Cyber Physical System for Continuous Evaluation of Fall Risks to Enable Aging-In-Place

by

VINAYAK JAGTAP

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Robotics Engineering
by

MAY 2015

APPROVED:

Professor Taşkın Padır, Thesis Advisor

Professor Sonia Chernova

Professor Michael Gennert

Abstract

Every year, one out of three adults over the age of 65 falls, and about 30% of the falls result in moderate to severe injuries. The high rate of fall-related hospitalizations and the fact that falls are a major source of morbidity and mortality in older adults have motivated extensive interdisciplinary clinical and engineering research with a focus on fall prevention. This research is aimed at developing a medical Cyber Physical System (CPS) composed of a human supervised mobile robot and ambient intelligence sensors to provide continuous evaluation of environmental risks in the home. As a preventive measure to avoid falls, we propose use of mobile robots to detect possible fall risks inside a house. As a step-up to that, we also define a control framework for intelligent, networked mobile robots to semi-autonomously perform assistive and preventive tasks. This framework is integrated in a smart home that provides monitoring and control capabilities of environmental conditions such as objects blocking pathways or uneven surfaces. The main outcome of this work is the realization of this system at Worcester Polytechnic Institute's (WPI) @Home testbed.

Acknowledgements

I would like to express my sincere gratitude towards Professor Taşkın Padır for giving me the opportunity to work on the projects mentioned in this thesis and his invaluable support and direction throughout the duration of this work. I would also like to the thank Professor Sonia Chernova and Professor Michael Gennert for being a part of my advising committee and providing insights into different aspects of the projects presented. Robotics and Intelligent Vehicles Research(RIVeR) laboratory has been a great place to collaborate and learn. Thanks to Ruixiang Du and Velin Dimitrov for their patience during my learning phase and guidance throughout the projects.

Thanks to Intel for sponsoring the Smart Home testbed and providing us with an opportunity to explore the vast possibilities of Internet of Things and its application in Cyber Physical Systems.

Lastly, I would like to thank my parents and my loving wife for the motivation and encouragement that was required for taking the tough decisions and choosing the right path forward.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As per current population projections, it is estimated that 20% of US population would be 65 years or over by year 2030 [18]. With this change in population, there would be an increased demand of nurses to take care of older adults. However, even today, there is limited availability of trained nurses to cope up with the existing demand to assist older adults. To cater for the needs and provide good quality assistance to the older adults, we need a framework that can provide a way to monitor well being of multiple people by limited staff.

1.1 Background

The age structure of the U.S. population is expected to change over coming decades [18]. As per the population estimates and projections, U.S. will experience considerable growth in its older population in the next 3 decades. In 2015, population comprises of a little less than 50 million people who are over 65 years of age which is about 15% of the population. These numbers are projected to increase to almost 70 million which would be 20% of the population by the end of year 2030. Another metric that is useful to understand the impact of this increase in number of older adults is Dependency Ratio. Dependency Ratio of old-age dependency is given by

$$\text{Old age dependency} = \frac{\text{Population of 65 and over}}{\text{Population 18 to 64 years}} \times 100 \tag{1.1}$$

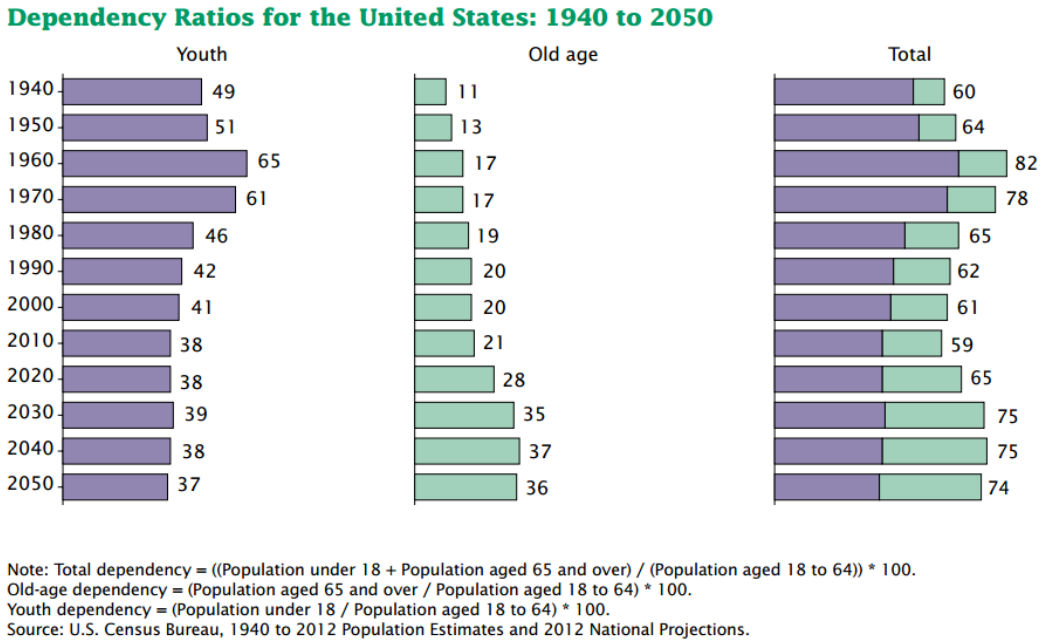


Figure 1.1: Dependency Ratio Estimates

As shown in the Figure 1.1, the old-age dependency ratio is projected to increase from 21 in 2010 to over 30 by 2030. This large increase in old-age dependency ratio would also result in increase of total dependency ratio projected to be almost 75 by 2030. This means that there would be less than one independent adult to every old age person by 2030. To improve the healthcare for older adults, it is important to increase the efficiency of existing systems such that available trained professionals would be able to assist multiple people simultaneously.

1.2 Closed-Loop Healthcare

Existing healthcare system comprises of a loose collection of disparate devices and systems that do not talk, or require manual intervention to transfer or receive data. This leads to inefficiencies and potentially dangerous conditions due to the lack of interoperability and coordination. Closed-Loop Healthcare (CLHC)[1] initiative introduces an in-home health

[1] http://smartamerica.org/teams/closed-loop-healthcare/

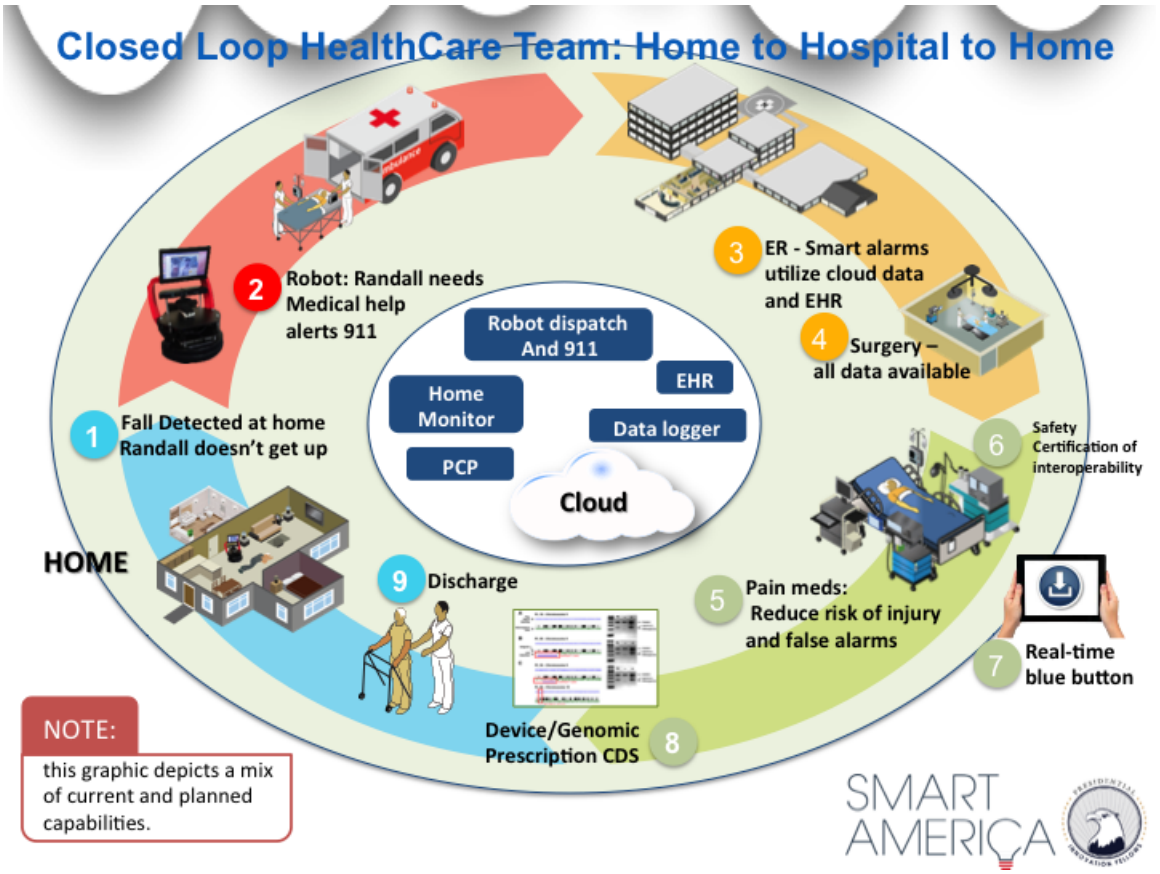


Figure 1.2: Closed Loop HealthCare [1]

monitoring and alert system to interconnected, interoperable components in the hospital to create a cohesive stream of patient-specific health information. The lifecycle of CLHC is shown in Figure 1.2 with the help of an elderly patient, Randall, who lives alone. When Randall falls and breaks his shoulder, a smart home system detects his fall and sends a request for help immediately. The fall is detected using a kinect sensor installed at home. The history of in-home sensor data helps facilitate personalized and better informed treatment in the hospital. Because the hospital devices and systems are interconnected, Randall receives more coordinated care with lower risk of complications. As Randall transitions between care units, sensor data are automatically delivered to his bedside by the integrated healthcare platform. When Randall returns home, the smart home system tracks his activity, behaviors, and vital signs with more sensitivity to avoid re-hospitalization. Randall

is able to stay in his own home, and his adult children are confident he will receive help if needed.

