Collaborative Robotics on Navigation Platform

A PROJECT REPORT

submitted by

G.Nikhil Chowdary AM.EN.U4ECE16122
Tammana Akhil AM.EN.U4ECE16157
Vignesh S Naick AM.EN.U4ECE16160

under the guidance of

Dr.Rajesh Kannan Megalingam

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

AMRITA SCHOOL OF ENGINEERING
AMRITA VISHWA VIDYAPEETHAM
AMRITAPURI (INDIA)
May - 2020

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITAPURI (INDIA)

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"Collaborative Robotics on Navigation Platform"** submitted by **Tammana Akhil(U4ECE16157),Gutlapalli Nikhil Chowdary(U4ECE1612)** and **Vignesh S Naick(U4ECE16160)** in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in Electronics and Communication Engineering is a bonafide record of the work carried out by them under my guidance and supervision at Amrita School of Engineering, Amritapuri.

Signature of Supervisor: **Signature of Examiner with Name**
Dr.Rajesh Kannan Megalingam
Director, Humanitarian Technology (HuT) Labs,
Asst.Professor, Department of ECE

Signature of Co-Supervisor:
Name of co supervisor
Designation
Department of ECE
Date:

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITAPURI - 690 542

DEPARTMENT OF ECE

DECLARATION

We, **Tammana Akhil(U4ECE16157), Gutlapalli Nikhil Chowdary(U4ECE 16122)** and **Vignesh S Naick(U4ECE16160)** hereby declare that this project report entitled **"Collaborative Robotics on Navigation Platform",** is the record of the original work done by us under the guidance of **Dr. Rajesh Kannan Megalingam,** Director of Humanitarian Technology(HuT) Labs and Assistant Professor, Department of ECE, Amrita School of Engineering, Amritapuri. To the best of our knowledge this work has not formed the basis for the award of any degree/diploma/ associateship/fellowship/or a similar award to any candidate in any University.

Place:Amritapuri **Signature of the Students**
Date:15-05-2020

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF THE UGC ACT, 1956

School of Engineering
Amritapuri Campus, Clappana P.O., Kollam - 690 525, Kerala, India
Ph: +91 (476) 280 1280, Fax: 289 6178, Email: ase@am.amrita.edu
www.amrita.edu/school/engineering

Internship Offer Letter

To Whomsoever it may concern,

This is to inform that Mr. Nikhil Chowdary of S8 ECE, is offered full time intern position at Humanitarian Technology (HuT) Labs of Amrita Vishwa Vidyapeetham of Amritapuri campus, for the period Jan 01 to Jun 30, 2020. Internship period may be extended beyond Jun 2020 depending on the completion of the project work assigned.

Thanking you,

Rajesh Kannan Megalingam
Director | HuT Labs
Asst. Professor | Dept. of ECE,
Amrita School of Engineering,
Amrita Vishwa Vidyapeetham
Amritapuri Campus,
Kollam, Kerala – 690525, India,
Ph: +91 476 2802720

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF THE UGC ACT, 1956

School of Engineering
Amritapuri Campus, Clappana P.O., Kollam - 690 525, Kerala, India
Ph: +91 (476) 280 1280, Fax: 289 6178, Email: ase@am.amrita.edu
www.amrita.edu/school/engineering

Internship Offer Letter

To Whomsoever it may concern,

This is to inform that Mr. Akhil Tammana of S8 ECE, is offered full time intern position at Humanitarian Technology (HuT) Labs of Amrita Vishwa Vidyapeetham of Amritapuri campus, for the period Jan 01 to Jun 30, 2020. Internship period may be extended beyond Jun 2020 depending on the completion of the project work assigned.

Thanking you,

Rajesh Kannan Megalingam
Director | HuT Labs
Asst. Professor | Dept. of ECE,
Amrita School of Engineering,
Amrita Vishwa Vidyapeetham
Amritapuri Campus,
Kollam, Kerala – 690525, India,
Ph: +91 476 2802720

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF THE UGC ACT, 1956

School of Engineering
Amritapuri Campus, Clappana P.O., Kollam - 690 525, Kerala, India
Ph: +91 (476) 280 1280, Fax: 289 6178, Email: ase@am.amrita.edu
www.amrita.edu/school/engineering

Internship Offer Letter

To Whomsoever it may concern,

This is to inform that Mr. Vignesh Naick of S8 ECE, is offered full time intern position at Humanitarian Technology (HuT) Labs of Amrita Vishwa Vidyapeetham of Amritapuri campus, for the period Jan 01 to Jun 30, 2020. Internship period may be extended beyond Jun 2020 depending on the completion of the project work assigned.

Thanking you,

Rajesh Kannan Megalingam
Director | HuT Labs
Asst. Professor | Dept. of ECE,
Amrita School of Engineering,
Amrita Vishwa Vidyapeetham
Amritapuri Campus,
Kollam, Kerala – 690525, India,
Ph: +91 476 2802720

Contents

Acknowledgement

First and foremost, we would like to thank God Almighty for the showers of blessings throughout the project to complete it within time successfully. Our humble pranams at the Lotus feet of Amma, Mata Amritandamayi Devi, Chancellor of Amrita Vishwa Vidyapeetham for blessing us throughout this project. We would first like to thank Dr. S N Jyothi, Principal, Amrita School of Engineering for providing us with the needful.It is delighting to express gratitude toward Humanitarian Technology Labs and the Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, Kollam who furnished us with a profoundly promising workplace and lab facilities, which was the spine for the culmination of this project.Authors thank Dr.Rajesh kannan Megalingam, Department of Electronics and Communication Engineering for all the support and encouragement. We would like to express deepest appreciation towards, Dr.Ravi Shankar, Head of Department of Electronics and Communication Engineering and Bri. Remya, Vice Chairperson ECE Dept, for their guidance, support and encouragement. We are thankful towards the staff of Humanitarian Technology labs for every bit of support for this project. At last we must express our sincere heartfelt gratitude to all the staff members of Department of Electronics and Communication Engineering who helped us directly or indirectly during this course of work.

List of Figures

List of Tables

List of Abbreviations

HSR	Human-Robot Interaction
DOF	Degrees of Freedom
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
Rviz	ROS Visualization
GPRS	General Packet Radio Service
Lidar	Light Detection and Ranging
UV	Ultra Violet
IMU	Inertial Measurement Unit
USB	Universal Serial Bus
MHz	Mega Hertz
DC	Direct Current
A	Ampere
RPM	Rotation Per Minute
VCC	Voltage Common Collector
GND	Ground
TxD	Transmit Data
RXD	Receive Data
DAQ	Data Acquisition
RBPF	Rao-Blackwellized Particle Filter
AMCL	Adaptive Monte Carlo Localisation
HMM	Hidden Markov Model
IP	Internet Protocol
SDK	Software Development Kit
URDF	Unified Robot Description Format
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
m	Mass of the Object
g	Gravitational constant
r	Perpendicular distance from the motor to the object
M	Mass of the object kept on the metal
y	Half of the height of the metal piece
I	Moment of inertia
Y	Young's modulus of the metal

Abstract

According to the World Health Organisation(WHO), about 2 billion people are aged 60 years, 125 million above 80 years, and 65 million are physically handicapped in this World[16]. These people are in need of assistance to do their personal activities. So, we came up with an idea of developing an Autonomous wheelchair with an integrated voice and robotic arm on it. In this project, we aim at developing our own library for navigation, the study of wheelchair dynamics of wheelchair combined with a robotic arm, and validating the results and library using hardware. This overall system aims at performing Human-Robot-Interaction and Cooperation, Navigation and Mapping in dynamic environments, Opening a Door, Exchanging the objects with a Companion.

Chapter 1

Introduction

1.1 Introduction

Society means all sorts of people belonging to different age groups. But in this era where people are living in their own world, elders are the ones who are cared less. There are people with physical disabilities who face difficulty in doing their daily activities. By 2050, the world's population aged sixty(60) years and older is expected to total 2 billion, up from 900 million in 2015 [16]. These people are in need of assistance for their regular physical activities. The normal wheelchairs which are available in markets couldn't solve this problem because of the obvious case that a person shall misuse a joystick or any mistakes without his notice to operate the wheelchair because of the physical and mental challenges. It would be helpful if a person is able to operate the robot using voice. A full-time caretaker to assist the elders and physically challenged is a costly affair. There is also difficulty in hiring caretakers due to labor scarcity. This type of integrated technology can solve this problem to a large extent where a wheelchair can follow the commands from the user as any human caretaker would do.

This also includes the human-robot interaction by communicating with people through voice as a medium or by using mobile phones. By 2020, the total number of smartphone users is projected to reach 2.87 billion[17].

In this modern scientific era Physically disabled and elders face difficulties in carrying out their daily activities at their residences. They need personal assistants to help them with their activities. But labor scarcity is a significant problem in many countries. We introduce Self e 2.0, a wheelchair to assist and interact with people by receiving commands either through speech or through a mobile phone. Self e 2.0 can help them with a glass of water, picking up a book, opening the doors, navigating to their destination, etc. In the case of picking an object, the robot uses the 6 DOF arm to pick the object and gives it to the user. The other advantage of self e is that it can be used for autonomous navigation or manual control by using a mobile app or through voice commands. This project can find applications in other places like airports, households, shopping malls, etc. Thus self e will take care of people and help for their well being by assisting them in all possible ways.

1.2 Literature survey

1.2.1 Topic: Navigation

For the navigation of the robot, the important thing is path planning. To do the path planning, the algorithm presented in the reference 1 is A* algorithm. With the help of this algorithm and grid map technology, the robot navigates to the goal position. This paper also explains about visual graph(V-graph) uses.

1.2.2 Topic: Navigation and voice

The controlling of the robot has changed to another level in the decade. This reference paper 2 explains about the control of robots through voice commands. This method can be used in many applications like autonomous wheelchair, service robot ect, where a person can be comfortable using voice command instead of buttons.

1.2.3 Topic: Dynamics

Reference 3 was considered for studying the dynamics of the mobile manipulator, which used the Lagrange method for calculating the dynamic equations. It also provides good information regarding the interactive control of the mobile base and robotic manipulator which we are trying to do.

1.2.4 Topic: Gripper

For the gripper based on reference 4 which presents a novel way of approaching the grasping by grippers. It presents a quasi static in-hand manipulation with end-effectors that have no degrees of freedom.

1.2.5 Topic: Arm interface

This paper research is based on the people who are having the upper limb disabilities. So the main goal of the research is to translate the natural shoulder and head movements to the movements of the robotic arm. This is achieved by using Body machine interface(i.e by using imu and surface electromyography (sEMG)

1.2.6 Topic: voice and robotic arm

Speech recognition is quite common in personal services like service robotics, but its use for industrial and medical purposes is rare because of the presence of motion ambiguity. This paper presents a design for a speech recognition interface for an HIWIN robotic endoscope holder. New intentional speech control is proposed to control movement over long distances

1.2.7 Topic: Voice and navigation

This paper[7] has proposed a method of integrating voice with the robot by the help of the Robot Operating System (ROS). This integrated system enables the user to navigate to the destination with the human-robot interface as speech. For example, when "FORWARD" is recognized, the Robot moves forward for a predefined distance. No training is required for the people to use this system thereby reducing the manual effort.

1.2.8 Topic: voice and wheelchair

In paper [8] speech is integrated with an obstacle avoidance wheelchair to drive the wheelchair to the desired locations using voice commands. Once the voice command is given by the user (ex: kitchen, living room etc.) the wheelchair will navigate to the destination by its predefined route.

1.2.9 Topic: Gazebo simulation

In this paper[9], the flexibility of a SLAM based mobile robot to map and navigate in an indoor environment is explained. The model robot is made using a gazebo package and simulated in Rviz. It is observed that the robot gives a good response time and also it takes only a reasonable time to cover the distance from the source to destination. As the distance increases, time also increases. In the case of a map with obstacles, the robot will find the shortest path. And if an extra obstacle is introduced, the robot will stop and recalculate the new path.

1.2.10 Topic: Control of Robotic arm

Paper [10] explains the keyboard control and RVIZ simulation of a 6 DOF robotic arm in ROS. Move-it has been used as the control interface. First three joints are used for the position last three joints is used for manipulation of the gripper

1.2.11 Topic: Arm's motion planning

Paper [11] came up with an idea of adapting the Euclidean cluster extraction algorithm to recognize the objects i.e. If the arm can't reach the position it will change its position and grasp the object. Implementation of vision and Arm as a part of a complete system in service robot is dealt in paper

1.2.12 Topic: Android app

In [12] paper researchers discuss how to operate the home appliances using an android app over a wireless network. A support vector machine is used for Speech recognition.

This total automation process happened through GPRS wireless technology

1.2.13 Topic: Dynamics

This paper presents the dynamics of the manipulator on a mobile platform. This paper also presents a good idea of the dynamics of the mobile manipulator in the Lagrange-Euler method, applied on to Mitsubishi PA10-6CE robotic manipulator mounted on a 2-DOF platform.

1.2.14 Topic :Gripper

This paper provides insight over the grasping is influenced by the gripper material and the amount of contact taking between the gripper and the object.

1.2.15 Topic: Arm and Gripper

This reference provides information about the kinematics of the manipulator and the gripper based on the Lagrange method taking the joint angles into consideration. This also provides insight into the mobile manipulator control.

1.3 Problem under investigation

Besides day by day increase in technology, the number of physical disabilities is also increasing parallelly. So there is some sort of a necessity in using this technology for efficient and effective use. Physically disabled and elders face difficulties in carrying out their daily activities at their residences.. These people are in need of human assistance to help them in their daily activities at home, hospitals, shopping malls, airports,

etc.with utmost safety, and care. Some of the elders who stay at home need 24-hour assistance. According to a recent report from WHO nearly 1 percent of the world population needs i.e 65 million people approximately need a wheelchair[18]. It will be very useful if one gives intelligence to a robotic arm via voice and a mobile app. Where those updated wheelchairs can be deployed in many places which is a boon for the wheelchair users where more tasks can be performed using this technology. In hospitals or at home, the patients need personal assistance to take care of them by serving them food, energy drinks, medicines, etc. on time. But labor scarcity and cost are major problems in many countries. So the need for this type of upgrade is increasing day by day. In addition to this, approximately 1 out of every 32,000 births Symbrachydactyly disease occurs. Most frequently, only one hand is affected. The majority of cases of symbrachydactyly can be identified shortly after birth. These people can't move their hands, and in some cases, their hands are paralyzed.

In order to solve this global problem, we came up with an idea of developing A wheelchair with a voice and robotic arm that can be a boon to such people in helping them as a friend, caretaker, etc. Self e wheelchair, a semi-humanoid autonomous robot with highly integrated architecture to make a robust product. The product is being integrated with the features like speech recognition, autonomous navigation with obstacle avoidance, and a 6 degree of freedom robotic arm to make the robot serviceable. So, the implementation of these types of wheelchairs would address the problem of assisting these people without human errors. Self e helps the disabled and physically handicapped people in serving them drinks, fruits, etc.

Chapter 2

Theory

2.1 Introduction to self e

Without any collaborations with foreign universities or companies, the first self-driving wheelchair in India, Self-E was built by the HuT lab of Amrita University. Self-E is unique in terms of controlling a wheelchair. To make ease of control, there are many modes/options by which users can control. The smartphone which is connected to the wheelchair has enabled voice control system by which a wheelchair can be navigated from one place to another and can control a robotic arm for placing or taking things. The other modes are joystick mode which is simple and easy to use for both wheelchair and robotic arm, whereas the touch mode, in which we touch the map screen by which the robot travels autonomously with an obstacle avoidance system. This Android application in a smartphone is not only controlled by the person sitting in a wheelchair but can be done by anyone who has that application by connecting to WiFi.

Figure 2.1: Self-E

2.2 Hardware architecture

2.2.1 Lidar

Lidar is a surveying method that helps to measure the distance from the body to the obstacles. It uses UV rays to map the environment. The instrument fires rapid pulses of laser light at a surface, some at up to 150,000 pulses per second.

