Product: N.E.A.T. (No Effort Automated Transporter)

Team: Phate I

Abstract

N.E.A.T. is an assistive relay robot, designed to carry and deliver items, for those with mobility issues which impede their ability to carry out this task as they walk. We will begin by implementing remote control movement of the robot base via an Android application, and an independently operated lift mechanism. We will then mount various sensors (including infrared, ultrasonic and gyro) to implement object and incline detection, and extend the functionality of the app, allowing it to control the lift mechanism. Later, we will begin to tackle object avoidance strategies, as well as implementing the follow feature using infrared sensors and beacons. We will improve the lift mechanism, offering set heights for ease of use. Finally, we will create the 'come to me' feature, where the robot will locate the user using the infrared sensors and beacons to make its own way to the user, avoiding any obstacles along the path.

1. Goal description

There is a large population who struggle to carry items while walking, whether this is due to physical disability, reliance on a walking device or loss of hand dexterity / coordination. To combat this, N.E.A.T. will consist of a tray which can be raised to various heights, to ease the placement and retrieval of items, attached to a moving base, allowing the robot to transport items to the user's intended destination.

1.1. Relevance of the system

Approximately one in five people in the UK are aged 65 and over, and this is projected to rise to one in four by 2050. (Sarah Coates & Scott-Allen, 2019) Additionally, there are over 11 million people in the UK alone with a limiting long term illness, impairment or disability. The most commonly-reported impairments are those that affect mobility, lifting or carrying; over a quarter of disabled people say that they do not frequently have choice and control over their daily lives. (Department for Work & Pensions, 2014) Our goal is to help to address this issue by building a robot which allows the user to have more control within their own home by easing their dependence on assistive living and the need for home adaptations, particularly because one in five disabled people find adaptations made to their accommodation unsuitable or unsatisfactory. (Department for Work & Pensions, 2014)

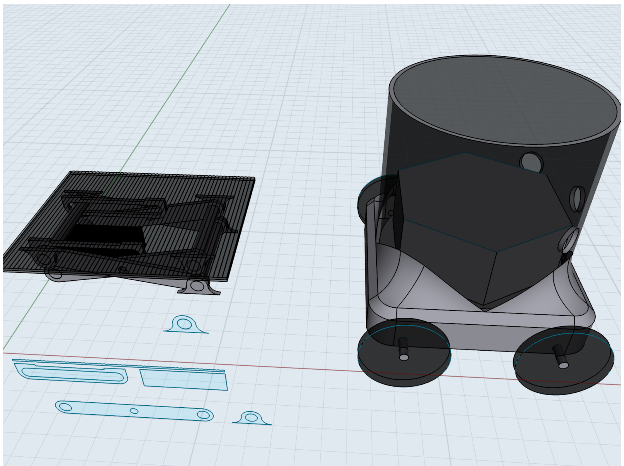

Figure 1. N.E.A.T. design sketch

1.2. High-level description

1.2.1. User story

- Actor
  Sam (has reduced mobility, which makes it very difficult for them to carry food whilst walking)
- Trigger
  Sam indicates that they would like the robot to carry a plate of food to the dining room.
- Main Success Scenario
  The robot lifts its tray so that it is inline with the kitchen counter so that Sam can place the food onto it with ease. The tray is lowered and the robot then follows Sam to the dining room. Upon arrival, the robot lifts the plate of food so that it is now inline with the table so that Sam can easily move the plate onto the table.
- Normal Flow
  1. The robot will start adjacent to the counter, with the lift at the base height
  2. Sam will use the app to raise the lift so that the tray is in line with the counter
  3. Robot remains still while Sam slides the food onto the tray
  4. Sam selects the 'Follow Me' option on the app
  5. The robot will automatically lower the tray to the base position
  6. The robot will follow Sam to the dining table
  7. The robot is docked beside the table

8. Sam will use the app to raise the lift so that the tray is in line with the table top
9. Sam moves the food off the tray and onto the table

- Alternative Flows

1A. The robot is elsewhere and Sam switches to manual control in order to drive it to them at the kitchen counter. The use case returns to step 2.

1B. The robot is elsewhere and Sam selects 'come-to-me' feature on the app and the robot locates the user beacon and finds its way autonomously to Sam at the kitchen counter. The use case returns to step 2.

