Design and Development of an Omnidirectional Mobile Base for a Social Robot

LIM HE WEI

NATIONAL UNIVERSITY OF SINGAPORE

2011

Design and Development of an Omnidirectional Mobile Base for a Social Robot

LIM HE WEI

(B.Eng.(Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2011

Acknowledgement

I would like to express my warm and sincere thanks to Professor Marcelo Ang Jr from the department of Mechanical engineering, National University of Singapore who introduced me to the systematic and analytical approach to robotic systems. His support and trust as supervisor and mentor has helped me in many aspects of my higher studies endeavor from engineering know-how to psychological approach towards understanding "mind-numbing" mathematics involved in robotics. Words cannot describe the "calmness" which I observed and learnt from him.

Professor Sam Ge from Electrical Engineering and also director of Social Robotics Lab have also helped and encouraged me in my partaking of this Masters of Engineering. His attitude towards engineering and the business surrounding engineering is positively radical. Despite the gap between our capabilities, he spent significant time and effort in educating and helping all whom worked with him. I respect him for being an agent of change and sincerely thank him for his mentorship which firmed up my decision to enter the industry early. The many members of the Social Robotic Lab have also been pivotal in shaping my engineering pursue path.

I would like to extend my gratitude towards Dr, John-John Cabibihan whom provided engineering advice and "listening ear", laboratory mates Haibin and Tiffany, former FYP and BTech students who worked on earlier powered castor robots and current social robots who have helped in both engineering and team support. Lastly, I would like to thank my family members and close friends who have been nothing but supportive in one way or another. Thank you very much.

Abstract

In this thesis, we examine the design, development and implementation of a mobile base for our social robot – Robotubby. Robotubby is a social robot envisioned to be a companion to children capable of social interaction and participating in meaningful edutainment tasks. The development of Robotubby involves other researchers investigating other areas of social robotics thus the mobile platform must be compatible and easy to integrate together.

Physically, Tele-tubby is a wheel-based mobile humanoid with movable torso, two robot arms and an expressive face in order to display gestures and facial expressions that potentially convey emotions and better communication. As a child companion, the size of the robot plays a critical role both for safety and approachability. Given the indoor environment of homes, the mobile platform requires good maneuverability to traverse its environment.

One key feature of Robotubby is the ability to conduct tele-presence which refers to a set of technologies allowing users to feel present or have an effect of presence at the particular event which are typically physically distanced from the user. A consideration of network technologies is necessary for the development of the mobile platform and the means of control for users.

A Powered-Castor wheel (PCW) design for the mobile platform is selected for the merits of Omni-directional travel as indoor environments such as homes are typically space-constrained. Compared to other solutions such as Omni-directional wheels, PCW maintains perpetual contact with the ground for smooth floors, providing smoother motion and less operation noise. Controlling of a PCW platform based on prior works provide the foundation for developing omni-directional control via a joystick with handle twist either locally or over the network. In experimentation, the mobile platform was able to perform the task of moving around on a level ground with relative ease except for some jitters due to the nature of the motor controller feedback.

Table of Contents

List of Figures:

List of Symbols

$\dot{X}$	=	Cartesian velocities
$\dot{x}$	=	Translation velocity in X direction
$\dot{y}$	=	Translation velocity in Y direction
$\dot{\theta}$	=	Rotation velocity in Z direction
$\dot{Q}$	=	PCW joint velocities
$\dot{\rho}$	=	PCW drive velocity
$\dot{\phi}$	=	PCW steering velocity
J	=	Jacobian matrix linking joint velocities to Cartesian velocities
V_a	=	Applied armature voltage, V
i_a	=	Armature current, A
Ra	=	Armature resistance, Ω
L_a	=	Armature inductance, H
V_b	=	Back EMF, V
T	=	Torque developed by the motor, Nm
J	=	Equivalent moment of inertia of the motor and the load referred to the motor shaft, kgm^2
b	=	Equivalent viscous-friction coefficient of the motor and load referred to the motor shaft, $Nmrad^{-1}s^{-1}$
m	=	mass of load, kg

Chapter 1

Introduction

Social robotics is increasingly relevant in today's world as more societies are maturing faster, reaching the graying population paradigm in a few short decades. Researches in the multi-disciplinary fields of social robotics have developed significantly over the past few years [1] with one of the goals of addressing the above real-world issues. In creating the "social" into more commonly known robots in industrialized settings, many researchers have undertaken the task of understanding and creating effective social interaction with robots also known as Human Robot Interaction (HRI). Examples of social robotic task include personal assistant, companion robot and handicap aid as seen in Figure 1-1.

Figure 1-1: Role of Social Robots in today's society, as companion to elderly, children and assistant people to overcome various handicap [2]

To understand the phenomenon of Social Robots and the field of HRI, we can refer to mankind's history for artificial beings development, robots medieval by today's standards but still capable of complex motion by clockwork engines. In the famous play by Karel Capek, the term robots was first coined in the title "Rossum's Universal Robots" (R.U.R) depicting moving mechanical machines as slave workers. The theme of robots continue to evolve with robots increasingly seem human despite not having human or humanoid form.

Physically, the design of the mobile base focuses on form and function that would make Robotubby a likable social robot with good usability. The environment Robotubby operates in is almost exclusively

indoors and has to traverse the dynamic grounds that filled with furniture, ornaments and even children's toys. The mobile platform thus has to be able to maneuver well around such environments in order to perform task such as navigate around to locate the child at home. Examples of different locomotion in robots are shown in Figure 1-2.

Locomotion allows social robots to navigate their surroundings and perform their assigned service to people. In the design of locomotive mechanisms [3,4], wheeled systems are most robust can observed in all modern land transports. Nevertheless, such designs on an indoor social robot must fulfill certain criteria like getting out of tight spots easily and navigating uneven terrain such as children's toys strew all over the floor. This thesis presents a caster-wheel platform design for a social robot that can navigate a home environment seamlessly and able to perform other task.

Figure 1-2: Various modes of locomation for social robots

Social-ness and sociability is a man-made concept we typically attribute to matter we come into contact with. In this respect, social robots can be applied to machines we interact with like computers, printers and even autonomous vacuum cleaners. Designing HRI into everyday machines we interact with can improve the wellbeing of people using these machines. Many experiments have been conducted with how humans react to robots or even machines and researchers test how to make the interaction experience more pleasant.

Designing the sociability of the mobile platform with respect to physical is mostly on the appearance and more importantly appeal such as the robots in Figure 1-3 with child-like features and smooth coverings. This particular aspect will not be covered as external design will be covered as the whole robot. Designing Robotubby's sociability will be more of software with respect to motor control.

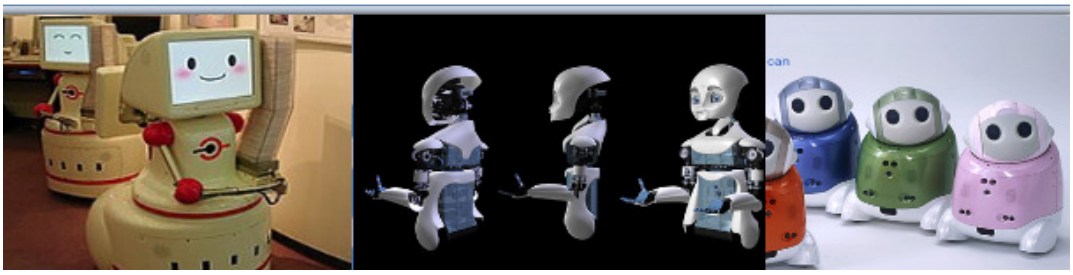


Figure 1-3: HRI in Social Robot Design, form and function [2]

Social robotics provide the integral bridge between the virtual information world and the physical world we live in thus allowing the provision of practical help to people in-need. Figure 1-4 shows examples of social robots performing various task of service for people in different scenarios. An active branch of robotics research is involved in building social robots are being developed to empower human caretakers, taking over mundane task so they can focus more on caring. The hardware aspect explores mechanical designs especially on ergonomics and rehabilitation purpose. Software aspect explores intelligent control that are able to adapt to specific users and remote operation, which for the purpose of social robotics, tele-presence.

Figure 1-4: Examples of Tele-robotics, task in communication, education and assistance

Tele-presence is increasingly gaining acceptance among robotics researchers and commercial entities. The term tele-presence was coined in a 1980 article by Marvin Minsky, who outlined his vision for an adapted version of the older concept of tele-operation that focused on giving the remote participation a feeling of actually being present [5]. The key difference of local control and long distance remote control lies in time delay and data integrity when it has to travel over long distance. Resolving this issue of tele-presence requires working with current networking technologies and performing control of the robot over the network.

With various attributes and potential task of social robots, mobility is an important feature that makes robots more service-orientated towards people by going to them. Of course, the motion can be autonomous with onboard or environment sensors or tele-operated which is similar to tele-presence.

The thesis presents the work done in creating the mobile base for Tele-tubby and the integration into the robot. A literature review of the relevant topics is conducted in Chapter 2. The design and implementation is covered in Chapter 3. The review and experimentation is covered in Chapter 4. Lastly, Chapter 5 summaries the project contribution and future works.

Chapter 2

LITERATURE REVIEW

The basic expectation of a mobile social robot is to be capable of maneuvering around the environment, following trajectories based on algorithms of path-planning and obstacle avoidance depending on implementation. Typical requirements of mobile robots include being easily operated via remote control. Depending on configuration of the mobility actuator/s, some remote controls are intuitive while some require mathematical models to reduce the complexity of control to such the human operator can handle. Other consideration for mobility optimization includes the efficiency of the system, generating smoother motion profiles and fault-tolerance [6,7].

2.1 Robot Architecture

Robots are typically task-specific and architecture is designed to match the expectations of the task [8]. For social robots, the architecture typically involves human interaction in vast degree and close interaction such as the behavioral based architecture in Figure 2-1. As such, many efforts are directed at higher levels towards meaningful communication with humans. Nevertheless, the creation of a successful social robot depends on all aspects of the architecture regardless of its layer of development during implementation. The mobile base plays a role in the architecture when the specification of the social robot such as Robotubby is required to navigate around the environment and locate a person.

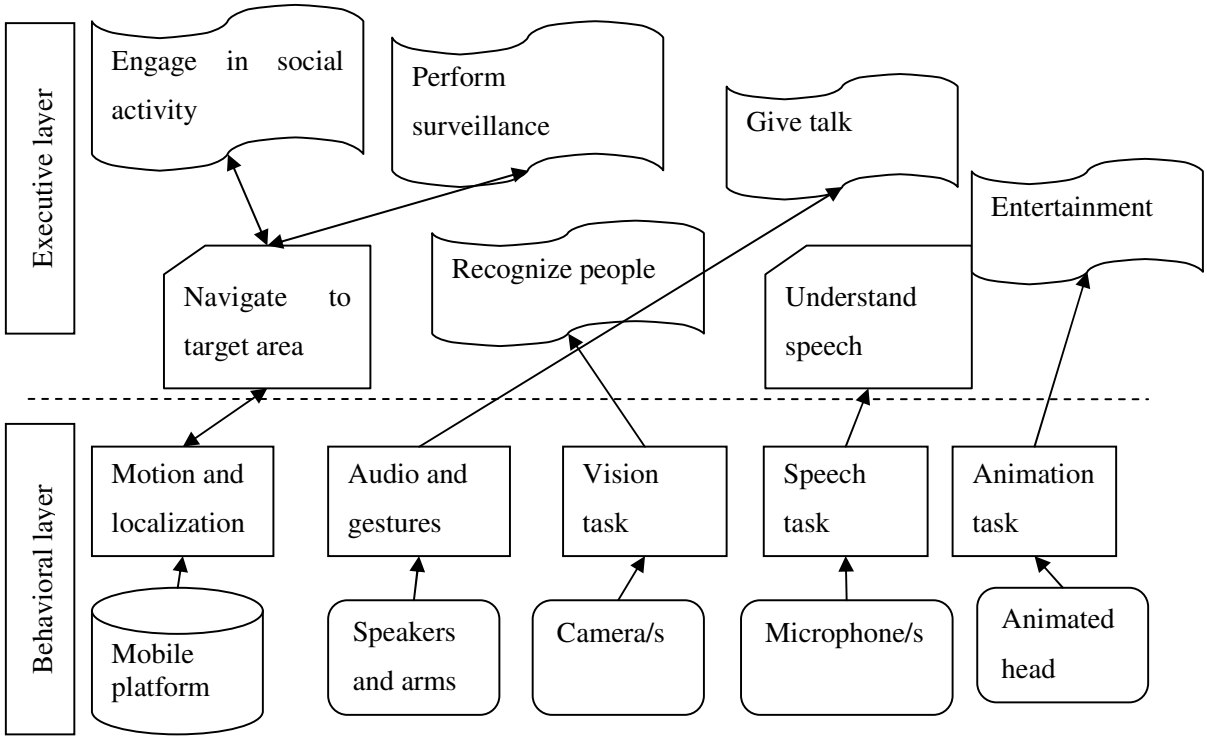


Figure 2-1: Example of social robot architecture for execution and robot behavioral layer

2.2 Review of mobile Robots

The prevalence of automation and mobile robotic platform has seen a rise of demand for high mobility platforms. While high mobility platforms are in demand, commercial systems have to balance many other factors such as price and turn-around time. Commercial systems typically go the way of differential drive systems which are capable of changing its orientation by pivoting at the center of the differential wheel pair. Extending on the concept, cars are almost exclusively differential drive on the front wheels thus requiring a minimum turning radius to achieve the orientation change.

2.2.1 Holonomic actuation:

Holonomicity in robotics refer to the relationship between controllable and total degrees of freedom for a given robot, for this case the ground mobility. The robot is said to be holonomic if the controllable degrees of freedom is equal to or greater than the total degrees of freedom. A robot is considered non-holonomic if the total number of controllable degrees of freedom is less than the total number of degrees of freedom in its task space. Conversely, a robot with more controllable degrees of freedom than its total degrees of freedom is considered redundant.

Most cars operating on the roads today are an example of a non-holonomic vehicle. Cars are designed to travel in 3 degrees of freedom namely the X and Y axis of the horizontal plane and Θ which represents the change of orientation as in the vehicle's heading. The car has only two controllable degrees of freedom which are accelerating or braking in the direction of travel and changing the orientation of vehicle via the angle of the steering wheel. Assuming no skidding or sliding, there are no other allowable paths in the phase space. As such, the non-holonomicity of most cars makes tasks like parallel parking difficult and rotation on the spot impossible.

2.2.2 Differential drive system:

Cars are the most common wheeled system and the design is based on differential drive for 2 of the wheels (mostly front 2) to provide change of direction. In robots, a similar differential drive mechanism is employed but most typically using 1 idler wheel instead. Nevertheless, all differential drive systems require a minimum turn radius, making it difficult for the robot to navigate home environments. If robots are equipped with castors much like office chairs, getting out of a tight spot would be easy.

Skid steering mobile platforms with form factor similar to cars. They have 2 driven wheels at the opposite sides of the vehicle where different rotational speeds allow it to turn with 0 turning radius. The motion profile of the skid steering is shown in Figure 2-2.

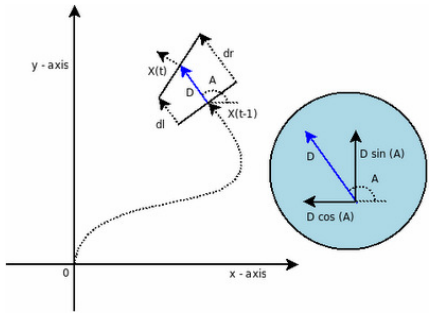


Figure 2-2: Differential Drive Modeling for Navigation

For these mobile bases, it is not possible to specify the desired orientation, θ, and position, X and Y as shown in Figure 2-3, in a single maneuver due to the fact that the translational and rotational motions are coupled.

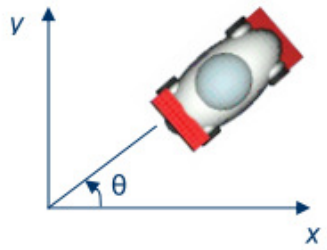


Figure 2-3: coupling of orientation and Position

To achieve such mobility, it is desired to design a mobile platform with full dexterity capable of achieving omni-directional control, i.e. the mobile robotic platform must be capable of independent translating and rotating motion. The main advantages of such an omni-directional system are:

i. Increase in mobility and
ii. No kinematic motion constraint.

