Auxilio Robotics

Alfred: A friendly indoor robot for low-risk elderly assistance

Team F: Critical Design Review Report

Abhinav Gupta
Atharva Pusalkar
Praveen Venkatesh
Shaolin Kataria
Shivam Tripathy

Sponsors: Prof Yonatan Bisk (LTI), Prof Zackory Erickson (RI)

May 3, 2023

Abstract

The Assisted Living industry is facing a major labor shortage. Since 2019, 238,000 caregivers have left the industry. The annual turnover rate is at an all-time high of 40-67%. This drop in the number of caregivers is facing a stark rise in the number of seniors seeking assistive healthcare. At present, there are more than 2.7 million seniors living in Assisted Living Facilities. It is predicted that in the future, 7 out of 10 people will require assisted living care in their lifetime. It is also projected that by 2040, seniors will comprise 22% of the US population, compared to the modern-day figure of 17%. This showcases how significant the gap between the demand and supply is. As for the problems people usually suffer from, according to Center for Disease Control and Prevention, the portion of people facing difficulty with mobility is 40% out of all the disabilities. This is where Alfred comes in with its mobile manipulator capabilities. Alfred is a friendly indoor robot for low-risk elderly assistance to make up for the drop in labor in the industry. Low-risk is defined as any form of care within the industry barring personal care like changing clothes, helping the person into the washroom, preparing a bath, etc. Within its low-risk definition, Alfred aims to collaborate with nurses to take care of patients by autonomously performing object pick-and-place tasks. It comes with some social interaction capability where a distant loved one can feel telepresent in the patient's vicinity in the form of a robot. It can also hold knowledgeable conversations with the patient. With these use-cases, Alfred aims to take the mundane workload off the caregiver so that they can focus on much more complex and rewarding tasks.

Contents

1 Project Description

Most US citizens above the age of 65 suffer from at least one chronic health-related condition [1]. A survey conducted by ACL [2, pp. 18-19] found that over 40% of elderly citizens have difficulties with mobility. When also afflicted with other health conditions that come with age such as diabetes, arthritis, heart disease, and cancer, senior citizens often enter the care of specialized assisted living facilities to tackle their loss of independence. While these facilities provide quality healthcare and attempt to improve the lives of elderly citizens, living in such facilities often adds to social isolation, and an increased dependence on another human being for carrying out daily routine tasks.

The COVID pandemic's consequences and the deteriorating state of the US labor market have had a significant negative influence on assisted living facilities as well. Over 238,000 people have quit their caregiving positions since the COVID outbreak began, according to AHCA/NCAL [3]. As a result, the market for assistive care is in desperate need of a solution to offer options for patient care and more effective use of labor.

Alfred is a friendly and autonomous mobile manipulator that aims to fill this gap in the market. Using state-of-the-art robotics and AI technology, Alfred can assist elderly citizens with low-risk pick-and-place tasks, and provide rich insights into a user's health. Alfred will be operated primarily using natural language commands in elderly care environments such as assisted living facilities.

2 Use Case

An example use case is described below for Alfred: the friendly assistive elderly care bot, along with illustrations. Pam is a senior citizen who lives in an assisted living facility. While she does have the capability to move around and carry out daily tasks on her own, she needs assistance from caregivers. However, she does not always have a caregiver beside her and often requires help with certain daily tasks such as taking her medicines on time and staying hydrated throughout the day. Her living center has purchased Alfred from Auxilio Robotics, to help Pam with her everyday routine when her caregiver is not around.

Pam's doctor has prescribed some multivitamin tablets that she has to take every day. However, she often needs to be reminded by her caregiver about the same. So, punctual as ever, at 6 pm sharp, Alfred goes to the shelf where the vitamins are kept, picks up the vitamins, brings it back, places them on a table near Pam, and reminds her with a friendly voice to take her medicine. This is just one use case where Alfred autonomously navigates to a pre-defined location, to execute pick-and-place tasks. This was made possible as a caregiver had previously set a scheduled engagement between Alfred and Pam on Alfred's mobile app, asking the robot to remind and deliver medicines to Pam.

On a different day, Pam is watching a soap opera and suddenly craving a nectarine. She realizes that it is on her dining table, but she has been having leg pain the whole week. She simply calls out to Alfred (”Alfred!”) and issues a simple voice command (”Get me a nectarine from the dining table”). Alfred has already been given a map of the environment and understands common locations in the environment such as ”dining table”, ”bed”, ”bedroom”, etc. Alfred interprets the voice command, goes to the dining table, autonomously determines the location of the nectarine on the dining table, picks it up, and brings it back to a table near Pam. Pam is happy as she can continue watching her TV show without any interruptions, and enjoys the sweet nectarine!

Later that evening, Alfred starts buzzing, saying that there is an incoming call from Pam's son John living in a different country. Pam accepts the call using a voice command and allows Alfred

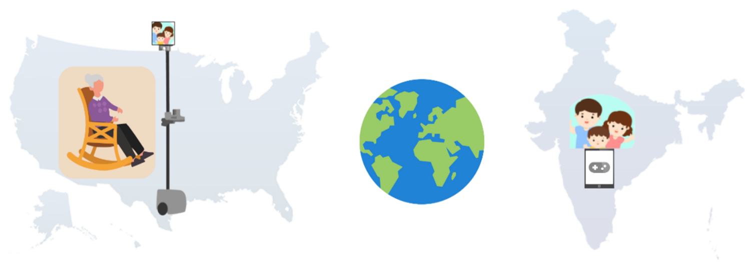
(a)

(b)

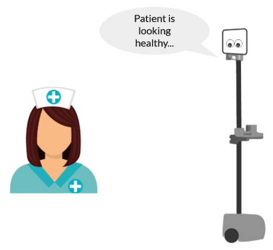
Figure 1: (a) Alfred brings Pam her vitamin tablets on time (b) Alfred provides low-risk assistance to Pam

to be remotely controlled by John. He can use a joystick interface on a handheld device (such as an iPad) to remotely operate Alfred. He is now ”telepresent” with Pam, and not only is he able to hold engaging conversations with her, but is also able to help Pam with her tasks. John sees some clothes to be loaded into the washing machine and uses the joystick interface on his iPad to pick up clothes and load them into the machine. In doing so, Alfred continuously ensures that John does not accidentally hit an obstacle by only accepting teleoperation commands that are safe. The same interface is also used by Pam when Alfred cannot execute his tasks accurately. For example, when Alfred was still learning about Pam’s house and habits, Pam had to often use the joystick interface to guide Alfred to pick up objects and navigate the environment more easily.

Apart from helping Pam with her day-to-day tasks, Alfred also keeps a check on her health. At various times throughout the day, Alfred checks up on Pam to ensure that she is going about her day in a normal fashion. At the end of the day, a report on Pam’s basic activities (gait, amount of movement, most common activity) is provided to the caregiver via a mobile app. Pam can now live worry-free knowing that Alfred can automatically alert caregivers in case there is an anomaly in her health (such as, if she accidentally falls down and requires assistance). By using Alfred, Pam now lives a much happier, healthier, and worry-free life!



(a)



(b)

Figure 2: (a) Pam’s son is now tele-present with her (b) Alfred provides health insights

3 System Level Requirements

After performing extensive needs analysis, and meeting with stakeholders and potential customers, we developed a three-level objective tree for Alfred as shown in Figure 21. Our system requirements are directly derived from our objective tree, literature reviews, and discussion with sponsors and previous teams. We elucidate the system requirements in the following subsections. Kindly note that while the scope of the project is beyond MRSD itself, we have determined our mandatory functional requirements from a sub-part of our primary objectives (Figure 21). These requirements are also not final and will evolve as the project progresses.

All of the system-level requirements have been captured in the following tables. The mandatory functional requirements of our system along with a description of each requirement can be seen in Table 1. Mandatory nonfunctional requirements are elucidated in Table 2. Keeping in mind the long-term vision of our project being successful deployment in an assistive care facility, we have developed a series of desirable requirements that are elucidated in Table 3 (functional) and Table 4 (nonfunctional).

Table 1: Mandatory Functional and Performance Requirements

Functional	Performance	Description
M.F.1 Receive commands from the user: preset speech primitives/handheld interface	**M.P.1.1** Interpret 10 speech templates as tasks. **M.P.1.2** Latency for control commands <5s	The robot will primarily operate using speech inputs. This should be seamless for the user.
M.F.2 Perform basic (pre-defined) social engagement with user	**M.P.2** Fallback rate: <20%	Robot provides feedback upon receiving commands, and should be able to automatically execute pre-defined tasks.
M.F.3 Localize itself in the environment	**M.P.3** Average error <25 cms	
M.F.4 Plan and navigate through the pre-mapped environment	**M.P.4.1** Plan global path to desired location within 2 minutes. **M.P.4.2** Navigate at a speed of 0.2 m/s	
M.F.5 Autonomously avoid obstacles in the environment	**M.P.5** Avoid 100% of static obstacles in range	The robot should not collide with any object as it may render the environment unsafe for users.
M.F.6 Detect objects for grasping	**M.P.6** mAP >= 80% for 10 object categories (e.g bottle, remote, medicines etc) under the following conditions: 1) Object is within 1m of body camera 2) Not kept on a white/transparent surface 3) Adequate Indoor lighting conditions	The conditions have been pre-specified keeping in mind that a robust system in a structured environment is more preferred than an unreliable system in a general environment
M.F.7 Manipulate predefined objects to/from planar surfaces at known locations in the environment	**M.P.7.1** Greater than 70% successful picks and places **M.P.7.2** Manipulation tasks should be completed within 8 min.	The objects will be picked up and placed on flat planar surfaces such as tables only. i.e, allowing ”the user to grasp and remove the object from the end-effector” is out of scope.
M.F.8 Allow approved operators to teleoperate the robot	**M.P.8** Communication latency <5s	
M.F.9 Provide user with robot metrics and video feed of the robot on a handheld interface	**M.P.9** Communication latency: <2s	

Table 2: Mandatory Non Functional Requirements

Requirements	
M.N.1	Appear non-threatening to the user
M.N.2	Be physically compliant to human interaction/contact
M.N.3	Have a simple UI/UX for the handheld interface
M.N.4	Have a modular software architecture for further development
M.N.5	Allow users to pre-schedule tasks/assistance

Table 3: Desirable Functional Requirements

Functional	Performance
D.F.1 Perform all required computation locally	–
D.F.2 Understand and interpret non-template natural language commands	**D.P.2** Fallback rate < 20%
D.F.3 Detect and alert caregivers if the primary user suffers a fall.	**D.P.3** <1 in 100 false positives

Table 4: Desirable Non Functional Requirements

Requirements	
D.N.1	Appear aesthetic to the user
D.N.2	Reasonable cost for the user

4 Functional Architecture

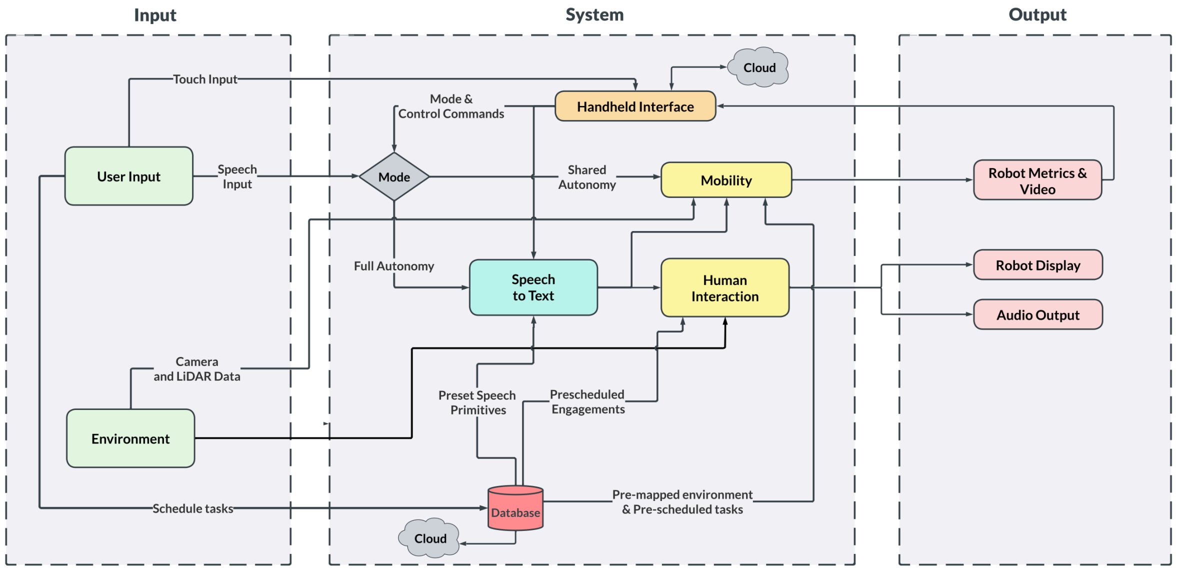


Figure 3: Functional Architecture

The functional architecture represents the high-level functions that our system will perform. It also represents the flow of information from the inputs, through various functional blocks that lead to the outputs of the system. The functional architecture is depicted in Figure 3. The inputs to our system are primarily of two types - User Input and Environment Input. The user input consists of the speech command provided by the user and the handheld interface input. The environment input consists of visual data captured by the robot's camera, and scene-depth information captured by the robot's LiDARs. Our system produces two types of outputs. The first is robot metrics and video, which consists of real-time information about the robot's health, operation mode, and system notifications. This information is displayed on the hand-held interface. The second type of output is the feedback provided by the robot to the user. This can be in the form of visuals (on the robot's display) and audio (through the robot's speakers). Information from the inputs passes through the following functional blocks before resulting in the outputs described previously.