1C. The robot is elsewhere and Sam finds it and uses the 'follow me' feature to lead it to the kitchen counter. The use case returns to step 2.

4A. (manual mode) Sam selects the manual mode. The use case returns to step 5.

4B. (come-to-me mode) Sam will make their way to the dining room. Sam will call the robot by selecting the 'come-to-me' option. The use case returns to step 5.

6A. (manual mode) Sam manually controls the robot towards the dining table using the direction controls on the app. The use case returns to step 8.

6B. (come-to-me mode)

6B1. The robot detects the user beacon and makes its own way towards the dining table. The use case returns to step 7.

6B2. The robot encounters an obstacle in its path. The robot alerts the user. The robot detects the user beacon. The robot moves in the direction that will bring it closest to the user beacon without encountering the obstacle. The use case returns to step 6B.

6C. (follow mode) Sam is not within range to be followed, in which case the robot switches to the manual control mode via the app. The use case returns to step 6A.

6D. (follow mode) The robot encounters an obstacle in its path. The robot alerts the user. The use case returns to step 6.

7A. (manual mode) Sam docks the robot beside the table using the rotation buttons on the manual mode interface of the app

1.2.2. Prototype constraints

We will not be handling stairs or inclines in this prototype design, it will traverse flat, even surfaces only. The user must place/retrieve the item on the tray themselves. Although this is designed for people who struggle with carrying items, we have dramatically decreased the distance they must move an item as placing it on the tray is far less movement than carrying it to their final destination. Additionally, the various heights of the tray will assist with this action as it won't require them having to bend down. For our prototype, we will limit the number of preset heights for the lift. Our prototype will implement a simple object avoidance strategy but it will not be designed to handle very complex settings with many obstacles. On this note, the 'come to me' feature, where the robot locates the user and makes its own way to them, will be designed for use primarily over short distances. The prototype will also have a weight limit for the items on the tray in order to ensure the robot only carries items it is capable of, reducing the risk of any potential equipment damage. There is an obvious space limit associated with the tray but this will be of a standard size as to avoid making the robot impractical due to underestimation. Although these constraints have been placed on the prototype, we have ensured that it is feasible to scale them up for a full-size product.

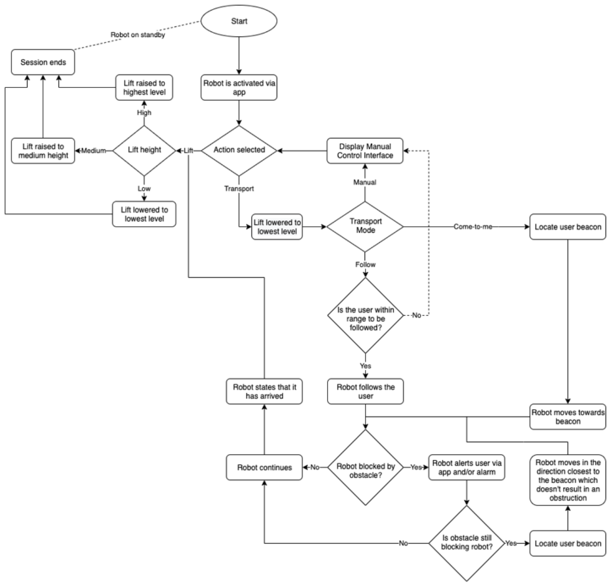

Figure 2. User case flow chart

2. Task planning

2.1. Milestones

- Milestone 1 - 05/02
  - Goal
    1. First iteration of robot base built with basic movement capabilities.
    2. Basic functionality of app: robot base movement.
    3. First iteration of a lift mechanism built and functioning, separate from the base.
  - Evidence
    1. The robot base is able to move forward, backward in a constant speed and turn to the desired direction.

2. The robot is able to move to a desired position on a flat floor under manual control.
3. The lift is constructed and able to move upward and downward on its own.
4. The lift can hold a plate or cup and maintain its balance while it's moving.
5. The base is stable and robust enough to hold more than the weight of the lift.
6. App has a clear, user-friendly menu and control interface.
7. App can communicate with the robot i.e. send control commands.