1.3 Fall Risk Assessment

According to the World Health Organization's (WHO) global report [1] on fall prevention in older age, there are four main risk factors for falls as shown in Figure 1.3. These risk factors are classified as biological, behavioural , environmental, and socioeconomic. In this project we are focusing on identifying and helping to reduce down the environmental risk factors. Motivation of the project lies in the facts that (1) aging in place improves the overall health and well-being of individuals, (2) falls are the leading cause of mortality in older adults, (3) home environmental fall risk assessment is an effective preventive strategy, and (4) extreme costs and shortage of trained personnel are huge barriers for effective and efficient delivery of fall risk home assessments by health care providers.

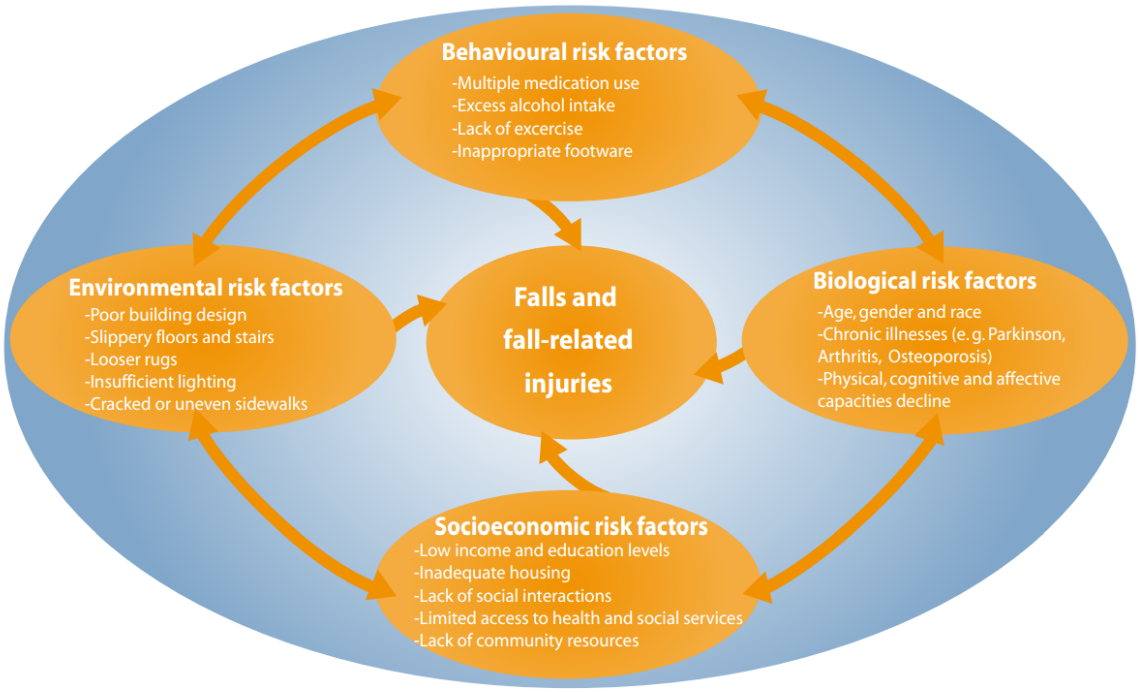


Figure 1.3: Risk factor model for falls in old age [1]

Considering the above facts, we present use of a semi-autonomous mobile robot to detect

potential fall risks through continuous assessment of home environment [8]. This mobile robot can be teleoperated by an operator from a remote location to help maintain risk free environment. The framework allows a single operator to control multiple robots located at different homes. In case of emergencies, the robot can be used to assess the seriousness of emergency and also to communicate with the injured person till help arrives. Nationally, $30 billion is wasted in needless MRI/CT imaging [3]. Placing the ambient sensors and mobile robot would provide a better picture of the incident obviating the need for unnecessary procedures. For example, if there was no direct impact on head and the fall was less than 3 feet high then CT scan might not be required.

1.4 Smart Home Environment

Home is a complex environment for robot operations due to the fact that objects and environmental parameters inside a house are not static. We propose use of smart homes [25] to provide necessary data/alerts to the robots for performing different tasks like initiating telepresence [7] session with health care provider, start or stop assessment of fall risks, navigating to a particular location to fetch more details, etc. A "smart home" can be defined as a residence equipped with computing and information technology which anticipates and responds to the need of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond [13]. The home is fitted with several sensors and intelligent devices to collect necessary environmental parameters. Data from these sensors is stored in a central database either on the cloud or at a local server. Data collection, storage and retrieval methods are defined in the central server that allows devices to store or fetch data. This data can be accessed by robots to get a holistic view of the operating environment.

1.5 Structure of the Thesis

The thesis is organized as follows. Chapter 2 provides the design requirements for the mobile robot and the smart home environment. Chapter 3 explains the hardware and

software components of the mobile robot used for fall risk assessment. This includes details of the robotic system, mapping and navigation capabilities of the robot, telepresence and teleoperation, GUI and screening of the environment for fall risks. Chapter 4 provides details of network architecture, devices and data management strategies for Smart Home and the thesis is concluded in the last chapter.

Chapter 2

Design Requirements

This system focuses on making it possible for the elderly to remain at home, safe and comfortable. There are two kinds of users for the system, robot operators at the healthcare center and the occupants of the house where the system is installed. Both types of users can be safely assumed to have little technical background and hence appropriate user-oriented design considerations are made as described below.

2.1 Mobile Robot

The mobile robot will be operated indoors for assessing fall risks. To assess the environment, the robot should be able to possess a list of features as described below.

2.1.1 Robot Features

1. Semi-autonomous Navigation

   The robot should be capable of navigating when provided a goal point. It should be able to modify its generated trajectories if dynamic obstacles are encountered while navigating.

2. Teleoperation and Telepresence

   Teleoperation is the ability to remotely control a robotic system that comprises of sensors and actuators [21]. Telepresence is the subclass of teleoperation that provides

the ability to interact with an environment that is physically away along with manifestation of one's presence in the remote location [7]. The telepresence required in our case should provide transmission of audio-video signals between both the users. For remote assessment of a house the robot should have teleoperation and telepresence. In case of emergency, the operator should be able to talk to the injured person to get preliminary information so as to facilitate better response.

3. Internet Connectivity

   The robot would be connected to the Internet using the available network at home, hence it should not block the bandwidth and at the same time should be able to send assessment information with available connection speed. Delays in sending or receiving packets from/to the robot should be handled by the system such that there are no negative impacts on the user experience.

2.1.2 Graphical User Interface

1. Software Compatibility

   The GUI should be accessible from most of the commonly used Operating Systems, specifically Windows and Mac OS. The required installation components on the user end should be minimal.

2. Easy to Learn

   The GUI should be simple and intuitive. Assuming that users of this interface use computers on a daily basis, we pose a hypothesis that the interface would be intuitive if they could learn 80% of the UI functionality in less than 15 minutes.

3. Delays in operation

   All the commands sent from the UI should be executed near real-time. If there are unpredictable delays, those delays should be handled elegantly without any side-effects on the surroundings.

2.2 Smart Home Requirements

Smart Home would involve addition of networked sensors or devices to existing home. As such, it should be able to record environmental parameters without impacting the appearance of the home.

2.2.1 Sensors

1. Record environmental parameters

   Most of the environmental parameters like temperature, humidity, motion, door opening/closing, etc should be recorded along with timestamp of the occurance.

2. Aesthetics

   The devices should be small, such that they can be hidden or placed without negatively impacting the aesthetics of the house.

3. Security

   The system architecture should provide security against unauthorized access of the data from the Internet. Security is one of the most important factors as there would be actuators and robots in the environment that can cause physical injury to occupants if exposed to wrong hands.

2.2.2 Data Collection, Storage, and Retrieval in Smart Home

1. Data Logging

   All the data generated by different ambient sensors should be logged in a central location. Ability to analyze sensor readings with respect to time or other sensor readings should be possible.

2. Interfacing with embedded devices/sensors

   Commonly used protocols should be used for lower deployment times of new devices. Existing off-the-shelf Internet of Things (IoT) devices should be integrable. It should support inserting or retrieving data from commonly used technologies like Arduino, Python, and NodeJS.

3. Robust data server

   The data server should be able to handle multiple requests asynchronously. This is required as multiple devices would be connected to the network and each device would be pushing some data independent of the other devices in the network.

2.2.3 Scalability Requirements

1. Scale up

   Adding new devices should not result in rework on existing devices unless new functionality is added to an existing device. Addition of new devices up to a maximum of 50, should be supported without the need to scale out the system architecture.

2. Robustness and Redundancy

   The system should be able to restore to normal state after resolution of failures caused by external factors like power failure, loss of internet connectivity, etc. Multiple sensors providing similar information can be used to provide a layer of redundancy for protection from failures within the system.

Chapter 3

In-Home Evaluation of Environmental Fall Risks

Fall risk assessment system comprises of a mobile robot, a service management layer, and the user interfaces. Turtlebot-2 [17] is the mobile robot that is teleoperated from a health care center to evaluate the safety of environment using the checklist provided by Health, Education, Research and Outreach for Seniors [1] (HEROS). Two interfaces are developed to operate the robot, one that runs in web browser and can be accessed by any authorized user and the other that is a standalone application designed for robot experts to fetch advanced details of the robot and the environment.

Next section provides an overview of entire system and different components of the system. After that we describe the implementation and design choices for every requirement of this project. The HEROS checklist can be divided into five major areas - lighting conditions, uneven floors, furniture placement, reading vital signs, and emergency response. Requirements for these evaluation aspects are shown in Table 3.1.