Figure 2.2: RPLidar A2

Inside the lidar, it contains a range estimation sensor that over and over transmits

a pulse of light. This pulse of light hits the environment and after that skips off and comes back to the range estimation sensor. By estimating the time of flight that is the time taken for the light pulse to travel out and return back, determine the distance to the object. The range of the Lidar A2 is 0.15 to 8 meters.

2.2.2 Encoder

To convert the rotation of the wheels into electric signals, we used a sensor named rotary encoders. These encoders are used as a feedback signal which can be used to determine speed, direction, position and count. As of now, there are two types of encoders. I.e., Incremental and absolute encoders. By generating a series of pulses, these incremental encoders will work. Coming to the absolute encoders, they have a unique code/value at shaft position. It's like every position of an encoder is different.

Figure 2.3: Rotary Encoders

2.2.3 Teensy v3.2

Teensy V3.2 is a 32-bit micro controller used in a wheelchair. This micro controller is equipped with a 72MHz Cortex-M4 processor and performs complex tasks at a faster rate. The sensors like ultrasonic sensors, rotary encoders, IMU, and Bluetooth slave modules are connected to Teensy v3.2. Serial communication to ROS over USB is established through a rosserial library in ROS. By this rosserial, all the information from Teensy v3.2 is published to different topics which will be used for different purposes. Some of the major features in this teensy are: It is Compatible with Software and Libraries of Arduino and USB can be used for any type of device. By using a single push button, we can program it and can be done using a teensy loader application.

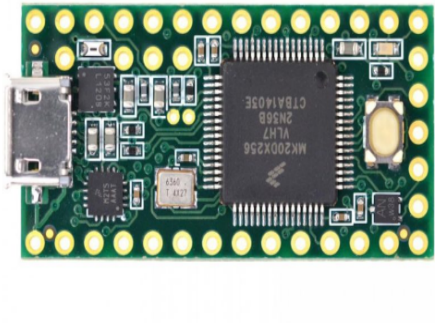

Figure 2.4: Teensy v3.2

2.2.4 PC

For recognizing speech and for path-planning in navigation, we need a high graphic PC for the wheelchair. So in this project, we have used Nvidia graphics GTX 1050 with ROS kinetic installed in it. The teensy 3.2v is connected to PC, such that all

the command generations will be taken care of here. Even the system for the Kinova robotic arm and LiDar is connected to a PC.

Coming to the specifications of Nvidia GTX 1050, it has a base clock of 1354MHz and a boost clock of 1455MHz.

2.2.5 Cytron Motor Driver

A motor driver is a gadget that serves to represent in some predetermined way the execution of an electric motor. A motor controller may incorporate a manual or programmed implies for beginning and halting the engine, choosing forward or switch turn, choosing and directing the speed, managing or constraining the torque, and securing against over-burdens and faults. MD30C is to drive medium to high power brushed DC engine with the current limit up to 80A pinnacle and 30A constantly

Figure 2.5: Cytron Motor Driver

2.2.6 Motors

The wheelchair should be able to have a payload of up to 120kg(both hardware and a human). So in this project we have used high powerful DC motors which work at 24V with a RPM of 4600. These two motors are attached to the back of the wheelchair. By changing the power supply for these two motors from motor driver, the wheelchair direction will be changed.

2.2.7 IMU- MPU6050

To get the data of orientation, velocity and gravitational forces from the various sensors like accelerometer, gyroscope and magnetometer, we use the integrated sensor named IMU. In an autonomous navigation system, IMU plays a major role in identifying the direction in which the robot is moving. For this, the values from the IMU are processed by the computer and the position, velocity, etc. are calculated. The IMU sensor that we used for this project is MPU6050. The power supply to be provided for the MPU6050

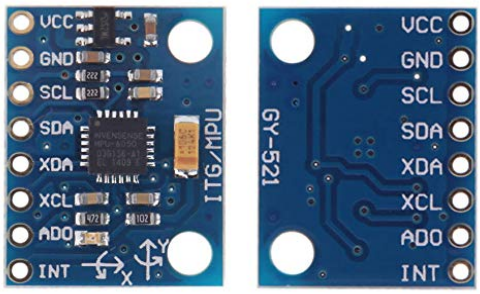

Figure 2.6: MPU6050

should be 3 5v DC power. The power can be directly provided from the teensy3.2v

micro controller. The size is very small such that it can be used in many projects. This MPU6050 module uses standard I2C protocol as the communication mode.

2.2.8 Bluetooth module (HC-05)

HC-05 is a wireless serial communication device that can be used in a Master or a Slave configuration with a packet-based protocol. The latest version of Bluetooth is Bluetooth 5 and it gives more output power than the previous versions. The operating frequency of Bluetooth is between 2402 and 2480 MHz or 2400 and 2483.5 MHz with a 2 MHz guard band. Bluetooth is used to replace wired-connections protocol with a low power low range and low-cost device. It is a 6 pin device with an operating voltage from 3.3V to 5V. Using AT Commands, the settings (Baud rate,Role, Name, Password, etc..) of the module can be changed. The 6 pins of the module are VCC, GND, KEY/EN, State, TXD, and RXD. The KEY/EN pin is used to make the module in command mode. By default, the module is in data mode.

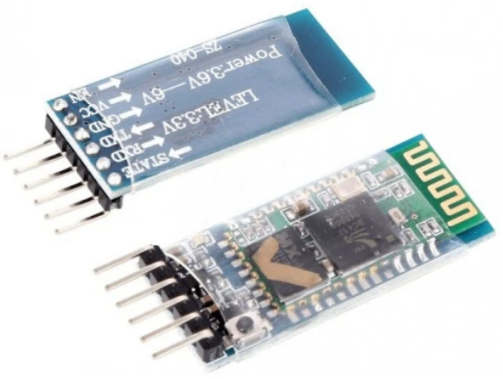

Figure 2.7: HC-05

To transmit and receive the serial data, the TXD of the Bluetooth pin will be

connected to the RXD pin of the micro controller and parallel, the RXD pin will be connected to the TXD of the micro controller. With the help of the EN pin, the data mode and command mode is selected. The data is exchanged between the 2 devices in data mode, while the command mode is used to change the configuration settings in the module. The stages in the data model are PAIRED, CONNECTED, and PAIRABLE.

2.2.9 Kinova Robotic Arm

Robotic arms are well implemented in the industry for manufacturing and are utilized in an outsized array of applications. Their controlled speed, precision, and their load capacity allow them to perform many tasks better, efficient, and faster than humans. They also allow humans to avoid many tasks that might be too dangerous or repetitive. These types of robotic arms typically have a limited number of sensors and decision capabilities preventing them from safely operating near humans. However, in many applications, humans would enjoy an immediate interaction with these artificial arms.

In order for this new collaboration to be possible, the arm must be safe, natural, and have a better understanding of their environment. In order to achieve this, sensors, motors, actuators data, and advanced algorithms are very important. This new era of human assistance is quite demanding in many applications. Such robots may assist people in various tasks such as eating, drinking, moving various objects, opening and closing doors One of the recent best examples we can see many assistive applications had been used for disinfectant spray, serving medicines for COVID 19 patients. It also serves to move the user's hand or feet, to cross legs or to stretch and is even used as a

Figure 2.8: Kinova Robotic Arm

rehabilitation device in clinics.

Actuator Data

JACO actuators use DC brushless motors with Harmonic Drive technology

1. K-75 actuators for joint 1 and 3
2. K-75+ actuator for joint 2
3. K-58 actuators for the wrist

Sensors in each actuator: Optical position encoder, Absolute position, Torque, Current, Temperature, Accelerometer XYZ

JOYSTICK CONTROL

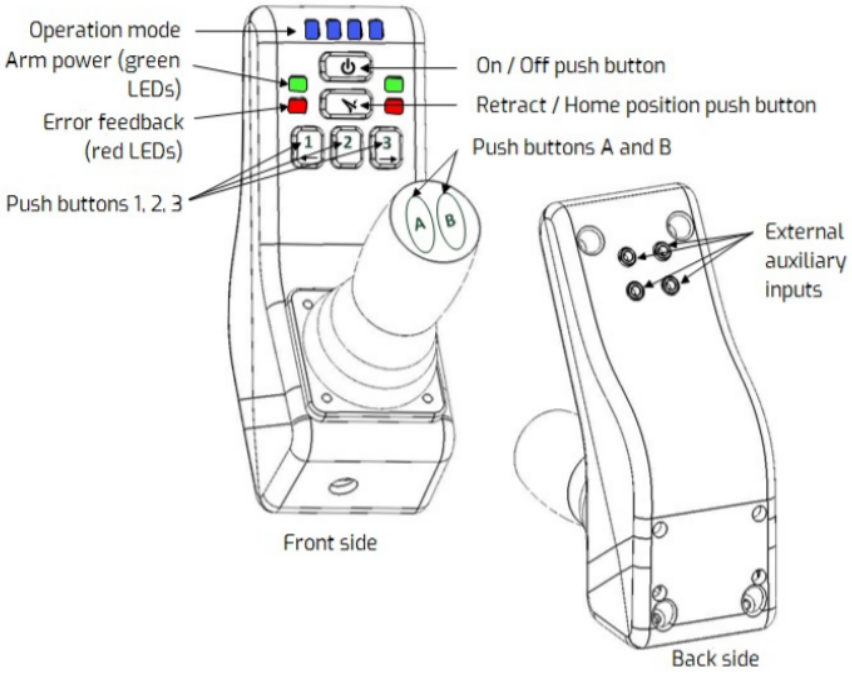

Figure 2.9: Joystick Control

Kinova's joystick enables the operator to control the robot quickly. It has three degrees of freedom stick and 7 buttons. The basic mapping allows performing translations and rotations.

CONTROL MODES

There are basically two basic control modes that are involved

1. Trajectory control: This includes position, velocity and admittance force control

2. Torque control: This includes angular and cartesian torque/force control

S.No:	Specifications	JACO 6 DOF Arm	Unit
1	Maximum Reach length	90	cm
2	Total weight	4.4	kg
3	Maximum Pay Load	2.2-2.6	kg
4	Power supply voltage	18744	7560
5	Average Power	25	W
6	Link Material	Carbon fiber	
7	Control frequency(High-level API)	100	Hz
8	Control frequency(Low-level API)	500	Hz
9	Maximum linear arm speed	20	cm/sec

Table 2.1: Specifications of Kinova arm

Trajectory(position, velocity, and admittance)

User can simply send the position to be reached. Where this can be done in two modes, one is angular and the other is Cartesian. Users are able to control velocity commands in two modes (Translational and orientation). As a safety factor, if kinova didn't receive any command for 10 ms, it would stop at that instant. admittance controllers accept a force as input (which is measured) and react with a displacement. The mapping between the displacement and force is normally explained through a mass-damper-spring model. The algorithm implemented in JACO uses the torque sensors available in every actuator, and the robot can then react to forces applied on any of its links. The best applications include manual trajectory teaching and safety features

- Angular: Each joint position must be specified
- Cartesian: desired position: Translation(x,y,z) and orientation needs to be given

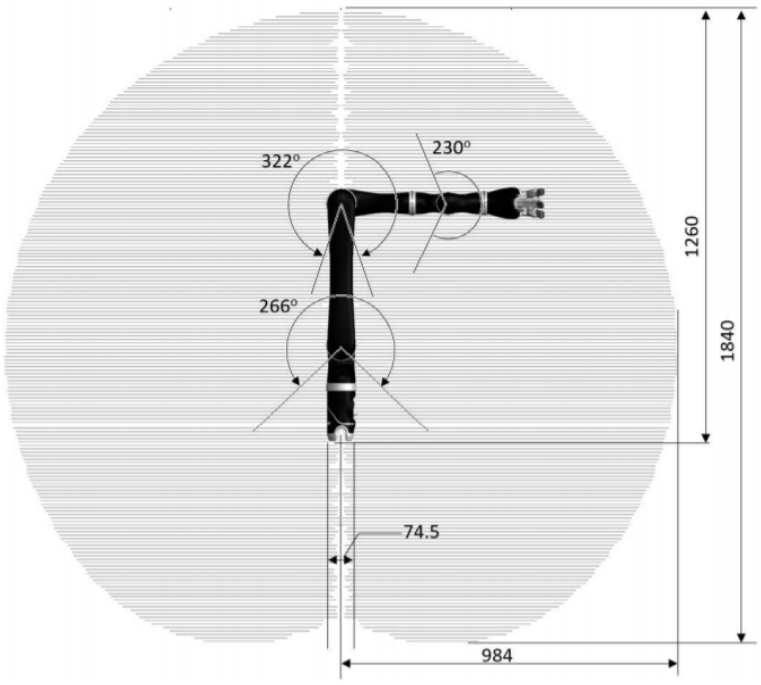

Figure 2.10: Kinova Angles for each DOF

Torque control

Torque control commands can be sent to each and every actuator. Cartesian commands (linear (X, Y, Z) and torque commands (rotation (X, Y, Z) can also be sent to the robot. As a safety purpose, the robot will stop working if it has not received any commands for a certain amount of time. Torque/Cartesian commands can be sent to the actuators through the high-level API and the low-level API. If a user uses a high-level API, the user can get access to some advanced algorithms such as vibration control, gravity estimation, etc.

There is even another facility to change the modes from Trajectory to torque or from torque to a trajectory. That process can be done by giving values to each actu-

ator(which accepts certain limits). As a safety measure if any fault is observed during the transmission it will remain in its previous state

Maximum reach length management

For each and every actuator we will assign a maximum and minimum value. Actuators 2 and 3 are limited to avoid collisions with their links. If in any case actuator exceeds its limitation it will come back to its position control and it will stop working

2.2.10 Gripper Circuit

The current circuit for gripper is restricted only to a servo converted DC motor which is controlled by the motor driver, the commands for the motor driver comes from arduino which processes the command which it gets from the master controller.

Figure 2.11: Gripper Circuit

2.3 Hardware Details

2.3.1 Robot Dimensions:

- Robot size: l x b x h = 120 x 50 x 80 cm3

- Robot weight = 57 kg (Estimation assuming presence of human being)

2.3.2 Power source and components:

- Two pairs of 3S 11.1v 10,000 mAh Lithium Polymer (Li-ion) Batteries.
- 2 Buck converters draw 5 v, and 7.4 v.

2.3.3 Manipulator Arm: (Manufacturer: Kinova)

- Model: Kinova j2s6s300
- Degree of Freedom: 6 DOF

2.3.4 Internal electronics and electrical working

The framework consists of a micro controller(Teensy 3.2) and a PC attached to the robot for processing. Our framework uses the Robot Operating System(ROS) as a platform. ROS is an open-source, meta operating system created mainly for autonomous systems. Due to its capability of node handling, it is the most highly used for Robotic applications. All the data obtained from the sensors is fed to the ROS framework for data acquisition(DAQ) in which a work space is created. The work space consists of different packages for localization, navigation, Speech recognition, Six DOF robotic arm, and gripper. Based on the sampled data obtained through DAQ the corresponding package is utilized. In the setup LIDAR(Laser Detection and Ranging) is considered as the heart of the robot as it provides 2D point cloud coordinates which can be used for mapping and obstacle detection of the robot. IMU(MPU 6050), encoders are used for precision.

