figs/logo.png

Product: Tadashi
Team: Detroit

figs/logo.png

Abstract

We propose an assistive healthcare robot, *Tadashi*, to automate simple tasks within a care home or supported living environment and allow nurses to spend more time caring for their patients. *Tadashi* will automate three key tasks in the nurse's day. Firstly, waking a patient up at a time specified by the nurse, by coming into their room and speaking to them. Secondly, bringing water and food to patients at specified times or on request. Thirdly, checking on the welfare of the patient while the nurse is occupied elsewhere, by coming to their room and asking the patient if they are okay and if they need a nurse to attend to them.

1. Goal description

Nurses are overworked: they spend a large amount of time on menial tasks, limiting the amount of time they can spend caring for patients. Our assistive healthcare robot allows nurses to automate certain menial and administrative tasks, allowing them to spend less time on the menial work, and more time doing what matters: caring for their patients.

1.1. Relevance of the system

1.1.1. The problem space

Tadashi is designed to tackle two problems facing nurses in assisted living situations:

1. High nurse:patient ratios (ie. many patients to one nurse), meaning nurses have little time to dedicate per patient;
2. High administrative loads on nurses, meaning they must spend more time on administration and menial tasks and less on caring for their patients directly.

There is no national guidance on staffing levels and nurse:patient ratios in care homes (**?**), but research shows that in care homes there is an average ratio of 18 patients per registered nurse during the day, and 26 patients at night (**?**).

With this many patients to take care of, nurses struggle to get the time they need with patients. In a 2017 survey on the impact of high nurse:patient ratios (**?**), 63.2% of nurses said that comforting or talking to patients was rushed, unfinished, not done to an acceptable standard, or missed entirely.

Equally, a 2013 survey showed that nurses spend almost 1/5 of their day on administrative tasks; and only 20% are satisfied with how they spend their time — preferring to spend less time on admin and more time on direct care (**?**).

By automating some tasks using Tadashi, nurses can decrease the amount of time they spend on administrative tasks, and increase the amount of time they spend caring for patients.

With elderly populations in particular, hydration and nutrition presents a pressing concern (**?**). A Care Quality Commission inspection in 2011 found that three out of 12 NHS trusts inspected failed to meet standards required by law in meeting patient needs regarding nutrition and hydration (**?**). A 2018 report from the Office for National Statistics found that more than 600 care home residents died suffering from malnutrition or dehydration between 2013 and 2017 (**?**).

Tadashi will allow for delivery of water or food to patients at a specified time — or patients to request these deliveries themselves — in order to ensure patients have access to regular food and water. Aditionally, by tracking these requests we will provide a dashboard for nurses to view when patients last had food or water: this provides added value to nurses along with reducing their workload through automation.

The problems mentioned above will only get worse in the future. It is estimated that by 2035, up to 190,000 more people aged 65 years or above will require some level of care, and increase of 86% from today (**?**). Meanwhile, ongoing staffing issues in the NHS mean that in 10 years time the NHS will have a shortfall of 108,000 nurses (**?**). This combination of factors will drive demand for innovative solutions — including assistive technology like Tadashi.

1.1.2. Existing solutions

The most relevant existing solution to the problems we have identified is the work of Fraunhofer IPA on service robots in residential care facilities. As part of the WiMi-Care project, they implemented and tested Care-O-bot 3, a 'robot butler' that tracks residents' hydration and brings them water if they have not drunk enough. The goal of this project was to automate certain service-related tasks in order to relieve pressure on care staff (**?**).

The key takeaways from this work that we will take into account in our project:

- Patient feedback to the robot was positive: "inhabitants understood the idea of a robot supporting the staff without replacing them and showed no fear to interact with the machine" (?).

- Adding speech output to address patients by name helped to improve perceptions of the robot and compliance with drinking the water it offered (?).
- Staff reacted positively to the introduction of the robot: "The overall reaction from the personnel ... was very positive" (ibid).

