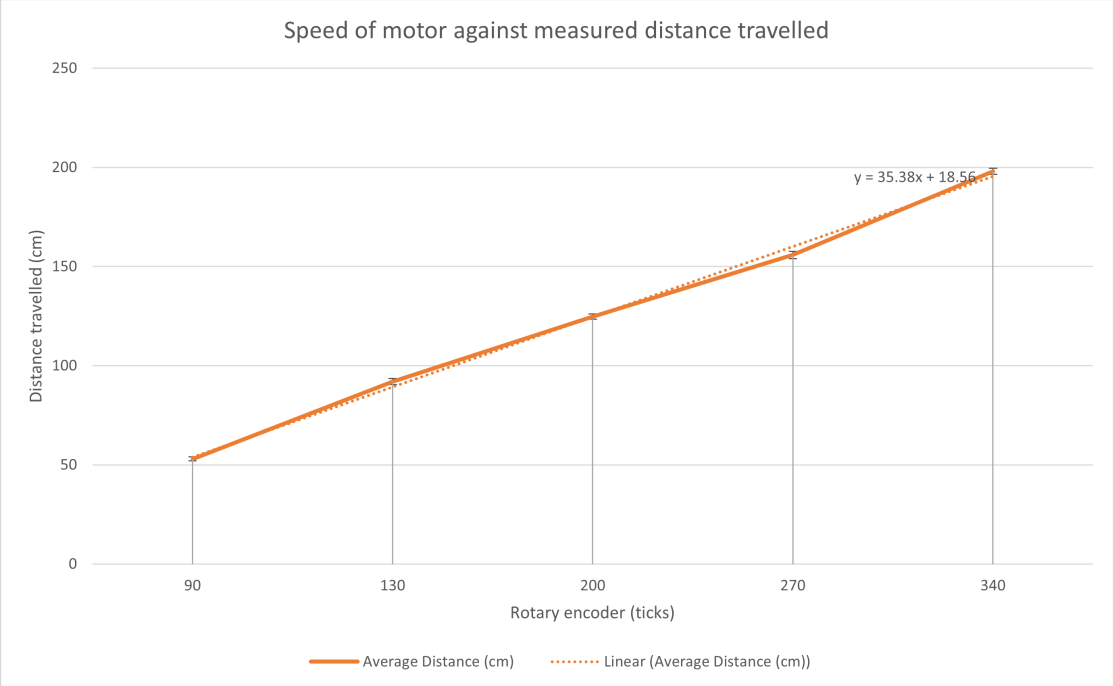

Figure 3. The distance travelled (cm) by the motor with varying levels of target encoder ticks