3.1 System Overview

The system implements a client-server architecture in which server is connected to the robots at patients' home and thin-clients (web based user interfaces) are located at the

[1] http://www.temple.edu/older_adult/

Risk Evaluation	Mobility	Sensing	Autonomous Navigation	Teleoperation	3D Interaction	Telepresence
Lighting Condition	✓	✓	✓			
Uneven Floors	✓	✓		✓		
Furniture Placement	✓	✓		✓	✓	
Reading Vital Signs		✓				✓
Emergency Response			✓			✓

Table 3.1: Requirement Matrix Based on HEROS Checklist

healthcare facility as shown in Figure 3.1. The server communicates with the robot to pull sensor data and provides it to the operators. A Service Management layer hosts Web service and Access Control to provide web based UI to the operators. Access Control takes care of authorizing users before providing access to the robots. Multiple robots can be added to

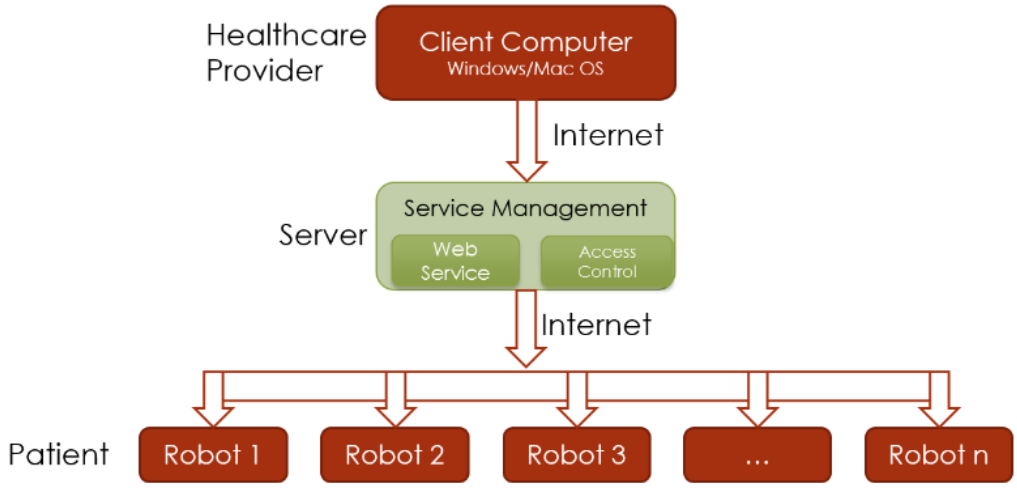


Figure 3.1: System Overview

the system and the operators can chose which robot to operate.

The server has an instance of Robot Management System [24] (RMS) installed that maintains the communication between robots and the web users. This forms the Service Management layer and hosts Web Service and the Access Control modules as displayed in Figure 3.1. RMS is a web-management system that utilizes model-view-controller framework using CakePHP [5]. RMS uses Javascript Object Notation (JSON) [15], a lightweight data-interchange format to communicate with robots running on Robot Operating System (ROS) [20]. ROS provides implementation of numerous algorithms and services required for robots to run. These services are accessed by RMS webserver and the requested data is sent to the web clients via HTTPS. The architecture of RMS server is shown in Figure 3.2.

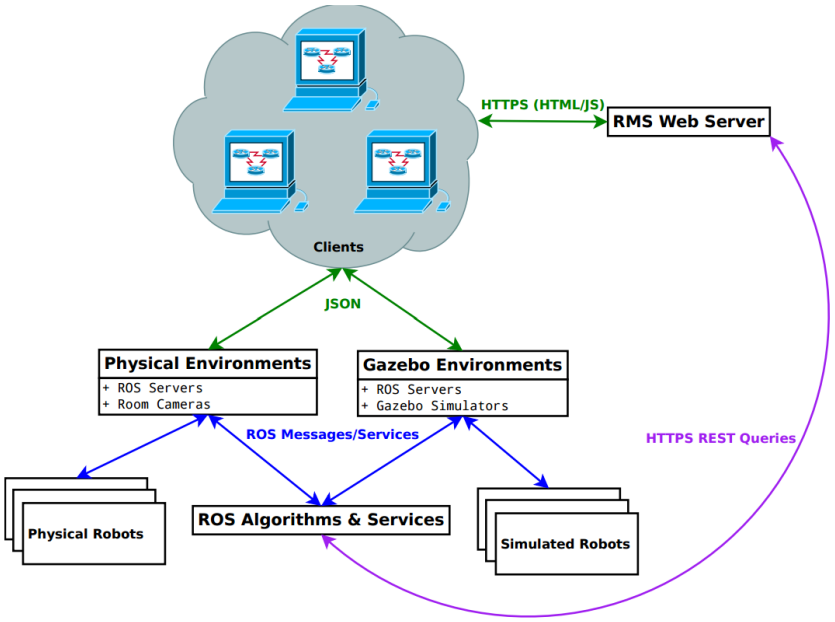


Figure 3.2: Robot Management System[24]

3.2 Mobility

We customized Turtlebot 2 for the mobility requirements of this project. Turtlebot 2 is a highly capable autonomous mobile platform for developing robot application. It

is commercially available and can be customized to one's requirements. The hardware structure can be divided in to three main parts: A Kobuki base, a netbook computer, and sensors. The Kobuki base is a differential drive mobile robot base with two passive caster wheels for balancing. The default netbook computer is replaced by a mini PC - Intel Next Unit of Computing (NUC) and a 30,000 mAh lithium-ion polymer battery. The NUC runs ubuntu OS and has ROS running on it which takes care of the communication between different sensors and actuators. The customized version of turtlebot is displayed in Figure 3.3.

Figure 3.3: Turtlebot 2 - Mobile Robot

Turtlebot 2 can be used to navigate around in the house to evaluate the lighting conditions, uneven floors, and furniture placement. The mobile base has the ability to navigate in cluttered spaces due to its small dimensions. Differential drive and circular base design provides a zero turn-radius making it easy to recover from obstructed positions.

3.3 Sensing

Specifications of all the sensors present in the mobile robot are provided in the Table 3.2. Kinect is a 3D vision sensor that provides RGBD data, i.e. depth information of every point in the image along with the color image of the scene. The color images acquired from Kinect are displayed in the web interface as a video feed and the depth data is used to display distance of the center point in the image. This enables the operator to understand the distances of objects in scene. The video feed is useful for tracking down irregularities on floor or identifying clutter caused due to wires, magazines, or other objects.

Sensor	Description
Microsoft Kinect	3D vision sensor
Odometry Sensors	52 ticks/enc rev, 2578.33 ticks/wheel rev, 11.7 ticks/mm
Gyro Sensors	1 axis 110 deg/s
Auxiliary sensors	3X forward bump sensors 3X cliff sensor 2X wheel drop sensor
Luminosity Sensor (custom add-on to Turtlebot)	0.1 - 40000+ Lux
Pulse Oximeter (custom add-on to Turtlebot)	Measures oxygen level & pulse rate

Table 3.2: Sensors in Turtlebot 2

We estimate the lighting condition of an environment by using the lux values measured by luminosity sensor. The relationship between the lux values and different lighting condition of the environment is shown in Table 3.3. We use these value as threshold for lighting condition detection and show the result on the user interface. When the lux value represents that the environment condition is in the dark range, a red light sign is displayed on the

corresponding item in checklist on the interface. Similarly, yellow light is used for medium condition and green light for bright condition. Operator thus can have a sense of the current lighting condition of the environment from these signs.

Luminosity [lux]	Lighting Condition
≥ 401	Bright
201 - 400	Medium
<200	Dark

Table 3.3: Lighting Condition for Different Levels of Luminosity

As an example of integrating small medical devices and sending signals to the operator, we have added a pulse oximeter to the robot. Pulse oximeter is a non-invasive medical device that calculates the oxygen saturation in a person's blood and changes in blood volume in the skin. When a person inserts his/her finger in the slot provided, the reading on the oximeter can be sent to the operator. Figure 3.4 shows an oximeter displaying oxygen saturation level and pulse rate of a person.

Figure 3.4: Pulse Oximeter - Displaying oxygen saturation level and pulse rate

Upper level of Turtlebot is replaced with a 3D printed mount and a 10" LCD Touch Screen. The touch screen can be used to initiate a telepresence session or to see the operator if the operator starts one. The screen also acts as the input device for onboard computer on the Turtlebot.

3.4 Autonomous Navigation

ROS provides packages for mapping and navigation of turtlebot. As a part of these packages, a wrapper for gmapping is available which is a highly efficient Rao-Blackwellized [12] particle filter to learn grid maps from laser range data and solve the simultaneous localization and mapping (SLAM) problem. 2D map of an environment is built using gmapping app on the robot. During map generation process, the robot is teleoperated around in the home environment, the system converts the point cloud from kinect and odometry information from the encoders to build a 2D map and stores this map locally on the robot.

After the map of the environment is generated, we can navigate the robot in the environment through web client or the standalone application using that map. Adaptive Monte Carlo Localization (amcl) [10] is used to determine the position of the robot on the map which in-turn is used for autonomous navigation of the robot. On the user interface, we can see current position of the robot and provide a navigation goal to it by clicking the desired position on the map. The robot will navigate to that position autonomously, avoiding any static or dynamic obstacles on its way.

The way ROS handles navigation is shown in Figure 3.5. ROS provides implementation of move_base, which is shown in the central rectangle, along with amcl and map_server. These implementations are generic and can be used for any robot. Sensor transforms data is combined with odometry source to provide tf and odom messages. tf message includes relationship between different coordinate frames and odom message provides the data related to frame positions and velocities. Sensor sources provide details of obstacles in the field of view in form of LaserScan or PointCloud. This information is used to assign costs for every point on the map locally and globally, thus building a local and global costmap.

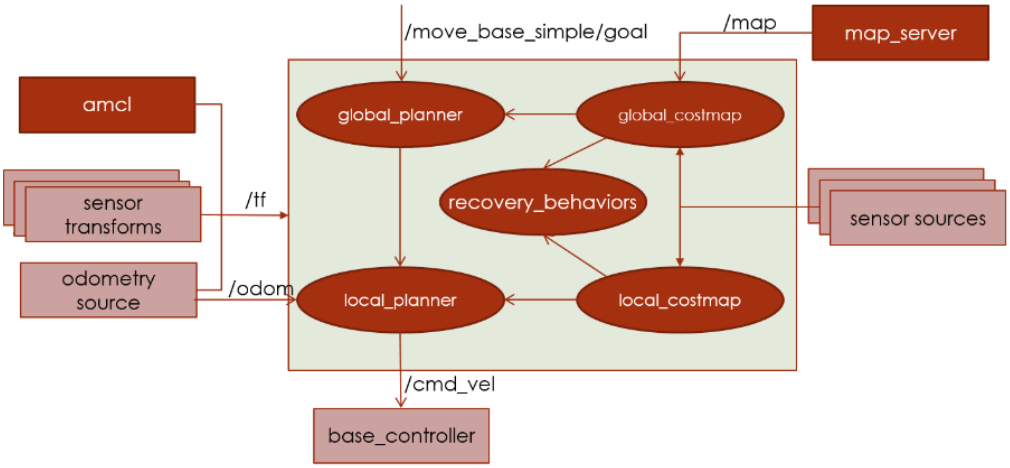


Figure 3.5: ROS Navigation stack

Navigation stack takes a goal as input in the form of geometry_msgs/PoseStamped message. This messages provides the coordinates of the final goal along with orientation. Global and local planners plan the trajectory to reach the goal and it is sent to the robot through /cmd_vel topic.