4.1 Speech-to-Text Module

In this module, the speech input from the user, captured by the robot's microphone array, is converted into text using a Speech-to-Text API.

4.2 Human Interaction Stack

The Human Interaction stack is represented in Figure 4. The inputs to this stack are the output text from the Speech-to-Text module. It then uses context parsing algorithms to interpret the context of the text and converts this to a command that can be executed by the robot. This module also generates a feedback response for the user which is sent as audio output and robot display visuals.

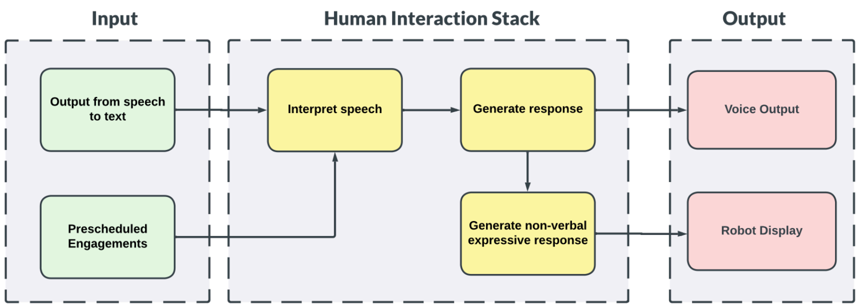


Figure 4: Human Interaction Architecture

4.3 Mobility Module

In this module, the robot uses the input command that has been interpreted from the speech-to-text module to plan and execute the task. As represented in Figure 5, the inputs to this module are various sensor data about the environment captured through the robot's cameras, LiDAR, IMU, odometry sensors, and teleoperation commands (if applicable). This information is then used by the navigation sub-block of the mobility module. In this block, the robot localizes within the environment, plans global and local missions to the destination of the desired object, and sends the appropriate actuation commands to the wheel motors to move the robot. Once the robot reaches the desired location of the object, the robot performs the functions listed in the Manipulation sub-block. This involves determining the object's location and pose, estimating grasp points, planning appropriate trajectories for the manipulator's arm, and finally actuating the manipulator motors to reach and grasp the desired object. Once the robot has successfully picked up the object, the robot needs to return the object back to the user. This involves following the above steps again from the navigation block to the manipulation block. Once the robot places the object near the user, the mission is completed.

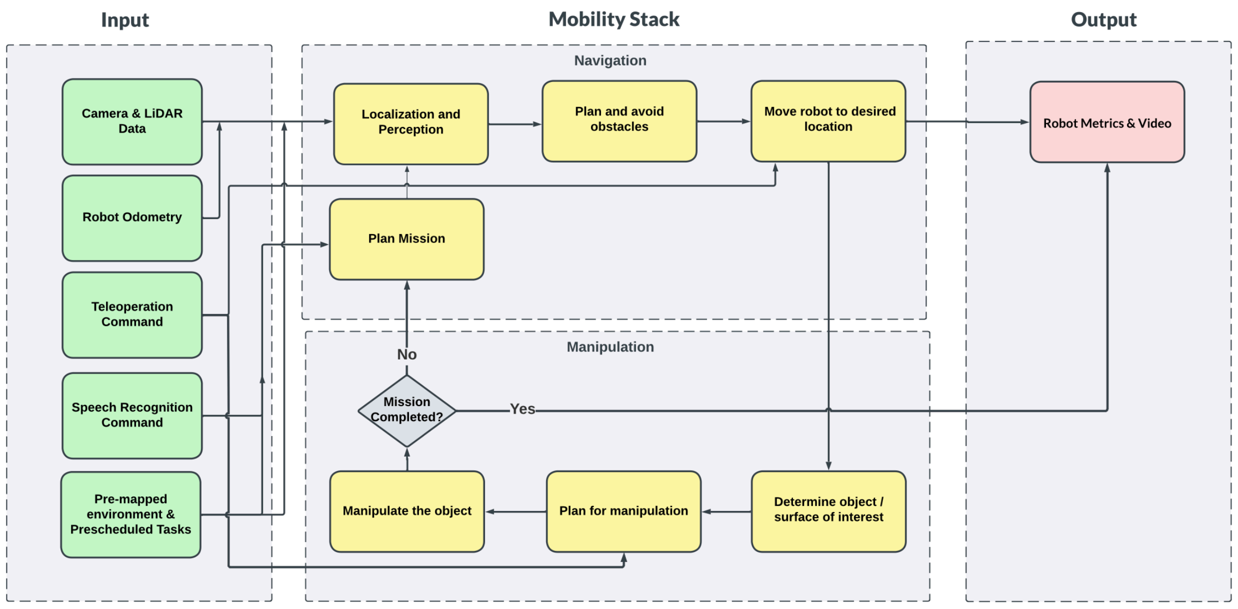


Figure 5: Mobility Architecture

4.4 Handheld Interface

Using the Flutter-framework-based software application on the handheld interface, the user can set the system's operation mode (teleop / autonomy). If in teleoperation mode, the user can manually control the robot through joystick commands. In addition, the handheld interface application consists of a video calling interface for the user to interact with family members and friends, as well as a section for viewing real-time robot metrics and video that is sent by the robot.

5 Cyberphysical Architecture

The cyberphysical architecture is designed considering the sequential flow of functions in the system's functional architecture. The diagram is shown in Figure 6.

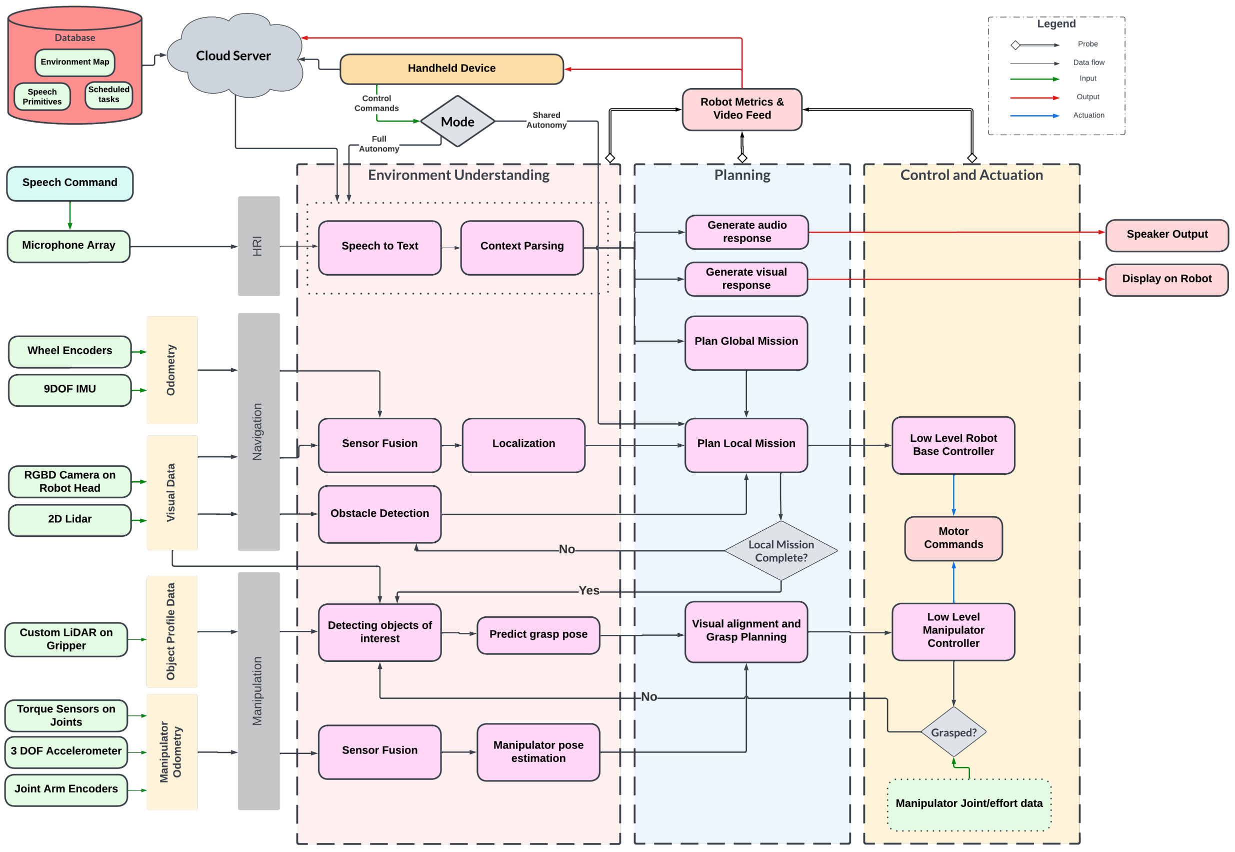


Figure 6: Cyberphysical architecture

The figure above depicts the cyberphysical architecture of the entire system. The architecture is composed of the following components: Environment Understanding, Planning, and Control & Actuation. These are described more in detail below.

5.1 Sensing

The Sensing block (input to the robot) consists of all the physical sensors that the robot utilizes to operate. This block represents the flow of information from each sensor as well as its primary functions.

Navigation odometry is achieved with inputs from the Wheel Encoders and 9-DOF IMU on the base of the robot. Manipulator pose estimation is achieved with inputs from joint encoders, Aruco tags on the gripper as well as a 3-DOF IMU on the gripper. For navigation, the primary sensors used for obtaining visual data are a 2D LiDAR by RPLiDAR and an RGBD camera mounted on the top of the robot. In addition, an RGBD camera on the manipulator has been added to improve manipulation performance.

5.2 Cloud Infrastructure

The system's cloud infrastructure is responsible for enabling the communication of information between the robot and the handheld interface. Since the user operates the robot using a handheld interface, the commands are routed through a cloud server to the robot. Additionally, pre-configured data for robot operation such as HD maps, speech primitives, scheduled tasks, and recorded data are stored on a server. The block also defines the flow of information to each function in the system.

5.3 Environment Understanding

The Environment Understanding block uses incoming information from the Sensor block to interpret the robot's state as well as its surroundings. For interpreting obstacles and objects of interest in its surrounding, the robot uses its visual sensors. Specifically, the robot fuses the data from LiDAR and an RGBD camera. The robot also receives a set of motion measurements from an IMU and wheel encoders at high frequencies and uses this information to estimate the motion of the robot. Using this motion data, an approximate pose of the robot is determined relative to the environment map. The robot is supposed to be operated indoors and hence, it runs GPS-denied navigation.

Since the robot interacts with humans, it listens to their speech and matches it with its own speech primitives from the cloud infrastructure. If a trigger word is detected, it then listens for a command. If a command is detected, the robot plans and executes the specified task. An appropriate response is generated for such tasks and interactions.

5.4 Planning

Once the robot localizes and receives a command from the user, the robot plans for executing the task. The planning block outlines this process on a high level, which involves the robot generating global and local mission plans to reach the goal position while avoiding obstacles. To manipulate the desired object, the robot also plans for manipulation and grasping tasks.

5.5 Control and Actuation

After generating a trajectory for navigation and manipulation in the Planning block, the Control and Actuation block converts this into motor commands which are then sent to low-level controllers to drive the motors. The primary actuators of the robot are the robot drive motors and manipulator joint motors.

5.6 Interface

Using the hand-held interface, the user is able to control the robot via teleoperation, understand the robot's operation mode and metrics, and visualize ongoing tasks through the live video feed from the robot's cameras. All telemetry between the user's handheld interface, the cloud, and the base robot is implemented using the MQTT protocol to enable real-time communication. The robot's operational metrics, such as battery level, ongoing tasks, motion data as well as parsed audio are logged to a database present on a cloud storage server, which can be accessed through the hand-held interface.

6 Current System Status

6.1 Targeted Requirements

For the Spring Semester, our team set a target to meet all system-level requirements, with some relaxations. For example, instead of running our object detector and grasp recognition for 10 objects, we only ran it for 3 objects. The complete list of targeted requirements is shown in Table 5.

