



DRAFT SYSTEM DESIGN DOCUMENT

MECHATRONICS 4TB6
GROUP 2

NATHAN FUJIMOTO
PRAKHAR GARG
JOSH GILMOUR
AARON JASS
TYLER JASS
FAUZIA KHANUM
JACK LIU
GAGAN SINGH


	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

Table of Contents

Table of Contents	2
Table of Tables	4
Revision History	5
Purpose	7
Scope	7
1 Context Diagram of Boundaries	8
2 System Data Flow Diagram	10
3 Monitored and Controlled Variables	11
4 Constants	14
5 Behaviour Overview	15
5.1 Overall System Behaviour	15
5.2 Navigation Goal Calculation	18
6 Component Details	20
7 Normal Operation	24
7.1 Autonomous	24
7.1.1 Moving to user	24
7.1.2 Moving to docking station	24
7.1.3 Docked	25
7.2 Manual	25
7.2.1 Regular Walking	25
7.2.2 Slope Change	25
7.2.3 Idle	25
8 Undesired Event Handling	27
8.1 Emergency Stop	27
8.2 User Detected During Autonomous Mode	27
8.3 Object Collision with Smartwalker	27
8.4 Walker has Fallen Over or Very high Incline/Decline	27
8.5 Goal location is unreachable	28
8.6 Walker cannot “see”	28

8.7 Walker is unable to move while autonomously navigating	28
8.8 Extremely low battery	28
References	29

Table of Figures


Figure 1 - Context Diagram of Boundaries	8
Figure 2 - System Data Flow Diagram	10
Figure 3 - Overall System Behaviour Diagram	15
Figure 4 - Autonomous Mode Behaviour Diagram	16
Figure 5 - Navigation Goal Calculation	18
Figure 6 - Manual Mode Behaviour Diagram	18

Table of Tables


Table 1 - Entity Descriptions and Interactions	9
Table 2 - Monitored Variables	11
Table 3 - Controlled Variables	13
Table 4 - State Behaviour	16
Table 5 - Autonomous Mode States	17
Table 6 - Manual Mode States	19
Table 7 - Component Descriptions	21
Table 8 - Derived Timing Constraints	22
Table 9 - Initialization	23

Revision History

Rev	Author(s)	Description of Change	Peer Reviewed	Date
-	T.Jass	Original Document	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg G.Singh F.Khanum J.Liu	01-Dec-2017
A	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg G.Singh F.Khanum J.Liu	Added initial rough draft of content.	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg G.Singh F.Khanum J.Liu	05-Dec-2017
B	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg G.Singh F.Khanum J.Liu	Added first draft of diagrams and added more content.	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg F.Khanum	14-Dec-2017
C	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg F.Khanum	Updated diagrams and refined content.	T.Jass A.Jass N.Fujimoto J.Gilmour P.Garg F.Khanum	20-Dec-2017
D	T.Jass A.Jass N.Fujimoto	Filling in missing content and finalized report format.		21-Dec-2017

	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

	<p>J.Gilmour P.Garg F.Khanum</p>			
--	--	--	--	--

	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

Purpose

This document's purpose is to define and portray the overall system design of all the SmartWalker capstone project. The designs in this document considers all system requirements, defined in the SmartWalker System Requirements document. The document will communicate all the necessary designs for each functional component of the SmartWalker. By the end, the reader should have a clear understanding of what each component is, and how its design meets and satisfies the requirements. The System Design document will be used to begin implementation of the SmartWalker.

Scope

The scope of this project is to design a walker with autonomous abilities to operate within a specific environment. The operating environment includes and is limited to the inside of a hospital (Patient rooms, staff lounges, hallways, common space). This does not include outside property belonging to a hospital. Outside of the operating environment the SmartWalker will operate as a standard wheeled medical walker.

This project will include (in scope) the following items:

- Autonomous driving (see Section 7.1 for details) between the user and docking station within one indoor room at a time. The docking station is a temporary storage location with charging capabilities.
- Obstacle avoidance in Autonomous mode.
- Assistive walking when using the SmartWalker is in Manual mode (see **7.2 Manual** for details).
- Emergency braking (See section 8 for details) if approaching a steep drop in elevation (such as a staircase).
- Location tracking of the SmartWalker within the operating environment.

This project will not include (out of scope) the following items:

- Autonomous usage in multiple indoor rooms simultaneously (autonomous movement is limited to one room).
- Global Positioning system (GPS) tracking

The end result of this project will demonstrate the students' cumulative learning through their academic career via a demonstration of the SmartWalker's autonomous ability in a predetermined location. The SmartWalker herein referred to as walker.

1 Context Diagram of Boundaries

The System Context Diagram gives a high-level overview of the information passed to the major system components. The entities and interactions between them are described in **Table 1 - Entity Descriptions and Interactions**. More detailed information can be seen in **2 System Data Flow Diagram**.