- Milestone 2 - 26/02
  - Goal
    1. Android app is able to control the lift mechanism, and has buttons implemented for 'follow me' and 'come to me' movement options.
    2. Sensors implemented on the robot base for object and incline detection, and lift mechanism integrated onto the robot base.
    3. Alert system implemented on the Android app when the robot encounters an object/incline.
  - Evidence
    1. App can move the lift up or down to a certain height within configuration.
    2. App provides interface for 'follow me' and 'come to me' mode.
    3. The robot is able to identify obstacles around it and stop if the obstacle is on the trajectory it is moving towards.
    4. The robot is able to identify inclination and stop once the incline is detected.
    5. When the robot stops due to an unexpected situation, the app can alert the user properly and timely, so that the issue can be addressed and resolved.
- Milestone 3 - 11/03
  - Goal
    1. The 'follow me' functionality is implemented.
    2. Object avoidance strategy designed and implemented onto the robot.
  - Evidence
    1. When the user stands near the robot and taps the 'follow me' option in the app, the robot will follow the user without colliding with the user or any potential obstacle.
    2. The robot detects, avoids and circumnavigates any obstacles in its path while moving.
- Milestone 4 - 01/04
  - Goal
    1. 'Come to me' functionality implemented.
    2. Preset lift heights installed.
    3. All features fully integrated to the Android app.
  - Evidence
    1. When the user taps the 'come to me' option in the app, the robot will move towards the user's location and stop once it is sufficiently close.
    2. The lift can be raised to several, predefined heights (consistent even with items on the tray), including base height, standard kitchen counter height and standard dinner table height.
    3. The Android app can be used for controlling the lift and for the three types of movement: manual control, 'follow me' and 'come to me'.

2.2. Task decomposition

Each milestone has been decomposed into a set of atomic tasks. The list of tasks is set out in Table 1. This table can be found at the end of the document.

The main tasks of each milestone have been detailed on a Gantt chart, which can be found in Figure 4, located at the end of this document. Estimation of the amount of time to be spent on a task and dependencies between tasks can be seen on the chart.

2.3. Resource distribution

2.3.1. Resource deployment

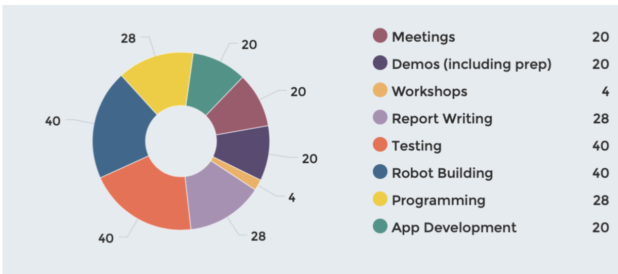

Figure 3. 200 hours per group member resource distribution

We have planned approximately how 200 hours per group member will be spent throughout the semester on this project. The allocation can be seen in the pie chart in Figure 3, above.

2.3.2. Resource List

1. Equipment
   (a) Arduino/Raspberry Pi for robot base movement control and wireless communication
   (b) LEGO EV3 brick for lift control
   (c) LEGO EV3/NXT Motors for lift mechanism

(d) Ultrasonic sensors for object detection/avoidance
(e) Infrared sensor + beacon for follow and come-to-me feature
(f) Gyro sensor for slope detection
(g) Omni directional wheels to enable smooth movement in all directions
(h) Bluetooth transceiver if using Arduino for Bluetooth
(i) Android WiFi connection support for indoor location acquisition
(j) Lego/wood for base
(k) 3D printer and printing supplies if printing lift, could also be Lego or wood
(l) £200 budget to be spent on extra materials
(m) 10 hours of technician time

2. Skills
   (a) Leadership
   (b) Team Management and Communication
   (c) Agile Methodologies
   (d) Back end Development
   (e) Front end Development
   (f) App Development
   (g) Lego Mindstorms EV3
   (h) Electrical Engineering

2.4. Risk assessment

1. Implementation (Mechanical)
   (a) Risk: Lift mechanism is too difficult / unreliable / unstable to implement.
   Solution: We'll choose the height that maximises convenience and stability and set the tray to this height at all times. In order to mitigate this risk, we will carry out extensive research, including building some Lego prototypes, before designing the lift in order to reduce the likelihood of our design failing.
   (b) Risk: The robot may be nearing the end of its power reservoir, at which point we cannot guarantee full control of the robot motor at all times.
   Solution: We will use feedback gathered from the sensors to control the robot while raising an alarm to let the user know about this.
   (c) Risk: Making a stable base large enough to fit the lift on it proves more difficult/time-consuming than expected. As other tasks are dependent on a working base, this could massively stall our progress.
   Solution: We will use the Turtlebot Waffle as a base, allowing us to progress on other features, while we work on building a custom base for our needs.