Table 5: Targeted Performance Requirements for Spring Semester

Requirement	Status	Subsystem
M.P.1.1 Interpret 10 speech templates as tasks. **M.P.1.2** Latency for control commands <5s	Passed	HRI
M.P.2 Social Engagement Fallback rate: <20%	Passed	HRI
M.P.3 Average localization error <25 cms	Passed	Navigation
M.P.4.1 Plan global path to desired location within 2 minutes. **M.P.4.2** Navigate at a speed of 0.4 m/s	Passed	Navigation
M.P.5 Avoid 100% of static obstacles in range	Passed	Navigation
M.P.6 Object Detection mAP >= 80% for 3 object categories (bottle, apple, banana) under the following conditions: 1) Object is within 1m of body camera 2) Not kept on a white/transparent surface 3) Adequate Indoor lighting conditions	Passed	Manipulation
M.P.7.1 Greater than 70% successful picks and places **M.P.7.2** Manipulation tasks should be completed within 8 min.	Passed	Manipulation
M.P.8 Teleoperation Communication latency <5s	Passed	HRI
M.P.9 Robot Metrics and Video Feed Communication latency: <2s	Passed	HRI

6.2 Overall System Depiction

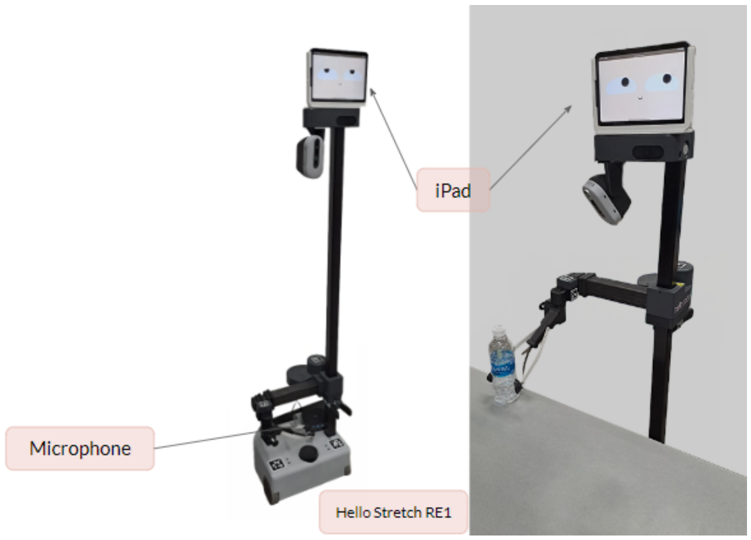


Figure 7: Overall System Depiction

The overall system comprises the Stretch RE1 mobile manipulator as the primary robot. For HRI capabilities, an external microphone is mounted on the base of the robot. Additionally, an iPad is mounted on the head of the robot for displaying the robot's face. Our overall software architecture comprises several different subsystems operating in tandem, coordinated by 4 different layers of finite-state machines. **Level 1:** A high-level mission planner coordinates the global status of tasks that are executed. **Level 2:** Each subsystem has its own local FSM that is used to execute sub-tasks. **Level 3:** Our manipulation subsystem has several complex algorithms that require their own FSMs executing within Level 2 FSMs. **Level 4:** All of our subsystems are orchestrated by action servers which are based on a low-level FSM.

We use a single script in order to launch our entire robot in a coordinated manner. When launched, our custom software automatically ensures robot calibration, runs system checks to ensure all sensors are operating nominally, and ensures that all ROS nodes are running nominally before the robot is ready to receive commands.

6.3 Subsystem Descriptions

6.3.1 Mission Planner

The Mission Planner subsystem enables the integration of the three primary subsystems of our robot: HRI, Navigation, and Manipulation. The Mission Planner manages the current state of the system through a finite state machine (FSM). This FSM triggers different sub-systems of the robot to be activated and perform their part in the larger task. Additionally, all the perception-related computation was performed on a remote GPU device, called the Brain. For low-level robot control, we have developed our own wrapper around the Stretch RE1's low-level controller, called Alfred Driver. The various architectures governing the Mission Planner's decision-making and the Brain / Alfred Driver's work are shown in Figures 8 and 9.

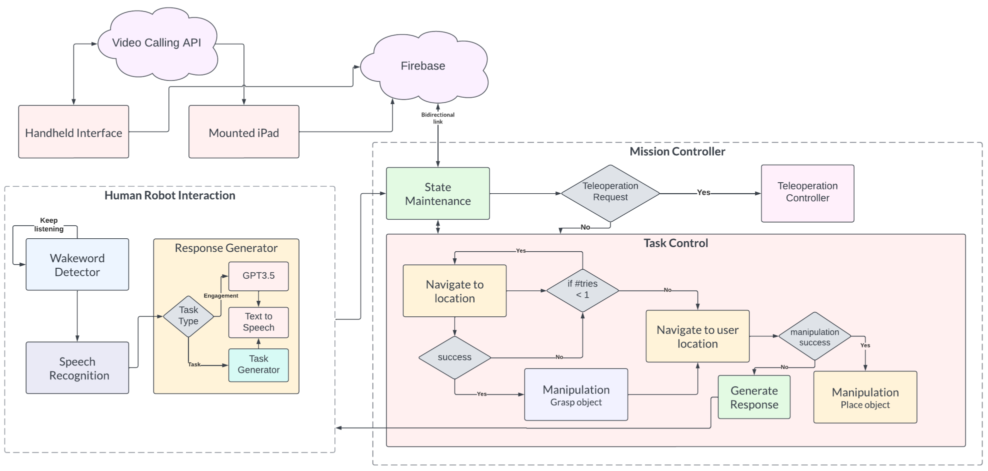


Figure 8: Mission Planner Architecture

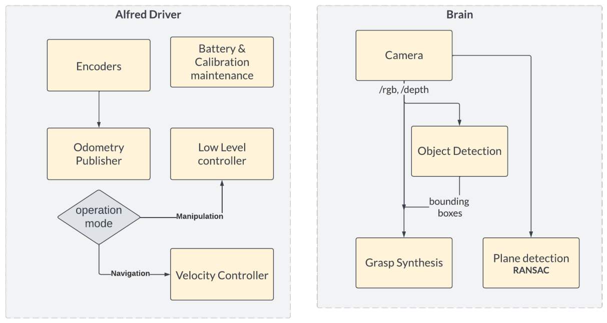


Figure 9: Alfred Driver Architecture (L) and Brain Architecture (R)

6.3.2 Human Robot Interaction

The Human-Robot Interaction subsystem (HRI) enables the Speech Engagement and Telepresence capabilities of the robot.

The Speech Engagement subsystem was implemented with Picovoice for trigger word detection, Google Cloud's speech-to-text API for speech parsing, and DeepMind's Neural2 API for life-like voice synthesis. Additionally, ChatGPT was integrated into the HRI subsystem to handle social engagement requests. The trigger-word detector is always running in the background. When the user says the trigger word, the speech-parsing API is activated. The user's subsequent speech is then parsed and the mission planner activates the respective subsystem to perform the desired task. If the user's request is identified as a pick-and-place command, the mission planner performs task allocation to appropriately identify the desired object to be placed. If the user's request is identified as a social-engagement command, the mission planner offloads the task to ChatGPT to generate a response. Finally, the robot gives feedback to the user through a speech command which appropriately addresses the user's request. A detailed implementation of the HRI subsystem can be found in the architecture below:

The HRI subsystem's telepresence capabilities include teleoperation using a handheld interface UI and video-calling implemented with the Agora API. For enabling wireless communication for video calling and teleoperation, the Firebase cloud service is used. Additionally, an iPad is mounted on

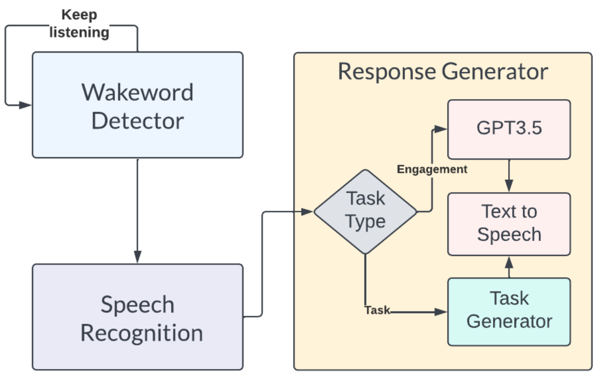


Figure 10: Human Robot Interaction Architecture

the robot's head to display the robot's face, as well as a video-calling screen. The HRI subsystem's telepresence architecture is shown in Figure 10 and the functioning can be observed in Figure 11.

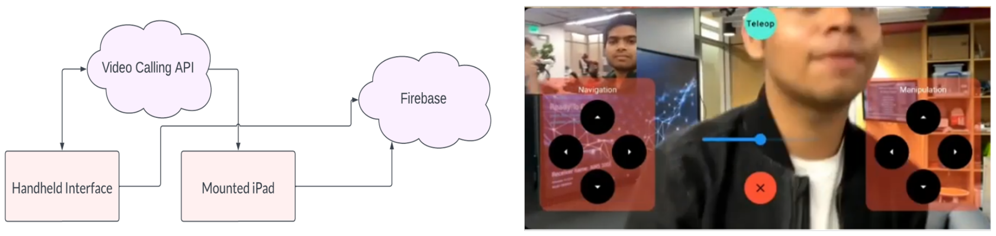


Figure 11: Telepresence Architecture and Handheld Interface UI

6.3.3 Navigation

The navigation subsystem is responsible for moving the robot in the operating environment from an initial location to the desired location of the object, and back to the initial location (where the object is to be placed). It is implemented in Python and C++ using the ROS Noetic platform. At a high level, the navigation subsystem performs functions such as Mapping, Localization, Motion Planning, Dynamic Obstacle Avoidance, Controls, and Actuation. A high-level architecture of the navigation subsystem is shown in Figure 12.

The development of the navigation subsystem started with the generation of a 2D map of the AI Maker Space (test environment). This was generated using SLAM with the help of the gmapping package in ROS. Given that the position of the Stretch RE1's 2D Lidar is at knee-height, several static objects in the environment, such as tables, were not included in the initial map generated. In order to prevent the robot from planning paths through these regions, the 2D map was manually edited using a software called GIMP. Specifically, regions, where the robot should not traverse, were marked as unknown regions (grey color) in the map (Figure 13).

For localization, we used the Adaptive Monte Carlo Localization algorithm, implemented in the amcl package in ROS Noetic. This algorithm would use the robot's real-time odometry and lidar data for the prediction and update steps, respectively. For robot path planning and controls, the movebase

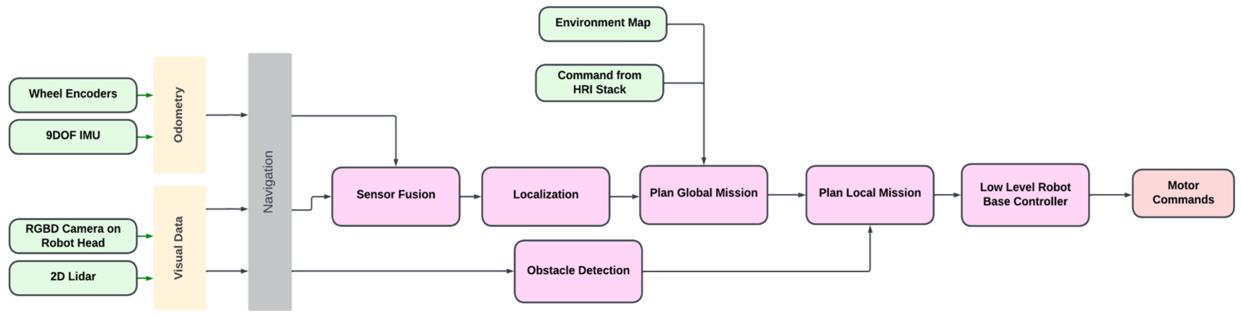


Figure 12: Navigation Subsystem

ROS library was used. We used Dijkstra's algorithm (Navfn ROS) for global path planning and the Dynamic Window Approach (DWA) algorithm for local path planning and controls. While these planners usually generated feasible paths through the test environment, there were times when the robot would struggle to navigate around an obstacle. In these situations, the robot would trigger its recovery behaviors. The recovery behaviors used by our navigation subsystem were Clear Costmap Recovery, Rotate Recovery, and Moveback Recovery. The Moveback Recovery behavior is a custom plugin developed by the team that commands the robot to move back by a set distance and replan a path, in cases where the other two recovery behaviors fail. Additionally, we extensively tuned ROS parameters in Costmap, Planner, and AMCL packages to optimize the robot's performance in the test environment. An image depicting the robot avoiding obstacles during navigation is shown in Figure 13.

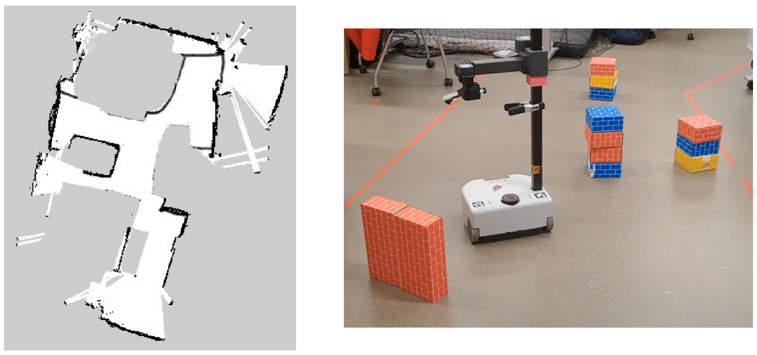


Figure 13: 2D Map of AI Maker Space (L) and Obstacle Avoidance (R)

6.3.4 Manipulation

Our manipulation subsystem handles both pick as well as place requests. The manipulation subsystem begins execution once the navigation subsystem has completed its task. Once a pick request is issued to the subsystem, we execute tasks in sequence as described in our architecture in Figure 14. They are described below:

Visual Servoing: The visual servoing algorithm is responsible for finding the object of interest in the environment and automatically aligning the robot to a graspable position. Each part of the algorithm is described below:

1. **Scanning**: Using the head camera assembly (Intel RealSense D435i) on the robot, we pan the

camera across the environment while parallelly detecting all objects in the environment using the Yolov8 [4] object detector. Using this, we estimate a ground truth location of each object in the world frame and employ a Maximum Likelihood Estimator (MLE) based on a weighted probability score that decays in a hyperbolic fashion with respect to the distance of the object from the camera. This helps filter out all objects that are improbable (such as bottles on the ceiling/floor, etc.). The perception capabilities can be seen in Figure 15.

2. **Visual Alignment**: Once we estimate the location of the object, we perform a sequence of closed-loop control steps that uses visual feedback in order to align the robot within a maximum graspable distance threshold. This allows the object to be within the workspace of the robot for manipulation.

3. **Recovery**: We employ several recovery schemes in order to ensure that the robot doesn't reach an irrecoverable state. These schemes are described in Figure 14.