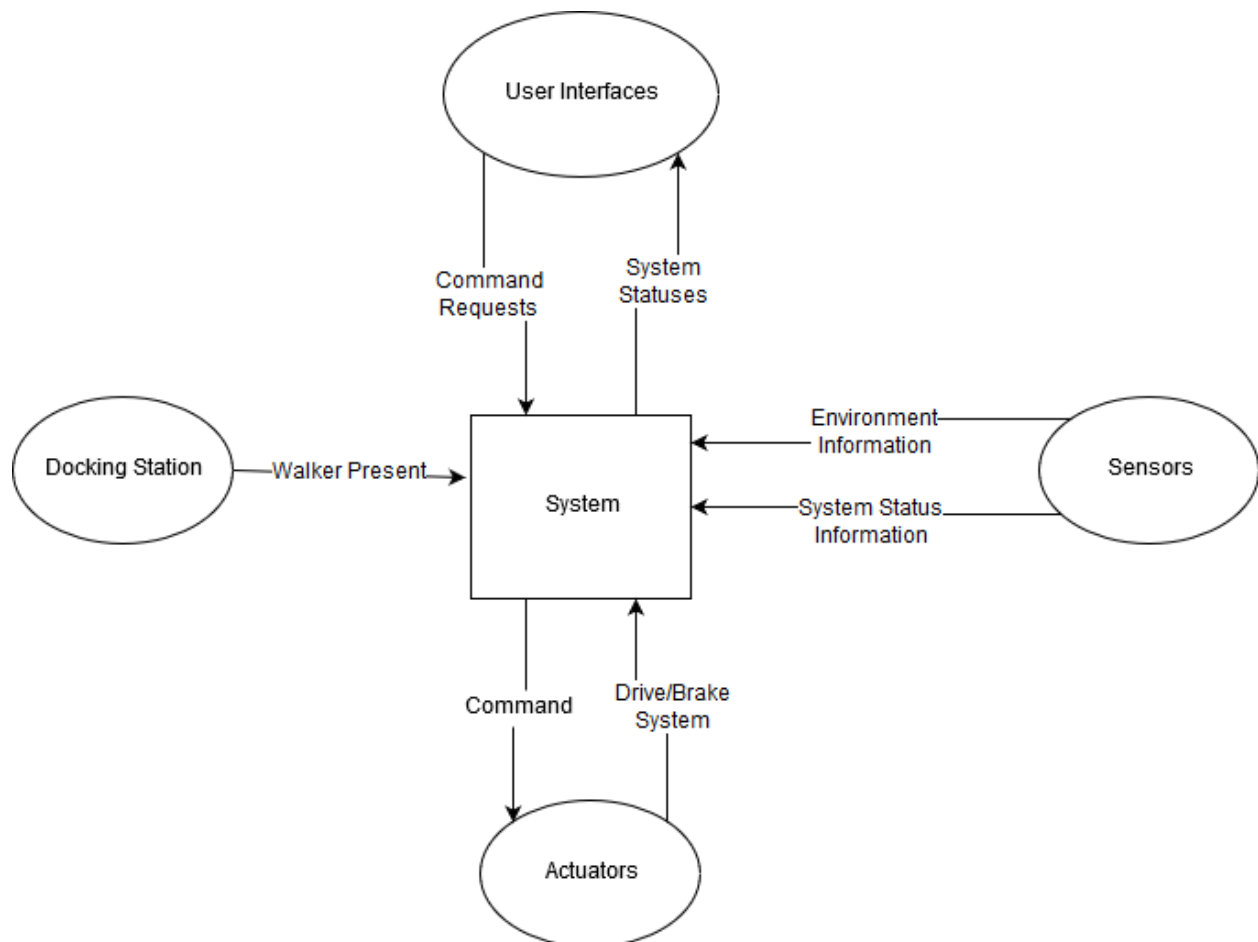


Figure 1 - Context Diagram of Boundaries

Table 1 - Entity Descriptions and Interactions

Entities	Entity Description	Entity Interactions	Interaction Description
<i>System</i>	The core of all operations. The "system" manages all of the information, computes required information, and outputs information needed by other entities.	See rows below.	
<i>User Interfaces</i>	These are the components that the user uses to interact with the system, this would include the phone application and the on-board touch screen.	<i>Command Requests</i>	This will usually be a request to move to a target location or to stop.
		<i>System Statuses</i>	This is any information about the system that may be useful to the user; Including the battery level, the operational mode of the walker, and the walker's current state.
<i>Docking Station</i>	The docking station is the walker's "Home", it is where it goes to charge.	<i>Walker Present</i>	Indicates to the system whether or not the walker is physically attached to the docking station.
<i>Actuators</i>	Components that execute a physical action. This would include the system's motors.	<i>Command</i>	Commanded operation to the actuator.
		<i>Drive/Brake System</i>	The output action of the actuator based on the command received.
<i>Sensors</i>	Components that monitor and measure environment or internal information. This information would include the system's camera vision sensor, proximity sensors, and encoders.	<i>Environment Information</i>	Information gathered by the depth and proximity sensors used in obstacle avoidance and navigation.
		<i>System Status Information</i>	Sensors that retrieve information about the system, such as encoder counts or battery level.

2 System Data Flow Diagram

The system Data Flow diagram shows how the different components of the system interact with each other, and with their respective inputs and outputs. The centre of all operations is the Raspberry Pi, which is effectively both the brain and the lesion controlling the flow of all information going in and out of the system.