2.3.5 Design

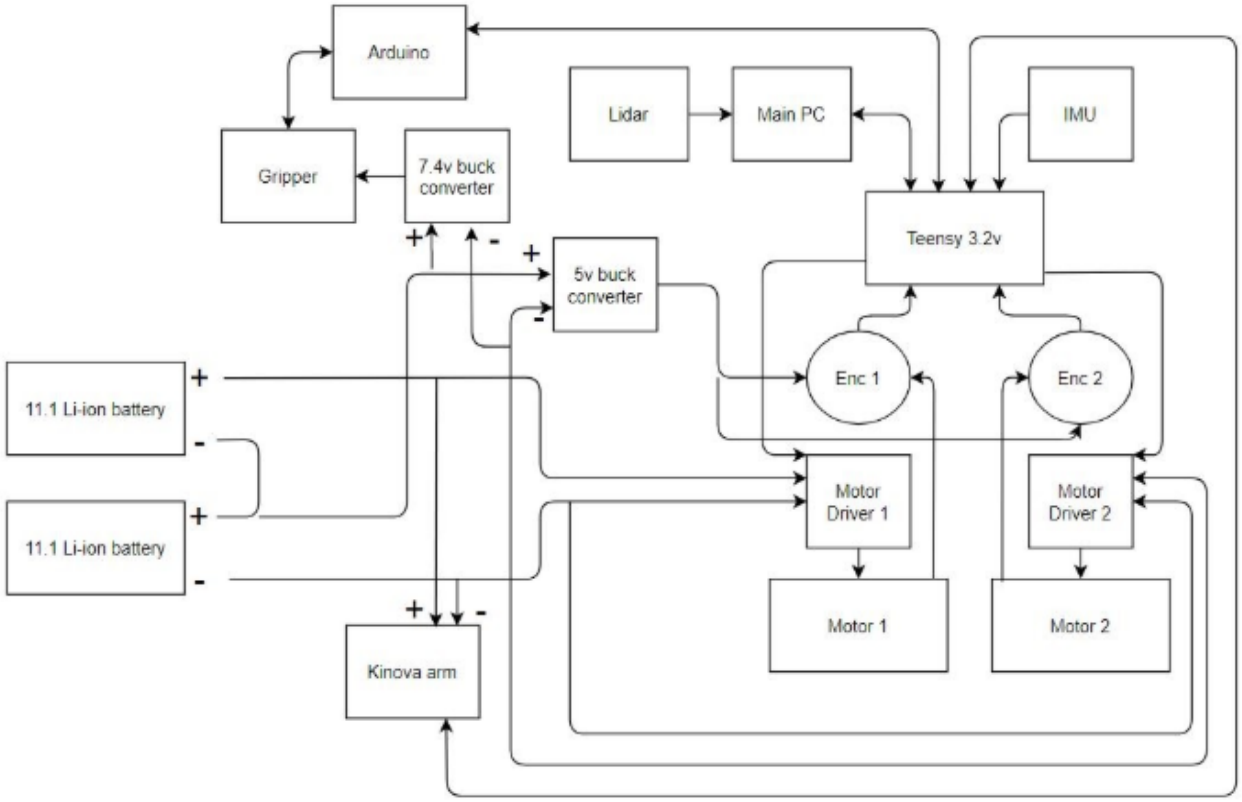

Figure 2.12: Circuit Diagram

The working of the robot can be explained as whenever the user gives the speech input/command the robot hears the command through a microphone and the input is processed on the laptop. Based on the command, the destination and task to be performed are identified and input signals are sent to the Teensy according to the user's input. Teensy sends control signals to the robot. LIDAR which is connected to the robot gives the laser data which is used for mapping the environment. IMU which is used to detect the orientation of the robot and rotary encoders which gives the position

of the robot is connected to the teensy. The values are processed in Teensy and given to the laptop to obtain odometry data. After getting the destination and odometry values the desired path is calculated through ROS signals are sent to motor drivers through Teensy which inturn runs motors. This is the outline of the navigation. The algorithm and clear understanding of the navigation is explained further. Basic operation for an arm is to pick or place an object. The Kinova Jaco(j2s6s300) version was used in the present robot. Kinova sets the standard in safe and responsible robotic technology to perform alongside humans in a structured and unstructured environment

2.4 Software Architecture

2.4.1 Approach followed

Ros is a robotic operating system which is empowering the world in the field of robotics. It is a meta-operating system built on Ubuntu platform providing the services of an operating system like hardware abstraction, package management and message passing between processes. Reuse of codes in Robotics and automation is the primary objective of ROS. This is highly efficient as communication of different nodes is done through subscribing or publishing required topics instead of reading or writing to shared resources thereby reducing the complexity in multithreaded systems.

2.4.2 ROS

ROS is a software framework meant to allow you to write applications that operate robotic hardware (hence Robot Operating System). It reduces complexity while in-

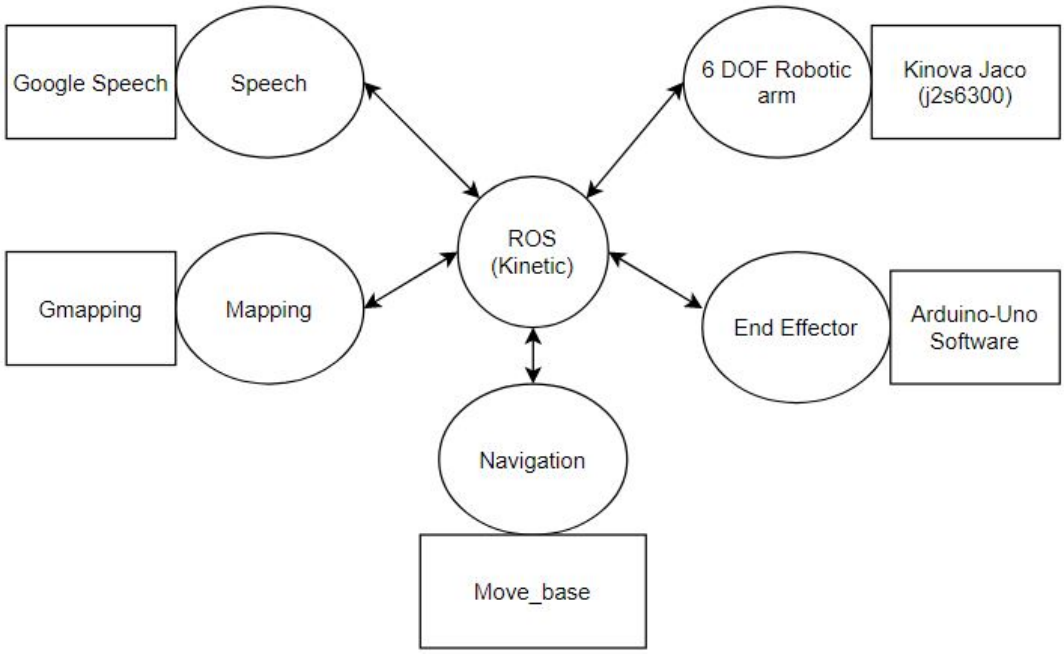

Figure 2.13: Software Architecture

tegrating the hardware. There are many companies using ROS are Pal Robotics, Clearpath Robotics, Fetch Robotics, etc. Ros is a robotic operating system which is empowering the world in the field of robotics. It is a meta-operating system built on the Ubuntu platform providing the services of an operating system like hardware abstraction, package management, and message passing between processes. The reuse of codes in Robotics and automation is the primary objective of ROS. This is highly efficient as communication of different nodes is done through subscribing or publishing required topics instead of reading or writing to shared resources thereby reducing the complexity in multithreaded systems. The basic computation graph consists of concepts master, nodes, messages, topics, services, and bags.

The overall system integration is done using the ROS (Robotic Operating System)

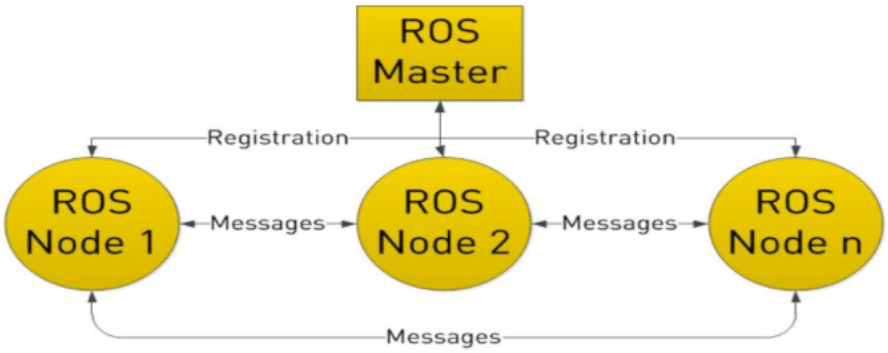

Figure 2.14: ROS Communication

framework, a meta-operating system that provides data transmission between the process, device control, and packet management where the workspace consists of different packages which have multiple nodes, while each node can publish and subscribe on multiple topics across the workspace. Two nodes communicate with each other when they are publishing and subscribing on the same topic. The communication between two nodes involves data or a message transfer. A message can be of different data types such as string, integer, or combination of two or more data types, etc.

Nodes

They are executable files within the ROS package, they use ROS client libraries to communicate with other nodes. Nodes will subscribe or publish to a Topic. Nodes perform the computation of the algorithms and each robot control system consists of many nodes. For example, localization consists of one node, speech recognition possesses one node and many more. Using ROS client libraries like roscpp nodes are written.

Topics

To receive messages, nodes should subscribe to a topic and should publish to a topic if they need to send the message. These topics can be used multiple times at different instances. Topics are named buses through which nodes communicate through messages. They comprise publish/subscribe semantics. Nodes are not aware of the nodes with whom they are communicating with. Nodes subscribe to the corresponding topic for data and publish to the relevant topic. For each topic, many publishers or subscribers can be accommodated.

Messages

It is a data structure containing typed fields and supporting standard types like integer, float, boolean, etc. Nodes pass messages to communicate.

Publisher Nodes

The Node which continuously publishes or sends messages through topics.

Subscriber Nodes

The Node subscribes to the messages published by the publisher node.

2.4.3 Mapping

It is the First step for navigation. To speak in general terms, for a robot to move in an unknown environment, it should know its current pose. To determine the current pose, it needs a good map of the environment. The SLAM(Simultaneous Localisation and Mapping) means generating and updating a map of the real unknown environment. This

should simultaneously keep track of the robot position in the environment. There are some types of mapping like Gmapping, Hector_slam, google _cartographer, core_slam, RGBD slam are some of the methods that can be used to solve this problem. On the usage of sensors, the Slam is differentiated. Here RGBD_slam generates 3D maps using RGB cameras whereas Gmapping and Hector_slam use LiDAR sensors which give laser_data for map generation. In this project, we are using Gmapping for mapping the environment. For this, we are using odometry data from encoders, IMU data from MPU6050, and laser data for generating a map. It creates 2D occupancy grid maps from the laser and poses data collected from the wheelchair. Implementing the Gmapping package is the best and most broadly used method to solve the SLAM problem. Here figure-4 shows what are the inputs for gmapping.

SLAM

SLAM(Simultaneous Localisation and mapping) means generating and updating a map of the real unknown environment. This should simultaneously keep track of the robot position in the environment. This means the current position of the robot should be known for navigating in the real environment. This initially appears as a chicken-egg problem which can be solved by many algorithms. Gmapping, Hector_slam, google_cartographer, core_slam, RGBD slam are some of the methods that can be used to solve this problem. Based on the usage of sensors the Slam are differentiated. RGBD_slam generates a 3D map using RGB cameras whereas Gmapping and Hector_slam use LiDAR sensors which give laser_data for map generation.

Types of mapping

Gmapping, Hector_slam, google_cartographer, core_slam, RGBD slam are some of the methods that can be used to solve this problem.

Based on the usage of sensors the Slam are differentiated. RGBD_slam generates a 3D map using an RGB camera whereas Gmapping and Hector_slam use the LiDAR sensor which gives laser_data for map generation.

Gmapping

Odometry_data obtained from encoders and IMU sensors and laser_data obtained from Lidar are used in gmapping to generate the map of the environment. Rao-Blackwellized particle filter(RBPF) occupancy grid_mapping is the algorithm used for gmapping. In this algorithm, particle-filtering is done thereby reducing the number of particles representing the SLAM posterior. Therefore precise maps can be constructed from resampling as computational exertion is reduced.

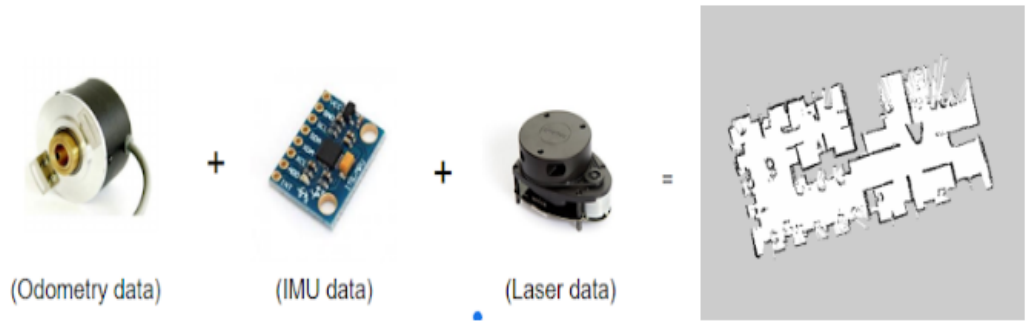

Figure 2.15: Gmapping

Understanding map

Figure-5 looks like the complete map after doing the mapping. We can categorize the map into 3 sections, namely, Free space, Unknown space, and Obstacles. The gray area is known as the Unknown area. In this area, the robot doesn't know what is present in that area. So mostly the robot doesn't go to that area. The white area in the map is a known environment in which it does not have any obstacles. So the robot moves only in this white space based on the dimensions of the robot. The remaining area of the map is the black area. This black area is referred to as static obstacles. So while the robot is navigating, the robot reaches the goal without hitting the black area.

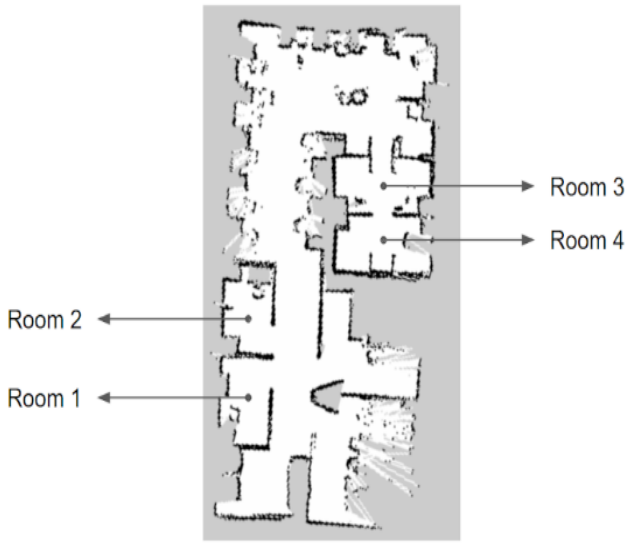

Figure 2.16: Sample Map

2.4.4 Localization

After the generation of the map, for the robot to navigate autonomously in the environment, it is necessary that the robot should have known its current position on the map. To obtain the current pose of the robot Adaptive Monte Carlo Localisation(AMCL) package is used. This determines the probabilistic location of the robot moving in 2D on the generated map. AMCL uses particle filters to track the position of the robot by matching the points on the map to the laserscan_data. Whenever a float is observed, the current position of the robot is determined by the previously obtained odometry_data which is known as Dead Reckoning. The float is pre-compensated by publishing a transform between the odometry frame and the map. AMCL maintains a set of highly probable poses such that whenever the robot moves, the odometry data is compared to the expected poses according to the map. The probability of the pose increases with the consistency of the readings with respect to the map. The probability decreases if the readings are inconsistent. As the robot moves, the poses are obtained based on the odometry estimations. When a robot is simulated in the virtual environment localization is not an issue because the starting position of the robot is always the origin. Whereas in the real world, localization plays an important role because of the dynamic applications.

2.4.5 Navigation Stack

After map generation and localization of the robot, navigation to the desired destination should be accomplished. For the navigation purpose, the move_base package is used

which consists of multiple nodes running in parallel to produce the new sub-goals in order to avoid obstacles based on shortest path algorithms. The different nodes of move_base are:

Data from all the sensors is taken as input to the navigation stack and gives the velocity commands as output. This output is fed as an input velocity to the microcontroller through the topic 'cmd_vel'.

Global planner

This package provides an algorithm for planning an optimum path to the destination by using shortest path algorithms like Dijkstra's algorithm which calculates the shortest path to the destination and is forwarded to the Local_planner.

Local planner

Taking odometry data, sensor information, and the path provided by Global_planner angular and linear coordinates along the path are generated and sent through 'cmd_vel' topic to the controller that drives the robot accordingly.