Autonomous navigation is useful for moving the robot to a particular location in the house to determine lighting conditions or to reach out to the occupant when a fall is detected. In case of a fall, the robot can navigate to the person and initiate a telepresence session with a doctor for immediate attention.

3.5 Teleoperation

Teleoperation is the ability to control a robot remotely. In traditional teleoperation, a robot normally follows the commands of the operator even if obstacles detected by the laserscan. It is the operator's responsibility to consider or reject the sensor readings and control the robot. This approach is of little use in cluttered dynamic environments with low connection speeds between operator and the robot. If given a command to move forward, the robot might collide with an object before the operator can see any possibility of collision. To overcome this problem we propose assistive teleoperation, a strategy to avoid obstacles

autonomously while being teleoperated by an operator. Related works can be found in [9] and [11]. Existing robotic systems such as PatrolBot, SpeciMinder and MapperBot also provide similar features to improve the robot teleoperation experience.

Local costmap of the area in which robot is facing is built using the data from Kinect and bump sensor. This data is stored in the form of point cloud and can be used to determine obstacles and distances between the obstacles. The costmap divides an area into a grid and assigns cost values to every square of the grid based on its distance from the observed obstacles. When the robot receives a teleoperation command, it calculates a trajectory to move to the desired location. If there is an obstacle present on the trajectory, the planner would calculate a new trajectory candidates based on the costmap information. The trajectory with the lowest cost is selected by the system and the robot follows the selected trajectory so that the obstacles are avoided autonomously.

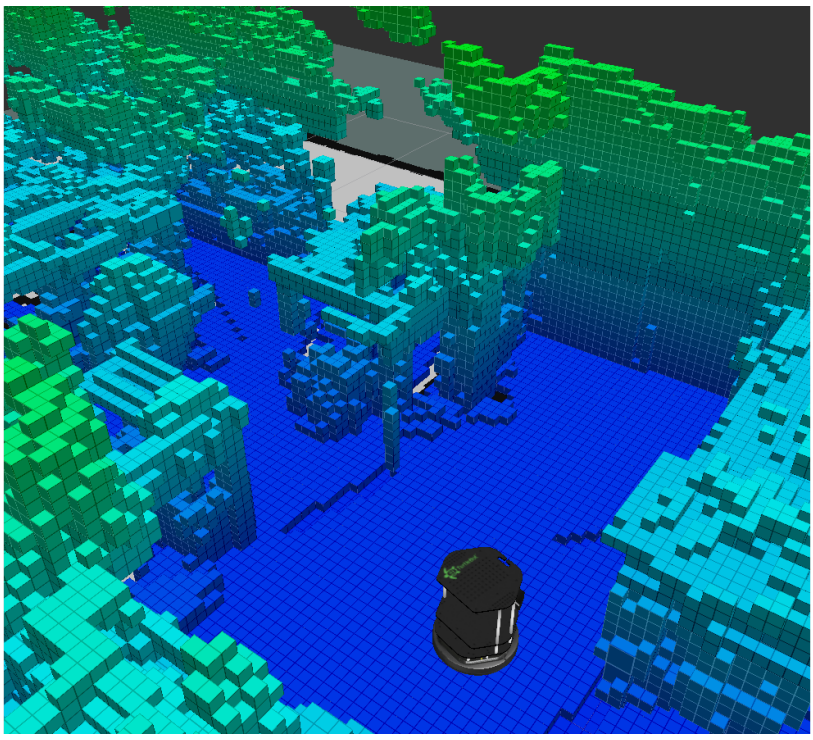


Figure 3.6: Building a 3D map with Octomap

3.6 3D Interaction

A 3D map is generated using octomap server [14]. Octomap server creates an octree that represents occupied three dimensional space into small cubes as shown in Figure 3.6. This map provides the ability to rotate or tilt the 3D space and analyze the conditions better. Once a 3D map is created, any part of the map can be accessed and analyzed by the operator at a later time depending on his availability. 3D map provides an ability to measure height of furniture or distances between objects for evaluating related fall risks.

3.7 Telepresence

Telepresence allows the person to interact with a doctor or operator from his/her home. In case of a fall, doctors can do a preliminary test by asking the patient to perform specific movements to analyze severity of the fall. Based on the results of these preliminary tests, unnecessary part of the diagnosis can be eliminated. For example, if the fall is from less than 3 feet height with no impact to head, a CT scan can be ruled out. Information about the fall can be collected from ambient sensors in the house. On the contrary, if there is a fall but the person does not report it, the doctors can setup a telepresence session and check the well being of the person. The telepresence feature is implemented using camera and microphone array present in kinect sensor. Audio and video is transmitted in the form of ROS messages and displayed on the screen of the user. This feature is embedded in the interface on both patient's and provider's side.

3.8 Graphical User Interface

User interface is one of the most important aspects of this project as it is the only method available for the operator to control the robot. Most existing robotic systems come with a complicated interface for the operator to accomplish various complex tasks. Thus operators for this kind of systems usually need to be trained or are professionals in engineering technologies. However, for this in-home screening robotic system, users on both ends, which are providers and patients respectively, generally don't have experience with

Figure 3.7: Telepresence Demo

robots and it's impractical to train all of them before they can really use the system. Hence the user interface should require minimum training. Technical details should be hidden and only necessary information should be shown to the users.

For the providers, a web interface is created. This interface is developed based on the Robot Web Tools [2].There are two display areas as shown in Figure 3.8. One is for the live video streaming from the robot where user can get a robot view as the visual feedback for tele-operation. It also can be used to monitor the environment. The video stream from the robot has a distance indicator. At the center of the image, a red circle is drawn and a number is present besides the circle, which shows the distance between the object at the center of the red circle and the robot. This distance indicator can give the operator a sense of relative position of the robot in the environment and improve the user experience of tele-operation. The other view is to display map of the environment. On this map, the position and orientation of the robot is shown as a triangle. The operator can send a command to the robot to move to a specific location by double clicking the the desired position on map. A checklist is provided at the top of the web page. It can help the operator to check if all items have been inspected. Patient's interaction with the robot is very limited. It includes

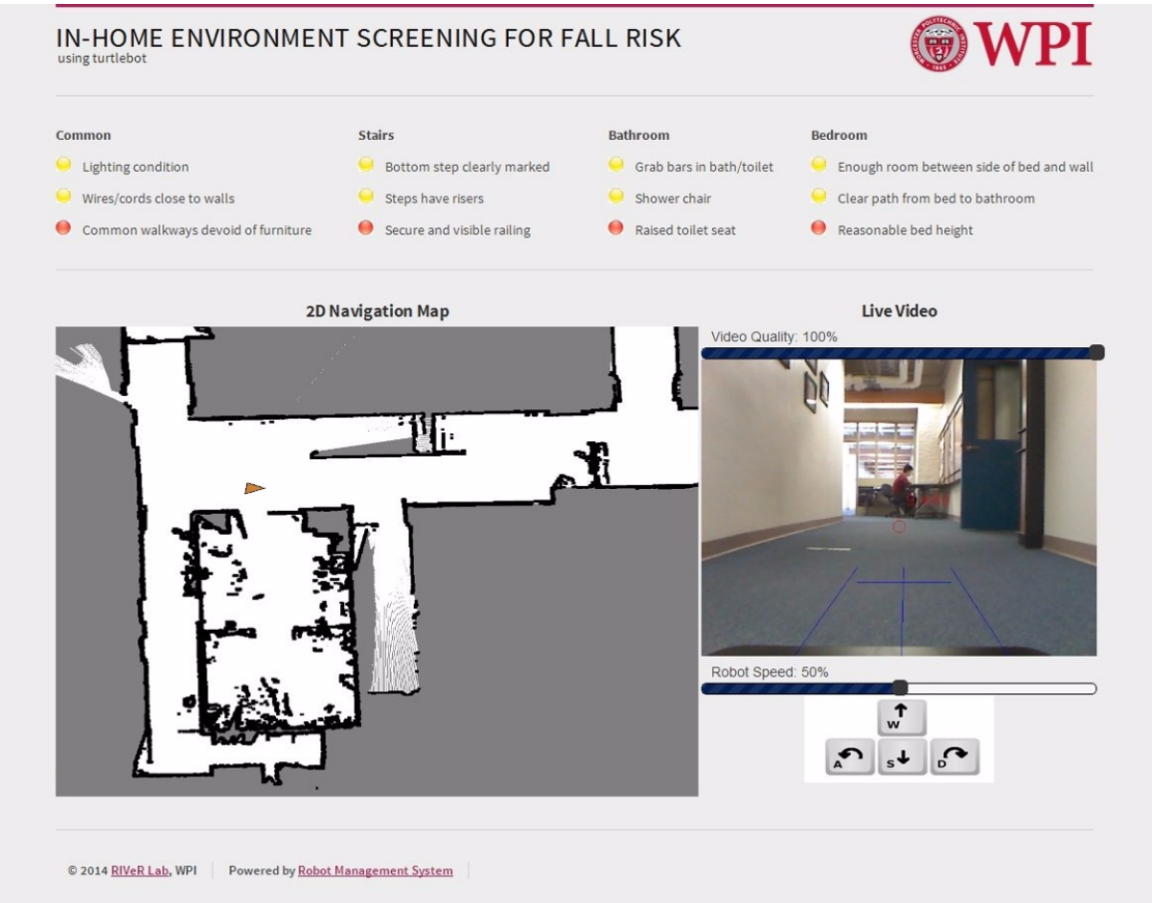


Figure 3.8: Web Interface[8]

basic control such as turn the robot on or off, command the robot to generate map and so on. All operations are done using the touch screen.