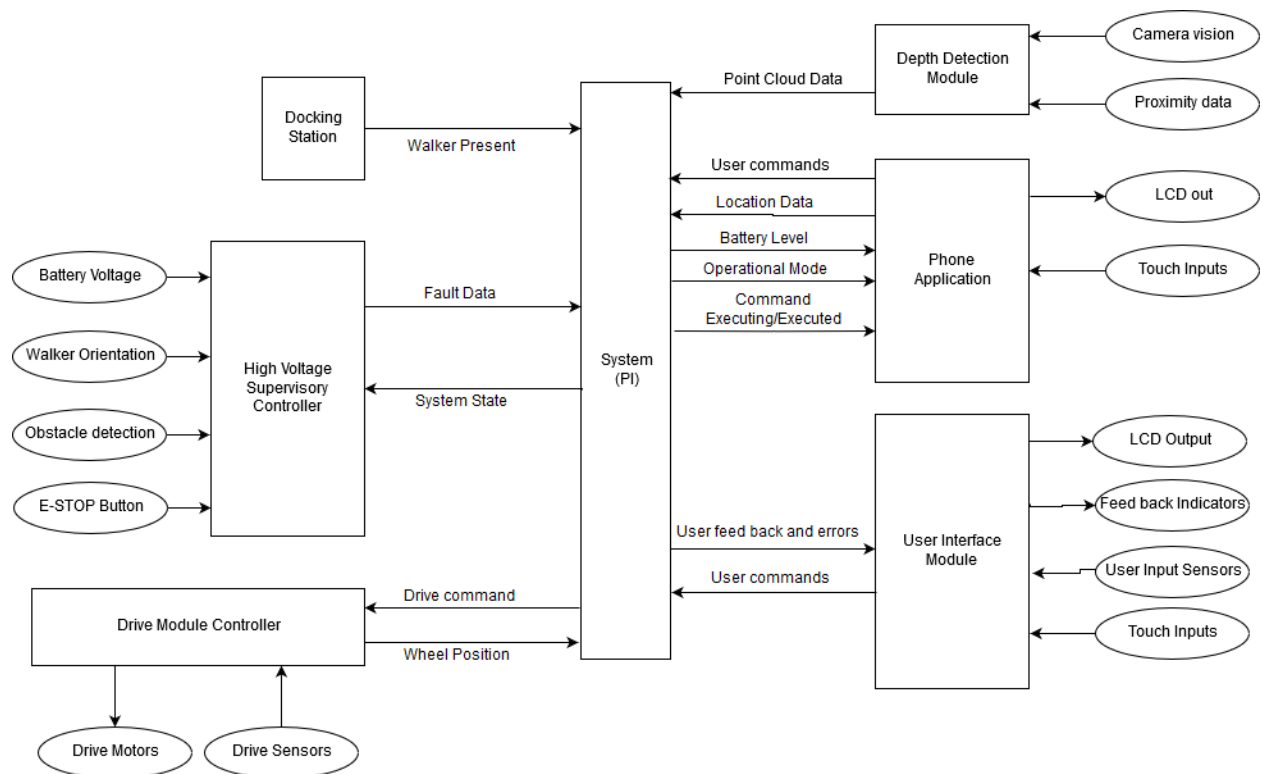


Figure 2 - System Data Flow Diagram

3 Monitored and Controlled Variables


Table 2 - Monitored Variables

Variable	Description	Units	Range
<i>m_battery_lvl</i>	Power level of battery.	Battery Sustain of Charge (%)	[0, 100]
<i>m_battery_rel</i>	Reliability of the battery.	Battery Health (%)	[0, 100]
<i>m_proximity_in_L</i>	Proximity to any objects in the environment (front-left side).	Distance (cm)	[0, 500]
<i>m_proximity_in_R</i>	Proximity to objects in the environment (front-right side).	Distance (cm)	[0, 500]
<i>m_depth_in</i>	Depth of the upcoming ground with respect to the bottom of the walker.	Distance (cm)	[-100, 0]
<i>m_walker_yaw</i>	Orientation of the walker (yaw) with respect to the walker's z-axis.	Angle (deg)	[-180, 180]
<i>m_walker_pitch</i>	Orientation of the walker (pitch) with respect to the walker's y-axis.	Angle (deg)	[-45, 45]
<i>m_walker_loc_out</i>	Global location of the walker.	[Longitude (deg), Latitude (deg)]	[-180, 180; -90,90]
<i>m_walker_loc_in</i>	Location of the walker inside the patient room, with respect to a preset coordinate system.	[x,y] on room map	[0, 100; 0, 100]

<i>m_human_loc_in</i>	Location of the human based on their phone, with respect to a preset coordinate system.	[x, y] on room map	[0, 100; 0, 100]
<i>m_human_yaw</i>	Orientation of the human (yaw) with respect to the human's z-axis.	Angle (deg)	[-45, 45]
<i>m_desired_loc</i>	User selected desired location for the walker in autonomous mode	Enum (states)	GoDocking, GoHuman
<i>m_dock_aligned</i>	Walker is physically aligned with the docking station	Boolean	0, 1
<i>m_user_in_L</i>	Desired forward assisted movement for the user, for the left motor.	Custom	[-100,100]
<i>m_user_in_R</i>	Desired forward assisted movement for the user, for the right motor.	Custom	[-100,100]
<i>m_emergency_signal</i>	Signal for monitoring an emergency situation.	Boolean	0, 1

Table 3 - Controlled Variables

Variable	Description	Units	Range
<i>c_motorL_speed</i>	Left motor speed.	Angular velocity (deg/s)	[0, 50]
<i>c_motorR_speed</i>	Right motor speed.	Angular velocity (deg/s)	[0, 50]
<i>c_brakeL_position</i>	Left brake position.	Brake Position (%)	[0, 100]
<i>c_brakeR_position</i>	Right brake position.	Brake Position (%)	[0, 100]
<i>c_charge_walker</i>	Charge the walker.	Boolean	0, 1
<i>c_dock_connected</i>	Walker physically connected to the docking station.	Boolean	0, 1