1.2. High-level description

We describe the following user stories to exemplify the functionality of the system:

1. Tadashi waking the patient
2. Tadashi bringing food or water to the patient
3. Tadashi checking on the patient
4. Nurse viewing patient nutrition / hydration on an app

1.2.1. Waking the patient

1. The nurse specifies in the app what time each patient should wake up.
2. At the specified time, Tadashi navigates to the specified patient's room.
3. Once in the room, Tadashi speaks to wake the user up. A button is accessible to the patient to press once they have woken up:
   (a) If the patient does not press the button, Tadashi sends an alert to the nurse's app, who can then go check on the person.
4. Tadashi returns to his starting spot to await the next command.

1.2.2. Bringing food or water to the patient

1. Either the nurse or the patient specifies a need for food or water:
   - The nurse specifies in the app for each patient:
     – The intervals at which they should be brought water (or another drink).
     – The times at which they should be brought food.
   - The patient presses a button on their controller to request Tadashi brings them food or water.
2. Tadashi navigates to the pick-up point for food or water.
3. Tadashi moves to the patient's room and passes over the food or water using its arm.
   - If the patient is at a different height to the robot, Tadashi will increase his height using a lift to get to an appropriate height for the patient and deliver food or water without spillage.
4. Tadashi returns to his starting spot to await the next command.

1.2.3. Checking on the patient

1. The nurse specifies in the app that Tadashi should go check on a specified patient.
2. Tadashi navigates to the specified patient's room.
3. Once in the room, Tadashi speaks to the user and asks if they are okay:
   (a) A button is accessible to the patient to press to reply yes or no.
   (b) If the user presses the okay button, Tadashi informs the nurse through the app.
   (c) If the user presses the not okay button, Tadashi informs the nurse through the app with an urgent alert.
      - The patient can alternatively, when Tadashi is not present, request the nurse's help by pressing a dedicated button on their controller.
4. Tadashi returns to his starting spot to await the next command.

1.2.4. Checking patient nutrition and hydration

1. The nurse opens the app to 'dashboard mode'.
2. The app displays for each patient a history of their nutrition and hydration deliveries.

2. Task planning

2.1. Milestones

We will split the work of each team into four milestones, each corresponding to the Sunday before the demo date on Wednesday.

2.1.1. Milestone 1

For the first demo on February 5th we plan to present our robot base frame with basic movement and tracking functionality, as well as a skeleton Android app.

Robot sub goals:

- The chassis is sturdy enough to carry a light load, initially 500g, representing future components.
- The robot can move around freely by remote control.
- Choose whether to use a TurtleBot or Lego frame for building the robot, based on which performs better in testing on battery life, turn radius, speed, and maximum load.

Once these sub goals are achieved, our robot base frame will be able to reliably and precisely follow along a straight line path of tape.

App sub goals:

- Create a functioning Android app which can run on a mobile device.
- Set up a basic user interface, consisting of five different activity screens in the app.
- Design and implement a database which can store a patient's information using Firebase.
- Implement GDPR features into the database.
- Implement functionality in the app to create and edit patient information in the database.

With these sub goals achieved, we can demonstrate updating the database from the Android app.

2.1.2. Milestone 2

For the second demo on February 26th we will showcase our robot with several new components and movement abilities, plus connectivity and interaction with the Android app.

Movement sub goals:

- Program the robot to recognize turns in a tape path
- Implement a memory system that allows the robot to remember its current state

With these new features, we will demonstrate the robot following along a bending line of tape on the floor, navigating around turns in the path.

Component sub goals:

- Attach a robotic arm, which will later be used to pass food and water to the patient along one axis.
- Equip the base frame with a lift system, which can be used to raise the robot's height in order to interact with patients in hospital beds.
- Implement external buttons which can be pressed by the patient to send a signal.

App sub goals:

- Accessible user interface allowing users to enter and show data from the database; and allow seeing alerts from the robot. Accessibility will be based on user testing.
- Receives alerts from the robot when its button is pressed.

2.1.3. Milestone 3

On March 11th for the third demo we will present the robot with more freedom of movement as well as app functionality.

Robot sub goals:

- Establish ability to move arm, lift, and robot on external command, to be integrated with app at a later stage.

Movement sub goals:

- Implement functionality to move around between rooms without needing a line of tape to follow.