The merit of the omni-directional mobile platform [4,5,9] is that it is possible to perform simultaneous rotation and translation. The increase in mobility is vital if the mobile bases are to be used in constrained environments such as narrow corridors in factories and buildings. Similarly, a system with no kinematic motion constraint will play a pivotal role in the development of motion planning and navigation algorithm whereby movement can be carried out easily.

2.2.3 Holonomic wheels:

Holonomic wheels are wheels with 2 or more degrees of freedom and commonly known as omni-directional wheels [4]. There are 2 main types of holonomic wheels, ones with peripheral rollers such as the Mecanum Wheels and specialized wheels such as a ball wheel mechanism. Mecanum wheels and those similar to it have small peripheral rollers attached to main driving wheel that gives the 2nd degree of freedom perpendicular to the main.

These wheels do not have kinematic constraints and fulfill the requirements stated above. However, there are significant limitations to each of the designs as follows:

i. Complexity in implementation
   a. Driving the main wheel in Mecanum wheel and similar design is straight-forward but the peripheral rollers require complex driving mechanism which are often left passive.
   b. Ball wheels require significantly more complex driving mechanism that still suffers from slippage given no direct means of actuating. Furthermore, like ball-wheeled computer mouse, the mechanisms are easily clogged by dust and dirt requiring more maintenance.

ii. Mecanum wheels, shown in Figure 2-4, also suffer from vibration especially in the main driving wheel as it has discreet numbers of rollers resulting in discontinuous contact with the ground.

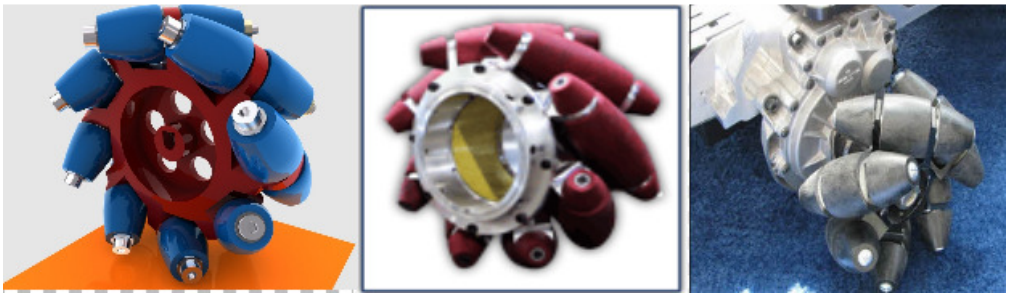


Figure 2-4: Mecanum wheels in different forms and number of rollers

2.2.4 Powered Castor Wheel:

With castors, robots can avoid the issue of singularity as the design allow for on the spot rotation and instant change of direction in terms of control. The additional benefit of castors are smooth locomotion while travelling on undulating terrain as it is always in contact with the ground compared to an omni-directional wheel which is a source of vibration. For example, the images captured by camera sensors mounted on the robot are not stable when the robot is moving. Furthermore the contact point with the ground is known thus exact control can be achieved [7,9].

The mechanism for Powered Castor Wheel typically consists of 2 motors driving the system, one for steering and the other for driving as shown in Figure 2-5.

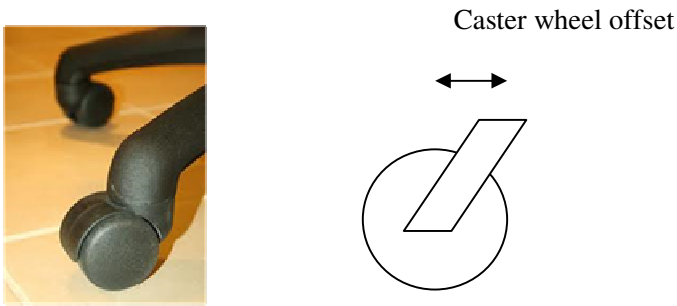


Figure 2-5: Caster Wheel and Operation Model

2.3 Review of kinematic Models

2.3.1 Robotics modelling and control:

The goal of kinematic analysis is to calculate the position, velocity and acceleration of all the linkages without consideration of the forces causing the motion. Specifically for RoboTubby, posture kinematic model can be found in [9,10,11,12,13] which also provides analysis based on other state space model like configuration kinematic model, configuration dynamical model and posture dynamical model. Robot kinematics are mainly of the following two types: forward kinematics and inverse kinematics. In forward kinematics, the length of each link and the angle of each joint is given and we have to calculate the position of any point in the work volume of the robot. In inverse kinematics, the length of each link and position of the point in work volume is given and we have to calculate the angle of each joint. Robot kinematics can be divided in serial manipulator kinematics, parallel manipulator kinematics, mobile robot kinematics and humanoid kinematics.

The forward position kinematics (FPK) solves the following problem: "Given the joint positions, what is the corresponding end effector's pose?" The same can be generalized for all forward kinematics which can be solved via Geometric or algebraic approach. Other than Cartesian coordinates, robot kinematics can also be represented in Denavit-Hartenberg parameters.

The inverse position kinematics (IPK) solves the following problem: "Given the actual end effector pose, what are the corresponding joint positions?" the key challenge for the generic inverse kinematics is that solutions are typically not unique if it exists at all. Similar to forward kinematics, solving the problem can be done by either geometric or algebraic method.

2.3.2 Forward Kinematics of single castor wheel

A single PCW can be modeled as a serial linked manipulator with one prismatic joint and two revolute joints. In this study, two different approaches of i) Inspection and ii) Transformation are used. The prismatic joint is obtained by relating the angular displacement of the wheel, ρ, to linear displacement, x. The linear velocity is also related by the same expression.

$$x = r\rho \tag{2.1}$$

$$\dot{x} = r\dot{\rho} \tag{2.2}$$

where: x = linear displacement, ρ = angular displacement , r = radius of wheel

$\dot{x}$ = linear velocity, $\dot{\rho}$ = angular velocity of wheel

The kinematics of the serial link manipulator can be modeled as a two dimensional planar robot as illustrated in Figures 2-6, 2-7 and 2-8.

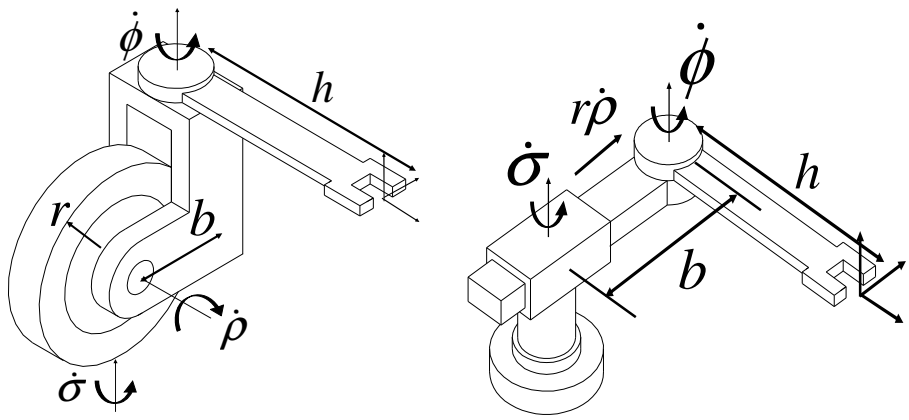


Figure 2-6: Joint Schematics showing relationship between various links

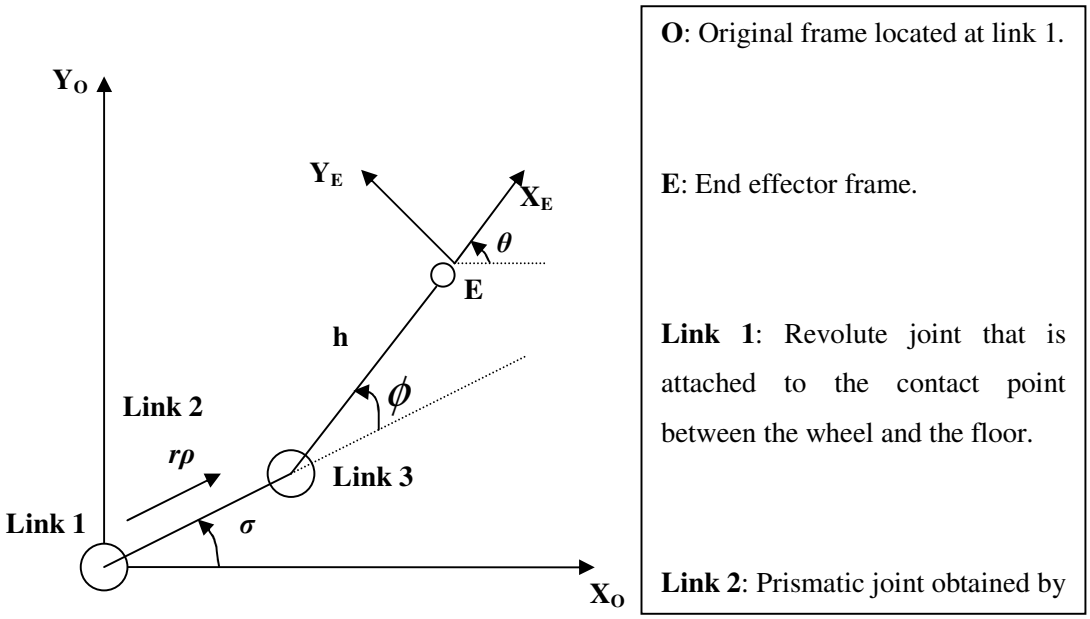


Figure 2-7: Modeling of the caster wheel as a 2 dimensional planar manipulator

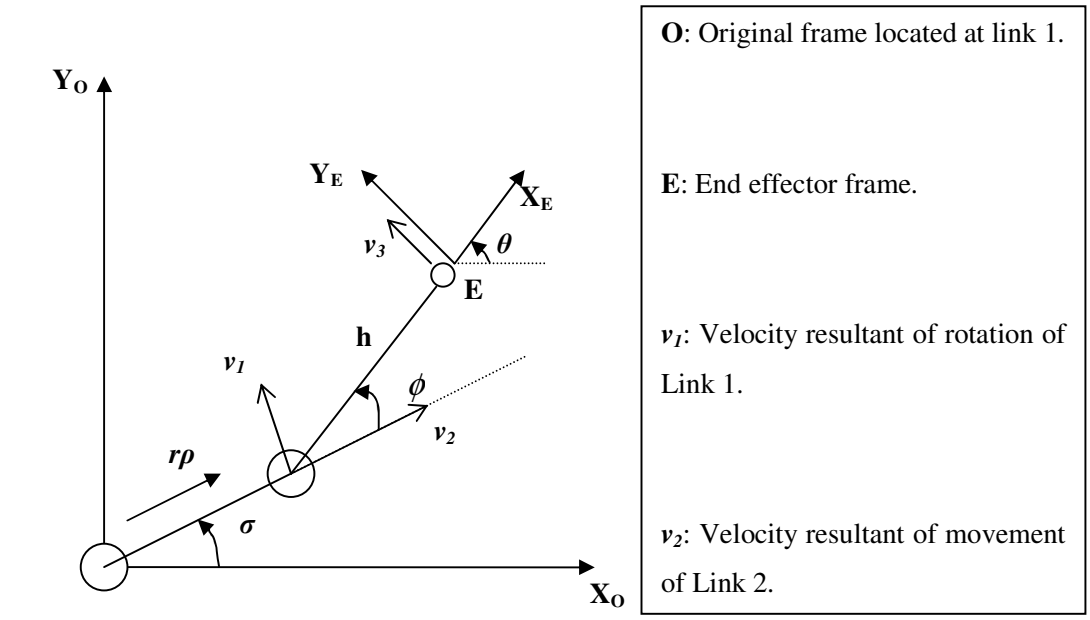


Figure 2-8: Resultant velocities of different links

The equations governing the position of the end-effector E with reference to frame O is:

$$
\begin{aligned}
{}^{O}x_{E} &= r\rho\cos(\sigma) + h\cos(\sigma+\phi) \\
{}^{O}y_{E} &= r\rho\sin(\sigma) + h\sin(\sigma+\phi) \\
{}^{O}\theta_{E} &= \sigma + \phi
\end{aligned}
\tag{2.3}
$$

Differentiating Eq 2.3, the velocities of the end-effector is achieved as such:

$$
\begin{aligned}
{}^{O}\dot{x}_{E} &= \dot{\sigma}[-r\rho\sin(\sigma) - h\sin(\sigma+\phi)] + \dot{\rho}[r\cos(\sigma)] + \dot{\phi}[-h\sin(\sigma+\phi)] \\
{}^{O}\dot{y}_{E} &= \dot{\sigma}[r\rho\cos(\sigma) + h\cos(\sigma+\phi)] + \dot{\rho}[r\sin(\sigma)] + \dot{\phi}[h\cos(\sigma+\phi)] \\
{}^{O}\dot{\theta}_{E} &= \dot{\sigma} + \dot{\phi}
\end{aligned}
\tag{2.4}
$$

From Eq 2.4 and setting $r\rho$ to b, the physical offset of the wheel, the Jacobian matrix, ${}^{O}J_{E}$, relating joint velocities to Cartesian velocities is derived.

$$
{}^{O}\dot{X}_{E} = {}^{O}J_{E}\dot{Q}
$$

Setting $r\rho = b$:

$$
\begin{bmatrix} {}^{O}\dot{x}_{E} \\ {}^{O}\dot{y}_{E} \\ {}^{O}\dot{\theta}_{E} \end{bmatrix} = \begin{bmatrix} -b\sin(\sigma) - h\sin(\sigma+\varphi) & r\cos(\sigma) & -h\sin(\sigma+\varphi) \\ b\cos(\sigma) + h\cos(\sigma+\varphi) & r\sin(\sigma) & h\cos(\sigma+\varphi) \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\sigma} \\ \dot{\rho} \\ \dot{\phi} \end{bmatrix} \tag{2.5}
$$

Eq 2.5 is a function of σ, ρ and ϕ. However, in the real world, σ is a passive joint and no odometry data can be obtained. To achieve a system that is a function of only ρ and ϕ, the expression of the Jacobian matrix must be obtained with respect to Frame E, the end-effector frame.

The velocities in the end-effector frame can be determined using two different methods which will essentially yield the same results. The first method is inspection method using the schematic diagram as shown in Figure 8. The three velocities are first resolved in the end-effector frame using simple trigonometry:

$$v_1 = r\rho\dot{\sigma}$$

$$v_2 = r\dot{\rho}$$

$$v_3 = h(\dot{\sigma} + \dot{\phi})$$

Setting $r\rho = b$:

$$\begin{aligned} {}^{E}\dot{x}_E &= v_1 \sin(\phi) + v_2 \cos(\phi) = \dot{\sigma}[b\sin(\phi)] + \dot{\rho}[r\cos(\phi)] \\ {}^{E}\dot{y}_E &= v_1 \cos(\phi) - v_2 \sin(\phi) + v_3 = \dot{\sigma}[b\cos(\phi) + h] + \dot{\rho}[-r\sin(\phi)] + \dot{\phi}[h] \\ {}^{E}\dot{\theta}_E &= \dot{\sigma} + \dot{\phi} \end{aligned} \tag{2.6}$$

Another method to obtain the Jacobian matrix in the end-effector frame will be to do transformation to the initial Jacobian in Frame O. In this case, the Jacobian is pre-multiplied with the rotational matrix showing Frame 0 in Frame E. This is done as such:

$$^{E}J_{E} = {}^{E}R_{O}\,{}^{O}J_{E}$$

where:

$$^{E}R_{O} = \begin{bmatrix} \cos(\sigma+\phi) & \sin(\sigma+\phi) & 0 \\ -\sin(\sigma+\phi) & \cos(\sigma+\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

Both Eq 2.6 and 2.7 will result in the same Jacobian matrix and hence verified the validity of the equation. The final expression in the end-effector frame is:

$$^{E}\dot{X}_{E} = {}^{E}J_{E}\dot{Q}$$

$$\begin{bmatrix} ^{E}\dot{x}_{E} \\ ^{E}\dot{y}_{E} \\ ^{E}\dot{\theta}_{E} \end{bmatrix} = \begin{bmatrix} b\sin(\phi) & r\cos(\phi) & 0 \\ b\cos(\phi)+h & -r\sin(\phi) & h \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\sigma} \\ \dot{\rho} \\ \dot{\phi} \end{bmatrix} \tag{2.8}$$

As can been seen in Eq 2.8 the Jacobian matrix is independent of σ cannot be determined.