Grasp Generation: We developed three methods for grasp generation, namely GraspNet [6], GGCNN [5], and Median Grasp. We primarily use the GraspNet algorithm in order to pick a location on the object to grasp. GraspNet is a model trained on a large-scale database of objects and grasps. We obtain a 6D grasp from point clouds observed from the camera focused on the object of interest. However, since we have constraints on the graspable region due to the design of the end-effector, we developed our own heuristics that allow us to generate grasps feasible by our robot. We use the Median Grasp strategy as a fallback in case GraspNet does not produce high-quality grasps. Median Grasp takes in the point cloud of the object and returns the median point in the object as the grasp location.

Plane Detection: We employ a plane detection algorithm based on the RANSAC scheme in order to estimate the plane from which an object is picked. We use this in order to estimate the height at which the object is grasped from the table. This is useful as it gives us an estimate of the clearance required for placing the object on a surface.

Motion Planning and Control: Our current system employs a basic control pipeline that does not consider obstacles within the workspace of the manipulator. We execute Cartesian motions that reduce the probability of collisions.

Grasp Success Validation: Our platform's Stretch Gripper has a parameter called "effort". This measures the current drawn in closing and opening the gripper. Naturally, the thicker the object, the higher the effort required to close the gripper around the object and grab it. Conversely, for thinner objects, we need the effort values will wary, and ultimately, they will differ significantly when the grasp is not successful and the gripper closes into an empty space. We verified this hypothesis by collecting data and plotting graphs and analyzing gripper width v/s effort values. There was a significant difference in the plotted graphs and we added a new label of grasp success to the parameters with Boolean values of 0 and 1. We trained a Logistic Regression model on this data and given a sequence of a grasp attempt, the system is able to tell if the grasp was successful or not.

Our **placing pipeline** is very similar to our picking pipeline. When a place request is received, we utilize a simplified version of our visual servoing module in order to align the robot with the table. We then use the RANSAC algorithm in order to estimate the plane of placing the object and estimate a placing location on the surface of interest. Post this, we use a contact sensing-based placing scheme that allows the robot to automatically sense when the object has been placed on the table and release the object from the end-effector.

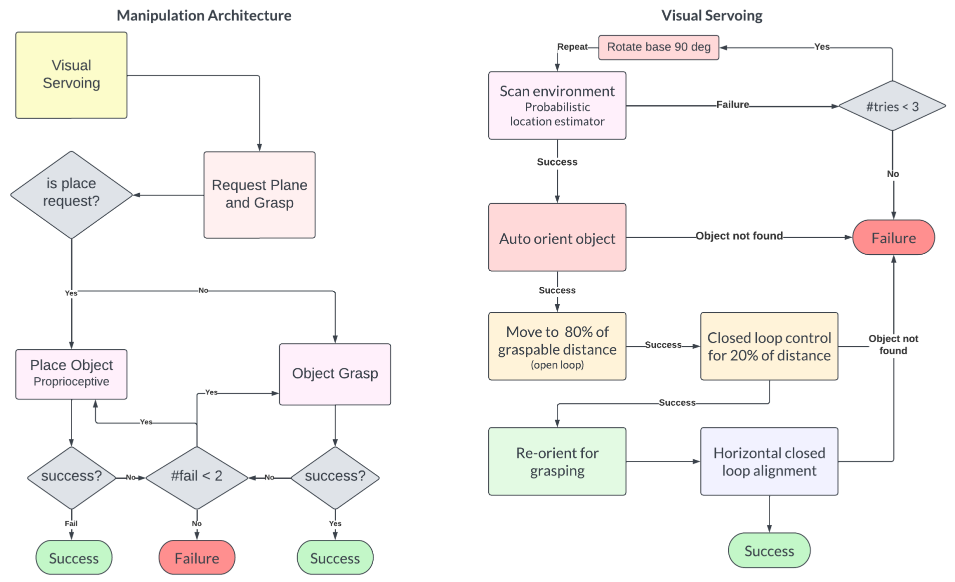


Figure 14: Manipulation Subsystem Architectures

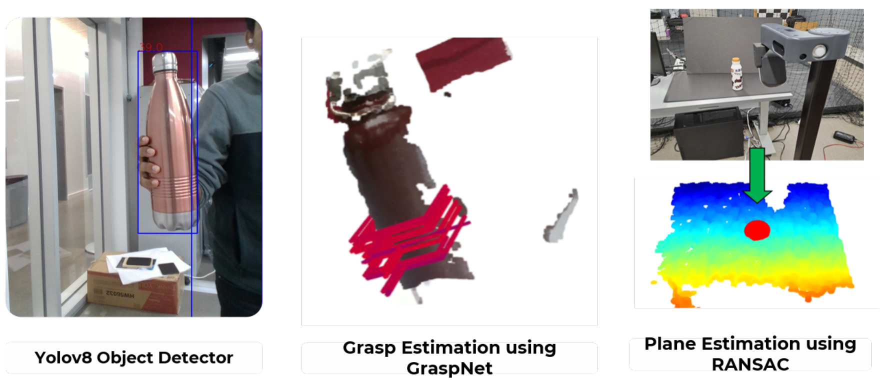

Yolov8 Object Detector
Grasp Estimation using GraspNet
Plane Estimation using RANSAC

Figure 15: Perception Capabilities of Manipulation Subsystem

6.4 Modelling, Analysis and Testing

To analyze the performance of various subsystems in meeting the desired requirements, we performed several tests throughout the course of the semester. The results of these tests are shown in Table 7.

Table 6: Targeted Performance Requirements for Spring Semester

Test	Success Criteria	Result
Speech Recognition Test	Average Transcription Rate should be less than 3.5 seconds	- Mean Transcription Rate: 2.5 seconds - Number of Attempts: 10
Object Detection Preliminary Test	The computed mAP is >= 80% for 3 objects.	mAP = 78.5% for 3 objects when using images from the Coco dataset
Localization Test	The average L2 norm between the true and estimated localization coordinates should be less than 25cms.	Average localization error is 3 cm
Path Planning Speed Test	The average time taken for path planning is less than 2 minutes.	Average time taken for path planning is 587 ms
Manipulation Speed Test	Each manipulation task (pick-and-place individually) complete within 8 minutes of issuing the command.	Performs manipulation task within 1 minute
Obstacle Avoidance Test	The robot was able to navigate in the desired path while avoiding 10 obstacles.	The robot avoids 100% of static obstacles in range
Teleoperation Command Latency Test	Teleoperation Command Latency should be less than 5 seconds	- Mean: 12.88 ms - Std Dev: 0.0096 - Number of Samples: 100
Manipulation Pick-and-Place Test	70% success criteria for picking and placing tasks	- Picking: 80% success rate - Placing: 100% success rate
Videocalling Latency Test	Videocalling Latency should be less than 2 seconds	- Mean: 188.75ms - Std Dev: 71.32ms -Number of Samples: 8

6.5 SVD Performance Evaluation

The set of system capabilities that were demonstrated in the SVD and SVD Encore are shown in Table 7. Our robot performed the set of tasks while meeting the desired system requirements. While the HRI fallback rate was being met, there were issues in the speech transcription accuracy of the HRI subsystem during SVD. This caused the HRI system to misinterpret the user's speech command.

Table 7: Targeted Performance Requirements for Spring Semester

Procedure	Success Criteria	Requirements
User asks Alfred to set up a video call with his/her family	Successful command interpretation and video calling	**M.P.1, M.N.3**
The user teleoperates the robot using the app on a handheld device to pick up an object. Robot metrics and video feeds are displayed on the device.	Communication Latency <5s (visual verification)	**M.P.8, M.P.9**

The user gives a speech command to Alfred to pick up an object. Alfred interprets the command, navigates to the desired location while avoiding obstacles, picks up the object, navigates back to the user, and places the object on a nearby table. All of this is done autonomously without any human intervention. This test is then repeated with a different user, different object, and different obstacle positions.	- HRI: Correctly interprets speech with communication latency < 5s - Navigation: Localizes with error <25cm, plans global path within 2 minutes, avoids 100% of detected obstacles, navigates at a minimum average speed of 0.2 m/s - Manipulation: Successfully detects and picks up the correct object within 3 tries - One cycle of the test is completed within 8 minutes	**M.P.1, M.P.3, M.P.4, M.P.5, M.P.6 (partial), M.P.7**

6.6 Strong/ Weak Points

Our progress through the semester led to a successful demonstration during SVD and SVD Encore. While there are several strong areas of our system, there are also areas that need improvement. The strong and weak points of our system are highlighted below:

Strengths:

- Fully integrated system: Our HRI, Navigation, and Manipulation subsystems are fully integrated and function synchronously to achieve desired pick-and-place / human-interaction tasks.
- Robust Navigation and Manipulation subsystems: Both of these subsystems are robust to unplanned deviations in unstructured environments. Each subsystem is equipped with recovery behaviors that handle input noise/failure cases to prevent complete failure of the system.

Weaknesses:

- Weak Scene Understanding: Our robot currently does not have a full model of its environment when operating, and thus cannot reliably plan for actions.
- Reliability issues in poor lighting conditions: In poor lighting conditions, the Yolo v8 object detector does not accurately detect objects.
- HRI subsystem not yet robust: At the moment, the HRI subsystem sometimes fails to correctly interpret the user's speech command. Specifically, errors in speech interpretation are more frequent when users with non-American accents communicate with the robot.

7 Project Management

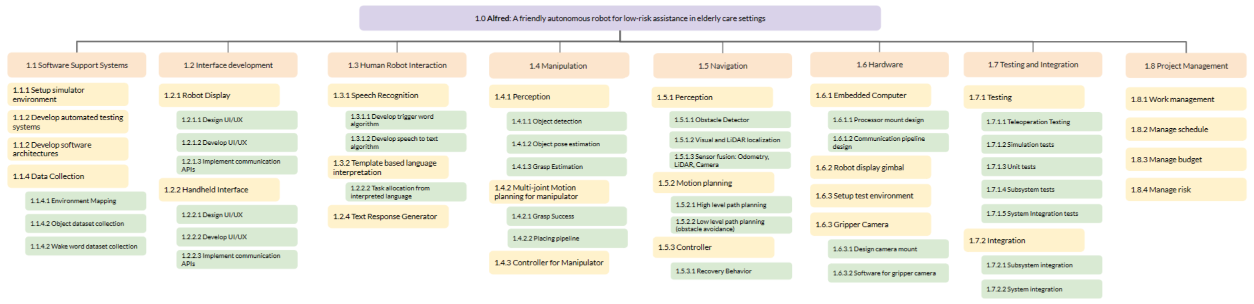

1.0 Alfred: A friendly autonomous robot for low-risk assistance in elderly care settings
1.1 Software Support Systems
1.1.1 Setup simulator environment
1.1.2 Develop automated testing systems
1.1.2 Develop software architectures
1.1.4 Data Collection
1.1.4.1 Environment Mapping
1.1.4.2 Object dataset collection
1.1.4.2 Wake word dataset collection
1.2 Interface development
1.2.1 Robot Display
1.2.1.1 Design UI/UX
1.2.1.2 Develop UI/UX
1.2.1.3 Implement communication APIs
1.2.2 Handheld Interface
1.2.2.1 Design UI/UX
1.2.2.2 Develop UI/UX
1.2.2.3 Implement communication APIs
1.3 Human Robot Interaction
1.3.1 Speech Recognition
1.3.1.1 Develop trigger word algorithm
1.3.1.2 Develop speech to text algorithm
1.3.2 Template based language interpretation
1.2.2.2 Task allocation from interpreted language
1.2.4 Text Response Generator
1.4 Manipulation
1.4.1 Perception
1.4.1.1 Object detection
1.4.1.2 Object pose estimation
1.4.1.3 Grasp Estimation
1.4.2 Multi-joint Motion planning for manipulator
1.4.2.1 Grasp Success
1.4.2.2 Placing pipeline
1.4.3 Controller for Manipulator
1.5 Navigation
1.5.1 Perception
1.5.1.1 Obstacle Detector
1.5.1.2 Visual and LiDAR localization
1.5.1.3 Sensor fusion: Odometry, LiDAR, Camera
1.5.2 Motion planning
1.5.2.1 High level path planning
1.5.2.2 Low level path planning (obstacle avoidance)
1.5.3 Controller
1.5.3.1 Recovery Behavior
1.6 Hardware
1.6.1 Embedded Computer
1.6.1.1 Processor mount design
1.6.1.2 Communication pipeline design
1.6.2 Robot display gimbal
1.6.3 Setup test environment
1.6.3 Gripper Camera
1.6.3.1 Design camera mount
1.6.3.2 Software for gripper camera
1.7 Testing and Integration
1.7.1 Testing
1.7.1.1 Teleoperation Testing
1.7.1.2 Simulation tests
1.7.1.3 Unit tests
1.7.1.4 Subsystem tests
1.7.1.5 System Integration tests
1.7.2 Integration
1.7.2.1 Subsystem integration
1.7.2.2 System integration
1.8 Project Management
1.8.1 Work management
1.8.2 Manage schedule
1.8.3 Manage budget
1.8.4 Manage risk

Figure 16: Work Breakdown Structure (full-size in Figure 22)

7.1 Work Breakdown Structure explained

Figure 16 shows the entire color-coded Work Breakdown Structure (WBS) required to enhance "Alfred". These work packages help us in exactly defining the schedule and work dependencies to achieve our desired milestones as shown in Table 8. We have roughly eight overarching technical work packages: Interface Development, HRI, Manipulation, Navigation, Hardware Setup, Testing and Integration, Software Support Systems, and Project Management. Interface development includes designing web/app interfaces and communication APIs for the handheld device, and the robot display. Hardware development involves designing gimbal mounts and designing PCB, procuring parts for it, printing mounts for relevant sensors, and setting them up on the robot. Software support systems include setting up the simulation environment, physical testing site, and data collection for perception. It should be noted that our core stacks like Manipulation, Navigation, and HRI come directly from our subsystems in the cyber-physical architecture as shown in Figure 6. Over the course of the Spring Semester, we performed unit tests and integration during the development of subsystems itself, and this is captured in the integration and testing package. Lastly, project management is an iterative process that will be consistently done by the project manager and will help us track our current progress, manage team finances, identify and mitigate risks, and help achieve our desired objectives timely. We plan to rotate our project manager once a semester to encourage a good personal learning curve of the team and manage the workload. One minor adjustment that was made to our Work Breakdown Structure throughout the Spring Semester was the addition of grasp success recognition and placing pipeline to the Manipulation work package and recovery behavior to the Navigation work package. This was essential since we needed to know if the grasp was successful or not for a particular object before executing the navigation and placing pipelines. Similarly, even though the obstacles were being avoided successfully, the system could not always plan a path around the object if it got too close. Hence, a recovery behavior was implemented. The reason why this Work Breakdown Structure remains largely the same for our Fall Demonstration Validation as well is that the subsystems and task divisions remain similar. We will be adding 7 objects and implementing 3D Navigation so it is iterating through similar base work with a small difference in the nature of the task. One important thing to note is that although we will discuss the possibility of the implementation of capabilities like Healthcare Analytics and Fall Detection, we would like to not commit to those requirements as our requirements may substantially change based on our customer feedback. The team and advisors alike are in agreement on this.