	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

4 Constants

1. Number of kinect sensors
2. Location and orientation of kinect sensors
3. Number of ultrasound sensors
4. Location and orientation of ultrasound sensors
5. Location and orientation of docking station
6. Reference orientation of the user and robot
7. Origin of estimate coordinate frame
8. Location of estimates in the room
9. Overall walker shape (parameter of differential package in ROS)
10. PID Coefficients for motor controllers
11. Voltage the walker runs on
12. Number of Arduinos
13. Number of Raspberry Pis
14. Number of motor-actuated wheels, motor control boards, and encoders
15. Maximum speed of the walker during autonomous navigation
16. Minimum allowable distance to obstacles while autonomously navigating
17. Distance in front of user that walker should initially navigate too
18. How close the walker should come to the user after initial navigation

5 Behaviour Overview

5.1 Overall System Behaviour

The behaviour of the walker system can be seen by the following behavioural state diagrams. To describe at the main behavioural components of the walker system, three state diagrams were outlined: the overall system behaviour diagram, the autonomous behaviour diagram, and the manual mode behaviour diagram. The latter two main sub states in which the walker primarily operates in. In any state, the walker is perceiving information from the environment for both functioning and state transitioning.

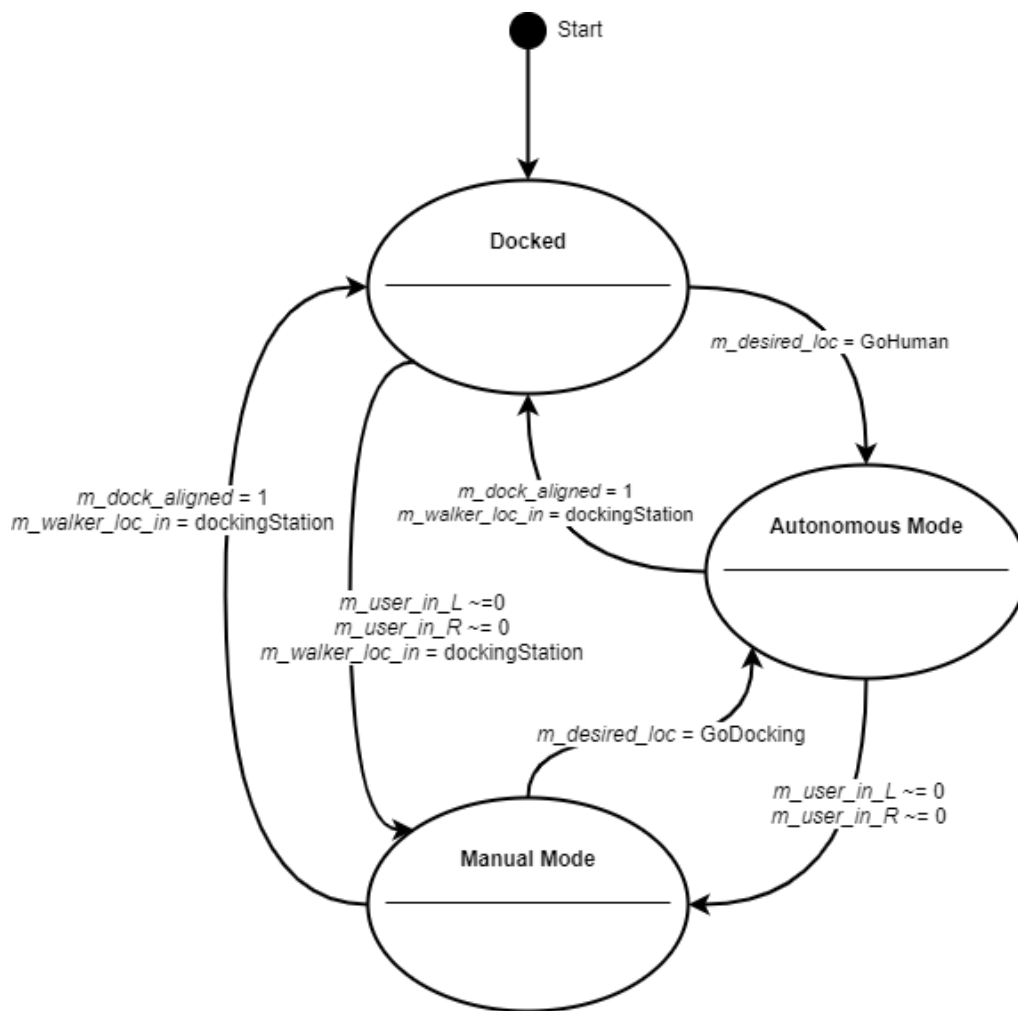


Figure 3 - Overall System Behaviour Diagram

Table 4 - State Behaviour

State	Description
<i>Docked</i>	In this state the walker is physically connected to the docking station. When connected, the walker is charging its onboard power supply.
<i>Autonomous Mode</i>	In this state the walker is autonomously navigating to and from the user and the docking station (or its current position).
<i>Manual Mode</i>	In this state the walker is being manually operated by the user. This is intended as standard walker usage, with additional safety features.