Having obtained the relationship between joint velocities and end-effector Cartesian coordinates is the first part of the formulation. In order to obtain the relationship of the joint velocities and the mobile base Cartesian velocities, it is then required to input the physical dimension in to kinematics equation and to orientate the end-effector frame of each individual wheel, Frame E, into the base frame, Frame B shown in Figures 2-9 and 2-10.

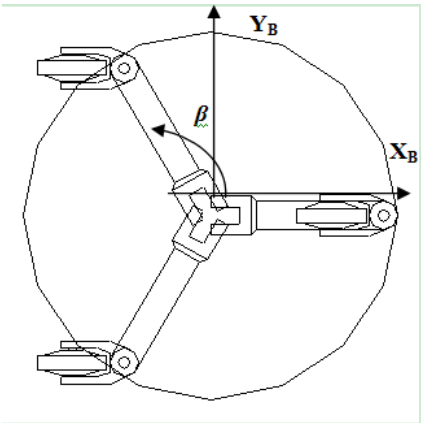


Figure 2-9: Orientation of base frame, B

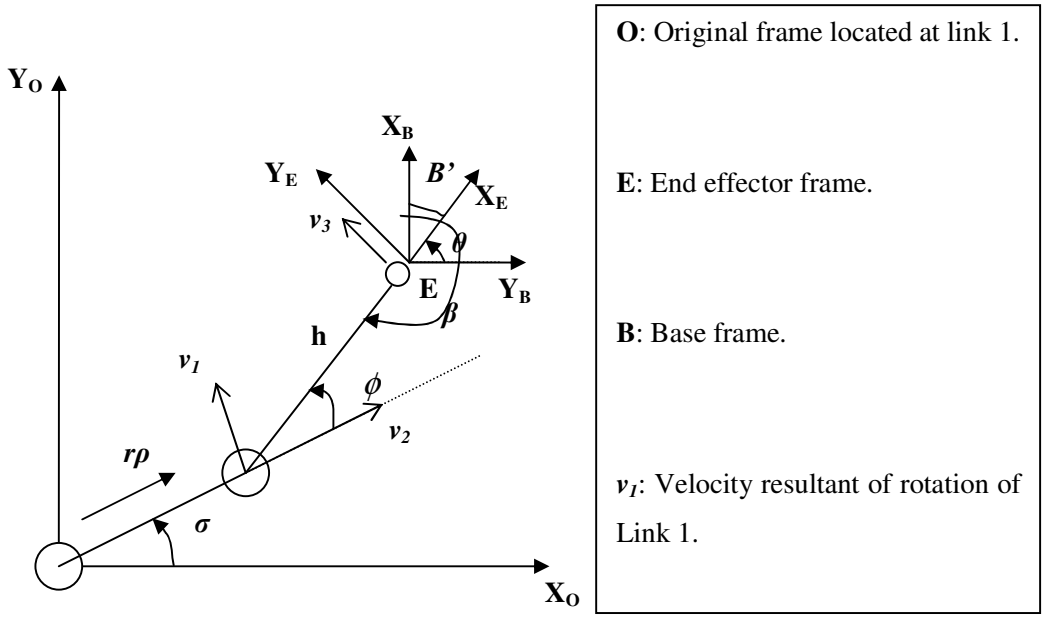


Figure 2-10: Position of base frame, B, with regards to end-effector frame

The values of v_1, v_2 and v_3 are the same as before. Using the inspection method, it is then possible to obtain the equations with respect to the base frame if we know the position of the PCW with respect to the base. Setting $r\rho = b$ and $\beta' = \beta - 180°$:

$$
\begin{aligned}
{}^{B}\dot{x}_E &= v_1(-\sin(\beta+\phi)) + v_2(-\cos(\beta+\phi)) + v_3(-\sin(\beta)) \\
{}^{B}\dot{x}_E &= \dot{\sigma}[-b\sin(\beta+\phi) - h\sin(\beta)] + \dot{\rho}[-r\cos(\beta+\phi)] + \dot{\phi}[-h\sin(\beta)] \\
{}^{B}\dot{y}_E &= v_1\cos(\beta+\phi) - v_2\sin(\beta+\phi) + v_3\cos(\beta) \\
{}^{B}\dot{y}_E &= \dot{\sigma}[b\cos(\beta+\phi) + h\cos(\beta)] + \dot{\rho}[-r\sin(\beta+\phi)] + \dot{\phi}[h\cos(\beta)] \\
{}^{B}\dot{\theta}_E &= -\dot{\sigma} - \dot{\phi}
\end{aligned}
\tag{2.9}
$$

Similarly, the same results could be obtained by a rotation of the Jacobian matrix in frame E to Frame B using the following:

$$
{}^{B}J_E = {}^{B}R_E \, {}^{E}J_E
$$

where:

$$
{}^{B}R_E = \begin{bmatrix} -\cos(\beta) & -\sin(\beta) & 0 \\ -\sin(\beta) & +\cos(\beta) & 0 \\ 0 & 0 & -1 \end{bmatrix}
\tag{2.10}
$$

The final equation from 2.9 and 2.10 are equivalent, hence verifying its validity.

$$
{}^{B}\dot{X}_E = {}^{B}J_E\dot{Q}
$$

$$
\begin{bmatrix} {}^{B}\dot{x}_E \\ {}^{B}\dot{y}_E \\ {}^{B}\dot{\theta}_E \end{bmatrix} = \begin{bmatrix} -b\sin(\beta+\phi) - h\sin(\beta) & -r\cos(\beta+\phi) & -h\sin(\beta) \\ b\cos(\beta+\phi) + h\cos(\beta) & -r\sin(\beta+\phi) & h\cos(\beta) \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \dot{\sigma} \\ \dot{\rho} \\ \dot{\phi} \end{bmatrix}
\tag{2.11}
$$

The Jacobian is always invertible, unless the offset b = 0. This shows the importance of having a non-zero offset.

2.3.3 Inverse Kinematics of single castor wheel

To obtain the desired joint velocities from a given base Cartesian velocities, the inverse kinematics of Eq 2.11 is used. Since the Jacobian matrix is a square matrix, the inverse of ${}^{B}J_{E}$ can be obtained, whereby ${}^{B}J_{E}^{-1}$ is determined to be as follows:

$$ {}^{B}J_{E}^{-1} = \frac{1}{rb}\begin{bmatrix} r\sin(\beta+\phi) & -r\cos(\beta+\phi) & -rh\cos(\phi) \\ b\cos(\beta+\phi) & b\sin(\beta+\phi) & bh\sin(\phi) \\ -r\sin(\beta+\phi) & r\cos(\beta+\phi) & r(b+h\cos(\phi)) \end{bmatrix} \tag{2.12} $$

As stated, σ cannot be measured physically. However the inverse kinematic equation can be made independent of σ. The inverse kinematics is determined with respect to the steer and drive velocities:

$$ \dot{Q} = {}^{B}J_{E}^{-1}\,{}^{B}\dot{X}_{E} $$

$$ \begin{bmatrix} \dot{\rho} \\ \dot{\phi} \end{bmatrix} = \frac{1}{rb}\begin{bmatrix} b\cos(\beta+\phi) & b\sin(\beta+\phi) & bh\sin(\phi) \\ -r\sin(\beta+\phi) & r\cos(\beta+\phi) & r(b+h\cos(\phi)) \end{bmatrix}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{2.13} $$

2.4 Kinematics of a multi-caster wheels system

2.4.1 Inverse kinematics of multi-caster wheels

The multi-caster wheels system is constrained by the following:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = {}^{B}J_{E_1} \begin{bmatrix} \dot{\sigma}_1 \\ \dot{\rho}_1 \\ \dot{\phi}_1 \end{bmatrix} = {}^{B}J_{E_2} \begin{bmatrix} \dot{\sigma}_2 \\ \dot{\rho}_2 \\ \dot{\phi}_2 \end{bmatrix} = \ldots = {}^{B}J_{E_N} \begin{bmatrix} \dot{\sigma}_N \\ \dot{\rho}_N \\ \dot{\phi}_N \end{bmatrix} \tag{2.14}$$

Where N is the n-th caster wheel

As such, the inverse kinematics of the multi-caster wheels system can be achieved by the following:

$$\dot{Q} = {}^{B}J_{aug}{}^{-1}\dot{X}$$

$$\begin{bmatrix} \dot{\rho}_1 \\ \dot{\phi}_1 \\ \dot{\rho}_2 \\ \dot{\phi}_2 \\ \vdots \\ \vdots \\ \dot{\rho}_N \\ \dot{\phi}_N \end{bmatrix} = \frac{1}{rb} \begin{bmatrix} b\cos(\beta_1+\phi_1) & b\sin(\beta_1+\phi_1) & bh\sin(\phi_1) \\ -r\sin(\beta_1+\phi_1) & r\cos(\beta_1+\phi_1) & r(b+h\cos(\phi_1)) \\ b\cos(\beta_2+\phi_2) & b\sin(\beta_2+\phi_2) & bh\sin(\phi_2) \\ -r\sin(\beta_2+\phi_2) & r\cos(\beta_2+\phi_2) & r(b+h\cos(\phi_2)) \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ b\cos(\beta_N+\phi_N) & b\sin(\beta_N+\phi_N) & bh\sin(\phi_N) \\ -r\sin(\beta_N+\phi_N) & r\cos(\beta_N+\phi_N) & r(b+h\cos(\phi_N)) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{2.15}$$

whereby β_i and ϕ_i are the wheel position and the steering angle, N is the n-th set of caster wheel

The above equation is used to determine the various PCW joint velocities when the Cartesian velocities are given. This is an instantaneous model that has its reference frame identical to the Base Frame. Detailed description of the hardware implementations will be covered in the subsequent chapters.

2.4.2 Forward kinematics of multi-castor wheels

The Forward Kinematics of the system computes the base position (x, y) and orientation (θ) from the number of rotations of the steering and driving axes of each wheel, hence providing odometry for the base.

2.4.2.1 Model 1

The forward kinematics of the system is derived from the inverse kinematics shown in Equation 2.15 by computing the left pseudo-inverse of the augmented Jacobian matrix. The governing equation is then:

$$\dot{X} = ({}^{B}J_{aug}{}^{-1})_{LPI}\dot{Q} \tag{2.16}$$

where : $({}^{B}J_{aug}{}^{-1})_{LPI} = (({}^{B}J_{aug}{}^{-1})^{T}({}^{B}J_{aug}{}^{-1}))^{-1}({}^{B}J_{aug}{}^{-1})^{T}$

This equation leads to a $\dot{X}$ solution that minimises the difference between the measured velocities and the desired velocities of the mobile base using the least-squares method. It should be noted that ${}^{B}J_{aug}$ is full rank and the pseudo-inverse always exists.

2.4.2.2 Model 2

In the second model, the governing inverse kinematics equation is separated into the mobile base PCW's physical parameters, *b* and *r*, and the non-constant variable, ϕ.

$$\dot{X} = ({}^{B}H_{aug}{}^{-1})_{LPI} B\dot{Q} \tag{2.17}$$

where: $({}^{B}H_{aug}{}^{-1})_{LPI} = (({}^{B}H_{aug}{}^{-1})^{T}({}^{B}H_{aug}{}^{-1}))^{-1}({}^{B}H_{aug}{}^{-1})^{T}$

$$B = \begin{bmatrix} r_1 & 0 & \cdots & 0 & 0 \\ 0 & b_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & r_n & 0 \\ 0 & 0 & \cdots & 0 & b_n \end{bmatrix}$$

$${}^{B}H_{aug}{}^{-1} = \begin{bmatrix} \cos(\beta_1 + \phi_1) & \sin(\beta_1 + \phi_1) & h\sin(\phi_1) \\ -\sin(\beta_1 + \phi_1) & \cos(\beta_1 + \phi_1) & (b + h\cos(\phi_1)) \\ \cos(\beta_2 + \phi_2) & \sin(\beta_2 + \phi_2) & h\sin(\phi_2) \\ -\sin(\beta_2 + \phi_2) & \cos(\beta_2 + \phi_2) & (b + h\cos(\phi_2)) \\ \vdots & \vdots & \vdots \\ \cos(\beta_N + \phi_N) & \sin(\beta_N + \phi_N) & h\sin(\phi_N) \\ -\sin(\beta_N + \phi_N) & \cos(\beta_N + \phi_N) & (b + h\cos(\phi_N)) \end{bmatrix}$$

β_N and ϕ_N are the wheel position and the steering angle, *N* is the *n*-th set of caster wheel

This equation leads to a $\dot{X}$ solution that minimises the difference between the measured velocities and the desired velocities of the contact points using the least-squares method. It should be noted that ${}^{B}H_{aug}$ is full rank and the pseudo-inverse always exists.

Chapter 3

Design and Implementations

The main motivation for this project is the realization of a wheeled mobile robotic platform that is capable of achieving the desired omni-directional capabilities through the use of the PCW mechanisms. The final objectives of the project are:

i. Assemble a mobile robotic platform using the PCW mechanisms,
ii. Complete the kinematics control algorithm and
iii. Development of user interface for commanding the mobile robotic platform including the network system.

Subsequently, this mobile platform will be controlled by the intelligence system which is part of the social robot RoboTubby. The data abstraction for the mobile robot will cater for pose-based and velocity-based control.

3.1 Mechanical design

The mechanical design of the castor structure is to design the driving mechanisms of the castors for rotation and translational motion as shown in Figure 3-1.

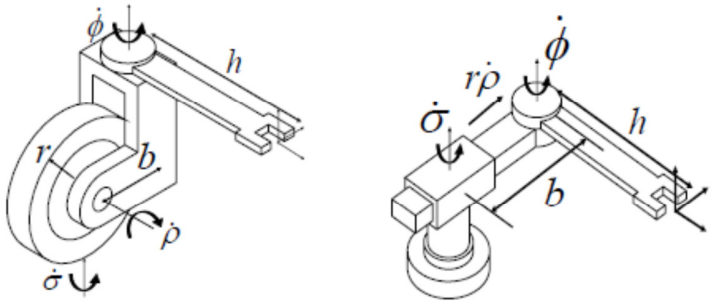


Figure 3-1: Caster Linkage Modeling

Several limitations are inherent in real-world physical systems like the finite turns of the castor steering mechanism given the multiple cables used. In most cases, a close to 360 degrees circle is sufficient for the steering, achieving the motion capabilities of the powered castor wheel design.

Skate scooter wheels such as those shown in Figure 3-2 are selected as wheels for the customized powered castor as its meets the size (100mm diameter), loading and general availability in the market. The wheels are also smooth, making it suitable for indoor environments reducing the risk of floor damage. These skate scooter wheels are capable of carrying even adults in the original skate scooter design and thus should have no issue for RoboTubby.

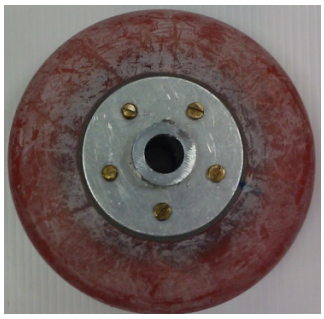


Figure 3-2: Skate Scooter Wheels with customised wheel hub

As for the castor frame, a standard industrial castor wheel frame was selected such that it is large enough for the offset to be the radius of the chosen wheel. Nevertheless, several complications arose from implementation of this PCW design in terms of drive and steer alignment given that standard industrial castors do not have stringent specifications.