7.2 Schedule

7.2.1 Biweeklies, Milestones, and Demos

Table 8 shows the major milestones for our project. The internal milestones were strategically decided such that we can better track our progress and achieve major external milestones. Broadly, we develop a basic skeleton structure for covering the requirements of all the subsystems in the early part of the fall semester and iteratively improve them to achieve our mandatory requirements. The system has been divided into three parts:
1) Pick-and-Place
2) Healthcare Analytics
3) Fall Detection
This has been done to allocate personnel to put maximum attention to detail in delivering the requirements of the respective subsystem. Our tentative objectives for some of the major milestones during the project are as follows:

Bi-weekly 1 - Define data points and crucial aspects of sensor modalities that we need in order to capture only the relevant data of the patients. Structure the Fall Detection pipeline, including software overhead and potential sensor/ camera additions. Gather a 3D Map of the environment of AIMakerSpace using RTAB-Map.

Internal Milestone 4 - 4 objects have been identified and trained on Yolo for object detection for pick-and-place tasks. Healthcare Analytics -> Data points are finalized and collection is complete for 1 person. A basic visualization for the nurses is developed. Pose estimation is implemented. Required clearances are discovered. SLAM is implemented.

Bi-weekly 2 - 4 Objects have been trained for the Grasp Success Recognition pipeline. The clearances are worked upon and feasibility is discovered and applied to adjusting the requirements of healthcare analytics. Pose Estimation is running successfully along with the floor detection pipeline. Optimize the 3D Navigation stack for Obstacle Avoidance.

System Development Review - A demo of all functionalities for the entire integrated system is performed on four objects, one patient, and one fall. All mandatory requirements for the Fall semester are expected to be achieved by this Review. The system should be able to pick and place 4 objects while traversing in a 3D Environment and avoid obstacles in its periphery. The system should also be able to detect any patient falls in the environment. In the case of schedule slack, the team re-delegates the work and focuses only on improving mandatory requirements from this point.

Internal Milestone 5 - Failsafe and recovery behavior is put in place. Work, if any is required, is put towards seeking clearances.

Bi-weekly 3 - Refined UI/UX for handheld interface and robot display. Baseline results for object detection for ten objects. Concrete technical details and basic working demo of non-template-based speech grounding and conversational AI.

Fall Validation Demonstration (FVD) - A working demo of the entire integrated system including patient fall detection and healthcare metrics. We expect most of our mandatory and desirable performance requirements to be met at this point. The detailed procedure for FVD is highlighted in Table 10 below.

Bi-weekly 4 - All findings, results, and analyses are documented. The robot demo and capabilities are advertised on respective social media channels.

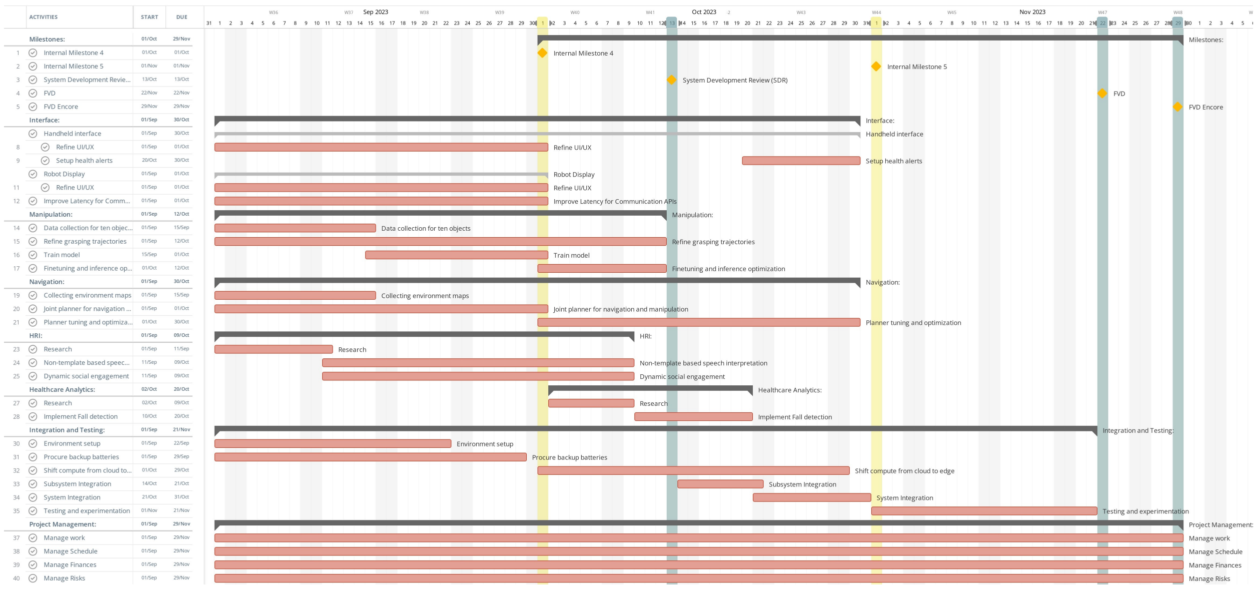
Table 8: Major System Development Milestones (bold are external, rest are internal)

Date	Milestones
15 September	Bi-weekly 1 (B1)
1 October	Internal Milestone (I4)
8 October	Bi-weekly 2 (B2)
13 October	**System Development Review (SDR)**
1 November	Internal Milestone (I5)
15 November	Bi-weekly 3 (B3)
22 November	**Fall Validation Demonstration (FVD)**
4 December	Bi-weekly 4 (B4)

Since we have met all our deliverables for the Spring Semester, it is safe to assume that we are on schedule. The next assessment of our schedule will be better evaluated in the Fall.

7.2.2 Schedule

Our Fall schedule can be seen represented in a Gantt chart format in Figure 17. This is the same as presented in the CODR as we are not officially committing to anything new other than what we presented for the CODR.



	ACTIVITIES	START	DUE
	Milestones:	01/Oct	29/Nov
1	Internal Milestone 4	01/Oct	01/Oct
2	Internal Milestone 5	01/Nov	01/Nov
3	System Development Revie...	13/Oct	13/Oct
4	FVD	22/Nov	22/Nov
5	FVD Encore	29/Nov	29/Nov
	Interface:	01/Sep	30/Oct
	Handheld interface	01/Sep	30/Oct
8	Refine UI/UX	01/Sep	01/Oct
9	Setup health alerts	20/Oct	30/Oct
	Robot Display	01/Sep	01/Oct
11	Refine UI/UX	01/Sep	01/Oct
12	Improve Latency for Comm...	01/Sep	01/Oct
	Manipulation:	01/Sep	12/Oct
14	Data collection for ten objec...	01/Sep	15/Sep
15	Refine grasping trajectories	01/Sep	12/Oct
16	Train model	15/Sep	01/Oct
17	Finetuning and inference op...	01/Oct	12/Oct
	Navigation:	01/Sep	30/Oct
19	Collecting environment maps	01/Sep	15/Sep
20	Joint planner for navigation ...	01/Sep	01/Oct
21	Planner tuning and optimiza...	01/Oct	30/Oct
	HRI:	01/Sep	09/Oct
23	Research	01/Sep	11/Sep
24	Non-template based speec...	11/Sep	09/Oct
25	Dynamic social engagement	11/Sep	09/Oct
	Healthcare Analytics:	02/Oct	20/Oct
27	Research	02/Oct	09/Oct
28	Implement Fall detection	10/Oct	20/Oct
	Integration and Testing:	01/Sep	21/Nov
30	Environment setup	01/Sep	22/Sep
31	Procure backup batteries	01/Sep	29/Sep
32	Shift compute from cloud to...	01/Oct	29/Oct
33	Subsystem Integration	14/Oct	21/Oct
34	System Integration	21/Oct	31/Oct
35	Testing and experimentation	01/Nov	21/Nov
	Project Management:	01/Sep	29/Nov
37	Manage work	01/Sep	29/Nov
38	Manage Schedule	01/Sep	29/Nov
39	Manage Finances	01/Sep	29/Nov
40	Manage Risks	01/Sep	29/Nov

Sep 2023
Oct 2023
Nov 2023
Shift compute from cloud to edge
Improve Latency for Communication APIs
Joint planner for navigation and manipulation
Non-template based speech interpretation
Finetuning and inference optimization
Planner tuning and optimization
Data collection for ten objects

Figure 17: Fall 2023 Schedule and Milestones (full scale in fig 23)

7.3 Test Plan

For the Fall Semester, not only do we have the additional capabilities, but we are majorly improving our current system's capability by introducing enhancements in the form of 3D Maps and improving our localization, mapping, and obstacle avoidance during navigation. For the sake of lack of redundancy, we are not including parts of the system that represent already satisfied requirements/ tests again as part of our Testing Activities.

7.3.1 Testing Activities

- **Accuracy Thresholds**
  - Our system's voice-detection accuracy needs to improve. In our demo, the task template matching was perfect. The only issue that arose was inaccurate speech detection. We plan to improve on this.
  - As for fall detection, the user pose should be recognized accurately. So should the floor plane that the robot detects.
  - Since we'll be testing the system's capability in a 3D map environment for the first time, the localization and mapping need to be as accurate as possible.
  - The sensor readings that we get from the user should be filtered thoroughly eliminating any kind of noise over time.
  - The grasps predicted for grabbing the object should take into account the orientation of the picked object as we don't intend for it to topple over while the placing pipeline is being executed, as was observed during our Spring Validation Demonstration.
  - The added modalities towards the end of the Spring Semester including Grasp Success and Plane Detection pipelines will be tested with more robust performance requirements.

- **Latency Tests**: Once the fall is detected, it needs to be reported to the nurse almost instantly. For this, the latency performance requirement needs to be on point. The communication latency between the nurse's request for user health reports and the displaying of data should be minimal.

- **Object Detection**: Yolov8 was very helpful to us in our Spring Validation Demonstration but testing its limits for repeatability and accuracy for 7 additional objects will be key to our success for the Fall Validation Demonstration.

- **Obstacle Avoidance**: Moving to a 3D Map environment means that we'll have a larger capability to recognize obstacles in the periphery of the robot since we won't rely on the LiDaR alone anymore. We plan to test the system's limit with dynamic obstacles introduced.

7.3.2 Fall Semester Capability Milestones

Table 9: Fall Capability Milestone

Progress Review	System Capabilities
PR 7	The team should have a solid understanding of what sensor setup we need in order to capture the relevant patient data. A structure should be laid out defining the task flow of the Fall Detection pipeline. The system should have at least one complete 3D map of the AIMakerSpace.
PR 8	The system is able to detect 4 new objects in addition to the three from Spring Semester. Pick-and-place is successful for these 4 objects. The system is able to make out the pose of the patient. The system can navigate in its 3D environment using SLAM. The system has testing healthcare data in place for one person.
PR 9	The system is able to make out successful grasps from unsuccessful ones using the Grasp Success Recognition pipeline on the new 4 objects. The system is able to detect falls using pose estimation and by detecting the floor plane in the environment.
PR 10	The clearances required for deployment in the Assisted Living Facilities are worked upon. The system is able to robustly navigate and perform pick-and-place tasks for 4 objects in its new environment.
PR 11	In addition to PR 10, the system is also able to detect if a patient has taken a fall in the robot's environment. The robot is also able to quickly alert the nurse if such a scenario comes up.
PR 12	In addition to the system capabilities displayed in PR 10 and P 11, the nurse can request health analytics feedback for a particular patient and a visualization will come up on the nurse's screen.

7.3.3 Fall Validation Demonstration

The test conditions are as follows:

- **Location**: AI Makerspace, Tepper
- **Equipment**: Stretch RE1 by Hello Robot with iPad display, handheld device
- **Operating area**: Elderly care setting-like environment reconstructed in AI makerspace spanning approximately 30-40m

The Fall Demonstration procedure can be found in Table 10.