App connectivity sub goals:

- Implement input and saving of:
  - Scheduling wake-up times for patients;
  - Visiting a patient for an impromptu check-up;
  - Scheduling meal and water times for patients.
- Implement encryption at rest and in movement of patient data, to support GDPR compliance.

2.1.4. Milestone 4

On the final demonstration on March 30th we will show our completed prototype with all components working together: we will fully integrate app functionality with hardware and robot software, being able to demonstrate how a nurse setting up actions for the robot can then be performed by the robot.

2.2. Task Decomposition

We have split our team into three groups to manage the respective tasks of robot building, robot programming and app development. For each milestone we have assigned the sub goals to the relevant team, and also planned how our sub teams will work together to implement these as working features.

For each sub-task we have assigned a difficulty rating ranging from simplest to most complex with XS, S, M, L, XL. We assign these values as estimations of the time and difficulty involved with implementing the sub-task, based on initial research, expert feedback, and gut feeling.

2.2.1. Robot Building

Milestone 1: We will build two early versions of the robot base frame with basic moving functionality (forward movement and turning). The base will be steady enough to hold a light load.

Sub tasks:

Robot chassis

- Build two frames: one from Lego and one using TurtleBot (M)

Testing

- Test base frames of Lego vs Turtlebot based on battery life, turn radius, maximum load, speed (L)

Milestone 2: Main robot components attached

Components:

- Build and add arm operating on one axis to deliver food and water to patient (S)
- Build and add scissor lift to raise robot up and down (XL)
- 3D print and add physical buttons (M)

Milestone 3: Interface with robot programming to allow components to be controlled through software

Milestone 4: Robot has all components attached and is sturdy enough to move between rooms and complete tasks according to its timetable (L)

2.2.2. Robot Programming

Milestone 1: Robot can move around and follow a straight line of tape.

Sub tasks:

Basic RC of robot:

- Connecting software to motor controls using ROS and EV3 (S)
- Moving forward and backwards (S)
- Rotating on the spot and turning to face a specified direction angle (M)

Robot pathfinding system:

- Robot can recognise tape underneath it and follow the tape in a straight line (L)

Testing:

- Testing out the ROS programming of Turtlebot vs the Python programming of Lego Mindstorms EV3 block to decide which is simplest and most customisable to find the best platform solution on which to build our pathfinding algorithms (M)

Milestone Two: Robot can follow a line of tape with turns and send alerts to app

Pathfinding

- Robot can follow the tape around turns (M)
- Robot has a memory system that remembers it's current state of position on the line (XL)

Connectivity:

- Robot can send an alert to app when it's button is pressed (M)

Milestone 3: Robot can maneuver through it's environment more independently and also receive remote controls

Pathfinding:

- Can navigate between rooms on it's own without line of tape (XL)

Connectivity:

- Robot can receive basic movement RC commands from app (L)
- Robot arm can be operated to assist patients (XL)

Milestone 4: Final prototype

Robot can follow it's timetable received from the app, travelling to rooms at given times and performing it's programmed tasks.

Pathfinding:

- Can navigate carefully around obstacles (XL)

App connectivity

- Robot can receive Calendar timetable and database information from the app to plan it's route through the rooms and to know who needs which medications at what times (L)

2.2.3. App Development

Milestone 1: Skeleton app

Database:

- Database server set up which contains patient name and basic information (S)
- Implement GDPR compliant design with ethical and security considerations (XL)

App:

- Skeleton Android app set up on a mobile device using Firebase (S)
- Updating database patient information through app (L)

Milestone 2: Basic robot connectivity

App:

- Basic RC control of robot from app (S)
- Show alerts from pressing button on robot (M)
- Creating a few different UI designs (L)

Demo	Date
1	Feb 5
2	Feb 26
3	Mar 11
4	Apr 1

Table 1. Demo dates.