Another QT-based graphical user interface is developed as an addition to the web interface. It is shown in Figure 3.9. This GUI provides all the information that is present on the web page along with more details of robot status comprising of battery power and sensor readings from bumper, cliff, and wheel drop sensors. 3D information such as point cloud and 3D map can be seen in this GUI which allows users to do more advanced operations such as measuring the distance between two specific points or the height of an interested object. Accordingly this QT-based GUI has higher requirements for operators because more knowledge of the robot's perception and control is required to interpret the data shown in

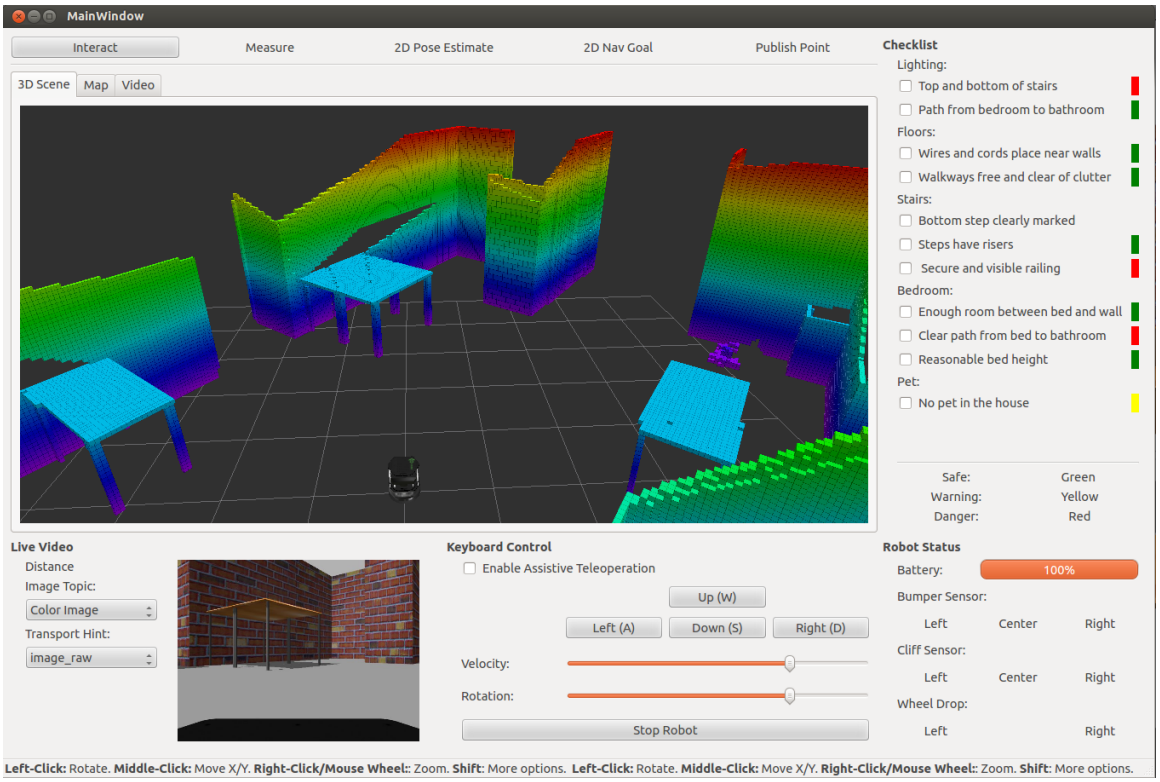


Figure 3.9: Alternative GUI for more technically adept users[8]

the GUI. Moreover, since the QT library needs to be installed and configured properly to run the QT-based GUI, this interface has few system prerequisites unlike the web interface which only requires a web browser that that is available on most computers. However, more low-level information of the system means this GUI is a more powerful tool for the developer or administrator of the system. This kind of information is very useful for debugging and troubleshooting when a problem occurs in the system.

Chapter 4

Smart Home Environment

Operating robots in home environment is a complex task due to the fact that most of the objects in this environment are dynamic and can serve different purposes based on time or context. To simplify the robot's understanding of the environment and also to reduce onboard processing for the robot, several sensors are installed in the house which can communicate with a central data server and store the sensor readings for other devices to access. Actuators or robots can fetch data from the server and trigger an action based on the readings. Actions can be either scheduled or event-based. Ambient intelligence could help in localizing the robot so as to improve accuracy of indoor navigation.

We describe the network architecture in next section followed by embedded controller used for building intelligent devices in @Home. Then we provide details on the data collection, storage, and retrieval strategies. Few projects implemented in @Home are described next with details of how they are integrated to work together.

4.1 Network Architecture

Network architecture diagram for WPI's @Home testbed is shown in Figure 4.1. It has a central server that separates all the devices from the Internet. All requests for data from the Internet are handled by the server and responses are sent to appropriate devices. This isolation along with a firewall on the server end ensures data security. All the devices are connected to a local network. A Ubiquity Access Point with 3x3 MIMO capability providing

450 Mbps at 2.4Ghz and 1300 Mbps at 5.8Ghz is installed for WiFi connectivity. High speed connectivity within the network helps in streaming HD images from camera. Ubiquity mFi sensors are added to provide data related to motion, temperature and door status. These sensors are connected to mPort which is responsible for pushing data from all these sensors to the central database. WiFi activated power strips that can be switched on/off based on some events or schedule are installed. These power strips also provide data related to power consumption at each port. IP cameras are installed that can provide videos to other devices/robots on demand.

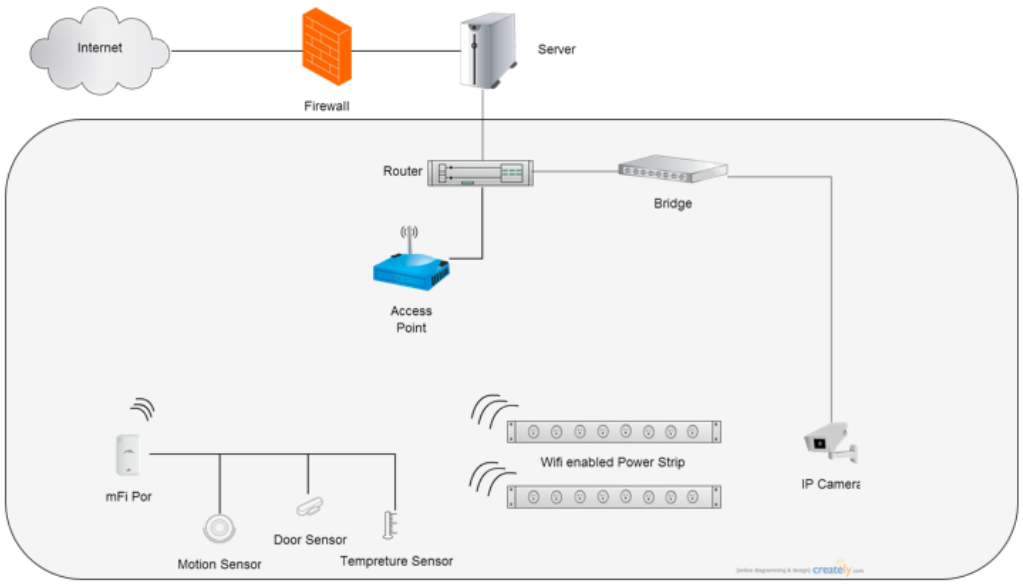


Figure 4.1: Network Architecture

4.2 Embedded Controller

Intel Galileo first generation boards were used to develop projects for this testbed as a part of a graduate course, Model-Based Design, in Fall 2014. Intel Galileo development board is a microcontroller board based on the Intel® Quark™ SoC X1000 application processor, a 32-bit Intel® Pentium® brand system on a chip (SoC). It is the first board

based on Intel architecture to be hardware and software compatible with Arduino Shields designed for Uno R3. The pins of Intel Galileo board are as per Arduino 1.0 pinout. Arduino IDE for Intel Galileo is available which enables uploading Arduino sketches to the board. Block diagram of the board is shown in Figure 4.2. Specifications of the board are provided in appendix at the end of the document.