A Robotis Servo (RX-28) is selected to be the driving motor since it is capable of continuous rotation. Being an intelligent servo, it is capable of relaying its operation information such as speed and orientation during its operation. When used as a servo motor, it has building PID control to achieve its target position

in user defined speed. Nevertheless, there are limitations to using Robotis Servo as position information is only for 300 degrees of operation. Figures 3-3 and 3-4 shows the modules to make the PCW with drive motor.

Figure 3-3: Robotis RX-28 servo and Standard Industrial castor wheel frame

Figure 3-4: Wheel driving unit of Powered Castor Wheel

Developing the steer mechanism is more complicated than the drive mechanism given the standard castor wheels do not have good alignment. One immediate future work is to fully fabricate the castor wheel instead of using standard parts to reduce misalignment which will case more noise, more wear and tear and requiring bigger motors to drive the system. Figure 3-5 shows the completed module with 2 pairs of PCW with steer and drive motors.

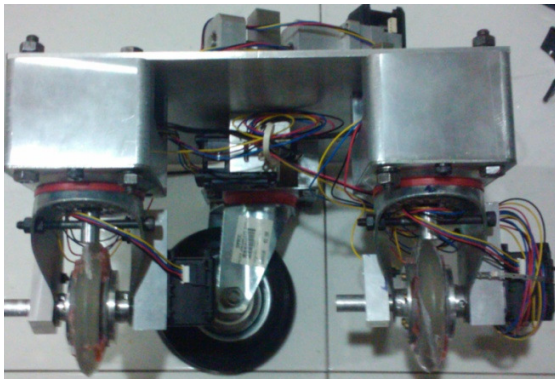


Figure 3-5: Powered Castor Wheels for RoboTubby

Table of Key Mechanical parameters:

Description	Value	Remark
Wheel radius	100 mm	
Wheel offset	50 mm	Horizontal distance from steer motor shaft center to wheel center
Platform radius	125 mm	From center of platform to steer motor shaft

Prior to software, the main computer systems that will power the current iteration of Robotubby is as follows:

Equipment	Purpose	Remark
X86 touch screen industrial PC	House main operating software including all high-level functions	The current PC is Intel Atom based which is relatively low power solution that runs a Windows OS for rapid development
USB Dynamixel controller	Provides hardware interface to connect to the motors which require RS-485 protocol or TTL	All the platform motors operate on RS-485 protocol
Atmel-based embedded controller	Perform other low-level functions such as gather sonar data for obstacle avoidance	Future versions include controlling the mobile platform, freeing the CPU to perform high level actions

Other features of the current iteration are as follows:

Equipment	Description	Remark
Battery pack	Provide direct 12V/24V supply to all systems.	For signal controllers and the industrial PC, regulators are necessary to ensure signal integrity and reduce risk of processor failures
Web camera	Captured images for high level processing	

3.2 Software system:

Microsoft CSharp (*C#)* was selected as the main running program of RoboTubby and as such will be used for programming of the PCW mobile base. The basis of the winform design is multithreading and the winform acts as the Graphical User Interface (GUI) capturing user input and providing updates of asynchronous events generated by algorithms. The general idea is presented in Figure 3-6.

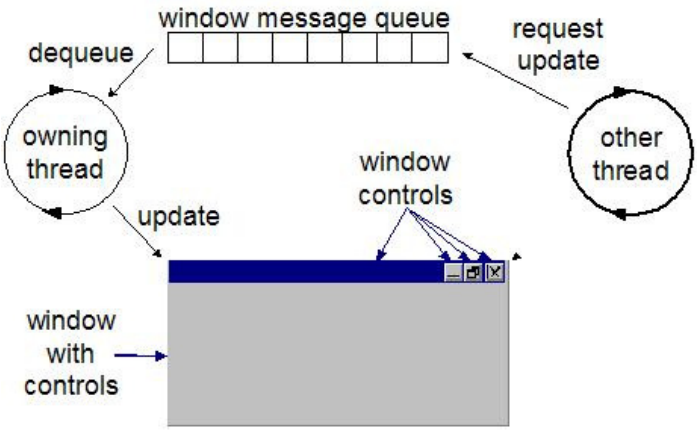


Figure 3-6 : Multithreading in Winform

The current implementation of the GUI is to test the mobility both locally and over the network. Figure 3-7 shows the current implementation.

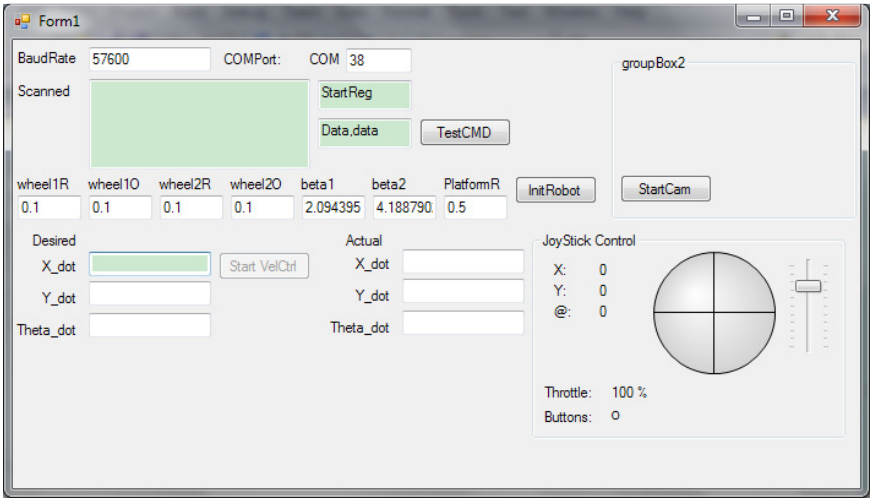


Figure 3-7: Current GUI for functional testing

Function	Feature	Remark
Velocity testing	Enter desired values	
Trajectory testing	Enter desired values	
Tele-operation	Joystick operation	Virtual and physical joystick differ in rotational is separate for virtual

The flowchart of the UI implementation is shown in Figure 3-8.

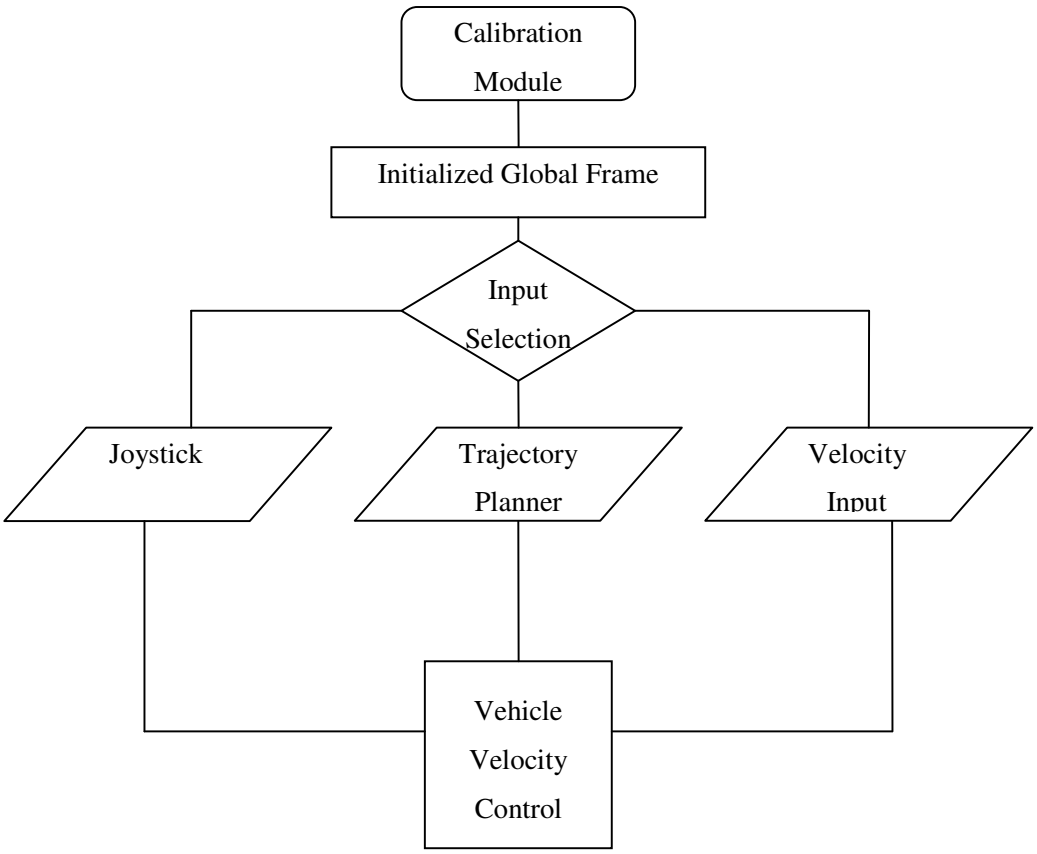


Figure 3-8: flowchart of operation

3.2.1 Global Coordinate Frame

The Global Coordinate Frame is initialized after the Calibration module and its origin is defined to be coincidental with the Base Frame at its starting position as shown in Figure 3-9. As the inverse kinematics of the mobile base is based on an instantaneous model, a Global Frame is important to allow the mobile base to translate and rotate simultaneously.

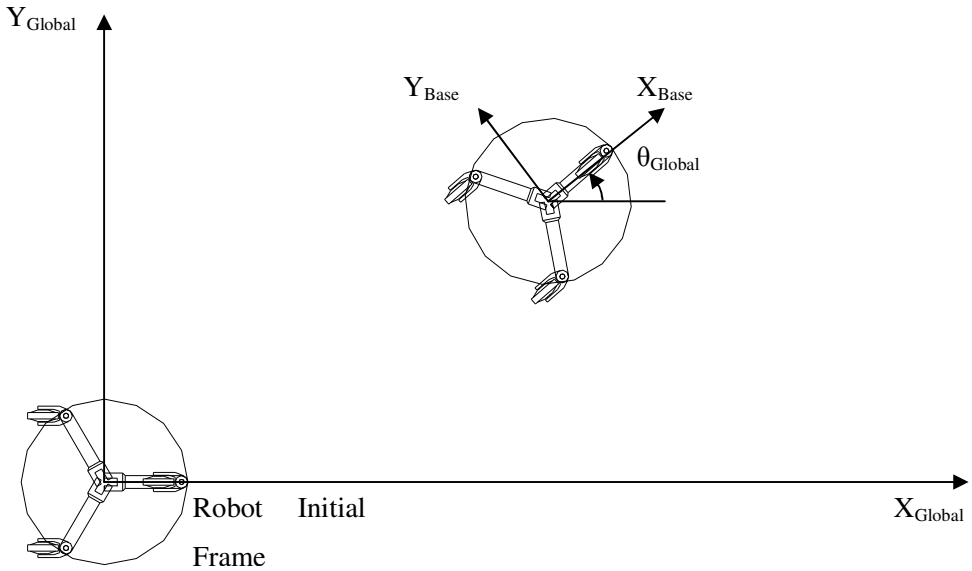


Figure 3-9: Relationship between Global Frame and Base Frame

Equations 3.1 and 3.2 show the mathematical relationship between the global frame and the Base frame. These equations are used in the Velocity Control Module and for odometry feedback.

$$\begin{bmatrix} \dot{X}_{Global} \\ \dot{Y}_{Global} \\ \dot{\theta}_{Global} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{X}_{Base} \\ \dot{Y}_{Base} \\ \dot{\theta}_{Base} \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} \dot{X}_{Base} \\ \dot{Y}_{Base} \\ \dot{\theta}_{Base} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{X}_{Global} \\ \dot{Y}_{Global} \\ \dot{\theta}_{Global} \end{bmatrix} \tag{3.2}$$

.

3.2.2 Velocity Input Mode

Velocity Input mode is the simplest input mode for the mobile base. It basically takes in the three Global Frame Velocities and the time duration for the mobile base to move in the specific velocities. It then sends the velocities to the Velocity Control Module using a step velocity input.

Though simple, this mode provides a comprehensive method for initial testing and verification of the system. This control method is also well-suited for use with the joystick for both local and network control.

3.2.3 Trajectory Planner

The above two methods allows the user to move the mobile base by specifying the velocities of the mobile base. While these methods allow control of the base, the user is not able to determine an accurate position for the movement of the mobile base. Determining accurate position is vital for autonomous functioning of social robots.

A trajectory is the path followed by the manipulator, plus the time profile along the path. Trajectories can be planned either in joint space (directly specifying the time evolution of the joint angles) or in Cartesian space (specifying the position and orientation of the end frame). Issues in trajectory planning include attaining a specific target from an initial starting point, avoiding obstacles, and staying within manipulator capabilities.

The Trajectory Planner function allows the user to determine the final position and orientation of the mobile base with regards to the Global Frame and the time duration required to accomplish the task. The Trajectory Planner utilizes the Cubic Polynomial system to achieve a smooth velocity profile that is fed to the Vehicle Velocity Control Module. Equations 3.3 to 3.5 show the formulation of the Cubic Polynomial System.

$$\begin{bmatrix} x_{(t)} \\ y_{(t)} \\ \theta_{(t)} \end{bmatrix} = \begin{bmatrix} a_{x0} \\ a_{y0} \\ a_{\theta 0} \end{bmatrix} + \begin{bmatrix} a_{x1} \\ a_{y1} \\ a_{\theta 1} \end{bmatrix} t + \begin{bmatrix} a_{x2} \\ a_{y2} \\ a_{\theta 2} \end{bmatrix} t^2 + \begin{bmatrix} a_{x3} \\ a_{y3} \\ a_{\theta 3} \end{bmatrix} t^3 \tag{3.3}$$

$$\begin{bmatrix} \dot{x}_{(t)} \\ \dot{y}_{(t)} \\ \dot{\theta}_{(t)} \end{bmatrix} = \begin{bmatrix} a_{x1} \\ a_{y1} \\ a_{\theta 1} \end{bmatrix} + 2\begin{bmatrix} a_{x2} \\ a_{y2} \\ a_{\theta 2} \end{bmatrix} t + 3\begin{bmatrix} a_{x3} \\ a_{y3} \\ a_{\theta 3} \end{bmatrix} t^2 \tag{3.4}$$

$$\begin{bmatrix} \ddot{x}_{(t)} \\ \ddot{y}_{(t)} \\ \ddot{\theta}_{(t)} \end{bmatrix} = 2\begin{bmatrix} a_{x2} \\ a_{y2} \\ a_{\theta 2} \end{bmatrix} + 6\begin{bmatrix} a_{x3} \\ a_{y3} \\ a_{\theta 3} \end{bmatrix} t \tag{3.5}$$

As there are four unknown results, there must be at least four separate results to obtain the required solutions. The four equations arise from the constraints on the system. The first two constraints take into account the initial and final values of the system while the last two constraints are required for the function to be continuous in velocity. Hence the following is obtained:

$$\begin{aligned} X_{(t=0)} &= 0 \\ X_{\{t=T_{Fina})} &= X_{Final} \\ \dot{X}_{(t=0)} &= 0 \\ \dot{X}_{(t=T_{Fina})} &= 0 \end{aligned} \tag{3.6}$$

The equations linking the coefficients to the constraints are as follows:

$$\begin{aligned} X_{(t=0)} &= 0 = a_{X0} \\ X_{\{t=T_{Final})} &= X_{Final} = a_{X0} + a_{X1}T_{Final} + a_{X2}T_{Final}{}^2 + a_{X3}T_{Final}{}^3 \\ \dot{X}_{(t=0)} &= 0 = a_{X1} \\ \dot{X}_{(t=T_{Final})} &= 0 = a_{X1} + 2a_{X2}T_{Final} + 3a_{X3}T_{Final}{}^2 \end{aligned} \tag{3.7}$$

Using the above, it is then possible to determine the coefficients:

$$
\begin{aligned}
a_{X0} &= 0 \\
a_{X1} &= 0 \\
a_{X2} &= \frac{3}{T_{Final}^{\ 2}}(X_{Final}) \\
a_{X4} &= -\frac{2}{T_{Final}^{\ 3}}(X_{Final})
\end{aligned}
\tag{3.8}
$$

With Eq 3.8, the position function obtained is smooth with respect to time while the velocity function is a parabola. The acceleration term has a linear profile.

