ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

DESIGN AND COMPARISON OF CONTROLLER PERFORMANCE ON FOUR MECANUM WHEELED MOBILE ROBOT

M.Sc. THESIS

Doğukan Taha TAYFUR

Department of Mechanical Engineering

System Dynamics & Control Programme

OCTOBER 2016

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

DESIGN AND COMPARISON OF CONTROLLER PERFORMANCE ON FOUR MECANUM WHEELED MOBILE ROBOT

M.Sc. THESIS

Doğukan Taha TAYFUR
(503121618)

Department of Mechanical Engineering

System Dynamics & Control Programme

Thesis Advisor: Assoc. Prof. Dr. Ayhan KURAL

OCTOBER 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK YÖNLÜ HAREKET EDEBİLEN MOBİL ROBOTTA DENETLEYİCİ TASARIMI VE PERFORMANS KIYASLAMASI

YÜKSEK LİSANS TEZİ

Doğukan Taha TAYFUR
(503121618)

Makine Mühendisliği Bölümü

Sistem Dinamiği ve Kontrol

Tez Danışmanı: Doç. Dr. Ayhan KURAL

EKİM 2016

Doğukan Taha TAYFUR**, a M.Sc. student of ITU** Graduate School of Science Engineering and Technology **student ID 503121618, successfully defended the** thesis **entitled "**Design and comparison of controller performance on four mecanum wheeled mobile robot**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.**

Thesis Advisor :	**Assoc. Prof. Dr. Ayhan KURAL** İstanbul Technical University	....
Jury Members :	**Assoc. Prof. Dr. Kenan Refah KUTLU** Istanbul Technical University	....
	Assoc. Prof. Dr. Semih SEZER Yıldız Technical University	....

Date of Submission : 09 September 2016
Date of Defense : 07 October 2016

Widmung für einen langen Weg zur erfolgreichen Wissenschaft…

FOREWORD

This thesis study is completed with highly great patience and ambition.

I would like to thank to my advisor Assoc. Prof. Dr. Ayhan KURAL who has given me a different point of view in the field of control and has guided me with his valuable information.

I also want to thank to my family for their great love, patience, moral and encouragement during all stages of my educational life.

October 2016 Doğukan Taha TAYFUR
(Mechanical Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

MPC	**:** Model Predictive Control
LQR	**:** Linear Quadratic Regulator
MIMO	**:** Multiple Input Multiple Output
GUI	**:** Graphical User Interface
TTL	**:** Transistor-Transistor Logic
GPIO	**:** General Purpose Input/Output
I2C	**:** Inter-Integrated Circuit
SPI	**:** Serial-Peripheral Interface
RPM	**:** Round Per Minute
PI	**:** Performance Index
MAC	**:** Model Algoritmic Control
GPC	**:** Generalized Predictive Control
DMC	**:** Dynamic Matrix Control

LIST OF TABLES

Page

LIST OF FIGURES

Page

DESIGN AND COMPARISON OF CONTROLLER PERFORMANCE ON FOUR MECANUM WHEELED MOBILE ROBOT

SUMMARY

In this thesis the main aim is to design and compare controller performance on omnidirectional mobile robot based on mecanum wheels and it can be use for further researches for elemination system errors. On the purpose of to increase orientation capability the system is equipped with four mecanum wheels which have twelve rollers around it. In the scope of this thesis to control dc motor speeds according to user demans and to minimize heading errors in the band of accaptable area.

Before start to develop controller for the large scale system experimental test rig has been equipped with system components which are mecanum wheels, dc motors, encoders and controller.

Also before start design controller all system elements should have mathematical models to take best controller results. Such as Dc motor model, system forward and backward kinematics, which help to minimize heading errors in real nonlinear environment.

For the developing linear quadratic regulator and model predictive controller transfer function of the system is needed. Especially for LQR controller owing to feedback constants. The dc motor model is implemented to the simulation enviroment to see system output according to user inputs. In order to validate designed model, different system scenarios are tested independetly. After validation of system outputs cross validation test is conducted to maximize dependability.

Designed model simulated in Matlab/Simulink environment with real dc motor coefficients. To improve controller performance for LQR controller different Q matrixes and R constants are tested and simulated. In the end admissible feedback constants are chosen. For MPC system inputs and outputs are simulated with modelled system. The controller performance is adjusted according to time interval, control horizon, weights of inputs and prediction horizon constants.

Based on simulated model the aim is to take smooth response from system without overshoot, minimal oscillation and minimize settling time. Simulated results will be compared to real system results in future.