Table 10: Fall Demo Procedure

Steps	Procedure	Success Criteria	Requirements
1	Pam (senior citizen) asks Alfred to set up a video call with his/her family	Successful command interpretation and video calling	**M.P.1(partial)** **M.N.3**
2	User (Pam's family members) teleoperates the robot using the app in handheld device to pick up an object for her	Communication Latency <5s	**M.P.8**
3	Robot prevents obstacles while moving in teleoperation and performs the manipulation.	Avoids 100% of obstacles	**M.P.5**
4	Pam gives speech commands to Alfred to pick up an object.	Correctly interprets speech with communication latency <5s	**M.P.1 (partial)**
5	Robot localizes itself in the pre-mapped environment	Localization with error threshold<25 cms	**M.P.3**
6	Robot interprets the command and plans a global path using predefined heuristics	Correct interpretation of the task Plans global path within 2 minutes	**M.P.1.1, M.P.4.1**
7	Robot navigates and reaches the goal location while avoiding static and dynamic obstacles.	Avoids 100% of obstacles Navigates at an average speed of 0.4 m/s)	**M.P.5, M.P.4.2**
8	Robot detects the object and estimates the grasping location	Successful object detection for chosen object category. mAP>=70% for 10 object categories	**M.P.6 (partial)**
9	Robot plans a path towards the grasping location and complete grasping	Greater than 70% successful picks Time to completion <=8min	**M.P.7.1,M.P.7.2**
10	Robot navigates back to home location and places the object on user's table	Avoids 100% of obstacles Navigates at an average speed of 0.2 m/s) Greater than 70% successful placements Manipulation task should be completed within 8 min	**M.P.5,M.P.4.2** **M.P.7.1,M.P.7.2**

11	Robot metrics and video feed of the robot are displayed on device of the user throughout the mission	Communication Latency <2s	**M.P.9**
12	User (facility staff) uses the app to preschedule tasks/reminders	Successful task allocation	**M.N.5**
13	Robot reminds Pam of this operation and performs it following the steps from 5) to 11)	Fallback rate <20% Successful task completion	**M.P.2** **same as steps 5 to 11**
14	Robot alerts user of mission status in app when object is not found in the scene	Correct alert Communication Latency <2s	**M.P.9**
15	User takes over the autonomous mode and looks around the scene to select the object	Latency for control commands <5s	**M.P.1.2**
16	User selects the desired item and robot follows the steps from 8) to 11)	Latency <5s Successful task completion	**M.P.1.2** **same as steps 8 to 11**
17	User pretends to take a fall near the robot. Robot gets alerted upon the noise of collision.	Fallback rate = 0% Successful noise detection	**- Stretch -**
18	Robot scans the area and detects user pose and floor plane.	Fallback rate <10% Accurate pose and floor detection	**- Stretch -**
19	If a fall is detected, the robot alerts the nurse.	Communication Latency <5s Successful task completion	**- Stretch -**
20	The nurse attempts to fetch the health report of the patient.	Response Latency <3s	**- Stretch -**

Other than the pick-and-place capability illustrated several times before, here is a visual depiction of the modalities our Fall Validation Demonstration will feature. (Figures 18, 19, and 20)

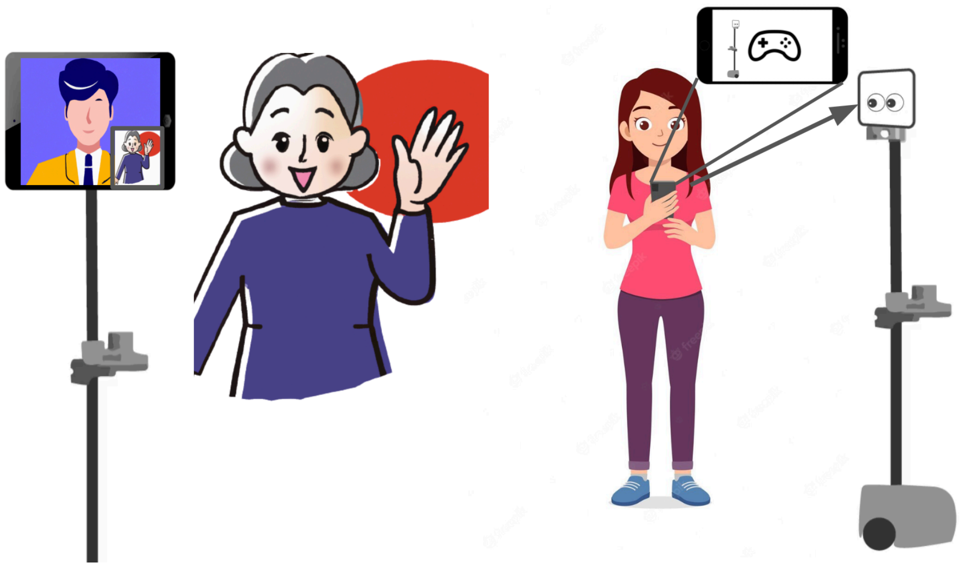


Figure 18: Tele-operation and Video-Calling

Figure 19: Nurse retrieving patient's status

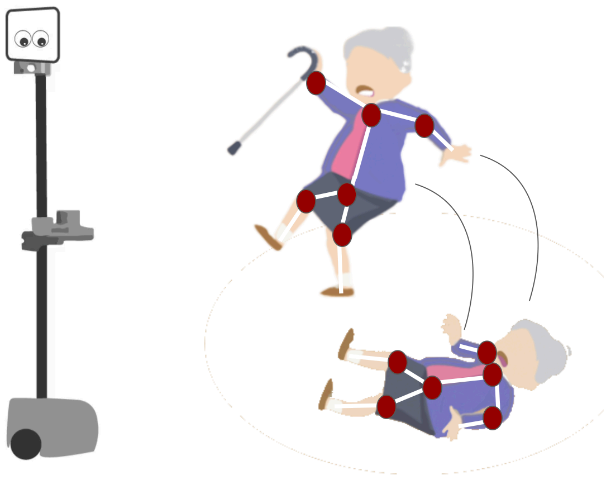


Figure 20: Robot detecting patient's fall

7.4 Budget

A majority of our big-ticket purchases include the microphone for the HRI subsystem. This is the biggest purchase yet ($84.79). The next biggest expenditure was on an advanced presentation pointer that we made good use of during our presentations throughout the semester. The other purchases were around the $20 mark including components acquired for PCB, and a couple of tablecloths. Table 11 shows the total expenditure as $373.36. We have spent a total of 7.47% to date.

Table 11: Budget: Our expenditure so far at a glance

S. No.	Part Details	Price
1	Logitech Professional Presenter R800	$49.68
2	4K HDMI Dummy Plug - Virtual Monitor Display Emulator, Headless Display Adapter Supports up to 3840x2160@60Hz, 1080@120Hz DVI EDID Emulator (Single)	$12.71
3	Seadream Coiled USB C Cable,Coiled USB C to USB 2.0 A Male Charger Cable	$8.79
4	ruthex 1/4" Threaded Inserts - 20 Pieces RX-1/4“-20x12.7 Brass Heat Set Insert for Plastic Parts	$11.65
5	binifiMux 1/4-20 x 2 Inch Flat Head Allen Bolt Joint Connecting Bolts Socket Cap Screws	$7.41
6	Audio-Technica ATR4697-USB Omnidirectional Condenser Boundary Microphone	$41.73
7	uxcell Knurled Insert Nuts - 25Pcs 1/4-20 Female Thread Brass Threaded Insert Embedment Nut for 3D Printer	$13.24
8	Anker PowerConf S330 USB Speakerphone	$84.79
9	Bussmann Division BP/ATM-7-1/2-RP ATM-7.5 Auto Fuse	$5.59
10	Adafruit VL53L0X Time of Flight Distance Sensor - ~30 to 1000mm	$14.95
11	Futaba S3003 Standard Servo	$15.74
12	PJ-067B	$2.03
13	PH1-03-UA	$0.20
14	PH1-05-UA	$0.10
15	UVR1H0R1MDD1TD	$0.28
16	UVR1HR47MDD1TA	$0.27
17	UVR1H0R1MDD1TD	$0.28
18	B32529C0183J289	$0.35
19	HC-49/U-S16000000ABJB	$0.55
20	UVR1E330MDD1TA	$0.29
21	ATMEGA328P-PU Arduino Bootloader	$6.90
22	LM3940IMP-3.3	$2.58
23	C503B-RAN-CZ0C0AA2	$0.18
24	K220J15C0GF5TL2	$1.62
25	RNF14FTD10K0	$0.60
26	Romanstile Rectangle Tablecloth	$13.24
27	UniQloth Rustic Farmhouse Rectangle Tablecloth	$18.01
28	Amazon Basics HDMI to DVI Adapter Cable	$8.10
29	DAOKI 50Pcs DIP IC Sockets 28Pin Narrow 2 Row 2.54 mm IC Sockets Adapter Solder Type Socket	$8.15

7.5 Risk Management

We identified six major risks last semester and assigned a risk owner for every single one of them. There were multiple ways in which these risks could have been triggered, but so were the ways of mitigating them. Our focus was and continues to be broadly on the risks that directly harm our project schedule. Table 12 elucidates the description, impact, type, and mitigation plan for said risks. Each risk has been further discussed in Tables 14, 15, 16, 17, 18, 19. The risk that has been experienced the most by us during the course of the Spring Semester was Hardware Failure. It began with the Intel Realsense failing on the robot which significantly hampered the progress of our manipulation subsystem as the perception modules of object detection, grasp recognition depended on it. We mitigated it using our pre-defined mitigation tactics of being in touch with Hello Robot and Team TouRI. We never faced the case of no support from Hello Robot as we constantly kept in touch with them and established good relationships with their team. The potential compute barrier was tackled by offloading our system's heavy computations onto a remote GPU. The other risk very slightly encountered was slack in the project schedule. Each time there was the unavailability of a teammate, or a task was getting too heavy, we were able to reallocate personnel and not let the schedule be hampered by it. How we keep track of our risks has been discussed below.

Table 12: Risk Management

S. No.	Name (Likelihood, Consequence)	Description and Impact	Type	Mitigation
1	No support from Hello Robot (1,3)	Loss of contact with Hello Robot due to unresponsiveness	Technical	- Connect with previous teams who have used Stretch - Consult online support forms adhering to Stretch
2	Testing/ development space becomes unavailable (1,3)	Testing stops indefinitely	Logistic	- Allocate some people to develop test environment/ transfer current work to backup environment - Allow remaining members to continue testing in-simulation until transfer is complete
3	Hardware Failures (4,3)	- Battery drains too quickly leading to poor performance - Mechanical vibrations in tablet causing poor video quality and customer experience	Technical	- Coordinate with sponsors and arrange a backup battery beforehand - Improve gimbal design with dampers and digital/optical stabilization
4	Slack in Project Schedule (3,4)	- A teammate being unavailable due to personal reasons - A team member finding it difficult to complete their task because of lack of skill or complexity of the problem	Personnel	- Assign tasks to people who are done with their tasks earlier than scheduled - Weekly scrums and daily stand ups to keep track of any technical assistance requirements so that a task doesn't get unreasoably stretched out
5	Compute insufficient for application (3,3)	Heavy algorithms leading to inability of system to operate at optimum capability	Technical	- Add a better onboard computer - Shift non-time critical requirements to cloud
6	Lack of availability of components (1,3)	Unavailability of cameras, battery and jetson that delays project schedule	Technical	- Switch to Realsense - Keep the robot plugged in

One way we were tracking our risks throughout the semester was by putting our mitigation tactics into practice and logging their effectiveness. We pivoted to a different tactic if a particular tactic didn't work. All of this was logged in the following manner, as can be observed in Table 13. Here, our major risks have been listed. We log its first occurrence in the column ”Occurred yet?” and then apply a mitigation strategy. If the strategy works, we mark ”Need of Action still?” as ”No” and log our findings in the reasoning column. As you can see, for our Hardware issues, starting from the fuse failure we were quickly able to get a replacement because of our contacts in Prof Zackory's lab. Our Intel Realsense pan/tilt and robot base's issues were constantly being tackled not just by us but by Hello Robot via Zoom. Although we were able to get our robot working because we managed to put working parts together, we haven't made the ”Need of action still?” to ”No” because we want the robot to work independently and right now we have two robots, one with the head not working, and one with the wheel not registered on the bus. In the future, if anyone wants to work on Stretch, or if we want to not rely on our main system forever without a backup, we would need those robots fixed and hence that mitigation strategy is still under process. Conversely, this has been answered ”No” for Risks 2 and 3 because Risk 3 was mitigated by offloading compute to a remote GPU and that works for us now so there’s nothing else we need to do. We have managed our entire lists of risks in a similar manner but since there were no logs filed for those, we are not showing them in this table to avoid redundancy.

Table 13: Risk Tracking

S No	Risk	Owner	Occured yet?	Need of Action still?	Risk Management effective? Cite reason
1	Hardware Failure	Shivam	Yes	Yes	Partly yes, because we were able to replace the fuse and buy backups almost instantly. We have constantly been in touch with the Hello Robot team to get quick insights on the hardware issues we continue to face. We have also been able to fetch 2 Stretch RE1s from Prof Oliver Kroemer and been able to transfer grippers between them to get them to work.
2	Slack in project schedule	Shaolin	No	No	
3	Compute insufficient for our application	Atharva	Yes	No	Yes, better onboard computer added. Mitigation strategy worked!

8 Conclusions

8.1 Lessons Learned

Integration is not a milestone, it’s a continuous improvement and optimization process.