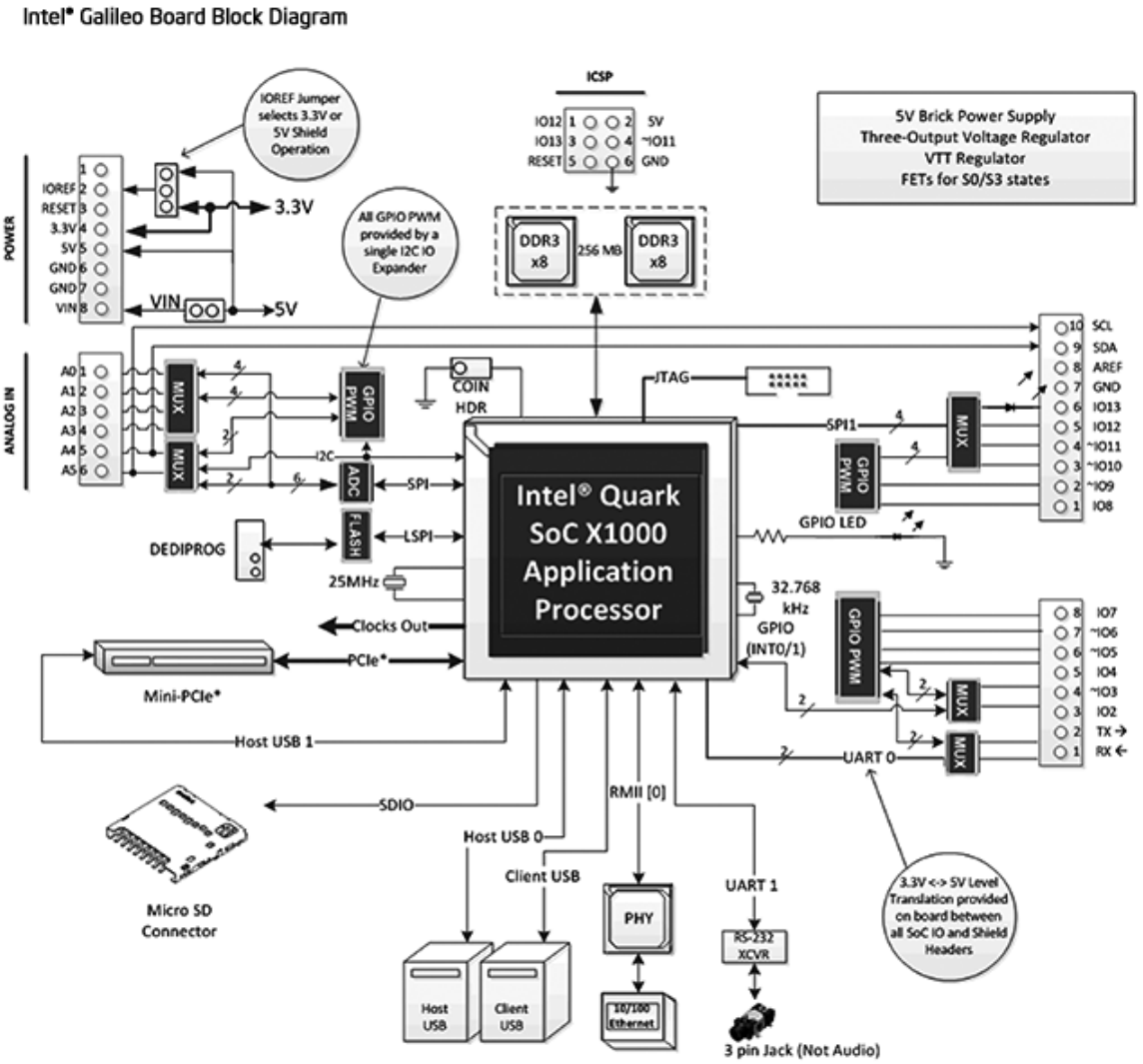


Figure 4.2: Block diagram of Intel Galileo

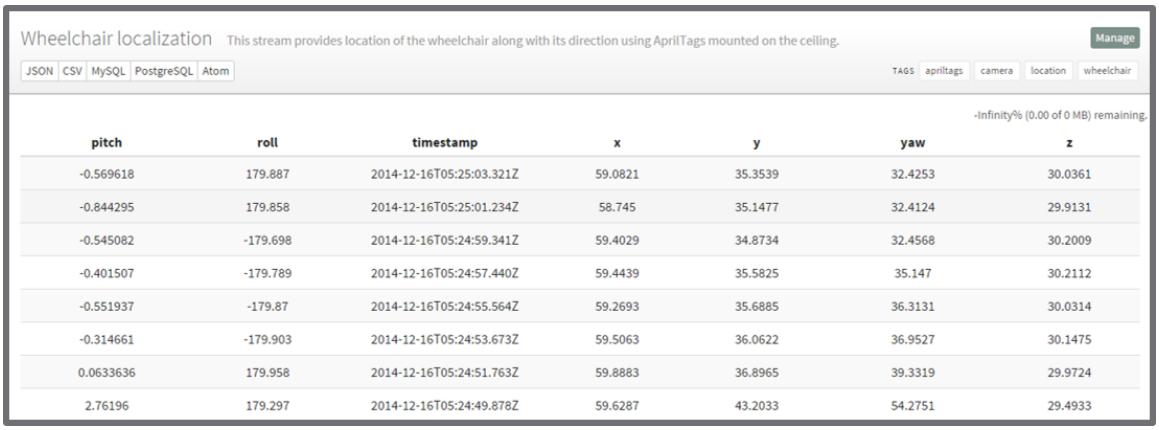
Full linux image can be installed on Galileo board which enables users to write programs

in many different languages such as python, javascript, Java, C/C++, etc. OpenCV library needs to be compiled by disabling SSE to make it work on Galileo. As a part of this project we ported OpenCV, PCL, and ROS hydro onto Galileo. Ros-Serial can be used directly without any modification to the OS image as it is compatible with Arduino.

4.3 Data Collection, Storage, and Retrieval

We have configured a Phant server [22] that accepts data insertion requests by http and stores the data in a MongoDB [19]. Phant is a modular node.js based data logging tool for collecting data from the Internet of Things. It is an asynchronous, event driven server that can respond to many connections concurrently using callbacks. A data stream is created for every device that would insert data. Data security is ensured through asymmetric cryptography in which two hash keys are generated, one public key and one private key, during the creation of every stream. The public key can be used to view or fetch data from the stream and the private key is required to insert data into the stream. Each stream contains one or more fields which are entered during creation of the stream. As an example, a data stream of temperature and humidity sensor can have fields 'temperature_F' and 'humidity_percentage'. Data of almost any form can be stored in these fields and data validation should to be performed at the device end while generating the HTTP request. Phant server adds a timestamp field to all the data streams which can be used for tracking the sensor values over time. As an example, wheel chair localization data from @Home is displayed in the Figure 4.3. The wheel chair sends position and orientation information to the Phant server. The position is stored in the fields named x, y, z and the orientation is stored in roll, pitch, and yaw. Phant server adds a timestamp field and logs the date-time at which a record was inserted in the database. This data can be read by other devices to find the position and orientation of wheel chair.

Once a data stream is created and keys are available, data can be inserted using a simple HTTP GET or POST request. While generating a GET request the public_key, private_key, variable names, and variable values are embedded in the URL that is sent to the server. In POST request, this information can be added to the HTML header instead of the URL.



Wheelchair localization This stream provides location of the wheelchair along with its direction using AprilTags mounted on the ceiling. Manage

JSON CSV MySQL PostgreSQL Atom TAGS apriltags camera location wheelchair

-Infinity% (0.00 of 0 MB) remaining.

pitch	roll	timestamp	x	y	yaw	z
-0.569618	179.887	2014-12-16T05:25:03.321Z	59.0821	35.3539	32.4253	30.0361
-0.844295	179.858	2014-12-16T05:25:01.234Z	58.745	35.1477	32.4124	29.9131
-0.545082	-179.698	2014-12-16T05:24:59.341Z	59.4029	34.8734	32.4568	30.2009
-0.401507	-179.789	2014-12-16T05:24:57.440Z	59.4439	35.5825	35.147	30.2112
-0.551937	-179.87	2014-12-16T05:24:55.564Z	59.2693	35.6885	36.3131	30.0314
-0.314661	-179.903	2014-12-16T05:24:53.673Z	59.5063	36.0622	36.9527	30.1475
0.0633636	179.958	2014-12-16T05:24:51.763Z	59.8883	36.8965	39.3319	29.9724
2.76196	179.297	2014-12-16T05:24:49.878Z	59.6287	43.2033	54.2751	29.4933

Figure 4.3: Localization data

The HTTP request in the form of GET or POST request can be generated and sent to the Phant server using any of the programming language supported by the sensing device. We used Python, NodeJS, C in arduino IDE, and C++ on Intel Galileo to insert and retrieve data from different sensors. Data from Ubiquity sensors is inserted in the same database using mFi controller provided by Ubiquity and hence Phant server is not required in this case. The data collected can also be used to create charts for analysis of environmental

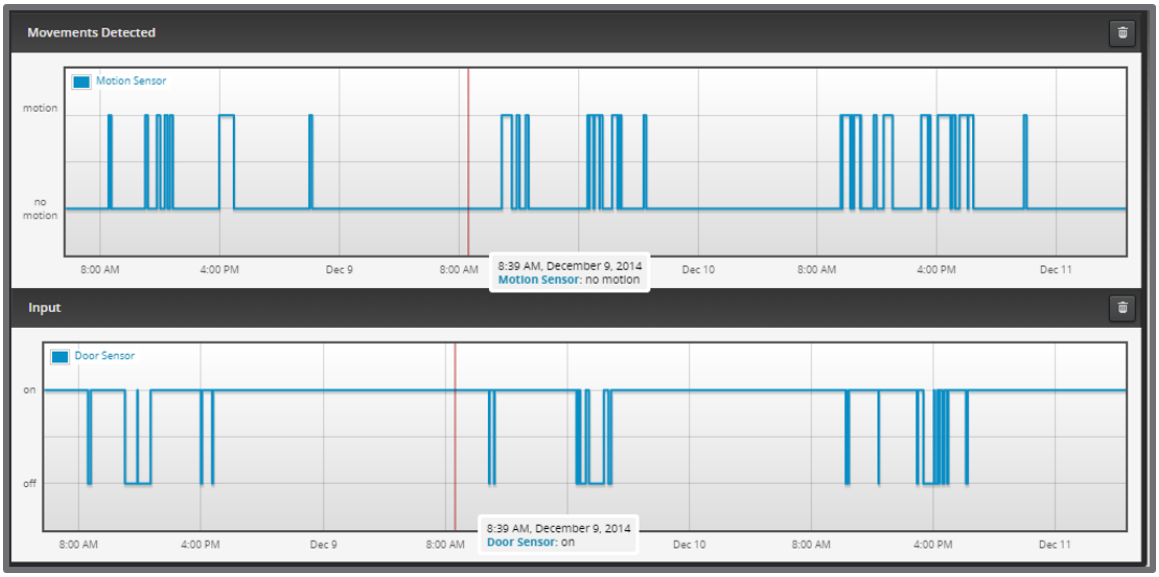


Figure 4.4: Motion sensor and door sensor data

parameters. Figure 4.4 shows motion sensor and door sensor outputs in a chart. Output from both the sensors is digital, hence we see either high or low in the charts. The motion sensor sends a high signal when motion is detected and low otherwise. Door sensor sends a high signal when the door is closed and low when the door is open. Combining data from both these sensors can provide an approximation of someone entering or leaving the house.

4.4 Projects

As a part of graduate-level course Model-Based Design, in Fall-2014, teams of students have developed several subsystems that were tested and deployed in the @Home. As a part of this course project, each team designed and implemented one subsystem that can be placed in @Home [6]. Information collected by all the devices built during these projects was stored in the central database and was used by other devices to perform their tasks. The projects aimed at providing assistive technologies to older adults and were developed using Intel Galileo boards. Description of few of those projects is given below.

4.4.1 Height Adjusting Cane

This project aims at revamping the idea of the walking cane by helping the elderly to move efficiently and with minimal efforts. It provides automatic height adjustment for the cane to suit different scenarios. The purpose of the cane is to facilitate and ease the movement of the user, especially on stairs, landings, and when bending over to pick up objects. A 1/3" per revolution and 30:1 gear motor are retrofitted in a milled slot on a regular metal cane.

The cane can adjust its height based on user commands or based on contextual location data. When the user enters the bottom of the staircase area, the cane retracts 7" to help the user up the stairs. When the user enters the top of the staircase, the cane extends 7" to help the user down the stairs. The retract and extend behavior of the cane is controlled by a finite state machine (FSM) which provides a desired length to a PID controller. The PID controller then spins the motor the correct amount to achieve the length.

Apart from the stair climbing scenario, the cane can also be beneficial to assist the user

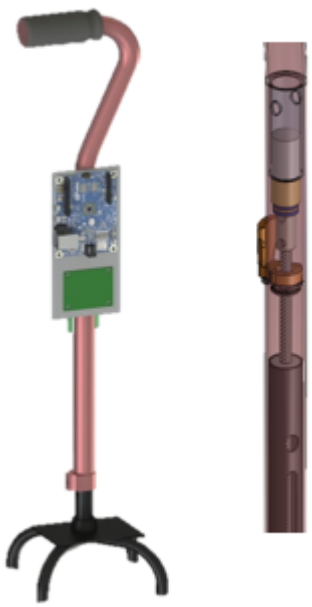


Figure 4.5: Height Adjusting Cane

while rising, either from a chair, getting out of bed, or bending over to pick up an object. The ultimate goal of the cane would be improve ease of use, so an untrained elderly user can understand how the cane operates in less than 30 seconds. The cane also provides a nice platform to integrate additional sensing such as heart monitors, fall detection, floor anomaly detection, and obstacle mapping within the home.