3.2.4 Vehicle Velocity Control Module

The Vehicle Velocity Control Module is the most important module in the control algorithm. It takes in three values, namely the translational velocity, $\dot{x}$ and $\dot{y}$, and the rotational velocity, $\dot{\theta}$. These values can be defined in either the Global Frame of reference or in the Base Frame Cartesian Coordinates.

After the velocities are input, a Proportional-Integral Velocity feedback controller is applied to compensate for errors in the process. Such a feedback loop is especially crucial when a Trajectory Planner is used to obtain the desired velocities.

When using the Global Frame Coordinates, Equation 3.2 is utilized to transform the Global velocities to the Base Frame velocities as the kinematics modeling is an instantaneous model at the mobile base own frame. After obtaining the Cartesian velocities, the Inverse Kinematics obtained in Chapter 2 is applied to determine the four PCW joint velocities, $\dot{Q}$ (Eq (3.9)). To determine the odometry data, the Forward Kinematics is applied.

$$
\begin{bmatrix} \dot{\rho}_{Lg} \\ \dot{\phi}_{Lg} \\ \dot{\rho}_{Sm} \\ \dot{\phi}_{Sm} \end{bmatrix}
=
\begin{bmatrix}
\cos(\beta_{Lg}+\phi_{Lg})/r_{Lg} & \sin(\beta_{Lg}+\phi_{Lg})/r_{Lg} & h\sin(\phi_{Lg})/r_{Lg} \\
-\sin(\beta_{Lg}+\phi_{Lg})/b_{Lg} & \cos(\beta_{Lg}+\phi_{Lg})/b_{Lg} & (b_{Lg}+h\cos(\phi_{Lg}))/b_{Lg} \\
\cos(\beta_{Sm}+\phi_{Sm})/r_{Sm} & \sin(\beta_{Sm}+\phi_{Sm})/r_{Sm} & h\sin(\phi_{Sm})/r_{Sm} \\
-\sin(\beta_{Sm}+\phi_{Sm})/b_{Sm} & \cos(\beta_{Sm}+\phi_{Sm})/b_{Sm} & (b_{Sm}+h\cos(\phi_{Sm}))/b_{Sm}
\end{bmatrix}
\begin{bmatrix} \dot{x}_{Base} \\ \dot{y}_{Base} \\ \dot{\theta}_{Base} \end{bmatrix}
\tag{3.9}
$$

3.2.5 PD Controller

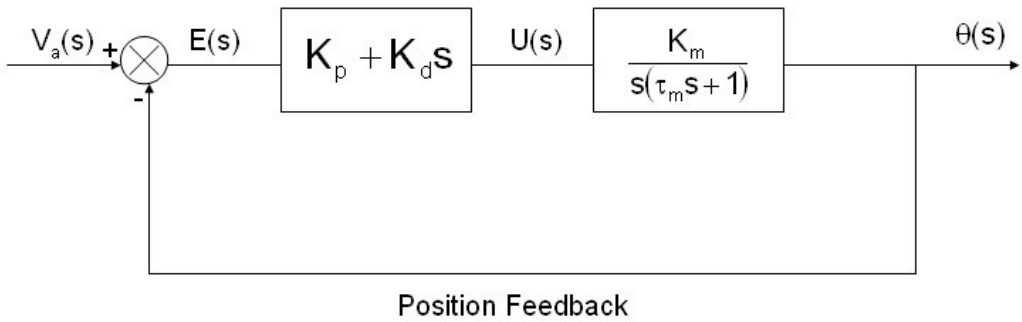


Figure 3-10: PD control for servo system

The transfer function of the PD controller in Fig 3-10 is

$$\frac{U(s)}{E(s)} = K_p + K_d s \tag{3.10}$$

- K_p = Proportional gain
- K_d = Derivative gain

The closed-loop transfer function of the system is

$$\frac{\theta(s)}{V_a(s)} = \frac{K_m(K_p + K_d s)}{\tau_m s^2 + (1 + K_m K_d)s + K_m K_p} \tag{3.11}$$

The characteristic equation is

$$\tau_m s^2 + (1 + K_m K_d)s + K_m K_p \tag{3.12}$$

The natural frequency is

$$\omega_n = \sqrt{\frac{K_m K_p}{\tau_m}} \tag{3.13}$$

and the effective damping ratio is

$$\zeta = \frac{1 + K_m K_d}{2\sqrt{K_m K_p \tau_m}} \tag{3.14}$$

The steady state error ess (ramp) is equal

$$e_{ss}(ramp) = \frac{1}{K_V} = \frac{1}{K_p K_m} \tag{3.15}$$

A PD-controller is used to improve the transient response of a closed-loop system. The overshoot is reduced by changing the damping ratio and the rise time is improved. It has no additional effect on the steady state error as compared to proportional controller.

3.2.6 PID Controller

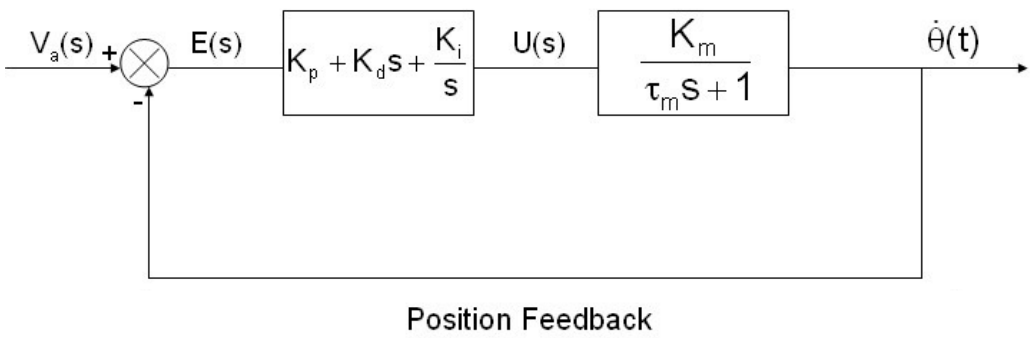


Figure 3-11: PID control for servo system

The transfer function of the PID controller in Fig 3-11 is

$$\frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{3.20}$$

- K_p = Proportional gain
- K_i = Integral gain
- K_d = Derivative gain

The transfer function of the closed loop system is

$$\frac{\dot{\theta}(s)}{V(s)} = \frac{\frac{K_m K_d}{K_m K_d + \tau_m} s^2 + \frac{K_m K_p}{K_m K_d + \tau_m} s + \frac{K_m K_i}{K_m K_d + \tau_m}}{s^2 + \frac{1 + K_m K_p}{K_m K_d + \tau_m} s + \frac{K_m K_i}{K_m K_d + \tau_m}} \tag{3.21}$$

The natural frequency is

$$\omega_n = \sqrt{\frac{K_m K_i}{K_m K_d + \tau_m}} \tag{3.22}$$

and the effective damping ratio is

$$\zeta = \frac{1 + K_m K_p}{2\sqrt{K_m K_i (K_m K_d + \tau_m)}} \tag{3.23}$$

and the steady state ramp error is

$$e_{ss}(ramp) = 0 \tag{3.24}$$

A PID controller is used to improve the response of a closed-loop system by increasing the type of the system to two such that the steady state step and ramp errors are both zero. The overshoot is reduced by changing the damping ratio by increasing K_D and the rise time can be improved by changing K_P.

3.2.7 PI Control Loop

A PI Feedback Control is chosen for the Velocity Control Module. It is governed by the following formula:

$$U = Kp(Error) + Ki(Integral\_Error) \tag{3.25}$$

where: $Error = \dot{X}_{Desired} - \dot{X}_{Actual}$

$$Integral\_Error = \sum (Error \times Sampling\_Time)$$

$$U = outputSignal$$

The PI controller is chosen to achieve accurate position control of the system especially when a trajectory planner is used for the system since the velocity errors will result in an accumulation of position errors. However, because of the nature of the governing equations, the main command inputs and feedback are in velocities instead of positions. The Integral Error term aims to correct the positional errors in the system since position is obtained through the integration of velocities.

3.2.8 Tuning

To achieve desired characteristic of the controller, it is vital that the controller constant are tuned accurately. The software algorithm for the controllers is as follows:

The PD controller used is of the form:

$$\text{Voltage Out} = K_p\,(\text{error}) + K_d\,(\text{error} - \text{error\_previous}) \,/\, \text{sampling time} \qquad (3.26)$$

The PID equation used is of the form:

$$\begin{aligned}\text{Voltage Out} = {} & K_p(\text{error}) + \\ & K_i(\text{sum of (error x sampling time)} + \\ & K_d(\text{error - error\_previous}) \,/\, \text{sampling time}\end{aligned} \qquad (3.27)$$

where: error = Desired Position – Current Position

sampling time = time duration of each control loop

K_p, K_i and K_d = controller constants that must be tuned

There are three main methods to tune the constants of the PD or PID controllers as shown below.

Table 3.1: Different methods of tuning controller constant

Analytical	Based on theory and accurate measurements of the physical parameters of the apparatus, derive a *transfer function* that describes the response of the system to changes in its input values. Chose parameters using standard techniques from control theory.
Empirical	Using bench-test results for the overall system, apply general-purpose procedures such as the Zeigler-Nichols method which compute parameters based on observed system behavior.
Trial and Error	Starting with an "initial guess", observe the behavior of the system and tweak the parameters until a stable solution is found.

The Ziegler-Nichols Tuning method is used as a guideline for determining the controller constants as it provides a basis for initial testing without elaborate experimentation to determine the exact motor constants.

Table 3.2: Ziegler-Nichols Tuning Guide

Type of Controller	K_p	T_i	T_d
P	0.5 x K_{cr}	-	-
PI	0.45 x K_{cr}	$P_{cr}/1.2$	-
PID	0.6 x K_{cr}	0.5 x P_{cr}	0.125 x P_{cr}

K_{cr} and P_{cr} are obtained experimentally. K_{cr} is the critical gain where the shaft exhibits sustained oscillations. P_{cr} is the period of the oscillations. For the PID controller, $K_p = K_p$, $K_i = K_p / T_i$ and $K_d = K_p \times T_d$. The following graphs show the experimental results of the individual motors when in critical state.

PID control

In earlier designs, it is necessary to consider the DC motor dynamics and perform the PID to regulate the speed in the case of the inner most PID control loop. In the current design, the dynamixel motors are supposed to perform that role which it did but only in positional control, not continuous turn for the drive motors. Consequently, a simple PID loop shown in Figure 3-12, is considered for the dynamixel servo motors which receive input of power and through the speed feedback, control the desired velocity.

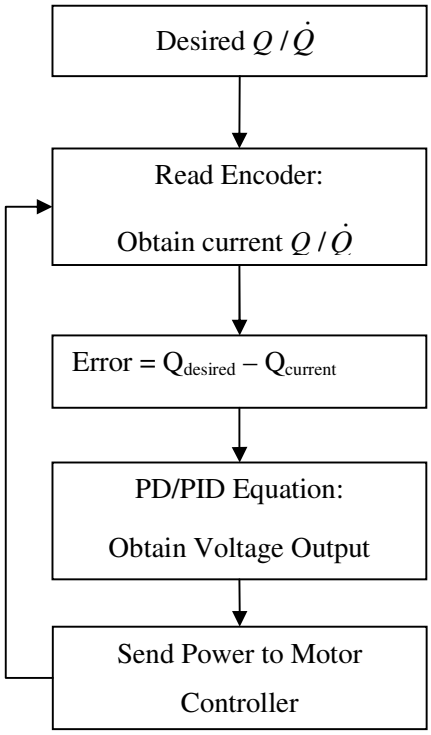


Figure 3-12: Motor Control loop

3.3 Network control and implementation

One fundamental purpose of social robots is tele-operation and for the case of Robotubby, parents can tele-operate our social robot and communicate with the child bridging physical distance. From implementations of video conferencing and remote surveillance, several potential issues with such an implementation can be observed, namely latency and when robots are concerned safety issue.

Prior to addressing latency and safety issues, implementation of global network systems requires knowing how the internet works and connections be made. Figure 3-13 shows the outline of the internet system - especially how computers are connected and can interact with each other.

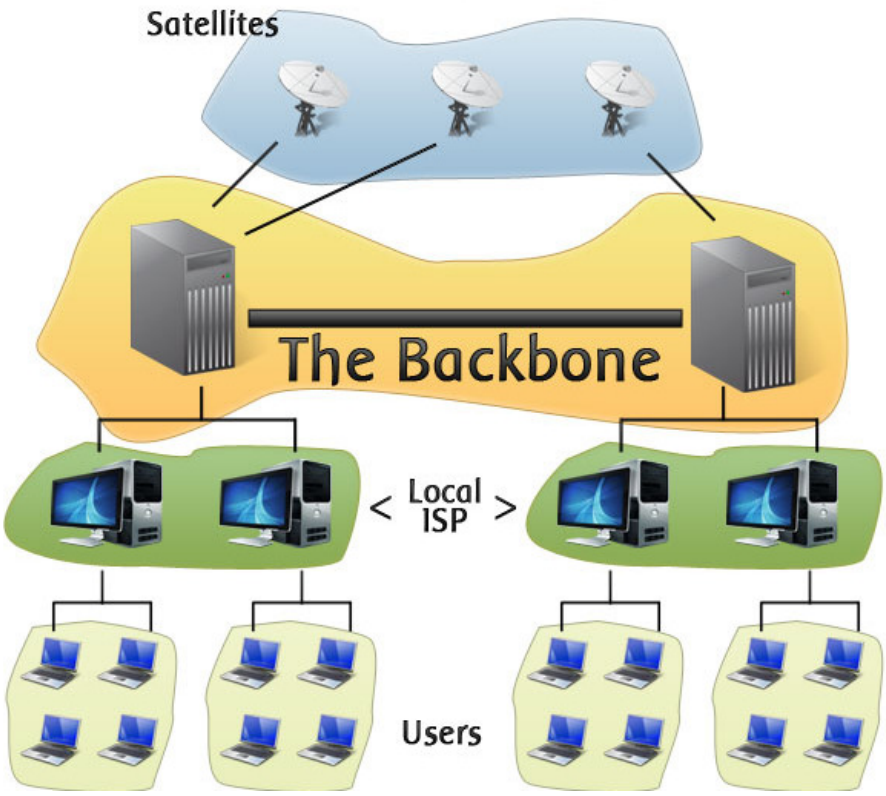


Figure 3-13: Basic Understanding of Internet Operation

3.3.1 Integration with Robotubby

Socket communication is one of the means for computers to connect to each other and pass information. The most common 2 protocols used are Transmission Control Protocol (TCP/IP) and User Datagram Protocol (UDP). The key tradeoffs of these 2 protocols are between reliability and speed. Other variants include Real-Time Protocol mostly used for video-conferencing. Figure 3-14 shows the GUI from the remote site.