Recovery behavior

If any dynamic obstacles are found then the robot rotates 360 degrees to find the gap. Though the robot could not avoid the obstacles then the robot stops by giving an error indication.

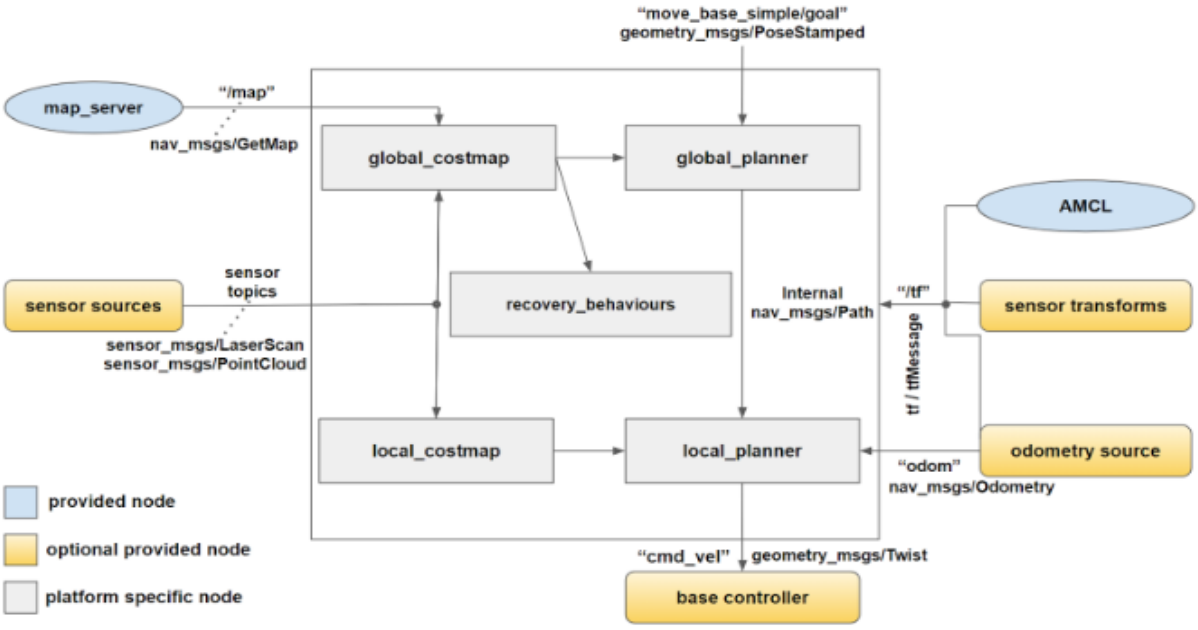

Figure 2.17: Navigation Stack

2.4.6 HuT Planner

To overcome the errors in navigation, we have implemented our own planner with the help of Dijkstra's algorithm with some modifications. The planner has tested in the scenario of dynamic obstacles. For taking the customer to the particular item, we are using way point navigation with the help of move_base in ROS. To simplify the tasks of navigation, we have developed our own systematic planner using Dijkstra's algorithm. The intention of designing this planner was to make the robot navigate safely to the destination. When the robot is given a goal position on the obstacle, the wheelchair always tends to reach that position though its a static obstacle. To overcome this

problem we have developed a planner such that with the help of some test cases, it will decide the type of obstacles and plan for another goal.

2.4.7 Speech Recognition

Speech Recognition is the ability to identify phrases and sentences and convert them into a machine-understandable format. To recognize more natural speech in a real-time environment sophisticated algorithms should be used. The algorithm mainly follows the Acoustic modeling and Language modeling approach. Speech is a composition of words and these words are built with phones. The phone is divided into parts, mostly three in number. The first part of a phone depends on the preceding phone, the middle part is often stable and the last part depends on the succeeding phone. The lightweight speech recognizer library "Google-android" is used for speech recognition. It takes input speech waveform and fragments it into utterances. The preceding word before the silence is considered as a single utterance. These utterances are recognized by trying all the possible combinations of words and matching them with input speech waveforms. The highest probable combinations are chosen by using the concept of models. This matching process uses important concepts as:

Model

Common attributes of the spoken word are gathered by these models. Speech uses a generic "Hidden Markov Model" that describes the black box channel. Here, the Sequence of states changes with a certain probability.

Algorithm for Markov model

In this model, the probability of kth word i depends on the preceding k-1th word. This probability in (n-1) can be observed approximately. The matching process itself is a highly sophisticated process. As every word cannot be compared with models the searching algorithm is optimized by maintaining the highest probable matching variants and extending them with time to give the best matching variants. The acoustic model and Language model are used in Android speech recognition for speech synthesis. To the system to be modelled, which is assumed to be a Markov process with some hidden/unobserved states, we use a statistical Markov model named Hidden Markov Model(HMM). This HMM can be represented by a dynamic Bayesian network. This Model basically starts at the initial state, and it is a collection of states which are connected by transitions. The output symbol is generated in the state, whenever a transition is taken into a new state in a discrete-time step. The transition and output symbols are generated randomly, which are governed by the probability distribution. This HMM model is assumed to be a black box, in which the outputs can be observable but the sequences of states visited over time are hidden, so they are named as Hidden Markov Model.

To the system to be modelled, which is assumed to be a Markov process with some hidden/unobserved states, we use a statistical Markov model named Hidden Markov Model(HMM). This HMM can be represented by a dynamic Bayesian network. This Model basically starts at the initial state, and it is a collection of states which are

Algorithm for Markov model:

$$n(\omega_1, \omega_2, \dots) = \prod_{k=1}^{L} M(\omega_k | \omega_1, \dots \omega_{k-1}) \approx \prod_{k=1}^{L} M\left(\omega_k | \omega_{k-(n-1)}, \dots \omega_{k-1}\right)$$

Figure 2.18: Algorithm for Markov model

connected by transitions. The output symbol is generated in the state, whenever a transition is taken into a new state in a discrete-time step. The transition and output symbols are generated randomly, which are governed by the probability distribution. This HMM model is assumed to be a black box, in which the outputs can be observable but the sequences of states visited over time are hidden, so they are named as Hidden Markov Model.

Acoustic model

It gives the relation between input data and phones. Recording of input data is their text transcriptions are taken for representing the sound in the statistical way that makes up each word for modeling of this speech recognition.

Language model

Restriction of search is done by the Language model. The present state is obtained by the preceding word and the least probable states(words) are stripped off.

2.4.8 Process involved in Speech Recognition

Speech is a complex phenomenon and it is dynamic in nature. Speech recognition systems are complex to build, train data, and use. So, the most efficient way for any person to communicate with a robotic system through speech as the medium of interaction is through a smartphone. According to the "Statista Research Development" survey 2019, there are over 373 million population in India who are using smartphones. It will be easy for an unskilled person to operate a system using his/her mobile phone. Voice is considered as another important part of this project. Processing of speech will be happening in these steps.

1. It should be able to hear the command what user speaks
2. Processing the command(understanding)
3. Based on the command it needs to perform the task

Speech first creates vibrations in the air and reaches the microphone. Now the analog signals are converted to digital signals using an analog to digital converter. In this step system filters are kept to remove any unwanted noise(Generally a Low Pass Filter) and the pace of the signal is corrected. Then the resultant signal is divided into small segments called phonemes in the appropriate language.

2.4.9 Android speech recognition

Speech recognition of the system is developed using an "Android Speech Recognition", android application. The scripting language for the application used is Java. It supports

all languages which google recognizer can handle such as English and German. The system turns active when the ON button in the application is selected. When the command from the user is given, it considers three best possibilities of which sentence it could be and locks the highest probability sentence as the final output using sequential recognition models such as Hidden Markov's Model (HMM) which is supported by Viterbi Algorithm to choose the most probable output.

The Android app will contain the Ip address and port number. That IP-address is our wifi IP address. Which address? The screen of the application has 2 buttons , On-Off, Speak log, sent. When the Recognise button is pressed on the android app, the app starts listening to us through the microphone. This input is published to Pc for further processing. This process is done using sockets. The process of tranferring info between a java and python is done through Sockets. The socket API is used to send messages across a network. They provide a form of inter-process communication (IPC). The network can be a logical, local network to the computer or one that's physically connected to an external network, with its own connections to other networks. The obvious example is the Internet, which you connect to via your ISP. Here the text data will be converted to the string data type.

Socket

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while another socket reaches out to the other to form a connection. Here the homepage for that

android application looks like in the figure shown in android sub section

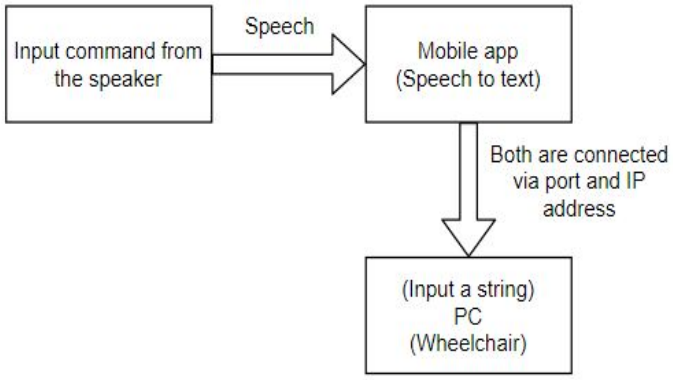

Figure 2.19: Android Speech Recognition

Viterbi algorithm

An algorithm is an approach to finding the most likely sequence of hidden states and the generated sequence of states is called the Viterbi path. The HMM-based approach to Speech Enhancement heavily relies on the Viterbi algorithm. It is a dynamic programming algorithm.

$$\delta_t(i) = \max_{q_1,q_2,\cdots,q_{t-1}} P[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t | \lambda]$$

i.e., $\delta_t(i)$ is the best score along a single path, at time t, which account for the first t observations and ends in state S_i, by induction

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(O_{t+1})$$

Figure 2.20: Viterbi algorithm

2.4.10 Kinova Software

Kinova, being one of the high-end assistive manipulators, uses inverse kinematics for its efficient motion planning. There are available packages in GitHub to control the manipulator that is implemented using inverse kinematics. This method is implemented by using the MoveIt motion planning framework to control the manipulator. MoveIt takes to input the pose values as (psy, theta ,pi, x, y, z) where first three values represents the orientation values whereas x, y, z represents the desired position values. The motion planning is implemented in such a way that the end effector moves to the desired position as the difference between current and desired positions. MoveIt follows the plan and executes mechanism i.e. once the input is published to MoveIt, it plans the path first then executes the motion. The orientation (psy, theta , pi, x, y, z) will vary based on the image parameters such as height, width, size, and shape. The position coordinates (x, y, z) are determined from the YOLO which are further transformed into the manipulator reference system for effective motion planning.

Kinova-Ros-Moveit interface

Step 1: launch the kinova file to start "kinova driver" and to apply some configurations within the arm. Kinova driver is one of the most essential files to run the kinova-ros Package. Inside the kinova-driver file, include folder, Kinova C++ API headers are defined in ../include/kinova, and ROS package header files are located in the kinova-driver folder. kinova-api source file is a wrap of Kinova C++ API. Some advanced accesses regarding force and torque control are only provided in kinova-api.

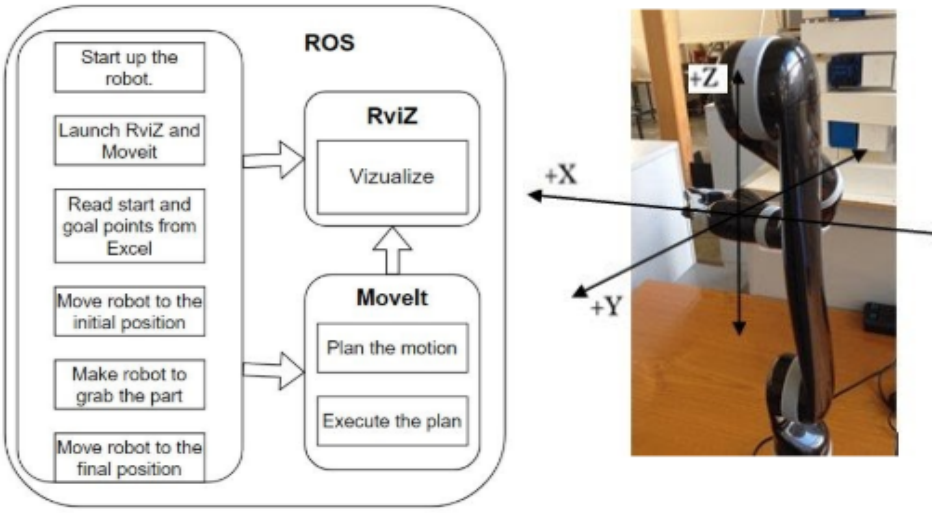

Figure 2.21: MoveIt

roslaunch kinova-bringup kinova-robot.launch kinova-robotType:=j2s6s300

Step 2: launches moveit file of j2s6s300. The demo file can be described as a python script for 'action libs' in both joint and cartesian space.

roslaunch j2s6s300-moveit-config j2s6s300-demo.launch

Cartesian Position Control

FOr cartesian position control, we need to call the node pose-action-client.py in the kinova-ros/src/kinova-demo package. The unit of position control command can be specified by using:- mq:meter-Quaternion , mdeg: meter-degree , mrad: meter-radian. The unit of position is always meter, and the unit of orientation may differ. Degree and radian are in relation to Euler Angles in XYZ order. inverse kinematics will be handled within the robot(with in kinematics it will have the collision avoidance)

rosrun kinova-demo pose-action-client.py -v -r j2s6s300 mdeg – +x +y +z 0 10 10

- "origin" is the intersection point of the bottom plane of the base and cylinder center line.
- +x axis is directed to the left when facing the base panel (where power switch and cable socket locate).
- +y axis is towards the user when facing the base panel.
- +z axis is upwards when the robot is standing on a flat surface.

Kinova SDK development center

It allows getting information from the robot (torque, current, position, etc.) and sending commands (position, velocity, or torque) both in angular and Cartesian mode. The robot controller operates at 100 Hz. The high-level API can be used in C++ applications, both under Windows and Ubuntu

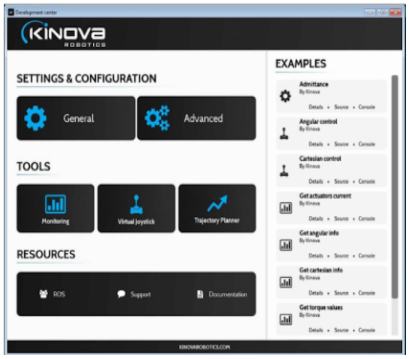

Figure 2.22: Kinova SDK

2.4.11 End Effector

The objective of the system is to meet the demands of voice-based pick and place applications with utmost accuracy. This is achieved with a good end effector or Gripper. Gripper which does the pick and place, event how a simple task has many challenges for achieving this in a good manner. Object parameters, gripping material, friction between material and the object, motor torque all affect the gripper. In a given set, the objects shall differ in their shape, size, weight, hardness, and fragility. The size and shape of an object will vary the orientation of the gripper required to pick or place it. The weight of an object should fall under the payload capacity of the manipulator. The hardness of the object determines the maximum force that a gripper can exert on it, to hold it properly. Next comes the fragility factor, though some objects are strong they are fragile enough to break if the applied force is more than the threshold force. That threshold force is the maximum force that can be applied to an object without breaking. This threshold force changes from object to object based on weight, geometry, brittleness, and smoothness of the object surface. To achieve this, the object details such as shape, size, hardness, and brittleness are also published to the ROS master along with object Id.