As we approached our previously planned Milestone 1 around the onset of the Spring Break, we realized we had many hardware issues limiting the successful full-system integration dry run of our system. This meant we had to offload integration to simulation and test it there. It was tough to get perfection during the full-system run as the robot would sometimes face issues in Navigation or Manipulation. The respective subsystem would then be worked upon till it’s better. As time progressed and we had our hardware problem fixed, we realized there was no way to meet the perfect ”Milestone” we had set for ourselves initially. We can only strive for it, never achieve it. This shift in mindset really geared us to fix failures, rather than getting demotivated by them. And every small success in a bug fix was celebrated. Until after our demo, we never quite met perfection. However, if you compare the state of the system now to what it was before Spring break, there have been substantial robustness improvements with some recovery behavior and fail-safe measures in place. This is a lesson we plan to carry forward for the rest of our professional lives.

Assume failure and design a schedule around that.

As taught to us in the Systems course, never ever plan a schedule for the ideal case. Always assume failure and design a schedule around it so that if your system works perfectly and finishes before the deadline, you've exceeded expectations but if it fails and causes slack in the project schedule, you've already accommodated it because you assumed failure at the beginning. Thus, it doesn't come off as a shock but as an expected error which we already have a plan in place for. And this was crucial in application to our team because if we didn't put the milestone for subsystem developments before Spring Break, we would've not begun our integration cycle then. This would mean that we would start our full-system integration around April and that would cut it really close to Spring Validation Demonstration. In hindsight, given the issues we faced with hardware and integration alike, it would be a near-impossible task to pull off a successful demo if we started our integration in late March or early April. Conclusively, planning around assumed failure really paid off for us.

Hardware fails, the team shouldn't, be patient.

Risk mitigation was stressed a lot and we made it a point to assess our risks and mitigate them really well. One of those was hardware failures. Our strategy to keep in contact with Hello Robot and resources in Robotics Institute where we can fetch a backup robot came to fruition as each time we would have a hardware hiccup, either one of these tactics would lead us to a solution. The one thing that led the team to success was being patient and calm at all times. Despite failures right on the day of our demos like the wheel coming off or the head pan/tilt not working again, we stayed calm, stuck to our fixing routine, and came out of it successfully on the other side. We were always aware of the gravity and the specifics of the root of the problem and did everything we could with our knowledge, awareness of resources, and contacts available to get it fixed. A lack of risk plan would've driven panic instead of strategy if we didn't define these things beforehand.

8.2 Key Fall Activities

Decide on the most relevant 10 objects

As part of our declared system requirements in the past, we plan to extend our system's current capability of pick-and-placing 3 objects to 10 objects. Deciding on the 10 relevant objects is crucial as the robot is supposed to be deployed in an Assisted Living Facility so we will converge on the ten most relevant objects for that environment based on our customer interviews and testing experience.

Converge requirements with customer demands

After our Spring Validation Demonstration Encore and meetings with faculties, we realized that extending the system's capability to 10 objects might be too trivial for the entire duration of a semester. So, with the trip to Vitalia - North Olmsted Assisted Living Facility and their unique requirements in consideration, we thought it would make sense to cater our requirements to their demands. This would not only add complexity to our Fall Semester workload but also aligns with our piloting goals as we hope to gain the trust of at least one such facility.

Pilot our robot in an Assisted Living Facility

With feedback from a little over 10 customer interviews, and discussing requirements and pain points of assisted living facilities, one thing is for certain. There is a severe labor shortage problem and it comes with its own implications for staff and patients alike. A robot can have multiple applications in such facilities without breaching the dexterity of personal care expectations. This ”low-risk assistance” feature of our system is what we plan to bring to reality as we hope to see our robot piloted and operating in these facilities. Moreover, we hope to learn from the feedback and see how the system can be made more robust and see if more use cases can be discovered.

References

[1] 2020 profile of older americans. https://acl.gov/sites/default/files/Profile%20of%20OA/2020ProfileOlderAmericans_RevisedFinal.pdf. Accessed: 2022-12-13.

[2] 2021 profile of older americans. https://acl.gov/sites/default/files/Profile%20of%20OA/2021%20Profile%20of%20OA/2021ProfileOlderAmericans_508.pdf. Accessed: 2022-12-13.

[3] New bureau of labor statistics data shows workforce crisis continues among long term care facilities. https://www.ahcancal.org/News-and-Communications/Press-Releases/Pages/New-Bureau-Of-Labor-Statistics-Data-Shows-Workforce-Crisis-Continues-Among-Long-Term-Care-Facilities.aspx. Accessed: 2022-12-13.

[4] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.

[5] Douglas Morrison, Peter Corke, and Jürgen Leitner. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *CoRR*, abs/1804.05172, 2018.

[6] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. *CoRR*, abs/1905.10520, 2019.

A Appendix

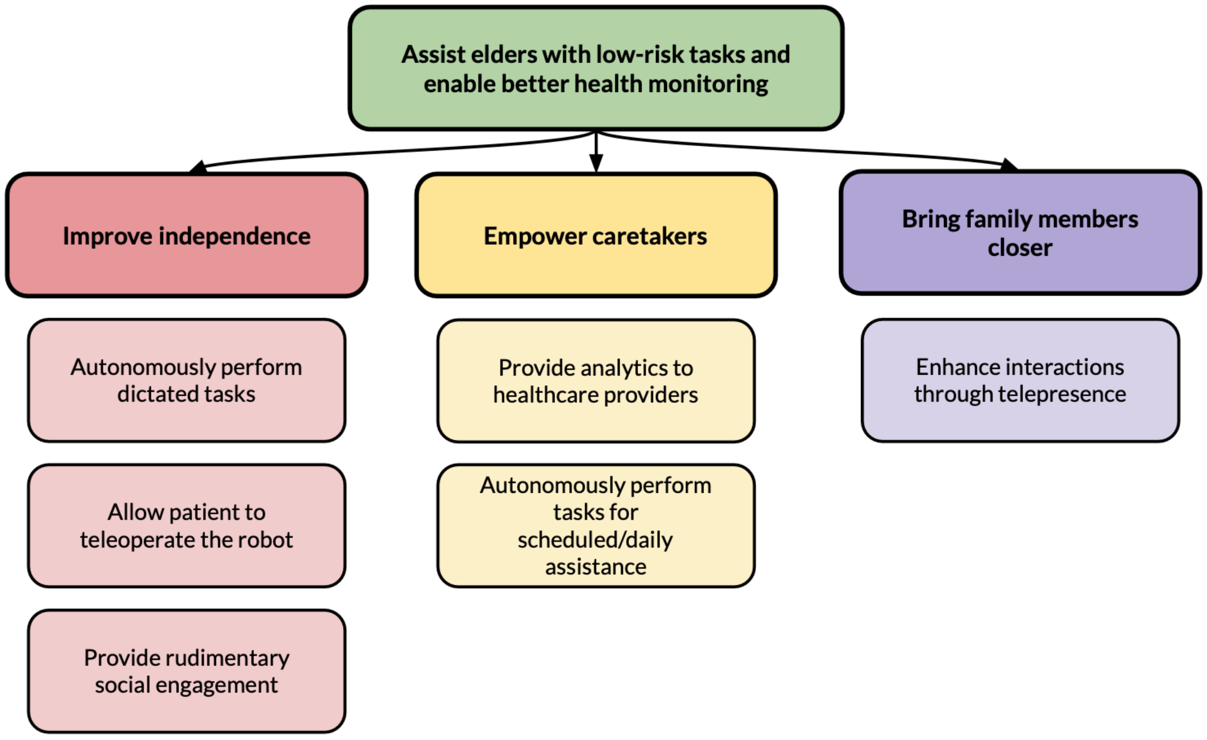
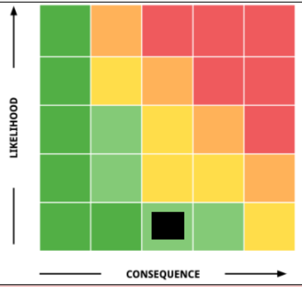

Assist elders with low-risk tasks and enable better health monitoring
Improve independence
Empower caretakers
Bring family members closer
Autonomously perform dictated tasks
Allow patient to teleoperate the robot
Provide rudimentary social engagement
Provide analytics to healthcare providers
Autonomously perform tasks for scheduled/daily assistance
Enhance interactions through telepresence

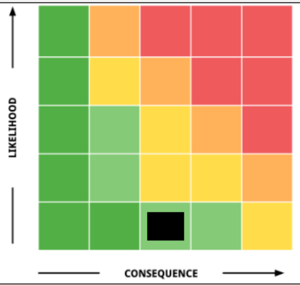
Figure 21: Objectives tree

Table 14: No support from Hello Robot

Risk Title	No support from Hello Robot Company	Date Submitted	Dec 1 2022
Risk Owner	Praveen Venkatesh	Date Updated	Dec 1 2022
Description	Risk Type		
We may lose contact with Hello due to several reasons that may be personal, economic, political, etc	Technical		
	Metrics (L, C, Exposure)		
	1, 3, 3		
Consequence			
Severe technical issues in case of problems with hardware / core software provided by Hello			

Mitigation Strategy			
Action	Date of Implementation	Expected outcome	Comments
Connect with the other stretch robot users for help via online forums and communities	-	Able to resolve technical difficulties	Connect in december for preventive action
Connect with previous teams using the stretch (Vertical Farming and TouRI)	-	Able to resolve technical difficulties	Maintain good relations with senior teams

Table 15: Testing/ Development space becomes unavailable

Risk Title	Testing/development space becomes unavailable	Date Submitted	Dec 1 2022
Risk Owner	Praveen Venkatesh	Date Updated	Dec 1 2022
Description	Risk Type		
The development space may become unavailable for a variety of reasons that we don't have control over.	Logistic		
	Metrics (L, C, Exposure)		
	1, 3, 3		
Consequence			
Progress of project is impacted as the testing site may have to be immediately shifted; Testing becomes significantly harder			

Mitigation Strategy			
Action	Date of Implementation	Expected outcome	Comments
Allocate a few people to develop/ transfer work to backup environment	-	Allow for a switft switch between the primary and backup setting to not delay integration or testing	Decide on backup environment before-hand
Allow remaining members to continue development in-simulation until switch is complete	-	Development speed is not hindered heavily	

Table 16: Hardware Failures

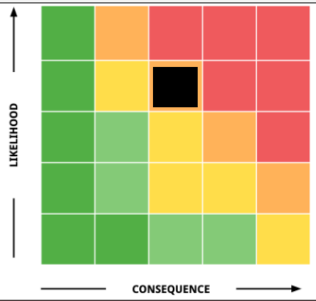
Risk Title	Hardware Failures	Date Submitted	Dec 1 2022	
Risk Owner	Shivam Tripathy	Date Updated	Dec 1 2022	
Description	Risk Type			
1) Battery drains too quickly leading to poor performance 2) Firmware issues in hardware causing problems in system integration 3) Mechanical vibrations in tablet causing poor video quality and customer experience	Technical			
	Metrics (L, C, Exposure)			
	4, 3, 12			
Consequence				
Progress of project is impacted as the testing site may have to be immediately shifted. Testing may become significantly harder.				
Mitigation Strategy				
Action	Date of Implementation	Expected outcome	Comments	
Stay in close contact with Hello robot and TouRI team to discuss hardware issues	-	Use their insight to tackle issues that may have already been solved before or to get a better understanding to be able to fix it		
Coordinate with sponsors and arrange a backup battery beforehand	-	Allows for a smooth transition to the backup battery without impacting the battery dependencies		
Well document all firmware updates and roll back to previous versions	-	Better understand where the fallacy originated and trace back to the previous correct state		
Improve gimbal design with dampers and digital/optical stabilization	-	No noticeable video disturbances		

Table 17: Slack in Project Schedule

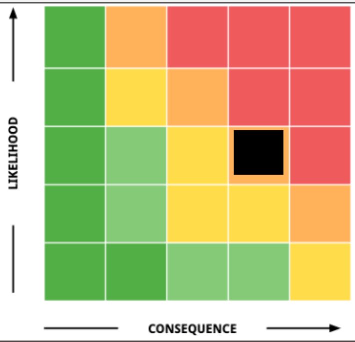
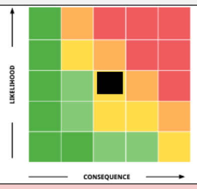
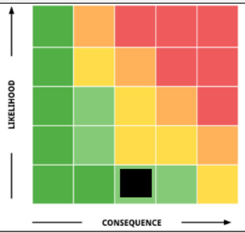
Risk Title	Slack in project schedule	Date Submitted	Dec 5 2022
Risk Owner	Project manager at the time	Date Updated	Dec 5 2022
Description	Risk Type		
Progress of project as per schedule is delayed because of either: 1) A teammate being unavailable due to personal reasons 2) A team member finding it difficult to complete their task because of lack of skill or complexity of the problem	Personnel		
	Metrics (L, C, Exposure)		
	3, 4, 12		
Consequence			
This sudden halt in progress impacts further tasks that are dependent on the task impacted by this risk. When the entire team's personnel resource is being focused on one task, it creates further slack in other tasks in the project, propagating a ripple effect causing more slack in those departments as well			
Risk Mitigation Strategy			
Action	Date of Implementation	Expected outcome	Comments
1) Reschedule activities to accomodate the slack and assign personnel or more working hours in the tasks that follow to still reach the goal in the same time	-	The major milestones of the project are still achieved	
2) Assign tasks to people who are done with their tasks earlier than scheduled	-	The project schedule is on track	Someone being done early makes no difference if they can not assist with the slack of others as the team is an aggregate of everyone's progress
3) Weekly scrums and daily stand ups to keep track of any technical assistance requirements so that a task doesn't get unreasoably stretched out	-	Instead of finding out a week later about someone being stuck at something, we can keep fixing things as they come daily	
4) Get help from people who have already done it before and experts (e.g. previous team)	-	A problem may have been encountered and fixed earlier too. It is better to seek expert opinion than to reinvent the wheel	

Table 18: Compute Insufficient

Risk Title	Compute insufficient for application	**Date Submitted**	Dec 5 2022
Risk Owner	Atharva Pusalkar	**Date Updated**	Dec 5 2022
Description	**Risk Type**		
Compute not able to handle the load of the software stack due to: 1) Heavy algorithms 2) Under estimation of of compute requirements	Technical		
	Metrics (L, C, Exposure)		
	3,3,9		
Consequence			
Robot not able to operate at optimum capability. Further software development is stalled.			