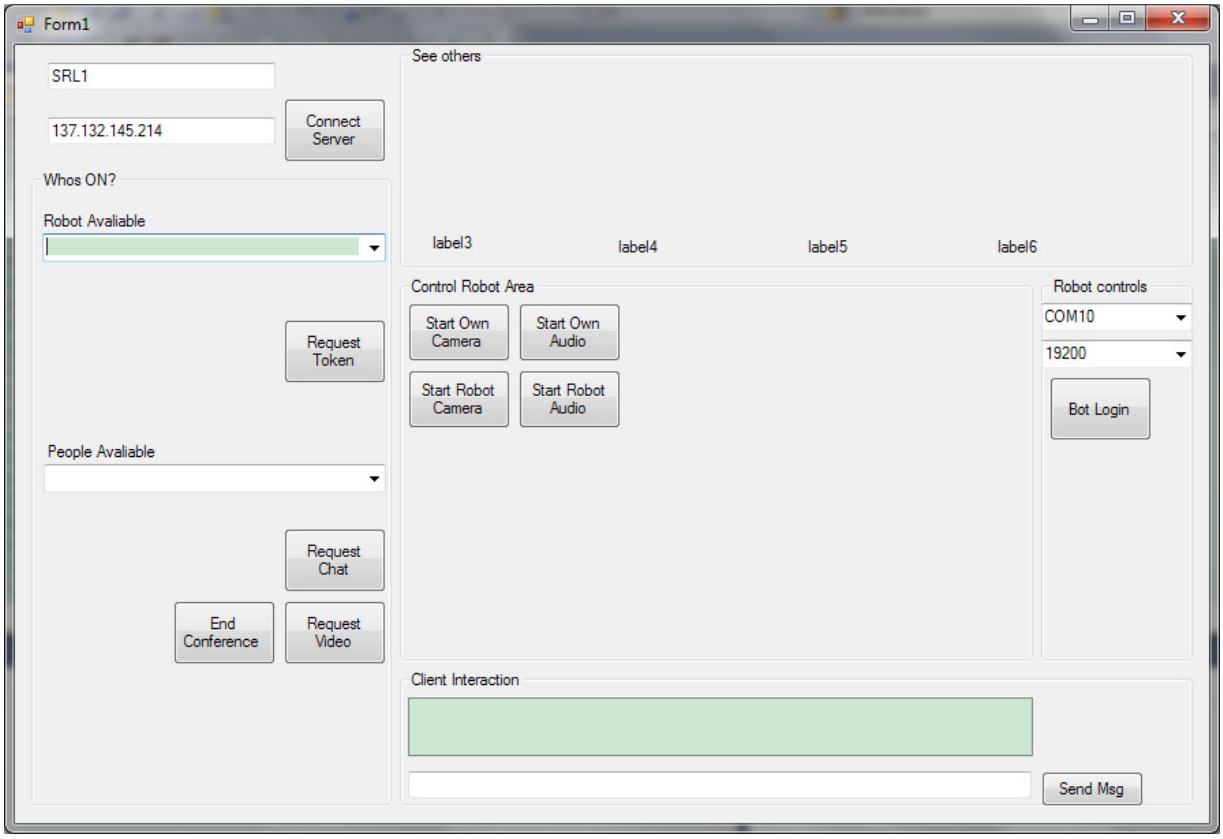


Figure 3-14: Trial implementation of Networked control

In order to solve the trap subnet where the parent's computer cannot connect to the robot, a server with a fixed IP is implemented to route data over the entire internet. The server may perform other functions such as verify user data to validate the connection and data transferred.

Chapter 4

Review of Implementations

Despite many prior efforts in research of social robots have been open for public and institutional learning, design and development of a social robot requires several iterations. At the current development, the mobile platform is the 2nd iteration of the PCW design but first iteration using current controllers which are also used in the head and arm modules. The dynamixel controllers are modular bringing the benefit of easy replacement. However, controlling of the specific model, RX-28, is rather cumbersome given little documentation especially in sensor feedback.

4.1.1 Hardware design and development

In the design of the Powered Castor Wheel module, using standard of the shelf castors as the foundation for modification was not a wise choice given the lack of stringent standards for mechanical tolerance. The current implementation of the module is sufficiently small for Robotubby but is too cumbersome and heavy for proper motion. One of the earliest design specifications for the robot to cross small ledges commonly found in homes and office or cables lying on the ground was not met given the complexity of driving a wheel not on a fixed plane.

The improvement of the hardware is direct coupling of the drive motor to the wheel which improves the efficiency of the motor and a smaller motor can be used. However the immediate drawback is the wheels cannot rotate continuously in the steer direction. Caution must then be taken in the initialization of the base given so that commands to drive the robot will not cause the connection cables to twist around the PCW and break. By observation, proper control of the PCWs can alleviate the limitations of such a design by setting a maximum rotation of slightly above a complete 360 degrees rotation and reverse the steer if necessary.

4.1.2 Experimental data: Servo

The Dynamixel servomotors were selected due to its compactness and relative ease of programming with the supplied library which is available for many languages. Fundamentally, these hobbyist robotic servomotors are "smart" - it contains its own PID implementation and communication with it is via serial. However, being hobbyist-grade servomotors, the data feedbacks are prone to errors despite setting different delays between read and write. Using Matlab to perform the tests, it can be observed the misinformation occurs at 2 main areas, namely very high values when the servomotors are moving and high values when motors are stationary. The most likely reason is the dynamixel servomotor did not perform integrity check prior to sending out the data and errors can occur when the internal microprocessor is updating the current status and a read out request was performed. The example of error readings is shown in Figure 4-1 with no delay in writing commands and reading feedback. The servo motor commands trace a sine wave of 0.5 Hz

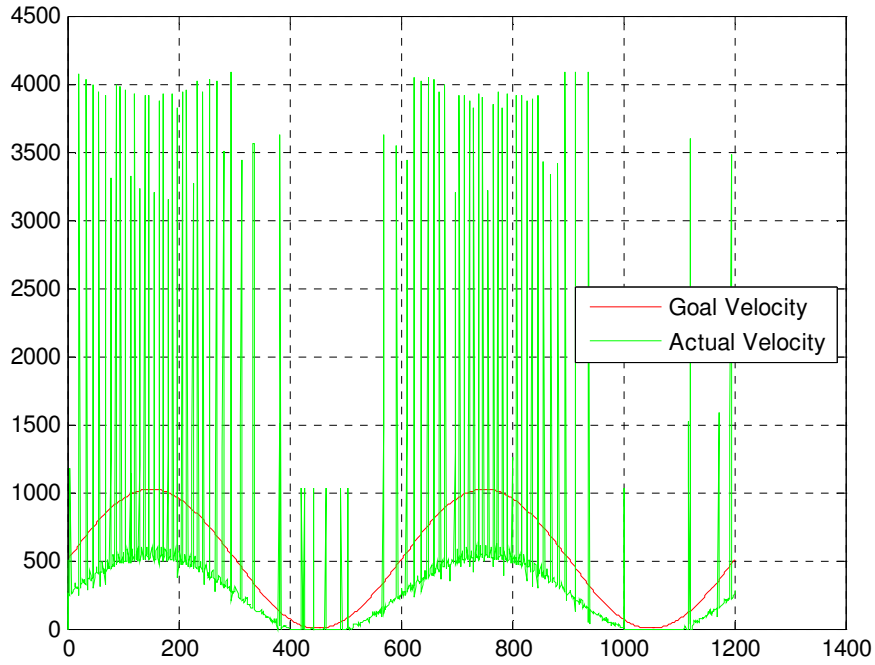


Figure 4-1: Sine wave setpoint for the motor to trace without delay

The high values of the servo motor signals (the valid data range is 10bits) are easy to filter as they are irrational for the servo motor. The main problem is when the motor speed reaches 0 and the data feedback records a high value which is a valid value. The most likely reason is the servo uses the same range for speed control on both directions and rapid swings of the servo may cause it to register an overflow value to the other direction. Figure 4-2 shows the Servomotor tracing the sine curve with bit masking (0x3FF).

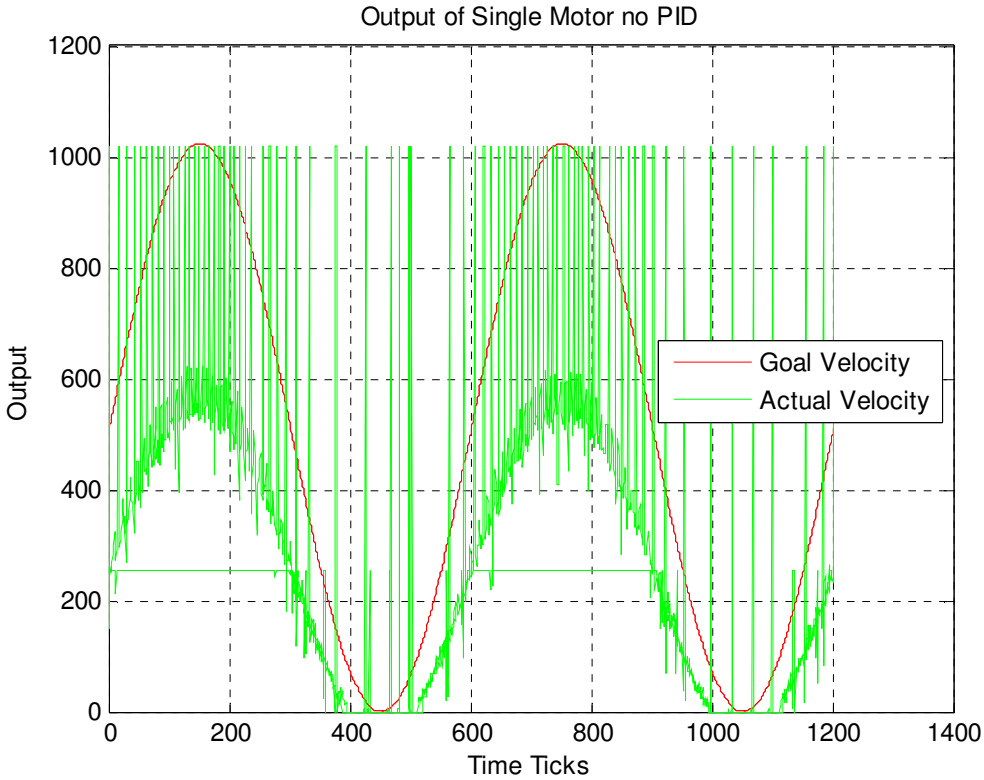


Figure 4-2: Sine wave setpoint for motor to trace with delay of 0.1ms

The impulse response of the servomotor is shown in Figure 4-3 and Figures 4-4 and 4-5 shows the Servomotor tracing the square wave input and the zoomed in view. Only an educated guess of the rise time is possible at current implementation given the systems used are not real-time. The rise is relatively linear and considered optimal.

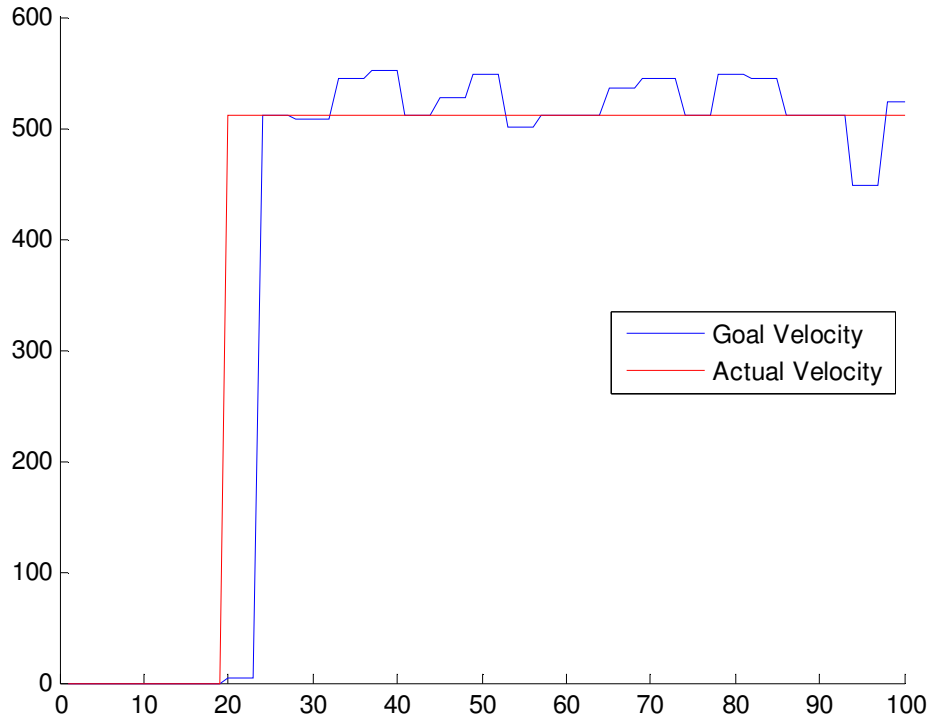


Figure 4-3: RX28 servo response to maximum speed impulse

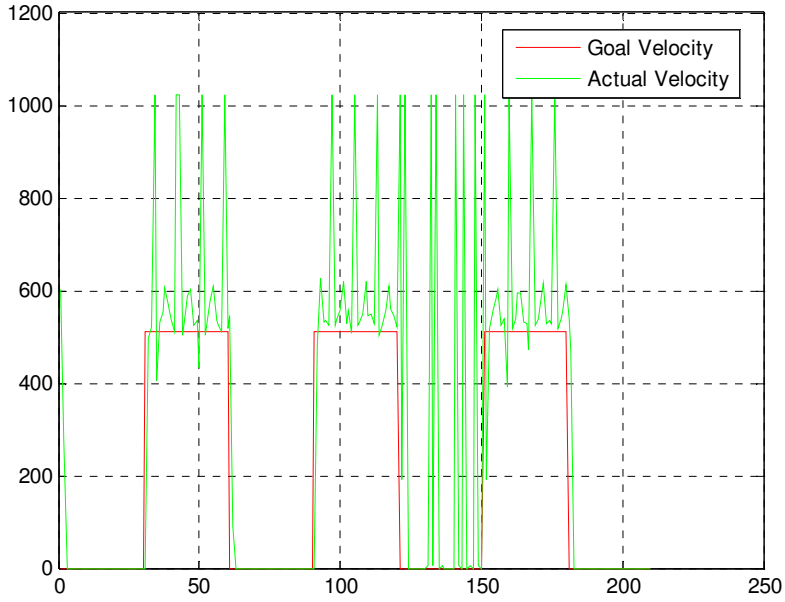


Figure 4-4: Rx 28 Servomotor Square wave input

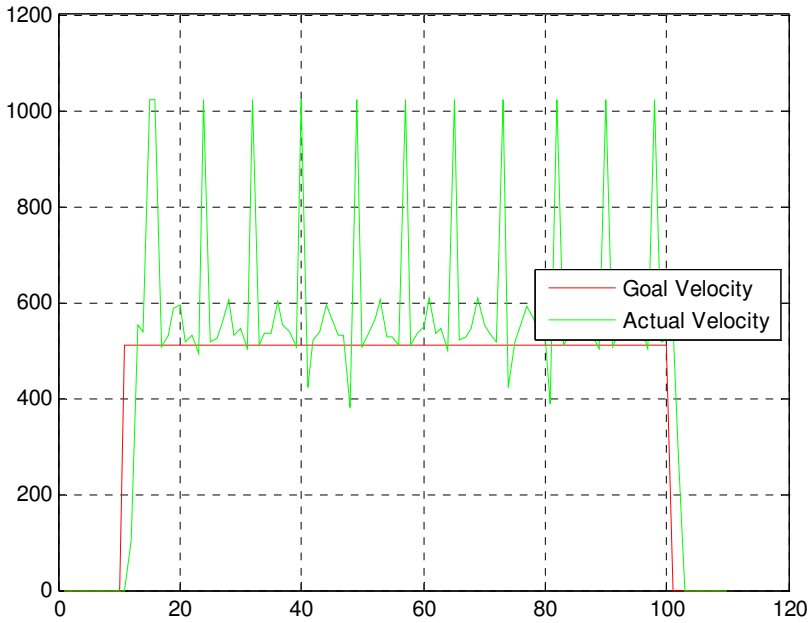


Figure 4-5: Zoomed in view of Square wave input

4.1.3 Experimental data: robot

The platform can be controlled by velocity control with little slippage. In the current design, the steering wheels have a dead band of 60 degrees can the current software will rotate the mobile platform in the reverse direction. A mechanical stopper also exists in each powered castor wheel, preventing the steering from performing a full turn which will result in damage to the wiring. Figure 4-6 shows the orientation and reference of the actual robot to the base frame and each PCW frame resolved to base frame.