4.4.2 Localization of wheelchair

Indoor localization is one of the biggest challenges for assistive robots, especially scooters and wheelchairs. Without good localization, the user needs to teleoperate the assistive device [4]. Sensors that have a global reference point do not suffer from the problems of accumulating drift and progressively worse localization estimates [23]. We wanted to demonstrate that with minimal modification to an environment, a cost effective device could enable a robot within the @Home testbed to accurately localize. For simplicity, we assumed the environment with the exception of the robot is static.

To implement the indoor localization, we placed a series of virtual reality April tags [16]

on the ceiling. An upward facing camera mounted to an embedded system calculated the pose of the robot with respect to the ceiling tags. While it is not reasonable to expect people to place tags that detract from the decor of the home on their ceiling, we have discovered a workaround. The tags could be applied using an infrared absorbing or blocking paint that is visually transparent. By swapping the camera with and infrared one, we can then detect the tags on the ceiling without them being a visual impediment.

4.4.3 Sleep detector

One of the most significant concerns about elderly aging-in-place and living independently is forgotten appliances that pose a fire hazard. This project proposes to utilize wearable biometrics in a smart home to monitor sleep. If a persons sleep can be reliably detected, an appliances connected to the WiFi controlled power strips can be automatically checked and turned off if they pose a fire risk. By monitoring the heart rate, respiratory rate, and acceleration, we can accurately determine if the user is sleeping. To eliminate false negatives (which carry a heavy cost in this scenario), only three of the conditions, heart rate, respiration rate, respiration rate, or accelerometer data need to be below their thresholds to indicate the user has fallen asleep. A simple FSM defines these thresholds and controls the power to the appliances.

The system was tested with the shirt on a user who was in the process of falling asleep. The sensor data was then monitored by the FSM and within a prescribed time period detected the sleeping state and shut off the connected devices. Because of the relatively low thresholds for the sleep state, no false positives were detected during the users daily routine. Future versions should have adaptive thresholds that eliminate the need to tailor them manually to each individual using the system.

4.4.4 Occupancy Detecting Floor Mat

The team integrated an array of force sensors on the bottom of a small area rug that would be common to a home environment. As an elderly persons senses and cognitive abilities degrade, one problem that occurs is they have a harder time recognizing if anyone is with them in a given room. Vision-based implementations to help track people throughout

Figure 4.6: Hexoskin - Sleep Detector

their homes have been developed before (in addition, we present one such project for tracking an assistive robot or wheelchair), but privacy concerns limit their applicability in real-world environments. Instead, a pressure sensitive rug can easily distinguish and track people throughout rooms of a home without the privacy concerns that cameras introduce.

Figure 4.7: Smart Carpet

To execute the pressure sensitive rug, an array of force sensors was attached to the bottom of an existing floor rug. A simple mass-spring-damper model was employed to model the forces of the users foot on top of the rug. In order to eliminate false positive detection from dropped objects, pets, etc, the action of a user stepping on and off the rug, from heel strike to toe off, was modeled in logic and detections that did not fit the forces associated with a step are rejected. This type of filtering proved very good at rejecting spurious and accidental detection.

Given the simplicity of the system, several assumptions were made including one person stepping on the rug at a time, the array of sensors is across the direction of travel (the array is parallel to a door opening for example), the person does not Uturn on the rug, and the person does not jump but walk onto the rug. Testing showed the system did not miss a single detection when a user walked onto the rug from within $\pm 45^{\circ}$ of the centerline of the rug. In addition, various objects were dropped onto the rug to test the false positive rejection, and not a single objects registered as a footstep.

4.4.5 Intelligent Stool

Short stools are a popular furniture item in many households.Elderly individuals like to use them because they are helpful for placing objects on top, seating, or using them as a footrest. The stool serves as an extension that helps the elderly by providing flexibility, comfort, and support as they need it. A serious issue though, stools present a tripping hazard when they are not in use. They are short, and may be more difficult to recognize than larger furniture such as tables making them a potential hazard.

The team implemented an intelligent stool that can be summoned or dismissed based on voice commands. When the stool is needed, it approaches the user and provides the necessary assistance. When the user is done using the stool, they can command it to move away. The stool is implemented using the commercially available TurtleBot 2 platform, which is a small differentially driven robot approximately the size of a stool. The robot has a preloaded map of the environment and uses a 3D Adaptive Monte Carlo Localization (AMCL) algorithm to localize within the map. Using the Bluetooth localization beacons to locate the user, the robot can navigate to the user and assist them.

4.4.6 Intelligent Wardrobe

This project implemented in the CPS testbed involved an adaptive system that would suggest wardrobe choices to the user based on the weather outside. Weather changes are not always foreseen easily by the elderly. A simple system to assist their choice in clothing in the morning can reduce the likelihood they are unprepared for the days weather. A series of hangers with visual tags are placed in the wardrobe and an embedded system holds a mapping of what articles of clothing are on each hanger.

A Mandani fuzzy inference system (FIS) provides a recommendation to the user based on the most recent data about the days forecast. The system takes into account temperature, rainfall, snowfall, and wind speed providing suggestions in categories such as shirts (both short and long-sleeve), pants, jeans, shorts,etc. In addition, the users preferences can be taken into account as the system functions over a longer period of time. A simple FSM controlled the operation of the embedded system to detect the hangers, poll the latest weather data, generate a suggestion, and confirm the users final selection.

4.4.7 Assistive Walker

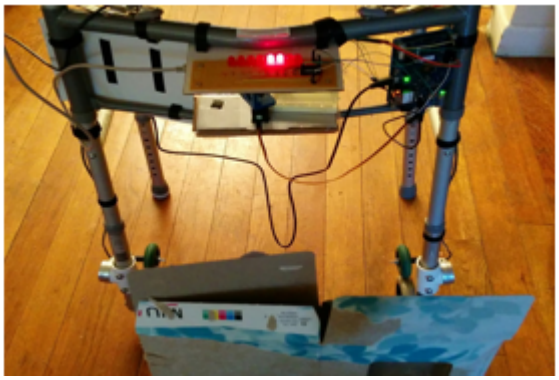


Figure 4.8: Assistive Walker

Walkers help the elderly by providing support and a platform to rest on when they have to stand for extended periods of time. The project team implemented a system that assists the elderly to navigate and move around the environment when using their walkers. Most commercially available walkers are simple aluminum frames with wheels and skids.

Some have brakes included as well, but none have any sort of built in intelligence. The team developed an assistive walker that has obstacle detection system which provides visual cues. A ring of LED lights can help the user navigate around obstacles or through doorways without getting their walker stuck.

The system consisted of a regular aluminum walker, retrofitted with wheel encoders to provide odometry information, ultrasonic sensors and LIDAR to detect obstacles in front of the walker, and a ring of LED on the front bar to guide the user to a given direction. In testing, the walker was able to correctly guide a user through the center of a doorway, preventing them from bumping the sides. While the passive system would be useful to many individuals, and system that also activates the brakes may be of more use to the elderly with more severely degraded physical capabilities.

4.5 Integration in @Home testbed

Total of nine projects were built and integrated into the @Home testbed. Though the projects were built independently, they form a part of the entire smart home environment. All the projects are connected to the local network and insert or fetch data from the Phant server. These projects use different technologies to communicate with the testbed and every device is independent of hardware or software used in other device. None of the devices are connected to Internet for security reasons. Any request that needs data from the Internet is sent to the central server and the server responds to that request by fetching data from Internet thus keeping an additional layer of security.

Chapter 5

Conclusions

We presented a framework for operating mobile robots at multiple homes to assess fall risks in the home environment. This included development of a prototype using a turtlebot platform. The robot is capable of screening homes and providing the collected information to the health care provider through the proposed framework. As this is a shared control system, an operator would be controlling the robot to collect necessary data. The user interface provides tools to navigate the robot, measure distances between selected points, set a navigation goal, and monitor the sensor outputs. During a typical screening, an operator would navigate the robot around the house while monitoring the checklist. The operator can also initiate or answer a telepresence call with the patient.

Next, we explained the architecture and design considerations of a Smart Home that would serve as a testbed for rapid prototyping of Cyber Physical Systems that can communicate with each other using the Smart Home network and assist the occupants. One such fully functional testbed is built at WPI @Home as a part of this project. We integrated nine projects running different technologies, ranging from off-the-shelf devices to custom built sensors, in this testbed.

Appendix A

Appendix

A.1 Kinematic Model of Turtlebot

Kinematics pertains to the motion of a body without considering the forces/torques that cause the motion. In mobile robotics, we need to understand the mechanical behavior of the robot both in order to design appropriate mobile robots for tasks and to understand how to create control software for an instance of mobile robot hardware. The Turtlebot is modeled as a two-wheeled differential drive robot with 2 additional points of contact as shown in Figure A.1

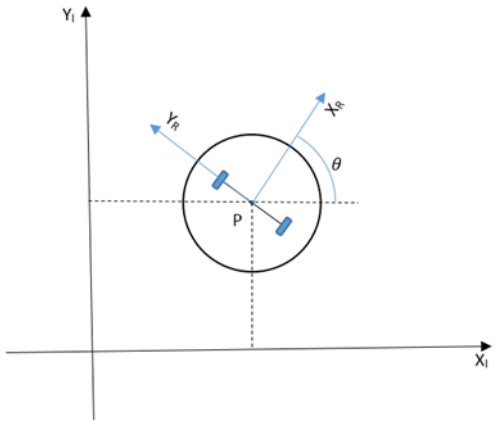


Figure A.1: Turtlebot - Reference Frame

Radius of Wheel : r

Rotation of the wheel : φ

The speed of each wheel : $r\dot{\varphi}_i$

Therefore, the translation speed is given by,

$$\dot{x_R} = \frac{r(\dot{\varphi}_1 + \dot{\varphi}_2)}{2}$$

The angular velocity of P for one wheel :

$$\dot{\theta}_1 = \frac{r\dot{\varphi}_1}{2l}$$

Hence, the total angular velocity of point P is given by,

$$\dot{\theta} = \frac{r}{2l}(\dot{\varphi}_1 - \dot{\varphi}_2)$$

The robot motion can be defined as

$$\xi_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\varphi}_1, \dot{\varphi}_2)$$

For an angular rotation of reference frame by θ, Rotation matrix is given by,

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore, the robot motion between frames is given by,

$$\xi_R = R(\theta)\xi_I$$

$$\xi_I = R(\theta)^{-1}\xi_R$$

$$R(\theta)^{-1} = R(\theta)^T = R(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\xi_I = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{r}{2} \begin{bmatrix} \dot{\varphi_1} + \dot{\varphi_2} \\ 0 \\ \frac{\dot{\varphi_1} + \dot{\varphi_2}}{l} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{A.1}$$

Equation A.1 is the Kinematic Model of a differential drive robot.