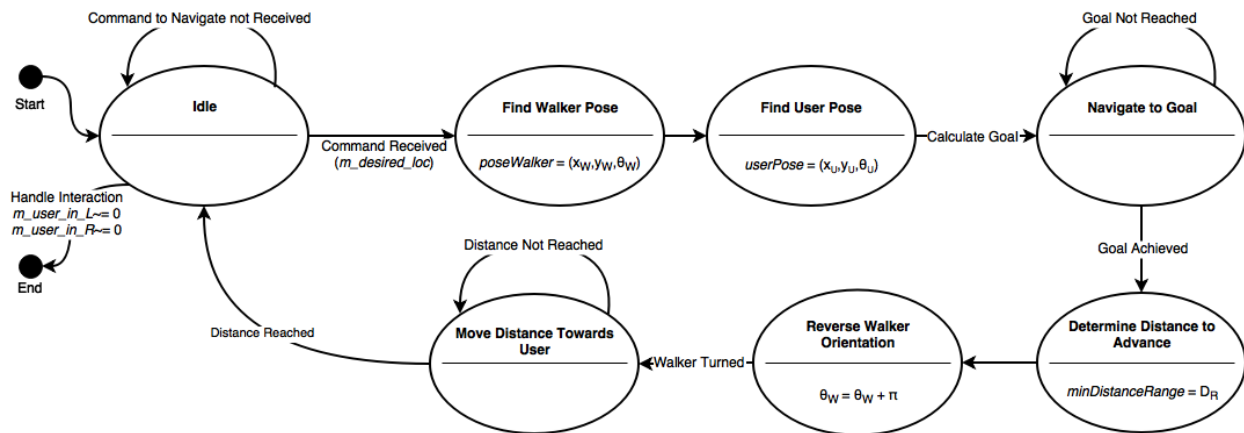


Figure 4 - Autonomous Mode Behaviour Diagram

Please note, whenever the walker is put into an unintended event, a safety check will be done of the status of the system, and the status will be fixed if possible and sent back to the idle position.

Table 5 - Autonomous Mode States

State	Description
<i>Idle</i>	In this state the walker is in a waiting phase where it is waiting on instructions on where to navigate by the user, via the user interface.
<i>Find Walker Pose</i>	In this state the walker determines its relative position on the constructed map grid, as well as its relative yaw angle on the map.
<i>Find User Pose</i>	In this state the walker system determines the relative position of the user on the constructed map grid, as well as the user's relative yaw angle on the map.
<i>Navigate to Goal</i>	In this state, considering the walker pose and user pose, the walker calculates the navigation goal and thus begins to navigate to the location.
<i>Determine Distance to Advance</i>	In this state the walker determines how far it can safely approach the destination.
<i>Reverse Walker Orientation</i>	In this state the walker orientates its yaw relative yaw angle so that its handles are facing the user (180-degree turn).
<i>Move Distance Towards User</i>	In this state the walker creeps until it has reached the safe approaching distance. The system then goes back to the idle state.

As described in the overall system behaviour diagram, if either the walker has reached the docking station or the user physically interacts with walker will the behaviour exit the autonomous mode.

5.2 Navigation Goal Calculation

The walker and user pose will both be determined by the beacons in the room. The goal is the location where the robot is supposed to travel based on where the user is. The goal is calculated to be some distance, K , in front of the user. It is calculated as:

$$\text{Navigation goal} = \{X_u + K * \cos \theta_u, Y_u + K * \sin \theta_u, \theta_u + \pi\}$$

Figure 5 - Navigation Goal Calculation

The π added to the user orientation ensures that the walker is facing the user at the end of navigation. This facilitates the measurement of distance between the walker and user in the next state. The walker is then to turn around and approach the user to a reasonable distance (to be determined).

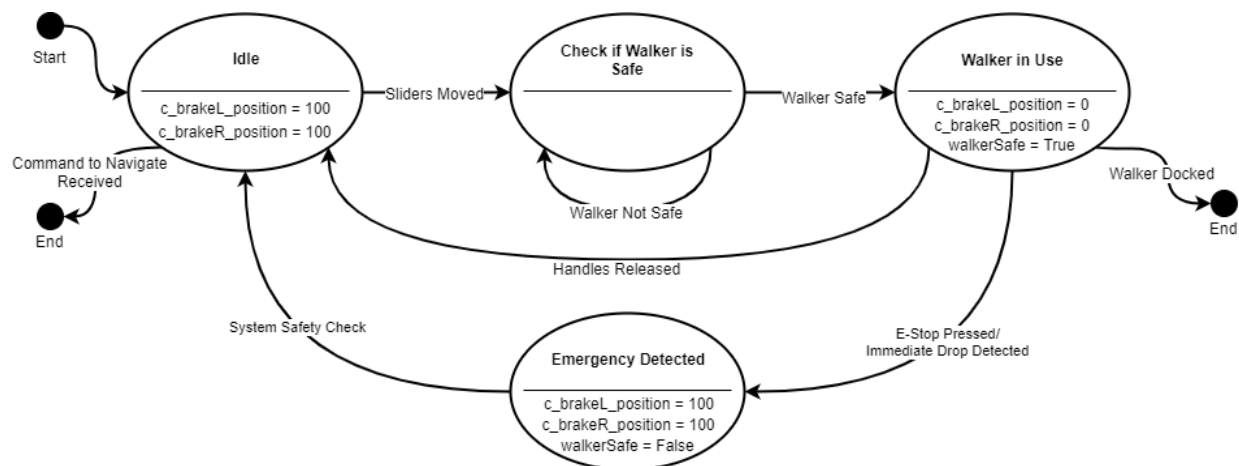


Figure 6 - Manual Mode Behaviour Diagram

Table 6 - Manual Mode States

State	Description
<i>Idle</i>	In this state the walker is in a waiting phase where it awaits physical interaction from the user, in the form of the handle sliders being moved.
<i>Check if Walker is Safe</i>	In this state the walker immediately checks the environment if there are any detectable dangerous obstacles or landmarks in the vicinity of the walker.
<i>Walker in Use</i>	Once the system deems it is safe, in this state the walker operates like a traditional walker, where it helps the user move him or herself.
<i>Emergency Detected</i>	In this state an emergency was detected and the walker brakes so it cannot be further moved. A safety check is completed so the emergency situation is no longer present.