2. Implementation (Environmental)
   (a) Risk: Obstacle avoidance strategy proves too difficult to implement.
   Solution: Upon encountering an obstacle, the robot will be programmed to just stop, alert the user, and defer to manual control or just wait until the obstruction is no longer there. To reduce this risk, we will plan a simple but effective strategy as the baseline for object avoidance, and if possible, improve the strategy once the baseline is successful.
   (b) Risk: Robot cannot deal with stairs or inclines and could therefore be damaged if it attempts to traverse either.
   Solution: The robot will be equipped with various sensors to detect when stairs or inclines are present and move to avoid them. In order to prevent accidental damage, we will test the robot in safe environments until we confirm that sensors are successful in avoiding danger.

3. System Malfunction
   (a) Risk: The 'follow' mechanism encounters difficulty.
   Solution: The user will able to switch to the manual control mode and use the app interface to remotely drive the robot towards their chosen destination.
   (b) Risk: The Bluetooth signal is broken and a direct connection fails to be established.
   Solution: We will have a backup system in place which uses WiFi signals as the mode of connection instead and revert to this, and vice versa when the WiFi signal is broken.
   (c) Risk: The robot runs out of batteries in the middle of the floor and the user could find it difficult to charge it.
   Solution: We will implement removable/replaceable batteries, and, if possible, raise an alert when the batteries get low.

4. Safety
   (a) Risk: The balance of the robot may be compromised when items are placed onto the tray.
   Solution: The tray will be lowered during transport to maintain balance. Thorough research and testing of the design for the robot and lift will take this into consideration to optimise the balance and maintain stability.
   (b) Risk: Liquids or food could leak onto the robot and cause damage.
   Solution: The tray will be made of waterproof material and will have edges to prevent spillage off the tray surface. Sufficient planning will be carried out for the design of the robot to ensure that liquids cannot reach and come into contact with the electronics of the robot.
   (c) Risk: Safety of transporting (hot) food.
   Solution: The tray will be covered with a non-

slippery surface, and will have edges on all 4 sides while transporting to prevent the food being dropped. When the tray is not at the base height, there will be one side without an edge, allowing the user to slide the food on and off with relative ease. Additionally, a heat insulating layer will be added to the holding tray.

(d) Risk: Crockery is breakable and could fall off or slide around on tray.
Solution: As above regarding hot food. Additionally, the insulating layer can be made of a material with high coefficient of static friction with typical crockery material to limit sliding during lifting and transportation.

(e) Risk: Safety of transporting (hot) liquids.
Solution: The acceleration and deceleration of the robot and lift mechanism will be smoother for the transportation of liquids to avoid spillage.

5. Practicality

(a) Risk: The material on the tray may make it difficult to slide the food onto/off it.
Solution: We will research suitable materials or alternatives to replace it if this proves to be a problem.

6. Resource Management

(a) Risk: We require important additional materials but we have already spent our budget, reducing our ability to deliver all our planned features.
Solution: We will practice smart budgeting, allocating our funds appropriately to ensure that we prioritise the most important materials.

7. Teamwork

(a) Risk: The team is not able to come to an agreement on the best way to implement, test a feature etc.
Solution: The project manager will mediate and help the group to come to a decision.

(b) Risk: The team struggles to balance the workload of the project.
Solution: We'll use daily check-ins and weekly meetings to monitor progress and ensure everyone is pulling their weight, and make changes to task and time allocation accordingly.

(c) Risk: Poor time management or difficulty of task estimation resulting in certain features not being delivered.
Solution: We will deliver the most basic implementation of each feature first and then build on this to deliver more difficult features. By delivering a MVP at each demo, even if the later features are not delivered, there is always a working implementation of our robot.

(d) Risk: Team members with key skills or particularly important responsibilities end up being absent at key points.
Solution: Throughout the life cycle of the project, we will share knowledge between team members to reduce dependence on a particular individual and revise task allocation to cover any absent member(s).