The end effector shown in the below figure is attached to the kinova arm for the gripping task. The gripper is designed in such a way that it is lightweight and is able to comfortably carry a payload of 1kg. First, the payload of the gripper was fixed with and around motor torque which was going to be on the gripper was also fixed, the

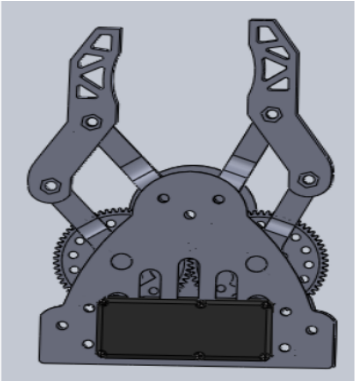

Figure 2.23: End Effector Solidworks Design

length needed for the gripper was taken from using the formulae

Formula 1

$$F = m * g \tag{2.1}$$

(force which is exerted on the gripper teeth for holding the object)

Formula 2

$$torque = r * F \tag{2.2}$$

(torque to be given by the motor,gear ratio should be taken into consideration)

Maximum payload : 1 kg

Gear ratio :(1:3)

The torque required: 5Nm(operating torque)

Link dimensions: 10*5 mm (upper link) and 10*3 mm (down link)

With the above values the length of the gripper was taken as 10cm with an additional 2cm for the motor to be fixed. The Solid Works model and the 3-d fabricated model

both are shown below. The arm on reaching the desired position will publish command to the gripper, and another acknowledgment from the gripper goes to the arm on which the arm starts moving to complete its task. Once the arm has its motion all through the and the gripper thus completing the task, upon which the arm shares a message which ends the task.

2.4.12 Stress analysis of the gripper

Stress analysis of gripper For achieving the autonomy in the gripper we need some sensor for giving the gripper a knowledge about what it is holding. For this purpose, we thought of keeping a transducer which will convert mechanical force acting on it to a small resistance change which is then mapped to a larger value for giving information to the gripper about the amount of force it is going to exert on the object, so the smooth pick and placing of the object happens.

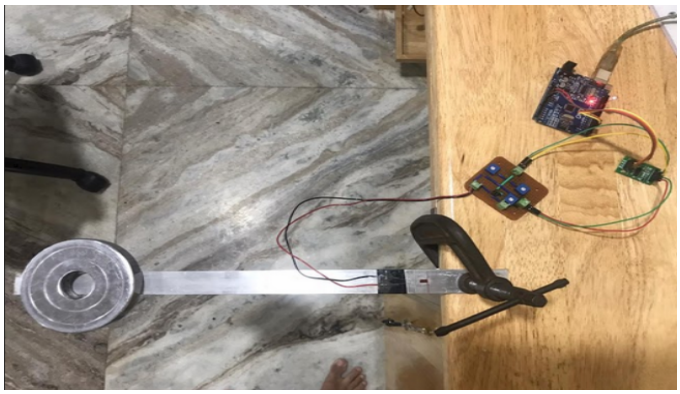

Figure 2.24: Stress analysis of the gripper

Before fixing the transducer we had done a study on the stress that we are going to get from the gripper. For doing this experiment, we took a thick aluminum metal

piece and clamped it to a surface and the transducer was installed at a known distance from one of the ends of the scale and several readings were taken and were compared with the theoretical values. The whole setup with the circuit diagram is shown in the below figures. The result which we get from the HX711 is the 24-bit raw data which needs to be scaled and the result is the strain acting on the point, From there stress is calculated using the below-given formulae.

Formula 3

$$Strain = my/I \tag{2.3}$$

Formula 4

$$Stress = Strain/Y \tag{2.4}$$

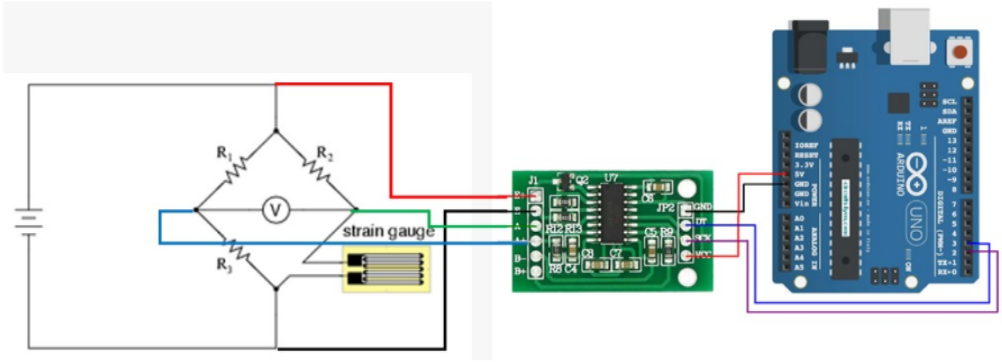

Figure 2.25: Circuit for doing Stress analysis

From the above figure, we can see that the strain gauge is attached to a Wheatstone bridge from where it is connected to the HX711 24-bit amplifier and from there the

amplified value comes to the Arduino. The value registered by the strain gauge is very small that we require some circuit to recognize that much small amount of change in resistance, for that purpose Wheatstone bridge, is used, the analog which is got from the Wheatstone bridge is then fed to an ADC and to a 24-bit amplifier which converts this value to a 24-bit digital value making it easier for the Arduino not recognize. Out of the several tests that is performed some of the important results are shared in the results section

2.4.13 Arduino uno for Gripper

Code for the gripper was written in ros Arduino format and with a rosserial library. The code is having one subscriber and publisher, first, the subscriber will subscribe to the arm topic where the arm passes the information with some parameter to calculate the force to be exerted by the gripper. Gripper then exerts the force and picks the objects and publishes to another topic to the arm which tells the arm that the picking/placing is successful now the arm can do its next task.

2.5 Gazebo simulation

Simulating a robot is important before testing the real hardware. It will be very helpful in debugging the code that we have written. So, we have taken Gazebo as the simulated software which is integrated with ROS. With the help of this software, we can manipulate or observe the changes in the system. By doing this we can check the performance and stability of the robot and change the source code according to it.

With the help of plugins, we can attach different sensors and can work on them, the same as in the real world. We can observe different properties like the moment in the joints, drive mechanisms, path planning, etc. To this project, we have created a virtual environment and tested an integrated wheelchair including mapping and navigation in the Robot Operating System.

To Control the wheelchair virtually, there are some steps to be followed.

1. Design the wheelchair in a Solidworks platform.

2. Exporting the SolidWorks to URDF.

3. Implementing the exported URDF to Gazebo.

2.5.1 Designing the wheelchair in Solidworks

When we want to simulate the robot, the main task is to create the 3D structure of it. So to implement this, we have used the software named SolidWorks. Solidworks is a program in which we can design the 3D structures with a solid modeling computer-aided design(CAD) and computer-aided engineering(CAE). It mainly runs on Windows only. With the help of this software, we created the structure of the wheelchair including a Kinova arm and gripper. The figure is the screenshot model of the wheelchair. So the designed wheelchair was designed into each section for the ease of control. This wheelchair consists of two differential drive motors, two caster wheels, one Lidar, and a robotic arm with a gripper.

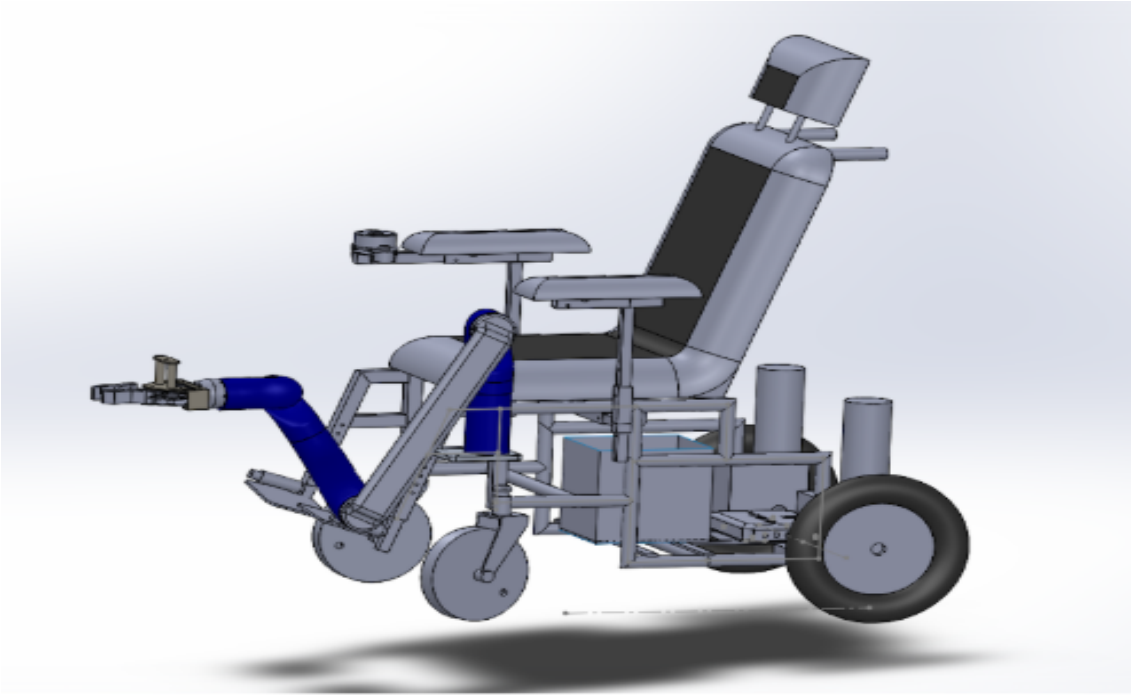

Figure 2.26: Wheelchair Design in Solidworks

2.5.2 Exporting the SolidWorks to URDF

URDF stands for Universal Robotic Description Format. With the help of this plugin, we can easily convert the SolidWorks part files or assembly files into URDF. This exporter comes with the different files like meshes that represent the robot parts, textures which represents color and .urdf file which consists of an origin and join positions. The joints, links, and axis can be automatically chosen by the exporter. In the exporter, we have two types of links, parent and child links. The parent link means the main link and the remaining links come under the child link. We can attach many child links to a parent link. The order we have taken for generating urdf is given in the figure. So we have taken the chassis as the main parent for the whole robot. To that chassis, we

have attached some child links. For some of the child links, we have attached other child links. For the parts like right_casterwheel_mount, the chassis acts as a parent link, and parts like right_casterwheel, right_casterwheel_mount act as a parent link. As we are using a 6 DOF robotic arm , the URDF data will look like the above figure. The link_1 will be the main link for the whole robotic arm and the remaining links add up and be as child links. By feeding all the data regarding axis and color, the URDF will be generated.URDF will generate 4 main folders, i.e.textures, meshes, urdf, and launch files. In the urdf file, the whole information about each part is provided.

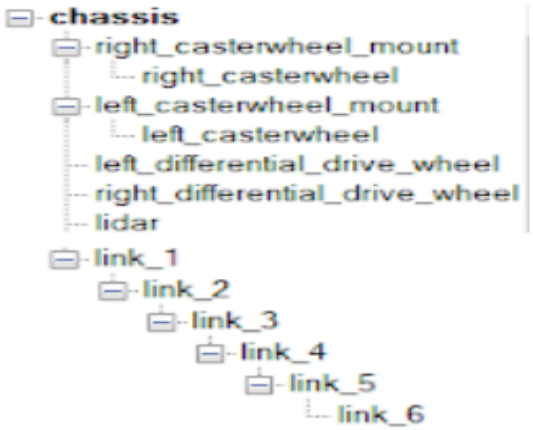

Figure 2.27: Parent - Child link

2.5.3 Implementing the exported URDF to Gazebo

After creating the URDF in windows, export them to Ubuntu for working on it. In the .urdf file, we will be adding plugins for the sensors we require. The plugins we added

in our .urdf file is for lidar and differential drive motor.

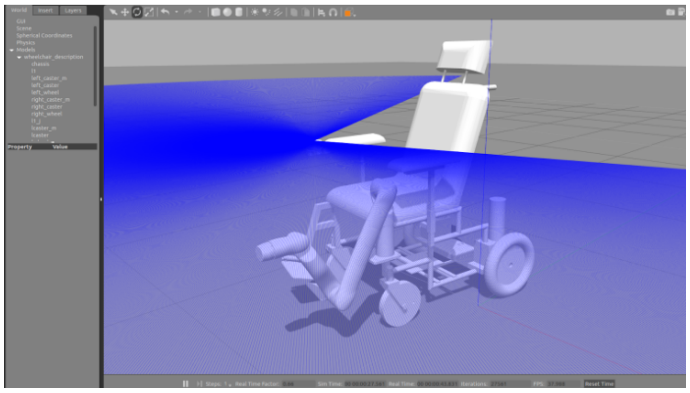

Figure 2.28: Wheelchair in Gazebo

2.6 Integration

2.6.1 Speech + Navigation

Here we mainly focus on integrating speech with self navigation. Befour that we had a module in a smartphone where the wheelchair automatically navigates to one place by touching the map. But here, instead of touching the smartphone, we directly give commands through voice. This feature in this system makes our wheelchair unique. The person operating the smartphone first needs to activate the voice command navigation module, such that the remaining modes will be deactivated. The smartphone is connected with the master PC through WiFi. In the PC we have a python code, in which all the voice recorded from the phone will be published here. The PC is equipped with a dataset of keywords, a set of locations and the 2D coordinate system of each location, which is extracted from RviZ with respect to origin.

The Pc also has a set of verbs like come, go, assist in database. After the user

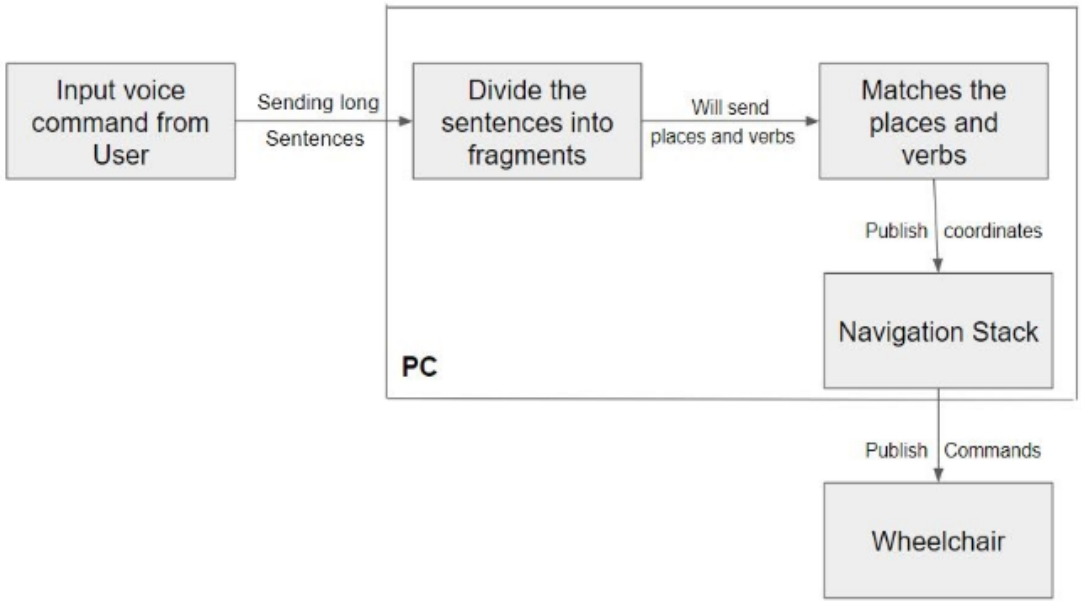

Figure 2.29: Speech-Navigation Flowchart

gives commands to the wheelchair in the long sentences/format, the programme will divide the long sentences into small fragments like place and verb. Then the places and verbs were sent to another file where it matches the prebuilt places and published the 2D coordinate goals to move_base. The goals published to move_base are subscribed by navigation stack. Now the algorithm chooses the best path to reach the desired location and the robot is directed to the goal with the obstacle avoidance system.