Please note, the Emergency Detected includes the detection of any hazardous situations the user is in the vicinity of (oncoming drops, large slopes, etc..). The safety check will enact functionality to aid the user if wished to proceed through the hazard, if possible (please refer to Section 7 - Normal Operation)

Manual mode behaviour is left when the user asks for the walker to navigate back to the docking station, or to a new position where the user would be.

6 Component Details

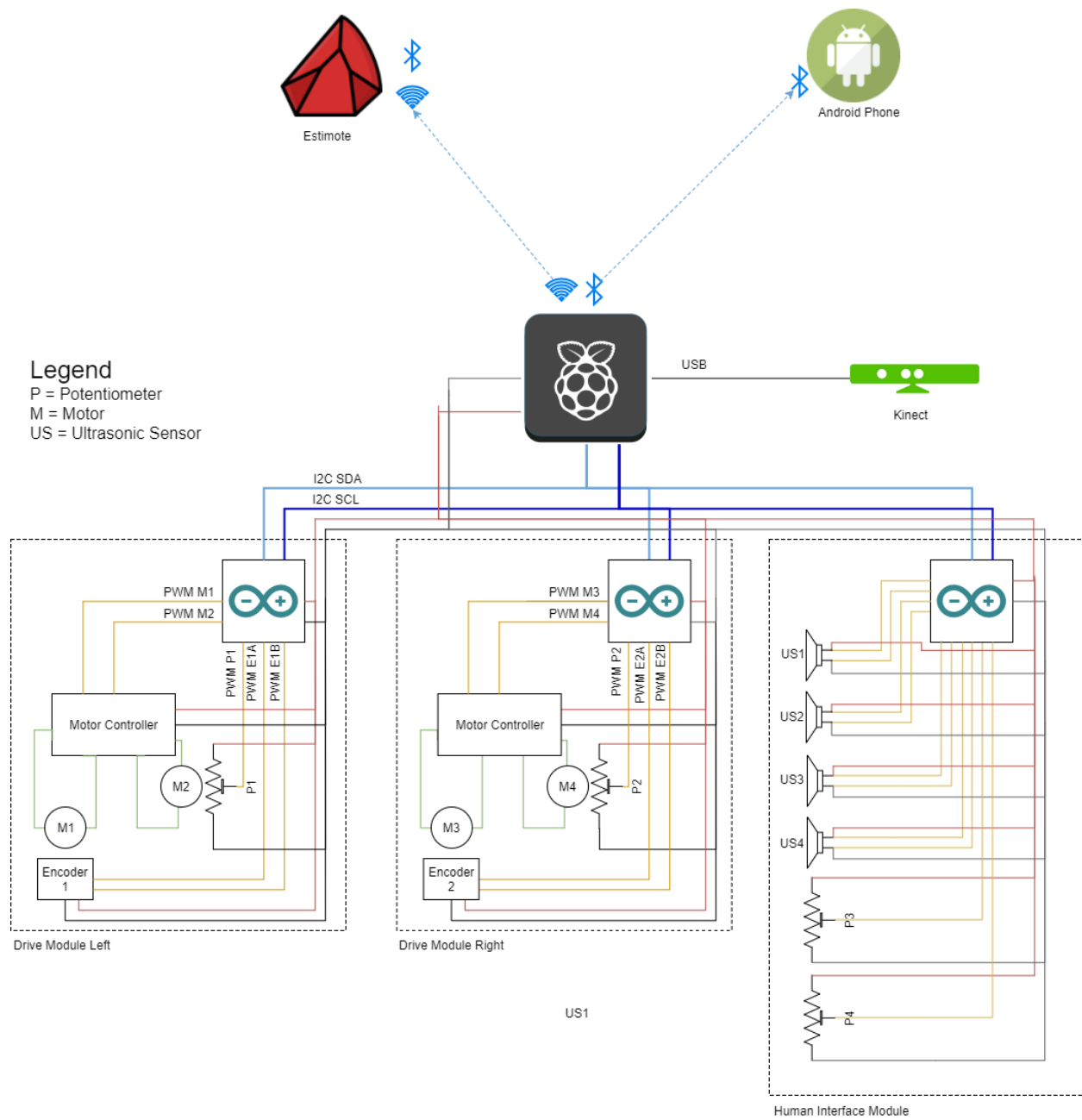


Table 7 - Component Descriptions

Component	Inputs	Outputs	Description
<i>Raspberry Pi</i>	Modules: <ul style="list-style-type: none"> • Microsoft Kinect • Estimotes • Android Mobile Phone 	Modules: <ul style="list-style-type: none"> • Drive Module Left • Drive Module Right • Human Interface Module 	The main supervisory module in the walker System. It directs data to and from the other modules to assure functionality.
<i>Estimotes</i>	Monitored Variables: <ul style="list-style-type: none"> • <i>m_walker_loc_in</i> 	Modules: <ul style="list-style-type: none"> • Raspberry Pi 	Wireless beacons to be used for indoor location and navigation of the walker System.
<i>Android Phone</i>	Monitored Variables: <ul style="list-style-type: none"> • <i>m_human_loc_in</i> • <i>m_human_yaw</i> • <i>m_desired_loc</i> 	Modules: <ul style="list-style-type: none"> • Raspberry Pi 	The assumed personal electronic device that is owned by the user. This device will host the walker Application that will transmit commands and user location information, and receive system status information.
<i>Kinect (Sensory Module)</i>	Monitored Variables: <ul style="list-style-type: none"> • <i>m_proximity_in_L</i> • <i>m_proximity_in_R</i> • <i>m_depth_in</i> • <i>m_walker_yaw</i> • <i>m_walker_pitch</i> 	Modules: <ul style="list-style-type: none"> • Raspberry Pi 	Sensor returning point cloud data about the environment. This sensor is vital to the system's object detection.
<i>Drive Modules</i>	Modules: <ul style="list-style-type: none"> • Raspberry Pi • Human Interface Module 	Control Variables: <ul style="list-style-type: none"> • <i>c_motorL_speed</i> • <i>c_motorR_speed</i> • <i>c_brakeL_position</i> • <i>c_brakeR_position</i> 	Modules that will do computations on the internal system data and external environment data from the sensors and use it to control the actuators (motors) appropriately.


<i>Human Interface Module</i>	<p>Monitored Variables:</p> <ul style="list-style-type: none"> <i>m_user_in_L</i> <i>m_user_in_R</i> <i>m_emergency_signal</i> <p>Modules:</p> <ul style="list-style-type: none"> Raspberry Pi 	<p>Modules:</p> <ul style="list-style-type: none"> Drive Modules 	<p>This module is the physical interface between the drive module and the user. It allows the user to manually drive the walker system's motors.</p>
-------------------------------	--	---	--

Table 8 - Derived Timing Constraints

Module	Derive Timing Constraints
Kinect	Cannot read data from the Kinect faster than the Raspberry Pi can process it whilst also maintaining other system functionality
Android Phone	Will communicate messages from the user at the standard Bluetooth 4.0 speeds to the Raspberry Pi (on the walker), and back to the user.
Estimates	The beacon responsiveness will adhere to the Raspberry Pi's scanning of the beacons while in Autonomous Mode.
Raspberry Pi	Must communicate with the arduinos at a rate they can handle and are configured to accept.
Drive Modules	Arduinos must communicate with the Pi and motor controllers at a rate they can handle and are configured to accept. The arduinos must run fast enough that it does not miss any encoder counts and are able to keep up with the rate of them.
Human Interface Module	The status of the human and environment interactions must be checked often enough that no user interactions are missed. Braking inputs and ultrasonic data will adhere to serial 'I-squared-C' speeds for sending data to the Raspberry Pi and to the Drive Modules.