(e) Risk: Team member doesn't know what they should be doing or which tasks take priority.
Solution: The tasks will be available on Trello and sorted in order of priority, making it clear what should be picked up next. Additionally, the team can be easily reached on Slack and group meetings at the start of each week will reiterate division of tasks and projected weekly goals.

(f) Risk: Conflicts arise within the team.
Solution: We will defer to the project manager for conflict resolution.

(g) Risk: We cannot get in contact with a group member.
Solution: We will notify the group mentor to devise a solution.

3. Group organisation

3.1. Group organisation and task allocation

We have elected a Project Manager who is responsible for overseeing development and we have split our group into two main sub-teams. These sub-teams will focus on software and hardware respectively. Tasks in the two sub-teams will be allocated according to level of expertise and may involve some pair programming to ensure the project is not reliant on one person. The members of either subgroup have opportunities to contribute to the other team's tasks which will allow for them to develop skills and experience in less familiar areas. This will be especially beneficial when we need extra members assigned to a task (which they may not have been previously familiar with) in order to keep it on track.

Weekly meetings will serve as an opportunity to distribute tasks and update the group as a whole on our progress. Additionally, the Slack channels for the two sub-teams will be used to allocate tasks throughout the week, particularly if a specific task requires an extra pair of hands, or if a group member is ready to pick up a new task and unsure of which task takes priority.

3.2. Meetings and communication

We have chosen to use Slack as our main form of communication. We have created various channels for handling general organisation, meetings, the hardware and software teams, and a check-ins channel used to keep the group apprised of the progress each member is making on a daily basis and what they plan to work on next - our answer to a daily stand-up, considering that it would be tricky to have everyone available at the same time for an actual stand-up meeting.

We plan to have at least one meeting per week with our group mentor, taking place at 11am on Mondays in Appleton Tower 3rd floor. Here we will monitor progress and plan tasks for the next week, as well as discuss any issues which have arisen. We will be flexible in organising additional meetings as and when the need arises.

3.3. Code-sharing

We plan to use GitHub as our code storing repository. This will allow all of us to collaborate on the code efficiently, maintain clear version control and store the code securely.

3.4. Progress tracking

We are currently using Trello to record outstanding tasks, keep track of which task each person is currently working on and to monitor our progress. The check-ins channel on Slack will also be used to track our progress, as we keep the group up to date on what has been accomplished.

We will also have weekly meetings, where we will review the previous week's progress as a group and decide whether we are on schedule and if not, what adjustments (in terms of job and time allocation) must be made to get the project back on track.

References

Department for Work & Pensions, Office for Disability Issues. Disability facts and figures. *gov.uk*, 2014. URL https://www.gov.uk/government/publications/disability-facts-and-figures/disability-facts-and-figures.

Sarah Coates, Priya Tanna and Scott-Allen, Eleanor. Overview of the uk population: August 2019. *Office for National Statistics*, 2019. URL https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/articles/overviewoftheukpopulation/august2019#toc.