Risk Mitigation Strategy			
Action	**Date of Implementation**	**Expected outcome**	**Comments**
1) Add a better onboard computer	-	The computer is able to handle the current load of the software stack	Should pre-plan the procurement of hardware
2) Shift to cloud development	-	Offload some non-time critical requirements to cloud	
3) Reduce requirements	-	Remove non-core requirements and keep must-have ones	
4) Simplify algorithms	-	Computational requirement of algorithms is reduced while still maintaining acceptable level of performance	

Table 19: Lack of components

Risk Title	Lack of availability of Components	**Date Submitted**	Dec 1 2022
Risk Owner	Abhinav Gupta	**Date Updated**	Dec 1 2022
Description	**Risk Type**		
1) Cameras not in stock 2) Battery not available 3) Jetson not available	Technical		
	Metrics (L, C, Exposure)		
	1,3,3		
Consequence			
This leads to delay in the progress schedule as cameras are key to perception, battery to the power supply and testing conditions, and lack of jetson hampers compute.			

Risk Mitigation Strategy			
Action	**Date of Implementation**	**Expected outcome**	**Comments**
1) Switch to realsense (existing cameras available within MRSD lab and RI)	-	Cut down on any resulting slack due to unavailability of camera	
2) Keep the robot battery plugged in	-	Avoid the reliance on any exernal battery alternative. Avoid running the risk of draining the battery onboard.	
3) Switch to existing compute available within MRSD lab and RI or run everything on cloud.	-	Not let the lack of Jetson impact the efficiency of the software deliverable	

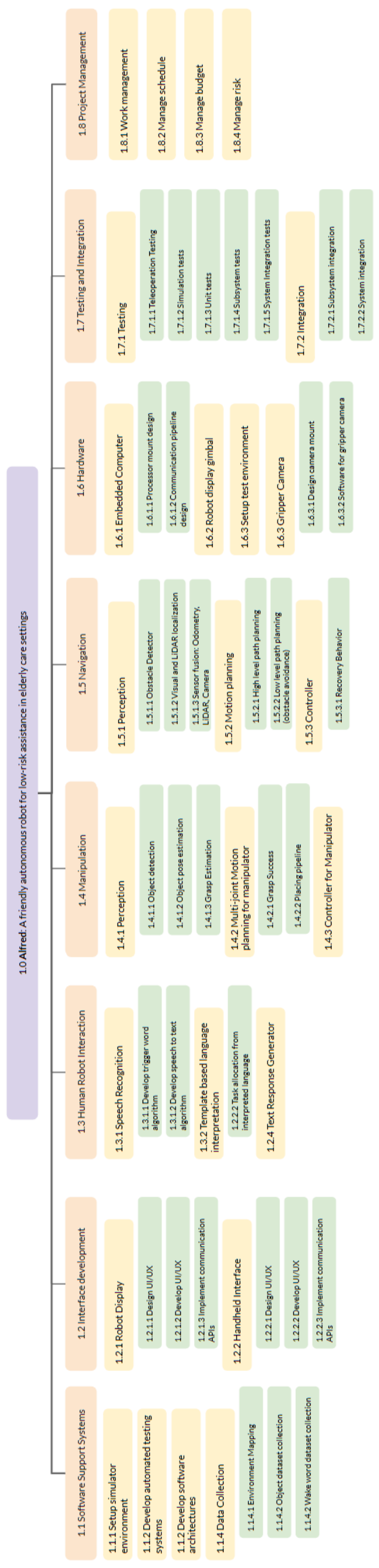
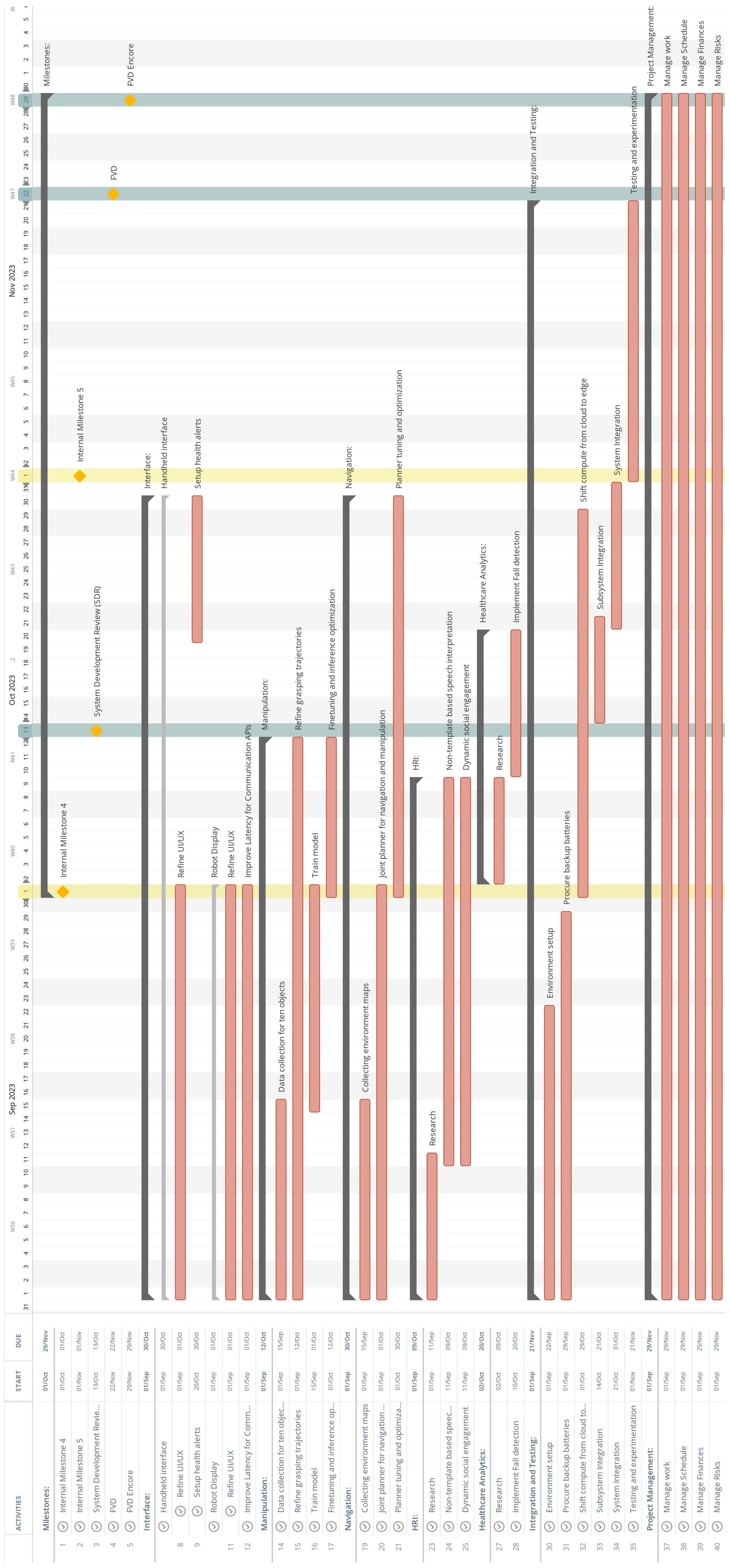

1.0 Alfred: A friendly autonomous robot for low-risk assistance in elderly care settings
1.1 Software Support Systems
1.1.1 Setup simulator environment
1.1.2 Develop automated testing systems
1.1.2 Develop software architectures
1.1.4 Data Collection
1.1.4.1 Environment Mapping
1.1.4.2 Object dataset collection
1.1.4.2 Wake word dataset collection
1.2 Interface development
1.2.1 Robot Display
1.2.1.1 Design UI/UX
1.2.1.2 Develop UI/UX
1.2.1.3 Implement communication APIs
1.2.2 Handheld Interface
1.2.2.1 Design UI/UX
1.2.2.2 Develop UI/UX
1.2.2.3 Implement communication APIs
1.3 Human Robot Interaction
1.3.1 Speech Recognition
1.3.1.1 Develop trigger word algorithm
1.3.1.2 Develop speech to text algorithm
1.3.2 Template based language interpretation
1.2.2.2 Task allocation from interpreted language
1.2.4 Text Response Generator
1.4 Manipulation
1.4.1 Perception
1.4.1.1 Object detection
1.4.1.2 Object pose estimation
1.4.1.3 Grasp Estimation
1.4.2 Multi-joint Motion planning for manipulator
1.4.2.1 Grasp Success
1.4.2.2 Placing pipeline
1.4.3 Controller for Manipulator
1.5 Navigation
1.5.1 Perception
1.5.1.1 Obstacle Detector
1.5.1.2 Visual and LiDAR localization
1.5.1.3 Sensor fusion: Odometry, LIDAR, Camera
1.5.2 Motion planning
1.5.2.1 High level path planning
1.5.2.2 Low level path planning (obstacle avoidance)
1.5.3 Controller
1.5.3.1 Recovery Behavior
1.6 Hardware
1.6.1 Embedded Computer
1.6.1.1 Processor mount design
1.6.1.2 Communication pipeline design
1.6.2 Robot display gimbal
1.6.3 Setup test environment
1.6.3 Gripper Camera
1.6.3.1 Design camera mount
1.6.3.2 Software for gripper camera
1.7 Testing and Integration
1.7.1 Testing
1.7.1.1 Teleoperation Testing
1.7.1.2 Simulation tests
1.7.1.3 Unit tests
1.7.1.4 Subsystem tests
1.7.1.5 System Integration tests
1.7.2 Integration
1.7.2.1 Subsystem integration
1.7.2.2 System integration
1.8 Project Management
1.8.1 Work management
1.8.2 Manage schedule
1.8.3 Manage budget
1.8.4 Manage risk

Figure 22: Work Breakdown Structure



		ACTIVITIES	START	DUE
		Milestones:	**01/Oct**	**29/Nov**
1	✓	Internal Milestone 4	01/Oct	01/Oct
2	✓	Internal Milestone 5	01/Nov	01/Nov
3	✓	System Development Revie...	13/Oct	13/Oct
4	✓	FVD	22/Nov	22/Nov
5	✓	FVD Encore	29/Nov	29/Nov
		Interface:	**01/Sep**	**30/Oct**
	✓	Handheld interface	01/Sep	30/Oct
8	✓	Refine UI/UX	01/Sep	01/Oct
9	✓	Setup health alerts	20/Oct	30/Oct
	✓	Robot Display	01/Sep	01/Oct
11	✓	Refine UI/UX	01/Sep	01/Oct
12	✓	Improve Latency for Comm...	01/Sep	01/Oct
		Manipulation:	**01/Sep**	**12/Oct**
14	✓	Data collection for ten objec...	01/Sep	15/Sep
15	✓	Refine grasping trajectories	01/Sep	12/Oct
16	✓	Train model	15/Sep	01/Oct
17	✓	Finetuning and inference op...	01/Oct	12/Oct
		Navigation:	**01/Sep**	**30/Oct**
19	✓	Collecting environment maps	01/Sep	15/Sep
20	✓	Joint planner for navigation ...	01/Sep	01/Oct
21	✓	Planner tuning and optimiza...	01/Oct	30/Oct
		HRI:	**01/Sep**	**09/Oct**
23	✓	Research	01/Sep	11/Sep
24	✓	Non-template based speec...	11/Sep	09/Oct
25	✓	Dynamic social engagement	11/Sep	09/Oct
		Healthcare Analytics:	**02/Oct**	**20/Oct**
27	✓	Research	02/Oct	09/Oct
28	✓	Implement Fall detection	10/Oct	20/Oct
		Integration and Testing:	**01/Sep**	**21/Nov**
30	✓	Environment setup	01/Sep	22/Sep
31	✓	Procure backup batteries	01/Sep	29/Sep
32	✓	Shift compute from cloud to...	01/Oct	29/Oct
33	✓	Subsystem Integration	14/Oct	21/Oct
34	✓	System Integration	21/Oct	31/Oct
35	✓	Testing and experimentation	01/Nov	21/Nov
		Project Management:	**01/Sep**	**29/Nov**
37	✓	Manage work	01/Sep	29/Nov
38	✓	Manage Schedule	01/Sep	29/Nov
39	✓	Manage Finances	01/Sep	29/Nov
40	✓	Manage Risks	01/Sep	29/Nov

Sep 2023
Oct 2023
Nov 2023
System Development Review (SDR)
Improve Latency for Communication APIs
Non-template based speech interpretation
Shift compute from cloud to edge
Joint planner for navigation and manipulation
Finetuning and inference optimization

Figure 23: Fall 2023 project schedule