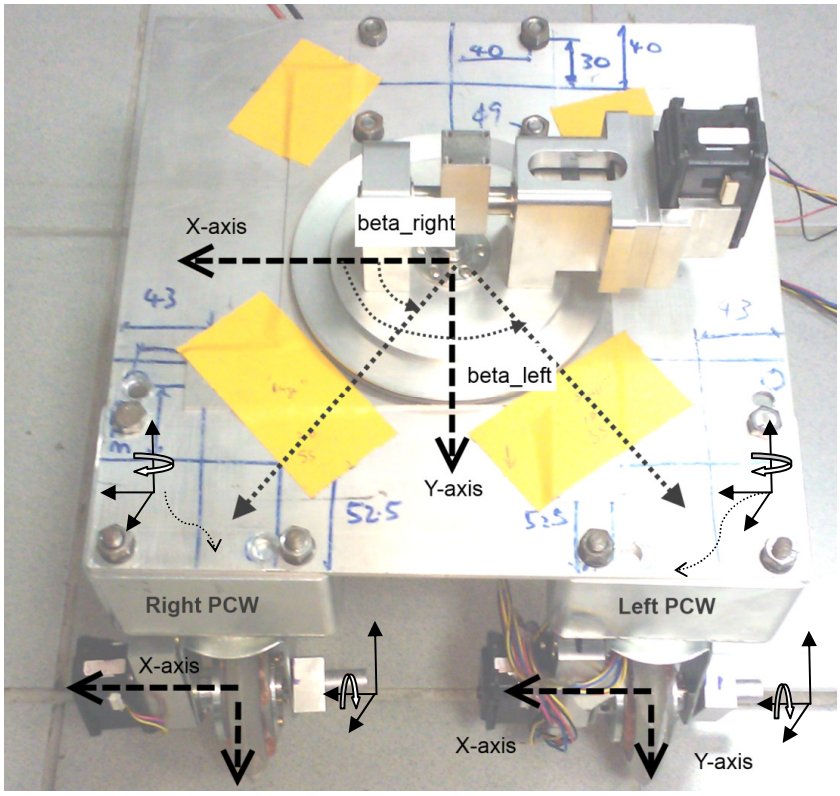


Figure 4-6: Current design of PCW platform

The motion of the steer mechanism can move from 0 degrees to 180 degrees with respect to the base. When the resultant angle of the steering is beyond this range, the angle sent to the steering servo is resolved via addition by 180 degrees for less than 0 degrees or subtraction by 180 degrees for greater than 180 degrees.

The first experiment was to perform translation motion on the platform. The usage of a servo motor provided good steering performance with no significant change in orientation. However, given the limitation of the servo motor steering angle detection, the platform experienced sharp charges in steering angles and drive direction when the direction switches more than 180°. There is no significant slippage noticed as well.

Translation in X and Y direction via velocity control is performed and the robot platform was able to follow in the path relatively well. The distance travelled is only indicative as the timer in the Winform is not real-time. The Figure 4-7 shows the graphs of translation command (1st and 2nd graph) values with 0 rotation (3rd graph) given to the robot and the resultant left rho, left phi, fight rho and right phi values in the last 4 graphs. The robot was given a positive Y translation then a negative X translation followed by positive X translation and lastly negative X and Y translation much like an inverted right angle triangle path.

A video of the robot translation can be downloaded here:

http://www.4shared.com/video/XjF_GOb9/TranslationPCW.html

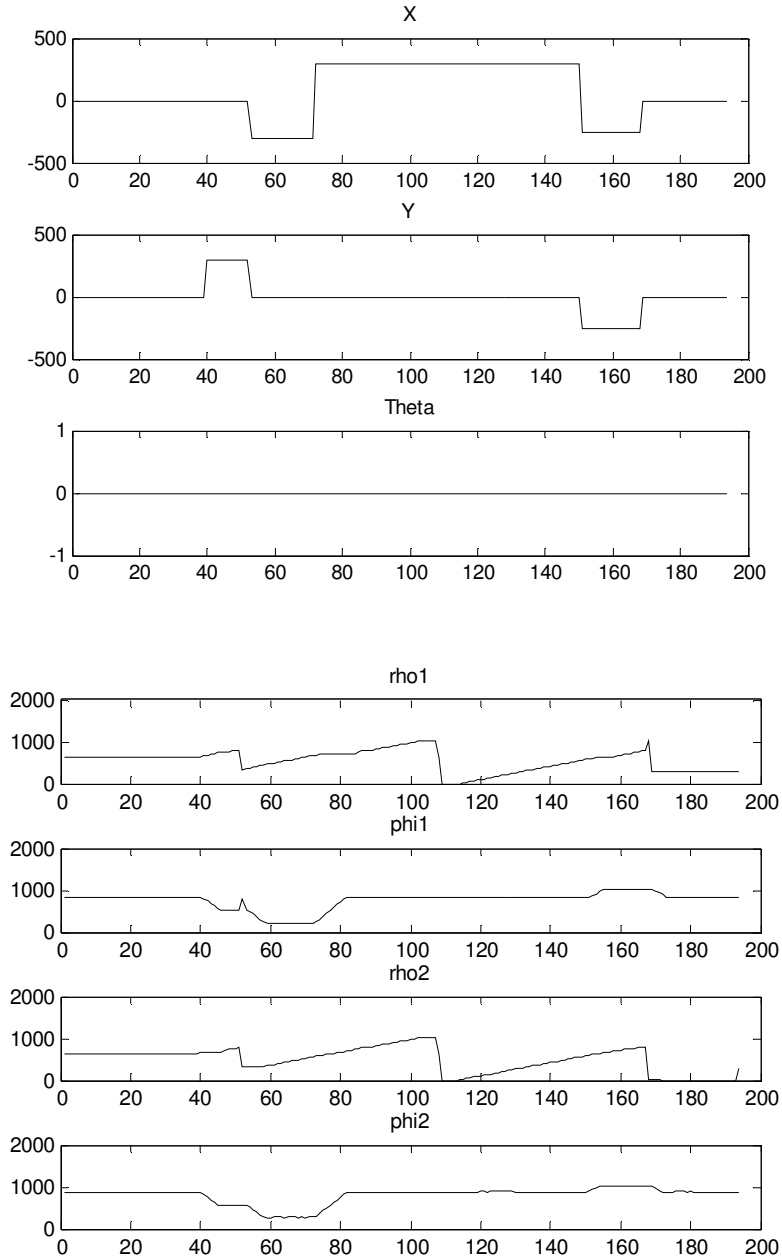


Figure 4-7: Omni-directional drive translation motion

Figure 4-8 showed the commands and resultant graphs of an X direction translation while performing a negative (with respect to the base frame) orientation change. A video of the test can be found:

http://www.4shared.com/video/_0bzLfyo/OrientationPCW.html

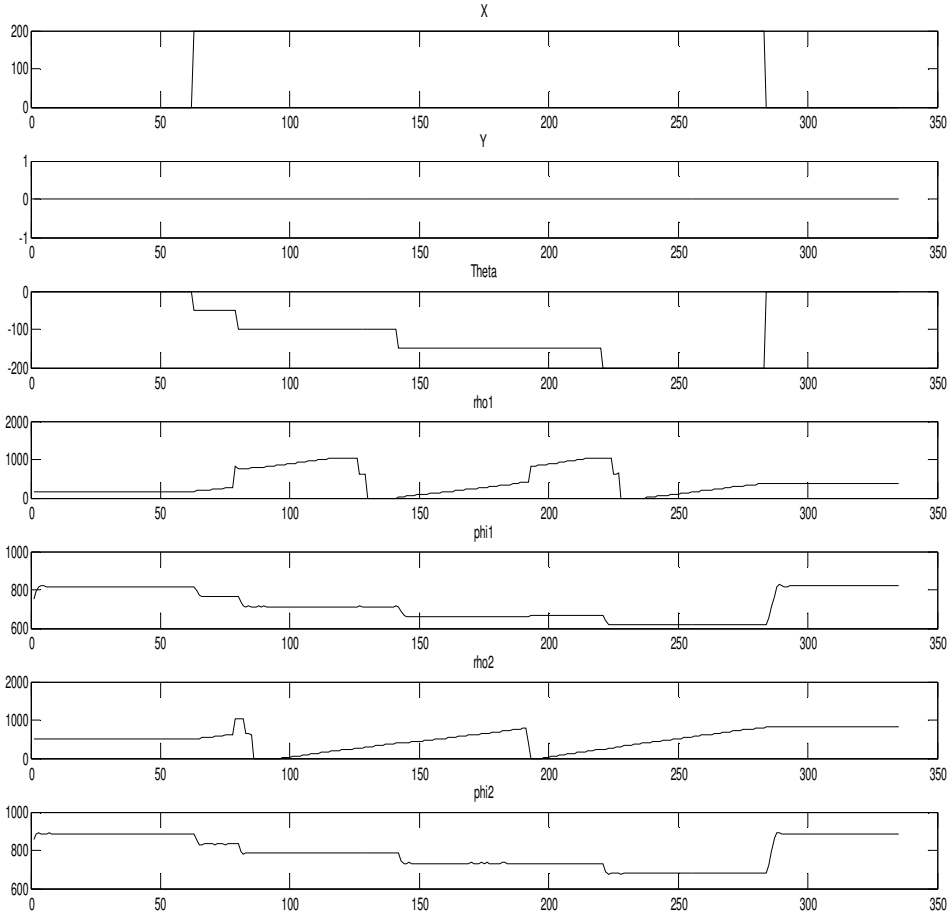


Figure 4-8: Omni-directional rotational and translational motion

The PCW platform was not able to perform a proper omni-directional orientation change drive as there was significant slippage when rotation was performed. However, the servo motors did perform its task from the last four graphs in Figure 4-8. The likely cause is the use of scooter wheels which although they performed well in translation motion failed to perform orientation. When each PCW was not aligned or when both drive velocities were different, slippage was significant.

The video of experiments using velocity control can be obtained from:

Video 1: http://www.4shared.com/video/VRbIvWF6/IMG_0380.html

Video 2: http://www.4shared.com/video/tHH3ulfQ/IMG_0381.html

Video 3: http://www.4shared.com/video/0tXWeDdg/IMG_0383.html

Video 4: http://www.4shared.com/video/IMhzQDTp/PCWTrippingOver.html (early result with no steer limitation)

Chapter 5

Conclusions

Using servomotors such as the RX-28 servos from Dynamixel makes controlling the servo simpler via serial commands. The servos have build-in controllers that can regulate their own speed, turn to a fixed position and most importantly feedback the data to the computer. Such a design facilitated development and integration for Robotubby. However, the key issues with using these Servomotors are the rates of communication available. At high communication baud rates, frequent erroneous readings occur and when the motor is stationary, more filtering is necessary. Applying the daisy chained design with these servomotors, higher baud rates will not work as the delay propagation becomes more significant. Another issue is with the current selection of wheels which introduced significant slippage during rotational test.

In conclusion, a mobile platform suitable for Robotubby was developed. The control system for the platform was implemented in software with necessary features for easy integration into Robotubby architecture. Velocity control was tested in the mobile platform and the kinematics was tested to achieve the desired omni-directional holonomic capabilities. The mobile platform constructed is suitable in size and function of Robotubby which is designed to interact with children and can maneuver around the house. At current implementation, the cost of the platform is also reduced with the use of off-the-shelf hobbyist servomotor. These servo motors can be daisy-chained for signal communication and power delivery making the system platform more compact. In addition, the parts of the mobile platform is also easy replaceable as every portion is modular.

5.1 Future works

The control software was developed in C# in order to facilitate communication with other modules of Robotubby such as vision, speech and networking task. However, the recommendation is to develop a generic PCW microcontroller as a standalone micro-processing unit such that better real-time control can be realized. Separating the mobile base controller from the main computer improves reliability as the main computer failure or task scheduling will not result is a runaway platform.

Physically, the weight of the platform can be reduced further with alternate material choice. Another wheel type should be selected or a suspension system should be designed to provide better contact with the ground especially during rotation. With the addition of a cover on the platform will make it more stable and less likely to tip over. Better wire management like wire looms can prevent the wires from getting caught in the wheels.

Other improvement includes low level collision avoidance systems such as contact and range sonar sensors. With the implementation onto a micro-controller, sensor data can be easily acquired for processing and integrating into the mobile platform. Sensors and the applications of other localizing systems and/or methods of resolving orientation was compensate better against slippage.

Bibliography

[1] Claudia Muhl1, Britta Wrede, Martina Hielscher-Fastabend, Gerhard Sagerer Frank Hegel1, "Understanding Social Robots," *Advances in Computer Human Interactions*, pp. 169-174, Feb 2009.

[2] Cynthia L. Breazeal, *Desgining Social Robots*.: MIT Press, 2004.

[3] O. Jahanian and G Karimi, "Locomotion Systems in Robotic Application," *Robotics and Biomimetics 2006. ROBIO '06. IEEE International Conference*, pp. 689-696, December 2006.

[4] Raul Rojas, "A Short History of Omni-Directional Wheels,".

[5] MARVIN MINSKY, "Telepresence," *OMNI magazine*, June 1980.

[6] Myung-lin Jung and Jong-Hwan Kim, "Developement of a fault tolerant Omni-directional Wheeled Mobile Robot Using Non-holonomic Constraints," *International Journal of Robotics Research* , pp. 527-529, May 2002.

[7] Oussama Khatib Robert Holmberg, "Development and Control of a Holonomic Mobile Robot for Mobile Manipulation Tasks," *International Journal of Robotics Research*, vol. 19, pp. 1066-1074, Nov 2000.

[8] Bruno Siciliano and Oussama Khatib, *Springer Handbook of Robotics*.: Springer, 2008.

[9] Akira TAKAGI and Shunji MORI Masayoshi WADA, "Caster Drive Mechanisms for Holonomic and Omnidirectional Mobile Platforms with no Over Constraint," *International Conference on Robotics & Automation*, pp. 1531-1538, April 2000.

[10] Georges Bastin, Brigitte D'Andreas-Novel Guy Campion, "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 12, p. 47, Feburary 1996.

[11] Jorge Angeles, John Dacovich Subir Kumar Saha, "The Design of kinematically isotopic Rolling Robots with Omni-directional Wheels," *Elsevier Science*, vol. 30, pp. 1127-1137, 1995.

[12] Whee Kuk Kim Byung-Ju Yi, "The Kinematics for Redundantly Actuated Omni-directional Mobile Robots," *IEEE International Conference on Robotics and Automation*, pp. 2485-2492, April 2000.

[13] K.H. Low, W.K. Loh Y.P Leow, "Kinematic Modelling and Analysis of Mobile Robots with omni-directional wheels," *Seventh International Conference on Control, Automation, Robotics and Vision*, pp. 820-825, Dec 2002.

[14] P.F. Muir and C.P.Neuman, "Kinematic modelling of wheeled mobile robots," *Journal of Robotic System*, pp. 281-340, 1987.

[15] Zhongwang Chua, "Design and Control of a Mobile Robot with Full Dexerity," National University of Singapore, Faculty of Mechanical Engineering, Singapore, Final Year Thesis 2008.

[16] A. El-Shenawy, A. Wagner, and E. Badreddin, "Dynamic Model of a Holonomic Mobile Robot with Actuated Caster Wheels ," *Control, Automation, Robotics and Vision, 2006. ICARCV '06*, pp. 1-6, Dec 2006.

[17] Eui-jung Jung et al., "Navigation of an omni-directional mobile robot with active caster wheels ," *Robotics and Automation*, pp. 1659-1665, May 2008.

[18] D. Oetomo, Yuan Ping Li, M.H., Jr. Ang, and Chee Wang Lim, "Omnidirectional mobile robots with powered caster wheels: design guidelines from kinematic isotropy analysis ," *Intelligent Robots and Systems*, pp. 3034-3039, Aug 2005.

[19] Jae Hoon Lee, Bong Keun Kim, T. Tanikawa, and K. Ohba, "Kinematic analysis on omni-directional mobile robot with double-wheel-type active casters ," *Control, Automation and Systems*, pp. 1217-1221, Oct 2007.

[20] Sungbok Kim and Ilhwa Jeong, "Systematic Isotropy Analysis of Caster Wheeled Mobile Robots with Steering Link Offset Different from Wheel Radius ," *Robotics and Automation*, pp. 2971-2976, April 2007.

Annex A: Dynamixel Servo Control

The Instruction Packet is the packet sent by the main controller to the Dynamixel units to send commands. The structure of the Instruction Packet is as the following.

Instruction Packet

OXFF 0XFF ID LENGTH INSTRUCTION PARAMETER1 …PARAMETER N CHECK SUM
The meanings of each packet byte definition are as the following.

0XFF 0XFF (Header)

The two 0XFF bytes indicate the start of an incoming packet.

ID

The unique ID of a Dynamixel unit. There are 254 available ID values, ranging from 0X00 to 0XFD.