ÇOK YÖNLÜ HAREKET EDEBİLEN MOBİL ROBOTTA DENETLEYİCİ TASARIMI VE PERFORMANS KIYASLAMASI

ÖZET

Bu tez çalışmasında çok yönlü hareket edebilen dört tekerlekli mobil robotun tasarımı ve farklı tipteki denetleyicilerin dizayn edilmesi ve bunların sonuçlarının kıyaslanması hedeflenmiştir. Denetim algoritması oluşturmaktaki amaç, gerçek sistem üzerinde oluşabilecek üretim hataları, motorların stabil olmaması ve diğer dış etkenler dolayısıyla meydana gelen hataları kabul edilebilir düzeye indirmektir. Çünkü herhangi bir kontrol algoritması kullanmadan sistemin istenilen stabilitede hareket etmesi ve farklı yüklerin motorlara binmesi durumunda hızların istenilen düzeyde tutulması zor bir durumdur.

Gerek dıştan gelen etkiler, gerekse içten kaynaklanan birtakım sebeplerden dolayı sisteme uygulanacak denetleyiciler seçilmiş olup, bunların sonuçları kıyaslanmıştır. Fakat denetleyici tasarlamadan önce sistemin matematik modelinin bilinmesi gerekmektedir. Bunun için sistemin ilk önce ileri ve ters kinematik denklemleri matris şeklinde çıkarılmıştır. Matematiksel modeli oluştururken sistemin hesaplamaları kolaylaştırmak amacıyla bazı kabuller yapılmıştır. Bunlara örnek olarak sistemin rijit olduğu varsayılmış, zemin ile tekerler arasında oluşan yuvarlanma direncinin tekerlerin kaymadan ilerlemesi varsayılmış, tekerlerin zemin ile sürekli temasta olduğu ve diğer sistem bileşenleri modellenirken lineer denklemler kullanılmıştır. Model de tekerlerde üretim hatalarının olmadığı göz önüne alınmıştır. Her tekerin açısı 45° olduğu ve sistem simetrik şekilde dizayn edildiği varsayılmıştır.

Oluşturulan matematiksel model ile sistemin istenilen doğrultuda gitmesi için herbir tekerin birbirinden bağımsız olarak hangi hıza sahip olacağı bu model sayesinde hesaplanabilmektedir. İleri kinematik denklemleri sayesinde sistemin kartezyen kordinat üzerindeki hızları ve her bir motorun devir sayısı elde edilebilmektedir. Oluşturulmuş olan ters kinematik denklemleri sayesinde ise sistemin dışarıdan kaynaklanan bozucu etkenlere karşı sistemin istenilen doğrultuda hareket etmesini sağlamak için yeniden motor hızları hesaplanmasına imkan vermektedir.

Matematiksel denklemleri elde edilen sistemin sonuçlarını almak için model simulasyon ortamına aktarılmıştır. Bu ortamda sistemin oluşturulan ters ve ileri kinematik denklemleri, dc motor modeli, denetleyici ve input output arasındaki sistemler arası dönüştürücü denklemler yer almaktadır. Oluşturulan model sayesinde simulasyonlar gerçekleştirilmiştir ve alınan sonuçlar neticesinde denetleyici için gerekli olan katsayılar belirlenmiştir. Kontrolcü performansını denetlemek amacıyla ise alınan bu veriler gerçek sistem üzerinde test edilmek amacıyla kullanılmıştır. Gerçek sistem ise dört adet dc motor, dört adet manyetik enkoder, iki adet dc motor sürücü kartı ve mikrodenetleyiciden oluşmaktadır. Sistemde kullanılmış olan dc motor 350 wattlık güce sahip ve bu motorlar 12v luk seri bağlanmış iki adet kuru tip akülerle beslenmektedir. Sistemin istenilen tork değerlerine ulaşması için redüktör kullanılmıştır. Kullanılan iki adet dc motor sürücü kartı kanal başına 30 amper çıkış verebilmektedir ve her kart iki adet motoru sürebilmektedir. Ayrıca kartı kullanmak için üretici firma C++ ve python2.x kütüphanelerini de vermektedir. İstenilen datalar