Task Name	Milestone	Estimated time	Dependency	Rough description
Design lift mechanism	1	1 day	-	Conduct research and design lift and tray
Create a CAD mock up	1	0.5 days	Task 1	Mock up a CAD of the lift mechanism and test
Prepare parts for lift	1	1 day	Task 2	Gather all equipment, print/cut any materials
Build lift	1	2 days	Task 3	Assemble the lift mechanism
Build tray	1	0.5 days	Task 3	Assemble the tray
Write lift control code	1	1 day	Task 4	Write code for initial control of lift
Test lift	1	1 day	Task 6	Test movement, ability to carry weight, etc
Record lift stats	1	0.5 days	Task 7	Record weight capability, average speed, etc
Design robot base	1	1 day	-	Conduct research and design robot base
Select platform for signals	1	0.5 days	Task 9	Conduct research and select best platform
Create a CAD mock up	1	0.5 days	Task 9	Mock up a CAD of the robot base and test it
Prepare parts for base	1	1 day	Task 11	Gather all equipment, print/cut any materials
Build base	1	2 days	Task 12	Assemble the base according to the design
Test load carrying of base	1	0.5 days	Task 13	Test ability of base to carry items, stability, etc
Write code for movement	1	1 day	Task 13	Write code for basic movement of base
Test base movement	1	1 day	Task 14	Test movement of base in each direction
Record base stats	1	0.5 days	Task 15	Record stats for base movement and carrying
Create skeleton Android app	1	0.5 days	-	Create a basic skeleton Android application
Create manual app controls	1	1 day	Task 18	Create page with buttons for manual movement control
Link buttons to movement	1	1 day	Task 19	Link buttons on app to control movement using Bluetooth
Integrate lift with base	2	1 day	Milestone 1	Attach lift mechanism to robot base
Test robot base with lift	2	1 day	Task 21	Test base movement, lift movement
Install sensors on robot	2	1 day	Task 22	Place appropriate sensors on robot according to design
Write sensor handling code	2	1 day	Task 23	Write code to handle input from sensors
Write sensor alert code	2	0.5 days	Task 24	Write code to stop movement at threshold and alert user
Test sensors	2	1 day	Task 25	Test alert thresholds, sensor sensitivity and placement
Improve sensor placement	2	2 days	Task 26	Improve location, number of sensors, re-test & improve
Test and record sensor stats	2	1 day	Task 27	Test sensors again, record sensitivity, new thresholds, etc
Design environment for demos	2	0.5 days	-	Design environment to show robot features in demo
Build environment for demos	2	1 day	Task 29	Build environment according to design
Create app homepage	2	1 day	Milestone 1	Create homepage with controls for lift and movement
Add other movement controls	2	1 day	Task 31	Create buttons for alternative movement options
Create lift controls on app	2	1 day	Task 31	Create buttons to control movement of lift
Update lift code for app	2	1 day	Milestone 1	Update lift code to allow app to control movement
Link lift controls to movement	2	1 day	Task 33	Link buttons on app to control movement using Bluetooth
Test lift controlled by app	2	0.5 days	Task 35	Test controls of lift on app
Research alternative signals	2	0.5 days	-	Consider alternatives to Bluetooth to use as backup
Implement follow sensors	3	1 day	Milestone 2	Sync infrared sensors with user beacon for following
Implement follow strategy	3	1 day	Task 38	Write code to handle sensor input to control following
Test follow feature	3	1 day	Task 39	Test at distances, different environments, sensor positions
Improve follow feature	3	2 days	Task 40	Improve sensor positions, strategy, re-test & improve
Re-test follow feature	3	1 day	Task 41	Extensively re-test follow as above
Record stats on following	3	1 day	Task 42	Record stats such as max distance, accuracy, etc
Enable app follow feature	3	0.5 days	Task 43	Enable button on app to select 'follow me'
Link app control to movement	3	1 day	Task 44	Link app control of follow me to robot movement
Implement object avoidance	3	1 day	Milestone 2	Write code to use sensor input to avoid obstacles, inclines
Test object avoidance	3	1 day	Task 46	Test feature in different environments, sensor positions
Improve object avoidance	3	2 days	Task 47	Improve sensor positions, strategy, re-test & improve
Re-test object avoidance	3	1 day	Task 48	Re-test avoidance after improvements
Record stats on avoidance	3	1 day	Task 49	Record relevant stats on distance, accuracy, etc
Set heights for lift	4	1 day	Milestone 2	Research useful height presets and update code to set them
Update app lift control	4	1 day	Task 51	Update app controls to allow for set heights
Link app control to lift	4	1 day	Task 52	Link app control of lift height to movement
Test lift movement	4	0.5 days	Task 53	Test lift at all set heights, record stats
Implement 'come to me'	4	1 day	Milestone 3	Combine follow feature and object avoidance for come to me
Enable 'come to me' on app	4	0.5 days	Task 55	Enable app control of 'come to me' feature
Link app to movement	4	1 day	Task 56	Link app control of 'come to me' to robot movement
Test object avoidance	4	0.5 days	Task 57	Use 'come to me' feature to test object avoidance
Improve object avoidance	4	1 day	Task 58	Take feedback to improve sensor placement, strategy
Test 'come to me' feature	4	1 day	Task 59	Test feature in various environments
Record 'come to me' stats	4	1 day	Task 60	Record relevant stats on 'come to me', accuracy, speed
Perfect app UI	4	1 day	Task 57	Improve UI with all controls
Test all robot movement	4	1 day	Task 60	Test manual, follow and come to me movement
Test robot carrying	4	1 day	Task 63	Test with various loads in all movement
Fix any outstanding bugs	4	1 day	Task 64	Tackle any bugs arisen in testing
Record all stats	4	1 day	Task 65	Record all stats from testing all features