Broadcasting ID
ID 0XFE is the Broadcasting ID which indicates all of the connected Dynamixel units. Packets sent with this ID apply to all Dynamixel units on the network. Thus packets sent with a broadcasting ID will not return any status packets.

LENGTH

The length of the packet where its value is "Number of parameters (N) + 2"

INSTRUCTION

The instruction for the Dynamixel actuator to perform.

Value	Name	Function	No. of Parameters
0x01	PING	No execution. It is used when controller is ready to recevie Status Packet	0
0x02	READ DATA	This command reads data from RX-64	2
0x03	WRITE DATA	This command writes data to RX-64	2 or more
0x04	REG WRITE	It is similar to WRTE_DATA, but it remains in the standby state without being executed until the ACTION command arrives.	2 or more
0x05	ACTION	This command initiates motions registered with REG WRITE	0
0x06	RESET	This command restores the state of RX-64 to the factory default setting.	0
0x83	SYNC WRITE	This command is used to control several RX-64s simultaneously at a time.	4 or more

PARAMETER0…N

Used if there is additional information needed to be sent other than the instruction itself.

CHECK SUM

The computation method for the 'Check Sum' is as the following.
Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)
If the calculated value is larger than 255, the lower byte is defined as the checksum value.
~ represents the NOT logic operation.

For most cases in the control of PCW robot, the instructions are mainly write register (0x03) and the parameters data form are as follows: identify initial register address for the specific data to be altered then input necessary data to be modified including subsequent register address data that require modification.

Address table:

	Address (hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM Area	0 (0X00)	Model Number(L)	Lowest byte of model number	R	28 (0X1C)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 [0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	240 (0XF0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
RAM Area	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	0 (0X00)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	0 (0X00)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Comliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered Instruction	Means if Instruction is registered	RW	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

Valid data range:

[Control Table Data Range and Length for Writing]

Write Address	Writing Item	Length (bytes)	Min	Max
3(0X03)	ID	1	0	253(0xfd)
4(0X04)	Baud Rate	1	0	254(0xfe)
5(0X05)	Return Delay Time	1	0	254(0xfe)
6(0X06)	CW Angle Limit	2	0	1023(0x3ff)
8(0X08)	CCW Angle Limit	2	0	1023(0x3ff)
11(0X0B)	the Highest Limit Temperature	1	10(0x10)	99(0x63)
12(0X0C)	the Lowest Limit Voltage	1	50(0x32)	250(0xfa)
13(0X0D)	the Highest Limit Voltage	1	50(0x32)	250(0xfa)
14(0X0E)	Max Torque	2	0	1023(0x3ff)
16(0X10)	Status Return Level	1	0	2
17(0X11)	Alarm LED	1	0	127(0x7f)
18(0X12)	Alarm Shutdown	1	0	127(0x7f)
19(0X13)	(Reserved)	1	0	1
24(0X18)	Torque Enable	1	0	1
25(0X19)	LED	1	0	1
26(0X1A)	CW Compliance Margin	1	0	254(0xfe)
27(0X1B)	CCW Compliance Margin	1	0	254(0xfe)
28(0X1C)	CW Compliance Slope	1	1	254(0xfe)
29(0X1D)	CCW Compliance Slope	1	1	254(0xfe)
30(0X1E)	Goal Position	2	0	1023(0x3ff)
32(0X20)	Moving Speed	2	0	1023(0x3ff)
34(0X22)	Torque Limit	2	0	1023(0x3ff)
44(0X2C)	Registered Instruction	1	0	1
47(0X2F)	Lock	1	0	1
48(0X30)	Punch	2	0	1023(0x3ff)

The Status Packet is the packet returned by the Dynamixel units to the main controllers. The structure of the Status Packet is as the follows:

OxFF 0xFF ID LENGTH ERROR PARAMETER1 PARAMETER2…PARAMETER N CHECK SUM

Each byte composing the packet means as below.

0XFF 0XFF This signal notifies the beginning of the packet.

ID It is the ID of RX-28 which transfers Status Packet.

LENGTH It is the length of Status Packet, the value of which is“the number of Parameters (N) + 2”.

ERROR It displays the error status occurred during the operatio of RX-28. The meaning of each bit is described in the below table.

Bit	Name	Contents
Bit 7	0	-
Bit 6	Instruction Error	In case of sending an undefined instruction or delivering the action command without the reg_write command, it is set as 1.
Bit 5	Overload Error	When the curren load cannot be controlled by the set Torque, it is set as 1.
Bit 4	Checksum Error	When the Checksum of the transmitted Instruction Packet is incorrect, it is set as 1.
Bit 3	Range Error	When a command is out of the range for use, it is set as 1.
Bit 2	Overheating Error	When internal temperature of Dynamixel is out of the range of operating temperature set in the Control table, it is set as 1.
Bit 1	Angle Limit Error	When Goal Position is written out of the range from CW Angle Limit to CCW Angle Limit , it is set as 1.
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control table, it is as 1.

For example, when Status Packet is returned as below

0xFF 0xFF 0x01 0x02 0x24 0xD8

It means that the error of 0x24 occurs from RX-28 whose ID is 01. Since 0x24 is 00100100 as binary, Bit5 and Bit2 become 1. In order words, Overload and Overheating Errors have occurred.

PARAMETER0…N It returns data except ERROR. For the usage of parameters, refer to “3-5 How to Use Packet".

CHECK SUM It is used to check if packet is damaged during communication. The below formula defines Check Sum. This formula is constructed in the same way as the Check Sum of Instruction Packet.

Check Sum = ~ (ID + Length + Error + Parameter1 + … Parameter N)

E.g.:

Example 17	Locates at the Position 180° with the speed of 57RPM.
Hint	Sets Goal Position (Address 30 (0x1E))= 511 (0x1FF) and Moving Speed (Address 0x20))= 512 (0x200). Instruction = WRITE_DATA, Address = 0x1E, DATA = 0x00, 0x02, 0x00, 0x02
Communication	Instruction Packet: FF FF 01 07 03 1E 00 02 00 02 D3 Status Packet : FF FF 01 02 00 FD
Status Packet Result	NO ERROR

Example 1 Reads the current internal temperature of RX-28 whose ID is 1.

Reads 1 byte from the value of Address 43 (0x2B) in the Control Table.

Instruction Packet : 0XFF 0XFF 0X01 0X04 0X02 0X2B 0X01 0XCC

ID LENGTH INSTRUCTION PARAMETERS

CHECKSUM

Status Packet returned is as follows:

Status Packet : 0XFF 0XFF 0X01 0X03 0X00 0X20 0XDB

ID LENGTH ERROR PARAMETER1 CHECKSUM

Data value read is 0x20 (i.e., 32 in decimal). Thus, the current internal temperature of

Important functions implemented:

Endless Turn

Endless Turn can be materialized when CW Angle Limit (Address 6,7) and CCW Angle Limit (Address8,9) are set as "0". It can be usefully applied to move wheels.

Endless Turn has no speed control function. Enter a desired torque value into Moving Speed (Addresses 32 and 33 (0X20 and 0X21)). The meaning of Moving Speed Address is as shown in the below picture.

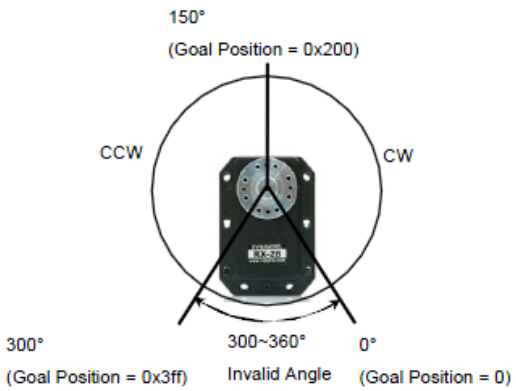
Data value in the table represents the ratio of output torque. For example, Data 1023 (0x3FF) means that 100% of torque should be generated in the current voltage state while data 512 (0x200) means that 50% of torque should be generated.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Turn Direction	Data (Torque Ratio)									

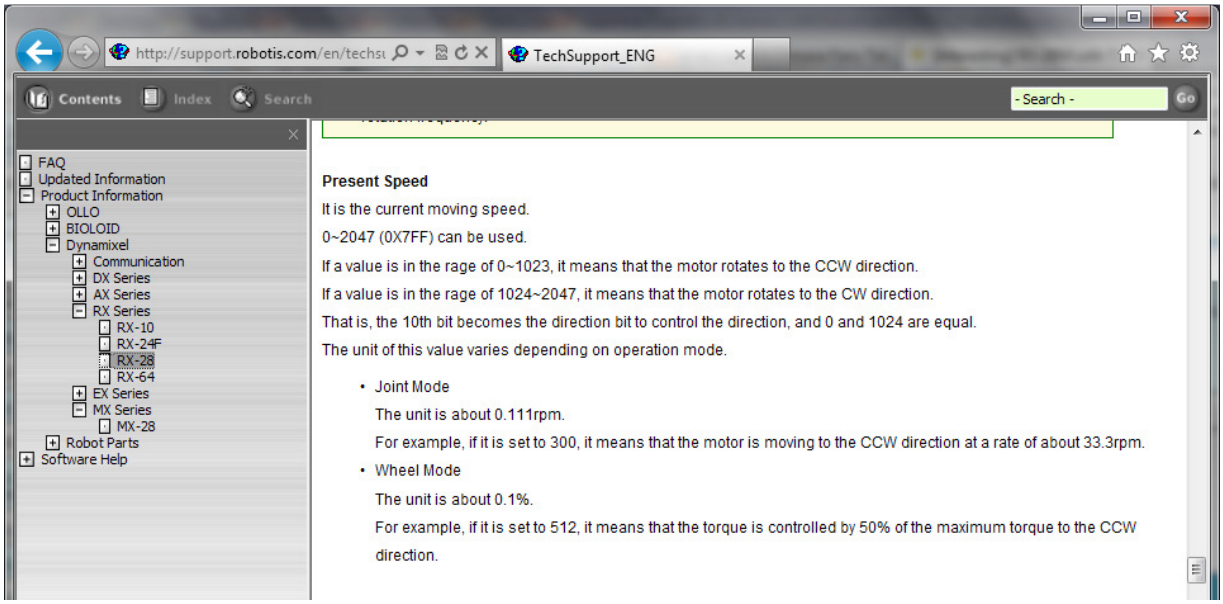
Turn Direction = 0 : CCW Direction Turn, Turn Direction = 1: CW Direction Turn

Goal Position

Address 30, 31 (0X1E, 0x1F) It is a position value of destination. 0 to 1023 (0x3FF) is available. Position values according to data values are as shown in the below picture. Goal Position should be used within the range of CW Angle Limit ≤ Goal Potion ≤ CCW Angle Limit; when it is out of the range, Angle Limit Error occurs.



Present Position Address 36, 37 (0x24,0x25) It is the current position of RX-28. The unit is the same as that of Goal Position.



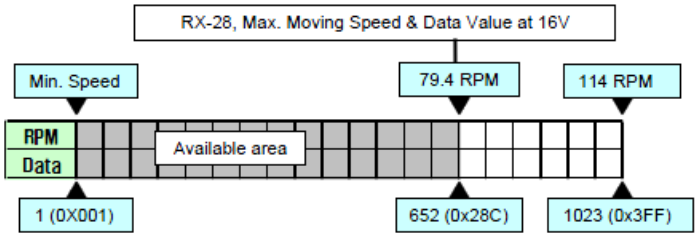
Moving Speed Address 32, 33 (0x20, 0x21) It is a moving speed to Goal Position. 0 to 1023 (0X3FF) can be set for the speed.

Present Speed Address 38, 39 (0x26,0x27) It is the current moving speed of RX-28. 0 to 1023 (0X3FF) can be measured.

Moving Speed and Present Speed can be converted into RPM when data value is multiplied by 0.111. For example, Data 1023 is 114RPM (1023x0.111=113.6). But, the maximum speed of RX-28 is less than 114RPM. Nevertheless, the range of speed data value is set up to 114 RPM since RX-28 can move faster than the maximum speed by outside factors.

The maximum speed of RX-28 is in proportion to the size of supplied voltage. In other words, the higher voltage it is supplied with, the wider range of speed it can control. For example, when RX-28 is supplied with 16V, it can reach to the speed of 79.4RPM and control the speed with 0 to 79.4 RPM. However, when it is supplied with 12V, the

maximum speed is reduced to 59.9RPM so that the speed with 0 to 59.9 RPM can be controlled. The relationship between data value and speed is as shown in the below picture.



Note	When Moving Speed is set as 1 (0X001), movement is made at the minimum speed. When Moving Speed is set as 0 (0x000), movement is made at the maximum speed which can be reached under the applied voltage. In other words, setting as 0 means that no speed control will be done.

Torque Limit

Address 34, 35, (0x22, 0x23) It sets the maximum output Torque. 0 to 1023 (0x3FF) is available. Torque related data is allocated in EEPROM (Addresses 14 and 15) and RAM (Addresses 34 and 35). And when the power is on, the EEPROM value is copied to RAM. Torque is restricted by the Torque Limit value located in RAM (Addresses 34 and 35) in driving. Data value represents the ratio of Torque that can be output under the currently applied voltage as described in Max Torque

Annex B: Robot Convention and Reference

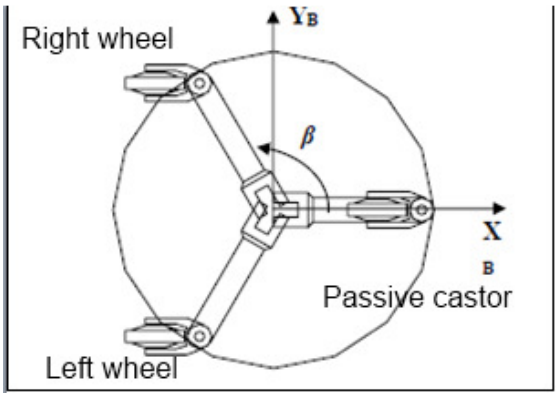


Figure 0-1: Layout of base frame

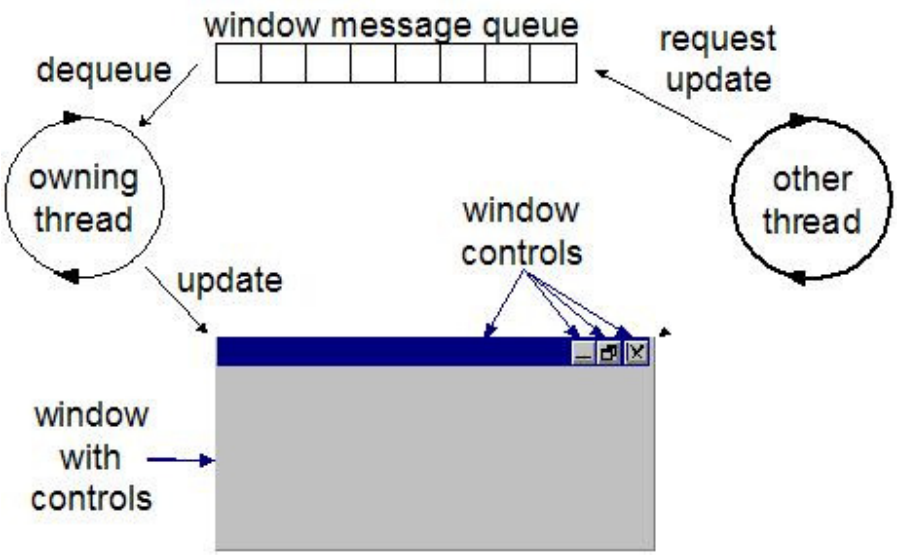
The dynamixel IDs are as follows:

Left wheel	Function	ID
	Steer	3
	Drive	1
Right wheel		
	Steer	5
	Drive	2

Annex C: Using the Graphical User Interface

The GUI requires the input of parameters from the target hardware to initialize the motion control. Controlling the robot can be realized by entering target values of the velocity or end-point (trajectory control) or via the software joystick.

[1]Multithreading with GUI and control task in winform



[1] http://msdn.microsoft.com/en-us/library/ms951089.aspx