Table 9 - Initialization

Module	Initialization
Kinect	Run the libfreenect driver to allow reading data from the Kinect.
Android Phone	The app must be running on the phone and a bluetooth connection established with the walker before several walker features can be used, such as autonomous navigation.
Estimotes	Physically the beacons required to be on the walls of the room, at the centre of each wall. Additionally, one would have to be inside the walker and connected to the Raspberry Pi at initialization.
Raspberry Pi	The Raspberry Pi will be turned on and have the necessary software pre-installed.
Drive Modules	The Arduinos, motor controllers, motors, and encoders will all have to be turned on via power-up. The arduinos will have the necessary software pre-installed.
Human Interface Module	The arduino and the sensors to interact with the human and environment (brakes and ultrasonic) will be initialized at power-up.

	<p style="text-align: center;">MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

7 Normal Operation

The following section will describe the different modes and states each mode has in more detail by stepping through possible positive normal scenarios.

Please note since this is the Draft System Design that we have not determined all details yet and have noted said details to be defined later.

7.1 Autonomous


As mentioned in our System Requirements document, autonomous mode is the ability to move out to a designated home location as well as return to the user when requested. Autonomous mode consists 3 states: moving to user, moving to docking station, and docked. The following sections will describe positive normal scenarios (behaviour the system should cover and everything proceeds as expected).

7.1.1 Moving to user

1. User is not detected by handlebar sensors (m_user_in_L and m_user_in_R)
2. User requests the walker to come to user via phone app
3. Phone app sends bluetooth signal to walker
4. If walker is not within bluetooth distance?
 - a. Display message to user “Walker out of reach.”
5. If walker is within bluetooth distance
 - a. Walker retrieves walker pose from estimates
 - b. Walker retrieves user pose from estimates
 - c. Walker calculates the goal pose and begins navigating to it, as described before
 - d. Once the walker reaches the goal, the distance to the user is measured
 - e. The walker turns 180 degrees
 - f. The walker backs up to within a reasonable distance from the user
 - g. Walker returns to idle state

7.1.2 Moving to docking station

1. User is not detected by handlebar sensors (m_user_in_L and m_user_in_R)
2. User sends signal to dock via phone app
3. If walker is not within bluetooth distance
 - a. Display message to user “Walker out of reach.”
4. If walker is within bluetooth distance
 - h. Walker retrieves walker pose from estimates
 - i. Walker retrieves docking station pose from estimates
 - j. Walker moves to docking station
 - k. Walker docks into charging port (m_dock_aligned)
 - i. Go to Docked state

	<p style="text-align: center;">MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

7.1.3 Docked

1. Walker is at the docking station and connected (c_dock_connected)
 - a. If battery is less than 100% (m_battery_lvl)
 1. charge (c_charge_walker)
 - ii. If battery is at 100% (m_battery_lvl)
 1. Disengage charging and go to Idle state

7.2 Manual

As mentioned in our System Requirements document, Manual mode is the ability to operate like a standard medical walker, with added smart features such as assisted braking and assisted driving. Manual mode has 4 states: regular walking, slope change, objects detected, and idle. The following sections will describe positive normal scenarios (behaviour the system should cover and everything proceeds as expected).