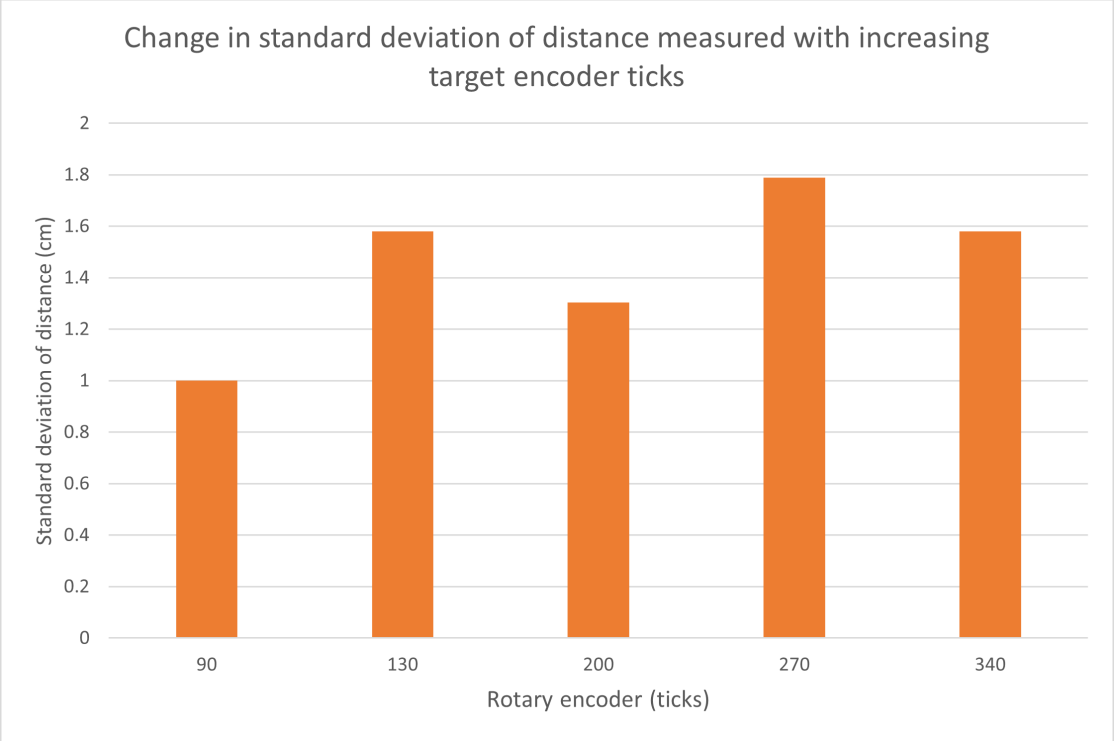

Figure 4. The standard deviation of the distance travelled (cm) by the motor with varying levels of target encoder ticks

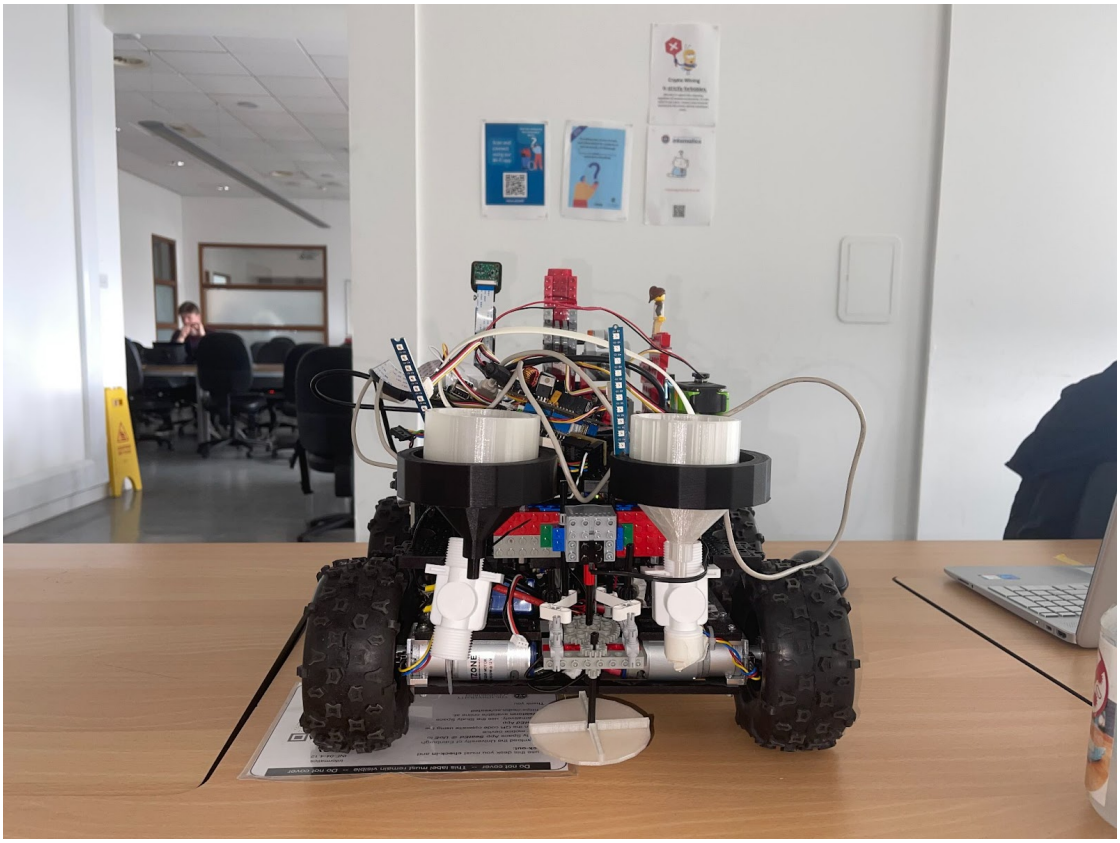

Figure 5. The robot prototype