Table 1. Task decomposition for the system

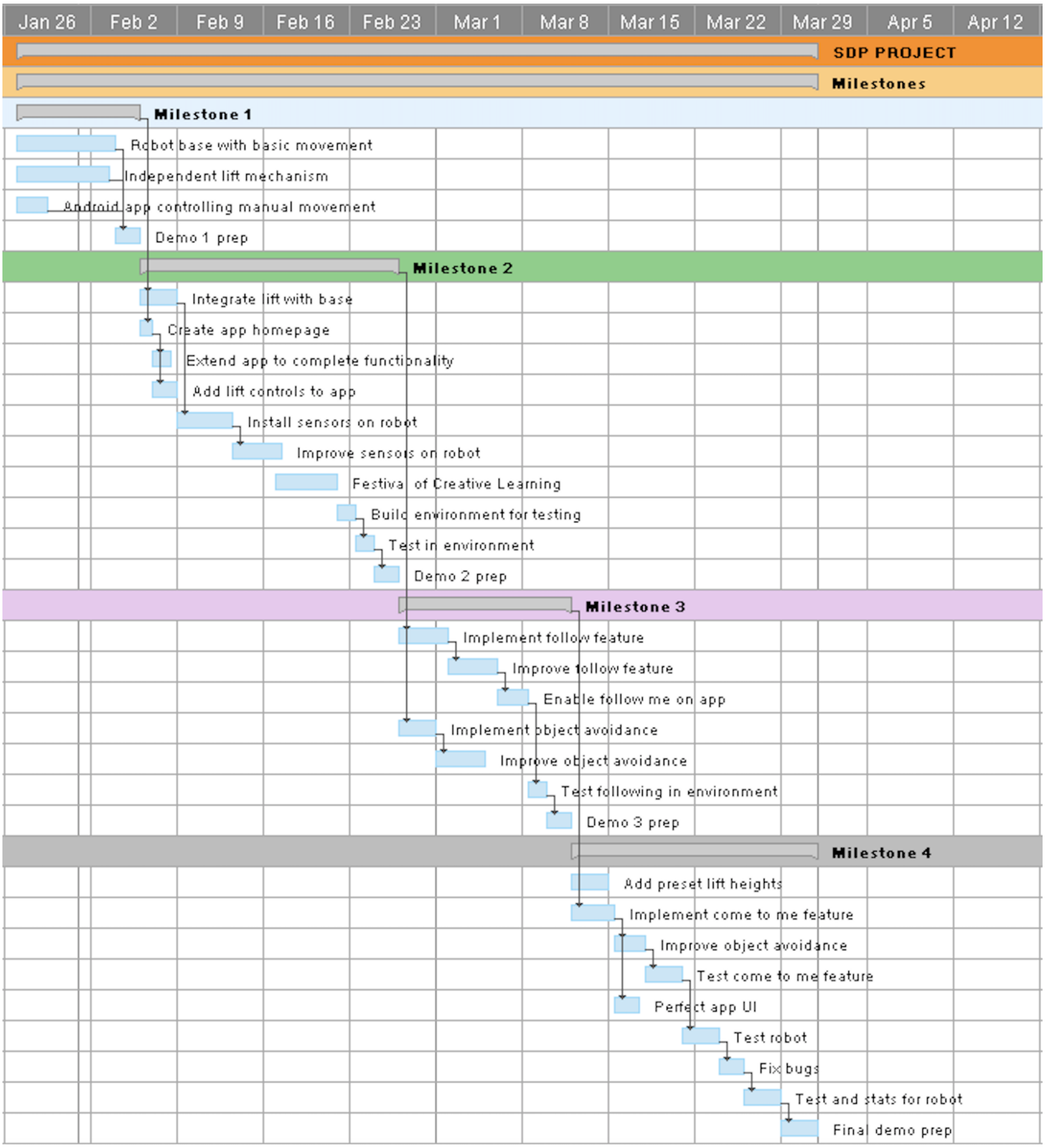

Figure 4. Gantt chart detailing the main tasks of each milestone and their dependencies