7.2.1 Regular Walking

This state is when the user is utilizing the walker as intended; aiding in regular walking.

1. User detected at handlebar sensors (m_user_in_L and m_user_in_R)
 - a. Detect if the walker is moving and surrounding environment
 - i. Slope change → Slope Change mode
 - ii. Objects in way → Objects detected
 - iii. User is utilizing walker for normal walking
 - b. User applies brakes manually (c_brakeL_position and c_brakeR_position)
 - i. reduce speed accordingly
2. If user is not detected (for more than a certain amount of time, to be determined)
 - a. Enter idle state

7.2.2 Slope Change

This state is for when the user is detected at the handlebar sensors, the user is walking around with the aid of the walker and the smartwalker has detected a change in slope of the ground.

1. Query the slope of the ground using the onboard sensors
2. Calculate the pitch of the walker (m_walker_pitch)
3. If the pitch of the walker is positive or negative 5 deg to a certain deg
 - a. This state has yet to be finalized, more details to come in the future

7.2.3 Idle


1. User is not detected at handlebar sensors (m_user_in_L and m_user_in_R)
 - a. If walker is not detected at docking station (m_dock_aligned)



MODERN MOBILITY
DRAFT SYSTEM DESIGN
DOCUMENT

Doc: DOC0004
Rev: D

- Walker waits for command from user
- b. If walker is detected at docking station (m_dock_aligned)
 - If battery is less than 100% (m_battery_lvl)
 - charge walker (c_charge_walker)
 - If battery is 100% (m_battery_lvl)
 - Sit idle (details of idle mode to be determined)

	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

8 Undesired Event Handling

Section 7 covered normal operation of the walker, this section will talk about abnormal scenarios and how the walker will handle them. Each section will describe the undesired event and the steps the system will take to ensure that the event is resolved in a safe manner.

8.1 Emergency Stop

Emergency stop refers to when the user pushes/applies the emergency stop or the system determines that there is an emergency and sends a signal at any point while the walker is in normal operation. This option is for when the user

1. Emergency Stop activated
 - a. Brakes applied to motors in a safe manner (c_brakeL_position and c_brakeR_position)
 - b. State of emergency (walkerSafe = False)
 - c. Display prompt for user to cancel emergency state
 - d. Wait for user input
 - i. Safe to continue OR enter manual mode (cancel initial request)

8.2 User Detected During Autonomous Mode

The following scenario is for the walker is in autonomous mode and a user is detected by the handlebar sensors.

1. Walker is in autonomous mode
2. User is detected by handlebar sensors (m_user_in_L and m_user_in_R)
 - a. Brakes applied to motors in a safe manner (c_brakeL_position and c_brakeR_position)
 - b. Walker enters manual mode

8.3 Object Collision with Smartwalker


The following scenario is for when the smartwalker is in autonomous mode and has collided with an object of a minimum size yet to be determined.

1. Walker location does not match expected location
 - a. Brakes applied to motors in a safe manner (c_brakeL_position and c_brakeR_position)
 - b. State of emergency (walkerSafe = False)
 - c. Wait for user input
 - i. Safe to continue OR enter manual mode (cancel initial request)

8.4 Walker has Fallen Over or Very high Incline/Decline

This state is when the walker has fallen over or is at a dangerously high incline or decline. This is a very undesired and not recommended.

1. Walker gyroscopes detect angle greater than 15°
 - a. Brakes applied to motors in a safe manner (c_brakeL_position and c_brakeR_position)

	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

- b. State of emergency (walkerSafe = False)
- c. Wait for user input
 - i. Safe to continue OR enter manual mode (cancel initial request)

8.5 Goal location is unreachable

When the walker is in autonomous mode, it has a goal of either the user or the docking station. In moments when the path is not clear to the destination, the following will occur:

1. Walker attempts to reach destination
2. Several attempts at reaching goal occur (attempt number and duration to be determined)
3. Walker stops attempting to reach goal
4. Go into idle mode
5. Message is sent to the user's phone "unable to reach destination."

8.6 Walker cannot "see"

The following state refers to when the walker sensors are not functioning properly either due to sensor malfunction or a blockage (items covering the sensors) and therefore reporting false data. The walker is in an environment it is not capable of reliably navigation through and thus navigation will cease autonomous mode and the walker will return to manual mode. The user is then able to move the walker to a safe location for troubleshooting.


8.7 Walker is unable to move while autonomously navigating

When the walker is in Autonomous mode and Autonomous navigation with cease and the walker will return to idle mode. The user will be notified via the phone application of the problem.

8.8 Extremely low battery

Since the walker is very dependent on power, when the battery level is very low (low level to be determined) the walker will take the following actions.

1. User will be notified of the low battery state via the phone application
2. The walker will shut down to avoid unexpected behaviour at low power and to avoid damaging the battery

	<p>MODERN MOBILITY DRAFT SYSTEM DESIGN DOCUMENT</p>	<p>Doc: DOC0004 Rev: D</p>
--	---	--------------------------------

References

Rodriguez, J. (2016, November 22). How An ADA Compliant Ramp Must be Built. Retrieved December 21, 2017, from <https://www.thebalance.com/ada-ramp-construction-844440>