Explanation through an example: COMMAND - Go to the kitchen This is divided into fragments like “Go”, “to”, “the”, “kitchen”. Now each of these words is compared to the words in the database. Go gets matched to a database of verbs and the Kitchen gets matched to the database of places. destination coordinate is published to the

Figure 2.30: Android Recognition

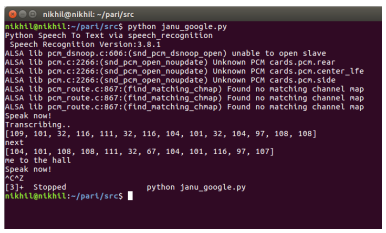

Figure 2.31: Publishing Co-ordinates

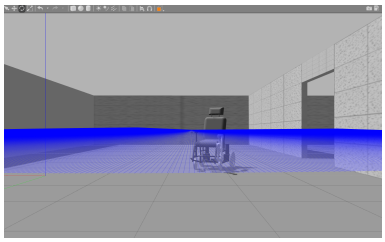

Figure 2.32: Wheelchair in Home environment

S.No:	Command	Action
1	Left	Moment of end effector to its left(i.e 5cm)
2	Right	Moment of end effector to its right(i.e 5cm)
3	Up	Moment of end effector to its up(i.e 5cm)
4	Down	Moment of end effector to its down(i.e 5cm)
5	Front	Moment of end effector to its front(i.e 5cm)
6	Back	Moment of end effector to its Back(i.e 5cm)
7	Grip	Starts Gripping
8	Hold	Starts Holding

Table 2.2: Testing of Speech Recognition

navigation stack.

2.6.2 Speech + Robotic arm + Gripper

Speech recognition of the system is developed using "Android Speech recognition". The trained data used for speech recognition is stored in the android package. The scripting language for the application used is Python. It supports both English and German language. The system can be even accessed using Mobile Phone away from the wheelchair. Speech can be recognized by using the mobile's microphone.

When the command from the user is given, it considers all the best possibilities of which sentence it could be and locks the highest probability sentence as the final output

using sequential recognition models such as Hidden Markov's Model (HMM), and this model is supported by Viterbi Algorithm to choose the most probable output. Now the output obtained is subscribed to the Master node (ROS node), depending on which task is being implemented

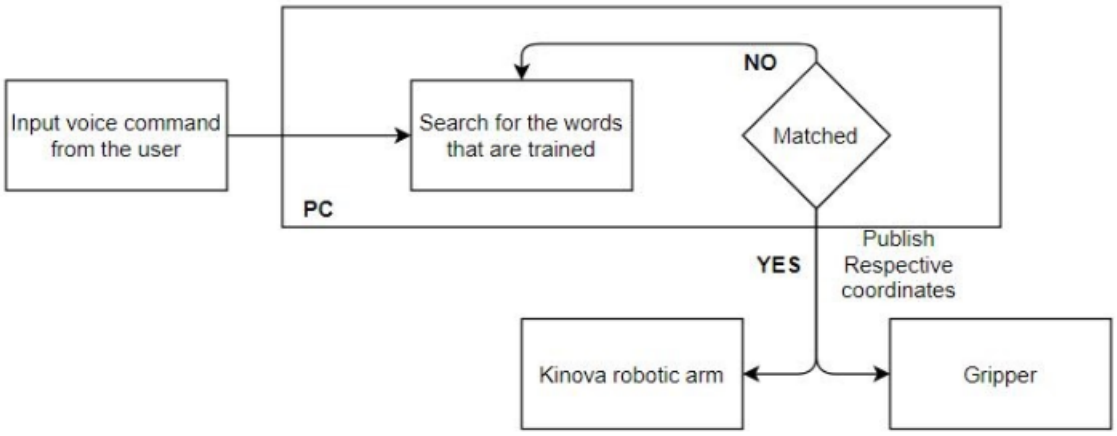

Figure 2.33: Flowchart for the Speech + Robotic Arm + Gripper

In this work "pose stamped" is a type of message that has 7 floats, a string, time and uint32 data types. Whenever the speech input is given, the microphone(inside the mobile) captures the input audio stream and sends it to the PC for processing. The processing of the audio signal is done through the "Android Speech Recognition" Package. (The data is published through the topic /speech node). This word is subscribed to node arm and the result of the processing will get published as the commands to the python file. i.e. Based on the given speech input the text data which is in the string format is compared with the predefined intents and entities. If the intents and entities get matched the predefined command will be published to the kinova arm.

NOTE: The limit of going front and back can be changed easily just by changing the variable in the python file available.

2.6.3 Robotic arm + Gripper

Kinematics and control of the manipulator mobility use some distinctive algorithms, which describe the freedom that it has in the space where it is located, are being used. Poseclient and motion plan analysis methods that use direct and inverse kinematic were implemented, this way allowing to describe the behavior and state of each joint in the given work space of the arm. Together with a control algorithm, we can accomplish simple and even complex tasks, as the spatial information on every movement of the arm can be captured through the sensors and are comprehended, so it can make decisions for new tasks regarding manipulators. For the effective gripper holding throughout the task, there will be effective communication between the arm and gripper without any fail. On reaching the desired point, the arm publishes an acknowledgment to the gripper. Parallelly, Gripper will also be publishing an acknowledgment throughout to

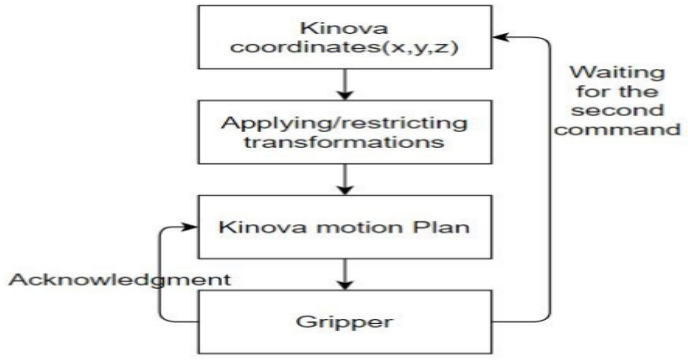

Figure 2.34: Flowchart for Robotic Arm + Gripper

the robotic arm, this acknowledgment carries the information on whether the gripper is in pick/place state. Currently, it is using the time delay pattern for holding the object. Several tests on the timings had been taken for a proper hold of the object. All readings are noted from its home position and are fed to the controller as a delay to the code for holding the object. As soon as the arm gets the locations, it will publish the coordinates to the gripper.

2.6.4 Android App

Adding to the convenience of the user an android app is developed having features of; switching to different modes for the navigation, speech-enabled navigation, and control of both vehicle and the robotic arm. The android application is Bluetooth enabled and gets activated on connecting it with the Bluetooth provided on the wheelchair. The user is allowed to choose the mode of operation once the app is activated. Users can change the mode from the given options of Fixed, auto, manual, and voice-command navigation. Each mode is unique in its features given. The communication happens between the Bluetooth and android applications which are then fed to the microcontroller for the further process. The app can be installed easily like other applications in android phones and tablets. Bluetooth will provide wireless communication between the system and the android phone.

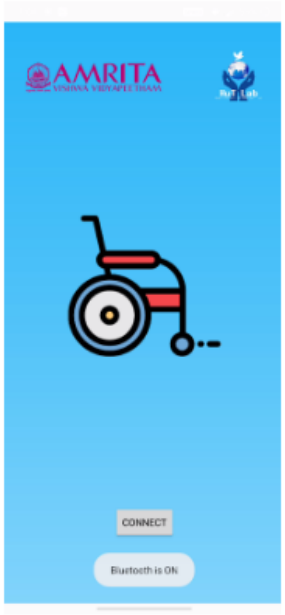

Figure 2.35: App opening Display

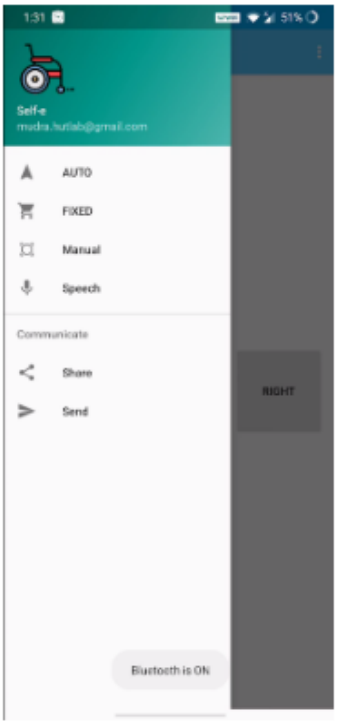

Figure 2.36: Options Screen

MODES OF OPERATION

Manual mode

The manual mode has the control of the wheelchair with predefined commands provided in the app. This mode is considered to be the safest among all the modes which is a benefit. The app is provided with commands like straight, back, right, left, stop commands for the operation of the wheelchair. Unlike other modes, this doesn't require any ROS for its working. In situations where the wheelchair goes out of control and all, the user can switch to this mode and control the vehicle.

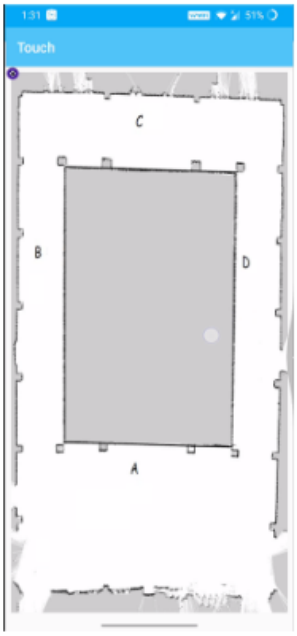

Figure 2.37: Auto Navigation

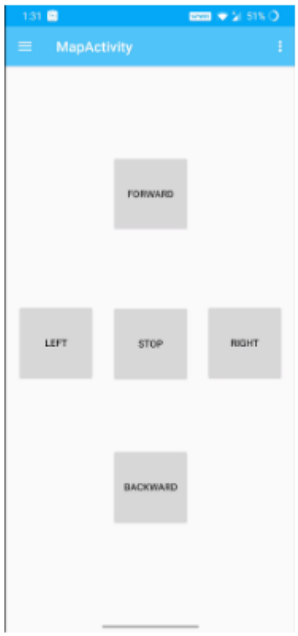

Figure 2.38: Teleop Control

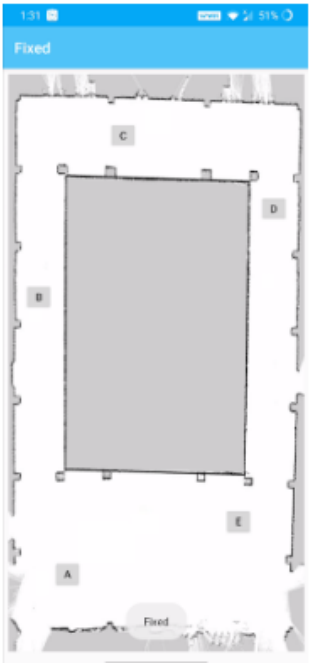

Figure 2.39: Fixed Navigation

Fixed Destination mode

In this mode, the user is provided with the map of the environment and certain predefined set of destinations in the map which are previously identified. The user can touch on these points and the destination coordinates will be passed to the wheelchair and the wheelchair will navigate to the destination. Although fixed destinations are provided, the path planning and all happens on a real-time basis, this mode will also respond to sudden and immediate obstacles. This mode is gives restricted to do navigation only on the predefined point and won't take any other destination coordinates for its operation.

Auto mode

This is an advanced mode where the user is able to move to any destination location of his wish from the map which was previously uploaded into the application. In this mode when the user clicks over a certain destination point, the coordinates from the map are sent to the sub-controller via Bluetooth and are mapped to the corresponding point on the real map. Once the point is mapped, path planning and navigation take place. This model is robust in overcoming the sudden obstacles, the machine is taking full control of the navigation. On giving the destination point each time the navigation and path planning happens from the first.

Voice command mode

In this mode, the user is able to give the voice commands and control both vehicle and the arm. Vocal commands given by the user are identified by google and are translated to text, the translated text is then identified and sent to microcontroller from act according to the recognized text. Here the user is given a certain set of words which he/she when recited, will be captured and processed. The predefined set of words will be mapping to unique and predefined locations in the environment that the user can access. The words also contain worsens like left, right, up, down, grip and release for smooth control of the robotic arm. The user is able to pick and place the action in this mode without any joystick.

Totally there are four steps that are involved in operating this speech app. In first step we need give a username in char client and that is named as selfe. Same IP address

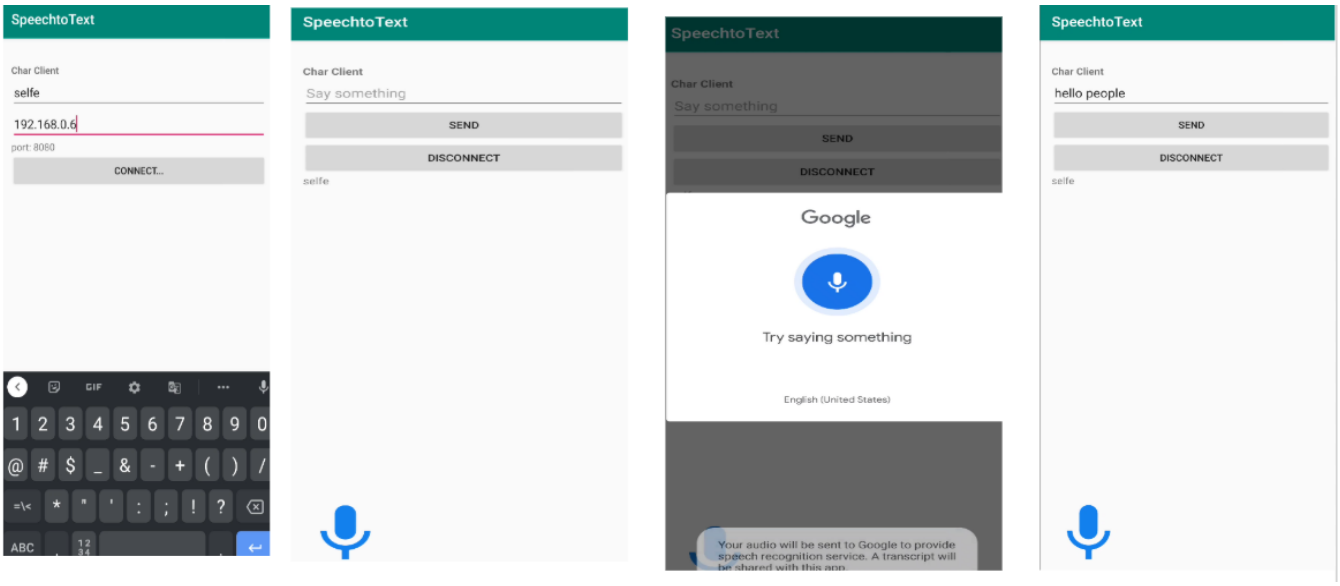

Figure 2.40: Voice Command Screen

and port numbers must be assigned to both java and python files available. This uses TCP/Ip protocol. once you click on connect after entering valid details you will taken to second picture in which process of speech recognisation starts once user clicks on mike button. Once mike button is clicked by using the offline google recognisation library it will converts speech to text. The text data will be printed on charclient assigned. if user is satisfied with the output data he can send that data to pc by clicking on send. Then the processing will take place based on the input command given by the user.

Chapter 3

Summary, conclusions and scope for further research

3.1 results and discussion

For the implementation of our technique we tested our robot in our research lab known as HuT lab

3.1.1 Hut Planner Vs Carrot Planner

To avail the advantages of this HuT planner, we have compared the results with the Carrot planner on the Gazebo platform. This carrot planner places a legal goal beside the obstacle when the given goal position on the obstacle of the global map. If you place a static obstacle on the local map and give a goal to the robot, this planner will fail to provide the goal. Here comes the difference to the HuT planner, where the legal goal is placed with the help of the local map. Here are the comparisons of HuT planner with Carrot planner.

S.No:	Scenario	carrot Planner	HuT Planner
1	When you publish goal for the first time	works	works
2	When you place an obstacle after publishing the goal	not works	works

Table 3.1: Comparison of HUT and Carrot Planner

Carot Planner

It automatically places a legal goal, if you mark a goal on an obstacle which are already on the map

HuT planner

We need to give a maximum limit for publishing goals if it struck up near a goal.