A.2 Intel Galileo Board Specifications

- 400MHz 32-bit Intel Pentium instruction set architecture (ISA)-compatible processor
  - 16 KByte L1 cache
  - 512 KBytes of on-die embedded SRAM
  - Simple to program: Single thread, single core, constant speed
  - ACPI compatible CPU sleep states supported
  - An integrated Real Time Clock (RTC), with an optional 3V coin cell battery for operation between turn on cycles.
- 10/100 Ethernet connector
- Full PCI Express* mini-card slot, with PCIe* 2.0 compliant features
  - Works with half mini-PCIe* cards with optional converter plate
  - Provides USB 2.0 Host Port at mini-PCIe* connector
- USB 2.0 Host connector
  - Support up to 128 USB end point devices
- USB Client connector, used for programming

  - Beyond just a programming port - a fully compliant USB 2.0 Device controller

- 10-pin Standard JTAG header for debugging
- Reboot button to reboot the processor
- Reset button to reset the sketch and any attached shields
- Storage options:
  - 8 MByte Legacy SPI Flash whose main purpose is to store the firmware (or bootloader) and the latest sketch. Between 256 KByte and 512 KByte is dedicated for sketch storage. The upload happens automatically from the development PC, so no action is required unless there is an upgrade that is being added to the firmware.

A.2.1 GPIO

GPIO capabilities of Galileo board are exposed through Linux Sysfs interface. To control a pin from linux(Yocto/Debian) we should write the GPIO number to /sys/class/gpio/export file. This creates a new directory under /sys/class/gpio. Parameters of the pin can be controlled by writing required values to the files inside this newly created directory. GPIO numbers are different than the on-board pin numbers. Reading sensor values or sending commands to actuators through these pins can be achieved by Read/write operations on these files. Pin mapping is given in the table A.1.

A.2.2 Digital Read/Write

As shown in the table A.1 Digital pin 7 can be controlled by setting parameters for gpio27. To activate it write the gpio number to /sys/class/gpio/export. In the following shell script examples, a line starting with $ is a command and a line without $ is the output of command on previous line. Assuming that the IP address of Galileo board is 192.168.0.6

```
$ ssh root@192.168.0.6
$ cd /sys/class/gpio
```

On-Board number	Linux sysfs number	Settings
Digital 0	gpio50	gpio40 = 1
Digital 1	gpio51	gpio41 = 1
Digital 2	gpio32	gpio31 = 1
Digital 3	gpio18	gpio30 = 1
Digital 4	gpio28	
Digital 5	gpio17	
Digital 6	gpio24	
Digital 7	gpio27	
Digital 8	gpio28	
Digital 9	gpio19	
Digital 10	gpio16	gpio42 = 1
Digital 11	gpio25	gpio43 = 1
Digital 12	gpio38	gpio54 = 1
Digital 13	gpio39	gpio55 = 1
Analog 0	in_voltage0_raw	gpio37 = 0
Analog 1	in_voltage1_raw	gpio36 = 0
Analog 2	in_voltage1_raw	gpio23 = 0
Analog 3	in_voltage3_raw	gpio22 = 0
Analog 4	in_voltage4_raw	gpio21 = 0 & gpio29 = 1
Analog 5	in_voltage5_raw	gpio20 = 0 & gpio29 = 1

Table A.1: Intel Galileo GPIO pins

```
$ echo -n "27" > export
$ cd gpio27
```

Set the I/O direction as input or output by writing in or out in the file named direction

```
$ echo -n "out" > direction
```

Drive can be set as pullup, pulldown, strong or hiz depending on the application. In most applications it would be set to strong.

```
$ echo -n "strong" > drive
```

To confirm the values are written correctly, the following command can be used

```
$ cat direction
out
$ cat drive
strong
```

To change the value of output pin, we can export either 1 or 0 to the value file. The output can be confirmed by checking the voltage at digital pin 7

```
$ echo -n "1" > value
$ echo -n "0" > value
```

To read the value of pin, the following command can be used

```
$ cat value
0
```

A.2.3 Analog Read/Write

Six Analog inputs are available on Galileo. The resolution of each channel is 12 bit, providing a range of 0-4095. Value 0 signifies input voltage of 0V and value 4095 signifies input voltage of 5V. To read the analog input in linux, we should first set/reset the corresponding GPIO pin given in the settings column in table A.1. Once the pins are set, the value can be read from /sys/bus/iio/devices/iio:device/in_voltageX_raw file. Example below shows how to read input from pin A4.
For reading input from A4, GPIO 21 should be 0 and GPIO 29 should be 1.

```
$ echo -n "21" > /sys/class/gpio/export
$ echo -n "out" > /sys/class/gpio/gpio21/direction
$ echo -n "0" > /sys/class/gpio/gpio21/value
```

```
$ echo -n "29" > /sys/class/gpio/export
$ echo -n "out" > /sys/class/gpio/gpio29/direction
$ echo -n "1" > /sys/class/gpio/gpio29/value
```

Read A4 value from sysfs

```
$ cat /sys/bus/iio/devices/iio\:device0/in_voltage4_raw
4090
```

A.2.4 Pulse Width Modulation

Six Pulse Width Modulation (PWM) Channels are available in Galileo. They are mapped as shown in table A.2

Pin Number	PWM Channel
3	3
5	5
6	6
9	1
10	7
11	4

Table A.2: PWM Pin mapping

Usage of PWM pins is similar to digital pins with minor modifications. To export a PWM channel to sysfs, use

```
$ cd /sys/class/pwm/pwmchip0
$ echo -n "3" > export
$ cd pwm3
```

Enable PWM

```
$ echo -n "1" > enable
```

Set PWM period in nanoseconds. The following commands sets the period as 1 millisecond

```
$ echo -n "1000000" > period
```

Set PWM duty cycle by writing its length in nanoseconds in duty_cycle file. The following command sets the duty_cycle as 50

```
$ echo -n "500000" > duty_cycle
```

Bibliography

[1] World Health Organization. Ageing and Life Course Unit, *Who global report on falls prevention in older age*, World Health Organization, 2008.

[2] B. Alexander, K. Hsiao, C. Jenkins, B. Suay, and R. Toris, *Robot web tools [ros topics]*, Robotics Automation Magazine, IEEE **19** (2012), no. 4, 20–23.

[3] Carlos Angrisano, Diana Farrell, Bob Kocher, Martha Laboissiere, and Sara Parker, *Accounting for the cost of health care in the united states*, McKinsey Global (2007).

[4] Guillaume Blanc, Youcef Mezouar, and Philippe Martinet, *Indoor navigation of a wheeled mobile robot along visual routes*, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, IEEE, 2005, pp. 3354–3359.

[5] CakePHP, *http://cakephp.org/*.

[6] Velin Dimitrov, Vinayak Jagtap, Mitchell Wills, Jeanine Skorinko, and Taşkın Padır, *A cyber physical system testbed for assistive robotics technologies in the home*, (Under Review).

[7] John V Draper, David B Kaber, and John M Usher, *Telepresence*, Human Factors: The journal of the human factors and ergonomics society **40** (1998), no. 3, 354–375.

[8] Ruixiang Du, Vinayak Jagtap, Yanren Long, Oke Onwuka, and Taşkın Padır, *Robotics enabled in-home environment screening for fall risks*, Proceedings of the 2014 Workshop on Mobile Augmented Reality and Robotic Technology-based Systems (New York, NY, USA), MARS '14, ACM, 2014, pp. 9–12.

[9] A. Enes and W. Book, *Blended shared control of zermelo's navigation problem*, American Control Conference (ACC), 2010, June 2010, pp. 4307–4312.

[10] Dieter Fox, *Kld-sampling: Adaptive particle filters*, Advances in neural information processing systems, 2001, pp. 713–720.

[11] P. Griffiths and R.B. Gillespie, *Shared control between human and machine: haptic display of automation during manual control of vehicle heading*, Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings. 12th International Symposium on, March 2004, pp. 358–366.

[12] G. Grisetti, C. Stachniss, and W. Burgard, *Improved techniques for grid mapping with rao-blackwellized particle filters*, Robotics, IEEE Transactions on **23** (2007), no. 1, 34–46.

[13] Richard Harper, *Inside the smart home*, Springer Science & Business Media, 2003.

[14] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, *OctoMap: An efficient probabilistic 3D mapping framework based on octrees*, Autonomous Robots (2013).

[15] JSON, *Javascript Object Notation http://www.json.org/.*

[16] E. Olson, *Apriltag: A robust and flexible visual fiducial system*, Robotics and Automation (ICRA), 2011 IEEE International Conference on, May 2011, pp. 3400–3407.

[17] Open Source Robotics Foundation, *Turtlebot 2.*

[18] Jennifer M Ortman, Victoria A Velkoff, and Howard Hogan, *An aging nation: the older population in the united states.*

[19] Eelco Plugge, Tim Hawkins, and Peter Membrey, *The definitive guide to mongodb: The nosql database for cloud and desktop computing*, 1st ed., Apress, Berkely, CA, USA, 2010.

[20] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng, *Ros: an open-source robot operating system*, ICRA workshop on open source software, vol. 3, 2009.

[21] Thomas B Sheridan, *Teleoperation, telerobotics and telepresence: A progress report*, Control Engineering Practice **3** (1995), no. 2, 205–214.

[22] SparkFun, *Phant Server http://phant.io/about/.*

[23] Suguna P Subramanian, Jurgen Sommer, Stephen Schmitt, and Wolfgang Rosenstiel, *Rilreliable rfid based indoor localization for pedestrians*, Software, Telecommunications and Computer Networks, 2008. SoftCOM 2008. 16th International Conference on, IEEE, 2008, pp. 218–222.

[24] Russell Toris, David Kent, and Sonia Chernova, *The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing*, Journal of Human-Robot Interaction **3** (2014), no. 2, 25–49.

[25] Ad Van Berlo, *A smart model house as research and demonstration tool for telematics development*, Proc. 3rd TIDE Congres: Technology for Inclusive Design and Equality Improving the Quality of Life for the European Citizen, 1998, pp. 23–25.