- Testing UI designs on consumers with ethical approval (M)

Milestone 3: Sending and receiving complex information to and from robot

App:

- Receiving photos from robot's camera (XL)
- Add calendar functionality for managing and updating a patient's medication timetable (M)

Database:

- Store additional information for patient, including their health conditions and medication timetable (M)
- Implement encryption security for this sensitive personal information (L)

Milestone 4: Final UI, features and full robot control

App:

- Shows GPS location of robot and alerts for if it is stolen (XL)
- Final UI with easy to use sections (L)
- App provides full RC control over the robot and it's components (M)
- App conveys relevant information about the robot's location, activity and timetable for the day (L)
- App allows user to plan a timetable for the robot's daily tasks (L)

2.3. Resource distribution

The 200 hours per member will include the following for every member: **at least** 15-20 hours worth of team meetings, 4 hours of demos with another 4 for preparation, 2 hours of workshops with each member attending at least one, and 1-2 hours for the individual reflection report. This leaves each member with 168 hours to dedicate to their work; roughly 500 hours per team. Each milestone has the same deadline across the teams - the date of the corresponding demo. They are shown in Table 2. Tables 3, 4, and 5 show the individual resource distribution for each of our three teams.

2.4. Risk assessment

2.4.1. Robot building

In terms of milestone 1, the main risk would be that the robot design is not able to fulfill expected tasks (particularly carrying loads). We have decided to develop 2 prototypes as a contingency plan for this exact issue. This allows us to compare two designs and ultimately pick the one which we find to be a better choice. 25 hours for the design and building for each robot may seem like a large amount of time when one of them will be discarded. However, ultimately this counters the many more hours that we would have to spend later down the line should we find that our bot was infeasible. Having to redevelop the bot as well as the software and some features in our app would require far more than 25 hours.

For milestone 2, our team are planning to design and build a scissor lift from Lego. This should be fairly risk free. We also plan to build a hardware controller for the bot. At least one of these should be achievable and whichever is not completed for this milestone can be worked on during the period of maintenance running up to the fourth demo.

This 30 hours of maintenance between milestone 3 and 4 will actually prevent mostly any risk that should arise with individual pieces of hardware, as it can be eaten into by the hardware team if need be.

2.4.2. Robot programming

Having the robot move without the aid of a line on the ground is definitely a risk the robot programming team will have to keep in mind. It could prove to be more difficult than anticipated. The fact that sensors are not always 100% accurate means that special care will have to be taken in the implementation of this. We will reduce this risk by re-planning regularly in software to make sure tolerances do not grow too much over time.

Failure of a sensor could be catastrophic in a project where they are the centerpiece of the functionality. It is therefore of vital importance that we constantly check that they are all functioning as they should be.

The algorithms used by the robot will also be a challenge facing our robot programming team. If implementing these turns out to be more difficult than we anticipate we will refer to advice from the expert consultant in electronics and software.

2.4.3. App

The use of Firebase should alleviate most of the risk in terms of the building of the app as it allows an easy way to get things off the ground. There are extensive resources to aid our developers in their progress should they hit a brick wall.

The UI will have to be usable for novice users and one of the risks is that this is not the case. The app team will use user testing in order to make sure they are putting out a

Milestone	Tasks	Equipment	Skills	Est. Hours
1	Build Lego bot	Lego	Lego building, Lego designing	25
	Build Turtlebot	Turtlebot		25
	Test and evaluate the two	-	Quantitative analysis	50
2	Install arm	Turtlebot arm/Lego arm	Design Lego arm	80
	Design and build scissor lift	Lego to build lift	Lego design/build	60
	Design and build controls	Electronic components	Circuitry	60
3	Integrate camera to send to app	Camera, RasPi	Interfacing cam with software	70
	Design 3D printed buttons	3D print design tools	Ability to use said tools	30
	Print and implement buttons	3D printer	-	20
4	Tidy up of the aesthetic	-	Design, getting feedback	50
	Maintenance	-	-	30
			Total	500

Table 2. **Robot building team** resource distribution.

Milestone	Tasks	Equipment	Skills	Est. Hours
1	Get robot moving	Access to robot	ROS (Turtlebot, Python, motors)	60
	Get working with sensor input	Sensors	-	25
2	Using sensor input to stop	Sensor readings	Analysis	120
	Using sensor input to detect environment	-	Real-time sensor analysis	115
3	Have lift working	-	Python for lift	40
4	Bot moves around to user defined paths/input	-	Sensor and/or camera analysis	140
			Total	500

Table 3. **Robot programming team** resource distribution.