3.1.2 Speech and Navigation

Here we have tested our integration of speech and navigation by generating some commands. The experiments are:

1. Initial position: Kitchen

   Final position: Dining area

   Input speech command: “skedaddle to Dining area”

2. Initial position: Bedroom

   Final position: Dining area

   Input speech command: “please go to Dining area”

3. Initial position: Dinning area(with obstacles)

   Final position: Hall

   Input speech command: “skedaddle to Hall”

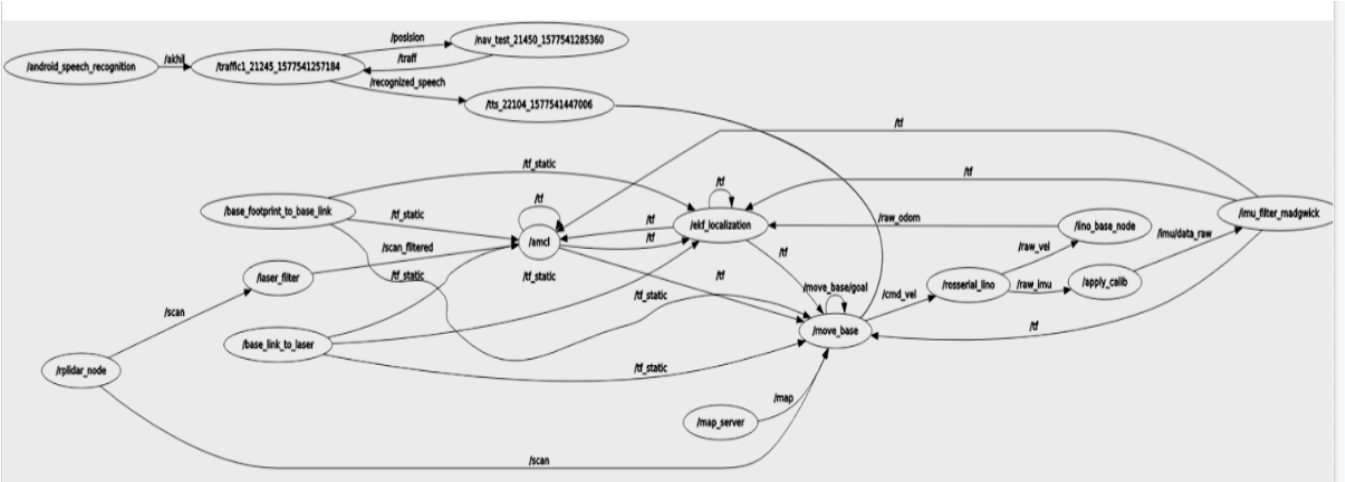

Figure 3.1: Rqt plot for speech navigation

4. Initial position: Kitchen(with obstacles)

   Final position: Dining area

   Input speech command: “skedaddle to Dining area”

5. Initial position: Hall

   Final position: Bedroom

   Input speech command: “Can you please go to Bedroom”

trails	command recognition	Navigation initialization time	Time taken to reach destination	Distance error	Orientation error
1	No	–	–	–	–
2	Yes	6 sec	108 sec	–	17°
3	Yes	5 sec	–	–	–
4	Yes	4 sec	102 sec	–	25°
5	Yes	5 sec	115 sec	20 cm	14°

Table 3.2: Test results for speech based navigation in Gazebo

Table 3.2 shows the observations noted while testing the whole system in the Gazebo environment. In English many words have multiple pronunciations. If a user pronounces any word in a different ascent other than trained ascent, the robot mightn't recognize the word. As a result, when ascent is changed the robot failed to detect the speech input in expt1. In expt3 though the input speech is recognised, because of the many obstacles in the arena the robot could not find any recovery behaviour and stops path planning by giving speech output "No recovery behaviours found." In the remaining experiments the robots detect the speech and navigate to destination. The error between 10 to 12 cm is observed in driving the robot to the desired goal as the tested arena has more obstacles which would impact the error. Slight delay is observed because of processing time for decision making and gmapping algorithm.

3.1.3 Testing of Manipulation and Gripper

The testing of this system requires a simple setup. The kinova arm with its end effector is fixed at the end of a table. The objects were placed on another table in front of the system setup where the trained objects were randomly changed in position and sometimes replaced with other objects. When a user inputs a valid coordinate, the task

starts as explained in the integration section.

Expt.	Object placed	coordinates	Reach	Motion Planning
1	bottle	(0.44, -0.21, 0.02)	0.4879	done
2	tomato	(0.32, -0.61, -0.31)	0.7553	fail
3	tomato	(0.34, -0.17, 0.16)	0.4124	done
4	cup	(0.164, -0.4, -0.31)	0.5319	done
5	bottle	(-0.06, -0.37, 0.17)	0.4115	done
6	cup	(0.57, -0.26, -0.2)	0.6576	done
7	cup	(-0.09, -0.15, 0.06)	0.1849	fail

Table 3.3: Test results for Manipulation and Gripper

The arm couldn't reach the recognized objects. This is because the reachlength of the arm is limited to 70 cm which is calculated from its 2nd joint. Hence thearm didn't reach the object in the experiment 2. Whereas in experiment 7, the armdidn't reach the object because the transformed pose value is within the self collision avoidance region of the arm.

Observations

As we are applying the restrictions on each joint and by giving a orientation based approach they are some of the observations done in this experiments. The arm is reaching object only when it is planned for two times. thereis some of the delay observed in the system there may be so many reasons one is a bulk package from kinova and other is due to the pc that is used

Start unix time	Stop unix time	delay(S)
1579843664.839909	1579843665.308468	0.4685599804
1579843875.584993	1579843876.12299	0.5379998684
1579843922.499685	1579843922.947342	0.4476599693
1579843972.607343	1579843973.090752	0.4834098816
1579844010.381798	1579844010.896823	0.5150301456
1579844043.651302	1579844044.129209	0.4779000282
1579844097.110755	1579844097.610489	0.4997301102
1579844135.532619	1579844136.007631	0.4750201702
1579844348.487575	1579844348.962976	0.475399971
1579846327.231606	1579846327.704246	0.4726400375
1579847832.516898	1579847833.039101	0.5222098827
1579855916.932142	1579855917.481708	0.5495598316
1579855978.059653	1579855978.546975	0.4873199463
1579928227.587549	1579928228.097825	0.5102801323
1579931352.395496	1579931352.901928	0.5064301491
1579934292.199474	1579934293.059497	0.8600199223

Figure 3.2: Delay Observed

3.1.4 END EFFECTOR

The aluminium fabricated gripper for the above discussed solid work model is shown below

The tests which were carried on for the sensor development of the gripper gave a deep knowledge on the scope of using strain transducer for the detection of the force, along with this we will also able to find some issues that can come across the sensor development in future which is; it was observed that the sensor value changed in a small range when the gripper was holding the object for a longer time. The effect of noise from due to the other electronic components on the strain gauge was also noted. Below we share with you some of the results we obtained in testing the strain gauge sensor and its interpretations.

Figure 3.3: Alluminuim Fabricated Gripper

The above figure shows the sensor values in raw format taken from HX711. The first spike of values is for the 0.8 kg weight. The second spike represents the 1 kg weight and the 3rd spike represents the 1.8 kg weight. The results were satisfactory for the weights it gave a proper and constant value from the sensor for the change in time. But when the same weight was held for a long time we can see that the sensor value decreases gradually, This is clearly visible in the below graph.

3.1.5 Testing Speech Recognition

In English many words have multiple pronunciations. If a user pronounces any word in a different ascent other than a trained ascent, the robot sometimes doesn't recognize the word.

Ex: Grip word is recognized as group

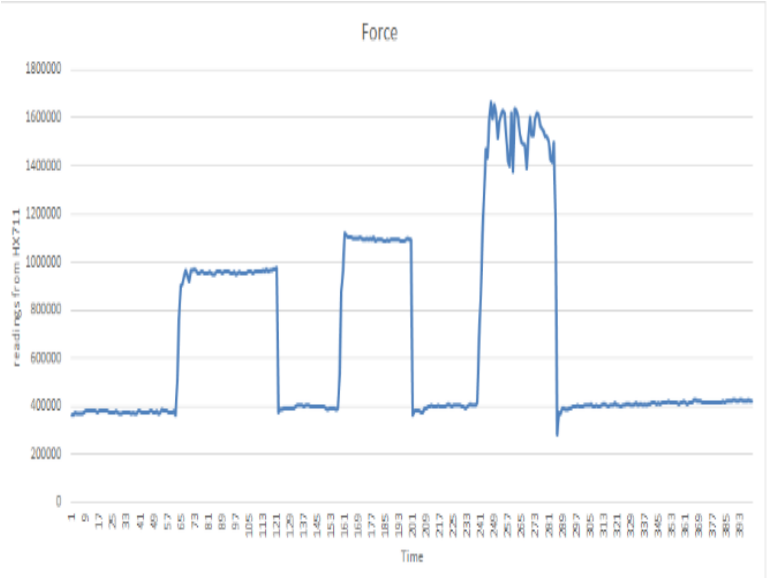

Figure 3.4: Raw readings from HX711 for 0.8kg, 1kg and 1.8kg

S.No:	Input	Output	Processing Time
1	“hello”	“hello”	1.3sec
2	“Go to the kitchen”	“Go to the kitchen”	2.4sec
3	“Go to the room”	“Go to the room”	2.3sec
4	“start”	“start”	1.2sec

Table 3.4: Testing of Speech Recognition

In the remaining experiments the robots detect the word which is input from the speaker.

3.2 conclusions

In this work, we present a complete working of the wheelchair mounted with robotic arm and gripper. The project was aimed at providing assistance to people with acute medical illnesses and disabilities. The prototype for the above-mentioned model is made with speech integrated onto it, so that the robot is able to accept vocal inputs

Figure 3.5: Raw strain gauge reading with the reading in y-axis and the time on the x-axis for 1kg weight

and act according to it. For better navigation, a new planner named HuT Planner is developed and tested, which made the navigation much robust to immediate obstacles. An android app is developed, making it more user friendly with advanced features of speech integration on the app. The gripper prototype is developed and is checked for mechanical stability. Sensor development for force detection is completed partially with the observation of results and outcomes from the sensor studied and discussed with challenges involved in it.

3.3 future works

The project can be further extended by having improvisations in the approach we took to achieve the purpose. Adding a camera for more applications like the camera can

Figure 3.6: oreintation of type 1

be integrated with the android app so that the user can click on the tablet, and the location will be identified which when then sent to arm, and arm will go to the location and perform the tasks as instructed by the user. The accuracy of the robot navigation to the desired location is tested in different environments and conditions where we observed that accuracy varies with the change in the Imu model. By using some sophisticated algorithms and machine learning techniques, the robot can be trained and tested simultaneously. In the next phase, we are going to make the 3D map for navigation purposes by incorporating the Visual data with the Odometry data to achieve high accuracy navigation.filters can be added for the microphone for the efficient output. For the arm, part process and code can be developed to control each dof of the robotic arms separately so that only certain dof can be enabled for performing certain tasks. For the gripper, we see from the result of developing a sensor that the reading can only be

Figure 3.7: orientation of type 2

accurate with a dedicated PCB for the strain gauge. A novel design can also be made for the gripper finger as well the links which will make possible the detection of even very small light weight items.

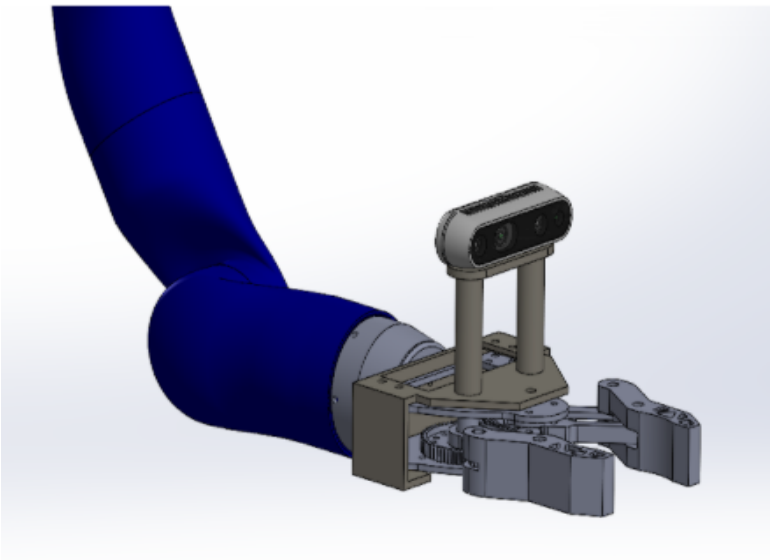

Figure 3.8: Camera mounted on gripper

References

1. B.Zhu, C. Li, L. Song, Y. Song and Y. Li, "A algorithm of global path planning based on the grid map and V-graph environmental model for the mobile robot," 2017 Chinese Automation Congress, Jinan, 2017, pp. 4973-4977.

2. "Human Robot Interaction on Navigation Platform Using ROS. (2020)", International Conference on Inventive Systems and Control (ICISC 2020), India.

3. Chung, J.H., Velinsky, S.A. Robust Interaction Control of a Mobile Manipulator – Dynamic Model Based Coordination. Journal of Intelligent and Robotic Systems 26, 47–63 (1999)

4. C. Mucchiani, M. Kennedy, M. Yim and J. Seo, "Object Picking Through In-Hand Manipulation Using Passive End-Effectors With Zero Mobility," in IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 1096-1103, April 2018

5. C. L. Fall et al., "A Multimodal Adaptive Wireless Control Interface for People With Upper-Body Disabilities," in IEEE Transactions on Biomedical Circuits and Systems, vol. 12, no. 3, pp. 564-575, June 2018. doi: 10.1109/TBCAS.2018.2810256

6. K. Zinchenko, C. Wu and K. Song, "A Study on Speech Recognition Control for a Surgical Robot," in IEEE Transactions on Industrial Informatics, vol. 13, no. 2, pp. 607-615, April 2017.doi: 10.1109/TII.2016.2625818

7. Y. Cheng and G. Y. Wang, "Mobile robot navigation based on lidar," 2018 Chinese Control And Decision Conference (CCDC), Shenyang, 2018, pp. 1243-1246.

8. R. K. Megalingam, R. N. Nair and S. M. Prakhya, "Automated voice based home navigation system for the elderly and the physically challenged," International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), Chennai, 2011, pp. 1-5.

9. R. Li, J. Liu, L. Zhang and Y. Hang, "LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments," 2014 DGON Inertial Sensors and Systems (ISS), Karlsruhe, 2014, pp. 1-15.

10. R. K. Megalingam, N. Katta, R. Geesala, P. K. Yadav and R. C. Rangaiah, "Keyboard-Based Control and Simulation of 6-DOF Robotic Arm Using ROS," 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018, pp. 1-5.

11. X. Zhao, Z. Cao, Q. Jia, L. Pang, Y. Yu and M. Tan, "A Vision-Based Robotic Grasping Approach under the Disturbance of Obstacles," 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, 2018,

pp. 2175-2179.

12. M. Tharaniya soundhari and S. Brilly Sangeetha, "Intelligent interface based speech recognition for home automation using android application," 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2015, pp. 1-11.

13. C.M. Wronka, M.W. Dunnigan,"Derivation and analysis of a dynamic model of a robotic manipulator on a moving base",Robotics and Autonomous Systems,Volume 59, Issue 10,2011,Pages 758-769,ISSN 0921-8890,

14. A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 348-353 vol.1, doi: 10.1109/ROBOT.2000.844081

15. Ata, A.A. Dynamic modelling and numerical simulation of a non-holonomic mobile manipulator. Int J Mech Mater Des 6, 209–216 (2010). https://doi.org/10.1007/s10999-010-9130-6

16. https://www.who.int/news-room/fact-sheets/detail/ageing-and-health

17. https://www.oberlo.in/statistics/how-many-people-have-smartphones

18. https://www.who.int/publications/i/item/guidelines-on-the-provision-of-manual-wheelchairs-in-less-resourced-settings

Publications based on the Project

Journal Papers:

1. "Modelling, Simulation and Testing of Altitude Hovering of the Quadcopter," Advances in Science, Technology and Engineering Systems Journal (ASTESJ 2020). (*Submitted*)

Conference Papers:

1. "ROS Based Control of Robot Using Voice Recognition," 3th International Conference on Inventive Systems and Control (ICISC 2019) IEEE. (*Accepted*)

2. "Human Robot Interaction on Navigation platform using Robot Operating System", 4th International Conference on Inventive Systems and Control (ICISC 2020) IEEE. (*Accepted*)

3. "Integration of Vision based Robot Manipulation using Robot Operating System for Assistive Applications", 2nd international conference on Inventive Research in Computing Applications (ICIRCA 2020) IEEE. (*Accepted*)

4. "Integration of Speech and Vision for Perception in Assistive Robots using Robot Operating System," 3rd International Conference on Computer Networks and Inventive Communication Technologies 2020 (ICCNCT 2020) Springer. (*Accepted*)

Appendix A

Navigation

A.1 URDF

```
<robot
  name="wheelchair_description">
  <link name="chassis">
    <inertial>
      <origin
        xyz="0.262285465630064 0.00775655751284371 0.48587315141618"
        rpy="0 0 0" />
      <mass
        value="44.2204895935843" />
      <inertia
        ixx="3.90150846928489"
        ixy="-0.125655599815589"
        ixz="0.504670789908843"
        iyy="4.76551098209845"
        iyz="0.0669171469527609"
        izz="3.01815232880521" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://wheelchair_description/meshes/chassis.STL" />
      </geometry>
      <material
        name="">
        <color
          rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
```

```
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/chassis.STL" />
    </geometry>
  </collision>
</link>
<link
  name="right_caster_m">
  <inertial>
    <origin
      xyz="-0.0203511857181596 3.08780778723872E-16 -0.0642021921749968"
      rpy="0 0 0" />
    <mass
      value="0.122527682330381" />
    <inertia
      ixx="0.000570464226341137"
      ixy="4.09116913523827E-19"
      ixz="0.000172881456061164"
      iyy="0.000587562122069702"
      iyz="-2.64274279543342E-18"
      izz="0.000145872604228965" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/right_caster_m.STL"
          />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
```

```
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/right_caster_m.STL"
          />
    </geometry>
  </collision>
</link>
<joint
  name="rcasterm"
  type="continuous">
  <origin
    xyz="0.627934962940585 -0.243517292495091 0"
    rpy="3.14159265358979 -8.11409419579269E-16 -3.12848106310099" />
  <parent
    link="chassis" />
  <child
    link="right_caster_m" />
  <axis
    xyz="0 0 -1" />
</joint>
<link
  name="right_caster">
  <inertial>
    <origin
      xyz="4.31929432950229E-05 2.01049294699662E-05 -6.86308880338757E-06"
      rpy="0 0 0" />
    <mass
      value="0.980320714803441" />
    <inertia
      ixx="0.00257869850096908"
      ixy="2.55863930097626E-06"
      ixz="-1.75043822489301E-06"
      iyy="0.0025694352948634"
      iyz="-4.47338630123438E-07"
      izz="0.00498313292895876" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
```

```
      <mesh
        filename="package://wheelchair_description/meshes/right_caster.STL"
            />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/right_caster.STL"
            />
    </geometry>
  </collision>
</link>
<joint
  name="rcaster"
  type="continuous">
  <origin
    xyz="-0.0508000000000002 0 0.0120000000000001"
    rpy="-1.5707963267949 -4.54886642756288E-18 -5.17641485231479E-15" />
  <parent
    link="right_caster_m" />
  <child
    link="right_caster" />
  <axis
    xyz="0 0 1" />
</joint>
<link
  name="left_caster_m">
  <inertial>
    <origin
      xyz="-0.0203511857181595 -8.32667268468867E-17 -0.0642021921749968"
      rpy="0 0 0" />
    <mass
      value="0.122527682330381" />
    <inertia
      ixx="0.000570464226341136"
```

```
      ixy="4.06575814682064E-20"
      ixz="0.000172881456061163"
      iyy="0.0005875621220697"
      iyz="3.3881317890172E-20"
      izz="0.000145872604228965" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/left_caster_m.STL"
          />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/left_caster_m.STL"
          />
    </geometry>
  </collision>
</link>
<joint
  name="lcaster_m"
  type="continuous">
  <origin
    xyz="0.627934962940584 0.245432707504909 0"
    rpy="3.14159265358979 -7.76438579476904E-16 -2.84665017791597" />
  <parent
    link="chassis" />
  <child
    link="left_caster_m" />
  <axis
    xyz="0 0 -1" />
```

```
</joint>
<link
  name="left_caster">
  <inertial>
    <origin
      xyz="4.31929432951339E-05 2.01049294700217E-05 -6.86308880382125E-06"
      rpy="0 0 0" />
    <mass
      value="0.980320714803439" />
    <inertia
      ixx="0.00257869850096908"
      ixy="2.55863930097679E-06"
      ixz="-1.75043822492536E-06"
      iyy="0.0025694352948634"
      iyz="-4.47338630132118E-07"
      izz="0.00498313292895875" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/left_caster.STL"
          />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/left_caster.STL"
          />
    </geometry>
  </collision>
</link>
<joint
```

```
  name="lcaster"
  type="continuous">
  <origin
    xyz="-0.0508000000000001 0 0.0120000000000001"
    rpy="1.5707963267949 0.983213864102094 -3.14159265358979" />
  <parent
    link="left_caster_m" />
  <child
    link="left_caster" />
  <axis
    xyz="0 0 1" />
</joint>
<link
  name="left_wheel">
  <inertial>
    <origin
      xyz="2.77555756156289E-17 -0.031115 1.11022302462516E-16"
      rpy="0 0 0" />
    <mass
      value="0.776379377303282" />
    <inertia
      ixx="0.00470924599389714"
      ixy="-4.99968244166719E-19"
      ixz="0"
      iyy="0.0090604235313858"
      iyz="-1.06158432795616E-19"
      izz="0.00470924599389714" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/left_wheel.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
```

```
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/left_wheel.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="lwheel"
  type="continuous">
  <origin
    xyz="-0.0155650370594172 0.292892707504908 0.0381000000000014"
    rpy="0 -0.750226906011315 3.14159265358979" />
  <parent
    link="chassis" />
  <child
    link="left_wheel" />
  <axis
    xyz="0 -1 0" />
</joint>
<link
  name="right_wheel">
  <inertial>
    <origin
      xyz="1.38777878078145E-17 0.031115 5.55111512312578E-17"
      rpy="0 0 0" />
    <mass
      value="0.776379377303283" />
    <inertia
      ixx="0.00470924599389715"
      ixy="-4.595438927656E-19"
      ixz="-5.42101086242752E-19"
      iyy="0.00906042353138581"
      iyz="-2.70368574737557E-19"
      izz="0.00470924599389715" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
```

```
        filename="package://wheelchair_description/meshes/right_wheel.STL"
          />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/right_wheel.STL"
          />
    </geometry>
  </collision>
</link>
<joint
  name="rwheel"
  type="continuous">
  <origin
    xyz="-0.0155650370594161 -0.290977292495091 0.0381000000000001"
    rpy="0 -0.407618862787816 3.14159265358979" />
  <parent
    link="chassis" />
  <child
    link="right_wheel" />
  <axis
    xyz="0 -1 0" />
</joint>
<link
  name="lidar">
  <inertial>
    <origin
      xyz="0.00634613668546278 -9.1926111087437E-05 -0.0224318722671598"
      rpy="0 0 0" />
    <mass
      value="1.33076913637072" />
    <inertia
      ixx="0.000614917940454092"
      ixy="-8.01984318975133E-07"
```

```
      ixz="0.00015824302591117"
      iyy="0.000960697335617224"
      iyz="-3.09162986587569E-06"
      izz="0.00109815200027136" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/lidar.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/lidar.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="lidarj"
  type="fixed">
  <origin
    xyz="0.680957038855197 -0.28643326884354 0.640699999999952"
    rpy="5.89434565890988E-16 -2.22044604925031E-16 1.5707963267949" />
  <parent
    link="chassis" />
  <child
    link="lidar" />
  <axis
    xyz="0 0 0" />
</joint>
<link
  name="l1">
```

```
  <inertial>
    <origin
      xyz="-2.27869412772197E-06 0.0819531846397067 -0.0154431442464013"
      rpy="0 0 0" />
    <mass
      value="0.113822392193168" />
    <inertia
      ixx="0.000365387884179077"
      ixy="1.06967942476891E-09"
      ixz="-3.87318753710054E-09"
      iyy="0.00015207591011146"
      iyz="7.64117882699427E-05"
      izz="0.000366226186980113" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l1.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0 0 0.752941176470588 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l1.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="l1_j"
  type="revolute">
  <origin
    xyz="0.67233 0.27956 0.3887"
    rpy="1.5708 -8.079E-16 3.0477" />
```

```
  <parent
    link="chassis" />
  <child
    link="l1" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-1"
    upper="1"
    effort="0.5"
    velocity="0.5" />
</joint>
<link
  name="l2">
  <inertial>
    <origin
      xyz="-4.36855553740401E-05 -0.0288101238855782 0.225634296548554"
      rpy="0 0 0" />
    <mass
      value="0.299300951509284" />
    <inertia
      ixx="0.00716004635236845"
      ixy="1.20133614254635E-07"
      ixz="2.55267092762312E-06"
      iyy="0.0072716757159213"
      iyz="-9.73900426259923E-08"
      izz="0.000268750621807733" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l2.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0 0 0.752941176470588 1" />
    </material>
  </visual>
  <collision>
    <origin
```

```
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l2.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="l2_j"
  type="resolute">
  <origin
    xyz="0 0.134 0"
    rpy="-1.5708 -1.3878E-17 2.6532" />
  <parent
    link="l1" />
  <child
    link="l2" />
  <axis
    xyz="0 -1 0" />
  <limit
    lower="-1"
    upper="1"
    effort="0.5"
    velocity="0.5" />
</joint>
<link
  name="l3">
  <inertial>
    <origin
      xyz="0.000117608698173344 -0.0171921633787411 -0.0831458948826457"
      rpy="0 0 0" />
    <mass
      value="0.136448774599551" />
    <inertia
      ixx="0.000743767289633624"
      ixy="1.30115740655641E-08"
      ixz="-1.13617761455231E-06"
      iyy="0.000759849573433189"
      iyz="0.000110312830903639"
      izz="0.000141769720688612" />
  </inertial>
  <visual>
    <origin
```

```
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l3.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0 0 0.752941176470588 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l3.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="l3_j"
  type="resolute">
  <origin
    xyz="0 0 0.45"
    rpy="-1.9138E-14 1.0151 3.1416" />
  <parent
    link="l2" />
  <child
    link="l3" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-1"
    upper="1"
    effort="0.5"
    velocity="0.5" />
</joint>
<link
  name="l4">
  <inertial>
    <origin
```

```
      xyz="-0.010618763569967 0.0939007952792743 1.86299276299096E-05"
      rpy="0 0 0" />
    <mass
      value="0.0735464655844353" />
    <inertia
      ixx="0.000156093216993758"
      ixy="2.54058907965856E-05"
      ixz="1.22715915193488E-08"
      iyy="6.51477382482669E-05"
      iyz="-5.61156675877574E-08"
      izz="0.000147755583106162" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l4.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0 0 0.752941176470588 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l4.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="l4_j"
  type="resolute">
  <origin
    xyz="0 0 -0.16929"
    rpy="-1.5708 6.5677E-15 1.4063" />
  <parent
    link="l3" />
```

```
  <child
    link="l4" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-1"
    upper="1"
    effort="0.5"
    velocity="0.5" />
</joint>
<link
  name="l5">
  <inertial>
    <origin
      xyz="-0.0106238043389506 -0.0438098225991777 1.74958006025072E-05"
      rpy="0 0 0" />
    <mass
      value="0.0735566402448327" />
    <inertia
      ixx="0.00015609581426092"
      ixy="2.54230465664775E-05"
      ixz="1.43045124971658E-08"
      iyy="6.51614695053811E-05"
      iyz="-5.75833320046815E-08"
      izz="0.000147770708635849" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l5.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0 0 0.752941176470588 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
```

```
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l5.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="l5_j"
  type="resolute">
  <origin
    xyz="0 0.13771 0"
    rpy="-1.0707 -6.9389E-18 3.1416" />
  <parent
    link="l4" />
  <child
    link="l5" />
  <axis
    xyz="-1 0 0" />
  <limit
    lower="-1"
    upper="1"
    effort="0.5"
    velocity="0.5" />
</joint>
<link
  name="l6">
  <inertial>
    <origin
      xyz="7.27800559119363E-06 0.027178689262767 -2.38113756489966E-09"
      rpy="0 0 0" />
    <mass
      value="0.0220961018790657" />
    <inertia
      ixx="1.23073936673388E-05"
      ixy="-2.7179114383126E-09"
      ixz="-5.69823500921333E-13"
      iyy="1.6773611361693E-05"
      iyz="-6.92273408175517E-13"
      izz="1.23259860802391E-05" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
```

```
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l6.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.792156862745098 0.819607843137255 0.933333333333333 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://wheelchair_description/meshes/l6.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="l6_j"
  type="resolute">
  <origin
    xyz="0 -0.10393 0"
    rpy="3.1416 -1.2698 5.8511E-18" />
  <parent
    link="l5" />
  <child
    link="l6" />
  <axis
    xyz="0 1 0" />
  <limit
    lower="-1"
    upper="1"
    effort="0.5"
    velocity="0.5" />
</joint>
<gazebo>
  <plugin name="differential_drive_controller"
     filename="libgazebo_ros_diff_drive.so">
    <legacyMode>false</legacyMode>
    <alwaysOn>true</alwaysOn>
    <updateRate>10</updateRate>
```

```
    <leftJoint>lwheel</leftJoint>
    <rightJoint>rwheel</rightJoint>
    <wheelSeparation>0.4</wheelSeparation>
    <wheelDiameter>0.32</wheelDiameter>
    <torque>50</torque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>chassis</robotBaseFrame>
  </plugin>
</gazebo>


<gazebo reference="chassis">
  <material>Gazebo/Grey</material>
</gazebo>



<gazebo reference="lidar">
  <sensor type="ray" name="rp_lidar_sensor">
    <pose>0 0 0 0 0 0</pose>
    <visualize>true</visualize>
    <update_rate>40</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>2000</samples>
     <resolution>1</resolution>
          <min_angle>-4</min_angle>
          <max_angle>0.5</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.2</min>
        <max>5.50</max>
        <resolution>0.05</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.00</stddev>
      </noise>
    </ray>
```

```
    <plugin name="gazebo_ros_head_hokuyo_controller"
       filename="libgazebo_ros_laser.so">
      <topicName>/wheelchair/laser/scan</topicName>
 <frameName>lidar</frameName>
    </plugin>
  </sensor>
</gazebo>
<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="l1_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="l1_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="l2_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="l2_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="l3_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="l3_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="l4_j">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
```

```
    </joint>
    <actuator name="l4_j">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran1">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="l5_j">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="l5_j">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <transmission name="tran1">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="l6_j">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="l6_j">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>
</robot>
```

Appendix B

Robotic Arm

B.1 YAML

```
wheelchair_description:
  # Publish all joint states -----------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 40

  # Position Controllers ---------------------------------------
  joint1_position_controller:
    type: effort_controllers/JointPositionController
    joint: l1_j
    pid: {p: 100.0, i: 0.0, d: 1.0}

  joint2_position_controller:
    type: effort_controllers/JointPositionController
    joint: l2_j
    pid: {p: 100.0, i: 0.0, d: 1.0}

  joint3_position_controller:
    type: effort_controllers/JointPositionController
    joint: l3_j
    pid: {p: 100.0, i: 0.0, d: 1.0}

  joint4_position_controller:
    type: effort_controllers/JointPositionController
    joint: l4_j
    pid: {p: 100.0, i: 0.0, d: 1.0}

  joint5_position_controller:
```

```
  type: effort_controllers/JointPositionController
  joint: l5_j
  pid: {p: 100.0, i: 0.0, d: 1.0}

joint6_position_controller:
  type: effort_controllers/JointPositionController
  joint: l6_j
  pid: {p: 100.0, i: 0.0, d: 1.0}
```