Milestone	Tasks	Equipment	Skills	Est. Hours
1	Set up app UI	Firebase	UI design, Android	70
	Set up database system	Database management server	Firebase	20
	Interface the database with the app	-	-	30
2	Set up robot control UI	-	-	40
	Test UI	More UI testing required	-	10
	Have app show alerts from bot	Information from robot	-	40
3	Implement calendar system	-	-	40
	Add wake up alarm feature	-	-	30
	Encryption of personal data	-	Network security	60
	Receive photo from bot	Access to bot	-	70
4	App shows location of bot	Information from GPS tracker	-	30
	App shows timetable/activity of bot	-	-	30
	Final design of UI	Memory to store information	-	30
			Total	500

Table 4. **App team** resource distribution.

Robot building	Robot programming	App development
Jakub	**Wojtek**	**Theo**
Luukas	Ben	Yuchen
Rebecca	Errikos	Ching Ling
Project management:	**Michael**	

Table 5. Team splits across the group. Names in bold are key points of contact.

usable product.

3. Group organisation

We will split the group into **three core teams**: robot building, robot software, and app development (table 5). Each team has a team lead (in bold), responsible for coordinating with the other groups as necessary during the development process and holding responsibility for ensuring the group is tracking to its milestones.

The project manager (PM) holds overall responsibility for keeping the project on track: primarily through helping each team plan, execute, and validate its progress against milestones. The PM is also responsible for time management; writing up reports; and helping teams with any blockers they encounter in their work.

Structuring the teams in this manner allows us to use the *functional* organizational structure. This has particular benefits in allowing everyone within each team to focus on developing expertise in their area, as well as improving efficiency in communication within the team.

The disadvantage of using a functional structure is that it can make communication between teams difficult. To minimize this, we have decided a key point of contact (POC), in bold, for each team: POCs will communicate with each other and the PM on key issues and when cross-team collaboration is needed — for example, in interfacing the robot hardware with the robot software. This eases communication between teams because it means there will only need to be four people meeting at once to represent all teams, rather than all ten group members needing to be present.

Each team will meet at a minimum once a week to discuss their progress against their milestones. Work will be done in collaboration with the PM following an Agile approach: at each weekly meeting we will follow **Plan, Develop, Review** methodology to iteratively update our approach and track our progress against our milestones.

In keeping with an Agile approach, **team members will give daily updates on their work** (how they are progressing, if they have any blockers, and whether they need any help from other team members) in online ‘standups’ on Slack. Additionally the team POCs will meet once a week to make sure that any cross-team integration issues are handled, as well as to support each other if needed.

Code-sharing will be done exclusively through GitHub. Using version control more generally allows us to track our work over time and easily deal with any merge conflicts or other issues that may come up in doing distributed development. GitHub was chosen primarily because all members of the group are somewhat familiar with it, meaning there will be a shallower learning curve to complete our project using it. When working to demos we will put in place a **code freeze** on midnight on the Sunday before the demo date to reduce risks to the system at the demo itself.

Using GitHub also allows us to do **task allocation and progress sharing** using GitHub projects. We will follow a Kanban approach, separating tasks into “to do”, “in progress”, and “done”, with one board per team. Using Kanban allows for team members to clearly see what tasks need to be done before the next demo; choose to begin working on tasks they feel they are suitable for; and to identify where there may be blockers within their work (ie. cards that are spending extended time in “to do”). Additionally, making these boards public allows for other teams to see how work is progressing in the rest of the group. Progress updates will also be discussed in weekly team meetings on a granular level and more broadly in POC meetings.

Additionally, in **task allocation**, teams will decide the complexity of tasks by evaluating them using **T-Shirt sizing**, where the size or difficulty of each task is graded on a scale: XS, S, M, L, XL. This will allow team members to clearly see the estimated difficulty of each task and to assess where they need to dedicate their effort.

Communication will be done primarily through a dedicated Slack workspace, with separate channels for each team (`robot-app`, `robot-building`, `robot-coding`, `report-writing`, and `general` for cross-team discussion). Using Slack allows us to integrate other apps, for example Doodle polls to decide group meeting times; and GitHub to track push or other notifications from our project repository.