bu kartlar üzerinden bilgisayara seri haberleşme ile aktarılabilmektedir. Kullanılan çokyönlü tekerler 12 adet dış dış tekere sahip olup bunların yanında tekerin hareketini kolaylaştırmak amacıyla da 24 adet küçük yarım tekerleklerde ilave edilmiştir. Amaç tekerlerin dönüşü esnasında süreklilik sağlamak.
Montaj sonrası sistemin matematiksel modeli oluşturulması için gerekli ölçümler yapılmıştır. Bunlara örnek olarak dc motor modelinin belirlenmesi ve bu modelde kullanılacak olan parametrelerin saptanması gerekmektedir. Bunları belirlemek için sisteme giriş verilerek çıkışlar gözlenmiş ve Matlab/Simulink system identification toolbox sayesinde sistem belirlenmiş ve parametreler belirlenmiştir. Daha sonra elde edilen bu değerler ile sistem tekrardan doğrulanarak değelerin güvenilirliği ölçülmüştür. Daha sonra sistemin diğer bileşenlerine ait olan tekerleklerin birbirine olan uzaklıkları redüktör oranları, maksimum motor devirleri ve bunun gibi önemli diğer parametrelerin ölçümleri yapılarak matematiksel model oluşturulmuştur. Motor için elde edilen transfer fonksiyonları ve durum uzay modeli denetleyici tasarımında büyük rol oynamaktadır.
Denetleyici olarak sisteme uygulanabilir denetleyicilerden lineer quadratic regulator ve model öngörülü kontrolcü kullanılmıştır. LQR denetleyici için sistemde durum geri besleme yapılmaktadır ve bu geri beslemeler için motorların transfer fonksiyonlarına ihtiyaç duyulmuştur. Bu tip denetleyicide optimal geri besleme katsayılarını bulmak için hali hazırda bulunan yöntemler kullanılmıştır ve ayrıca mükemmel sonuçlar için simülasyonlar gerçekleştirilmiştir. Geri besleme katsayıları belirlenirken Matlab'de bulunan fonksiyonlar yardımıyla bulunmuştur. Fakat Q matrisi ve R katsayısı denetleyiciyi tasarlayan kişi tarafından seçilmektedir. Bu katsayılar performansı etkileyen en önemli kriterlerdendir.
Kullandığı yazılım ve donanım itibariyle MPC ileri düzey bir denetim yöntemi olarak sınıflanabilir. İleri denetim tekniği olması denetim sinyallerini oluştururken optimizasyon algoritması çalıştırarak ilgilenilen süreç çıkış sinyallerini tasarımcının arzu ettiği optimizasyon ölçütüne uygun olarak sağlayan yapıya sahip olmasındandır. MPC yönteminden daha önce analog kontrol yöntemleri ve nümerik optimizasyonlar kullanılarak kontrol sağlanmıştır. Bugünkü teknolojiye bakarak işlemci teknolojilerini göz önüne aldığımızda bu denetleyicinin başarılı olabilmesini mümkün kılmıştır.
Lineer olmayan sistem modelleri için az sayıda MPC algoritması varlığına karşın, lineer sistem modelleri için geliştirilmiş olan hali hazırda çeşitli algoritmalar bulunmaktadır. Model öngörülü kontrolcü ideal çalışma için denetlenmek istenen sistemin kesin modeline gerek duyar; ancak sistem modelindeki belirsizlikler durumunda dahi uygun geri besleme konfigurasyonları kullanılarak MPC algoritması başarılı olarak çalıştırılabilir.
Başlangıçtan itibaren T saniyelik süre boyunca arzu edilen süreç çıkış yörüngesini daha önceden kullanıcı tarafından hazırlanabilir. Diğer denetleyici olarak model öngörülü kontrol seçilmiştir bu denetleyicide ki amaç büyük sistemlerin sisteme giriş verilmeden önce kumanda sinyalinin tespit edilmesi amacıyla kullanılmaktadır. Sistemi bir kapalı kutu olarak algılayıp bu sistem ile arka planda çalışan algoritma yardımıyla kullanıcı istekleri doğrultusunda sistem davranışı belirlenmiş olur. Sistem denetleyicisi adım cevabı veya darbe cevabı modeliyle belirlenebilir. Ayrıca Matlab/Simulinkte yer alan toolbox sayesinde denetleyici parametreleri ayarlanabilir. Bunlar kontrol ufku, saniyedeki hesaplama sayısı, tahmin edilen adım sayısı, sistem cevap süreleri gibi parametrelerdir.
Bu denetleyiciler belirlenirken amaç motorun farklı senaryolar altında kullanıcı tarafından belirlenen doğrultudaki hızını sabit tutmasıdır. Fakat LQR denetleyicide geri besleme katsayıları sabit olduğundan belli her senaryo için adaptif olarak

çalışamamaktadır. Bunun yanında MPC denetleyicide ise sistemin o anki davranışına göre ileride gerçekleşecek olan davranışı tahmin ettiğinden anlık verilerden yeni bir kumanda sinyali hesaplanır. Temelde kullandığı kontrol ufku ve tahmin ufku iterasyon sayılarından dolayı o andaki sistemin durumunu ölçüp gelecekte olabilecek durum hakkında bir kumanda sinyali üretilmektedir. Tahmin ufkunun fazla olması bir kaç adım sonra karşılaşabilecek durum hakkında daha stabil bir kestirim yapılabilmektedir. Ayrıca kumanda sinyallerinde oluşturulan kısıtlamalar sayesinde simulasyonlardan istenilen kumanda çıkışları ve sistem cevapları alınmıştır.

1. INTRODUCTION

Recently, several mobile robots and mobile platforms have been commonly developed. And it is expect that they will be used in a sort of applications, such as service robots in homes, flat surfaced factories and supermarkets. So forth the environments, there are lots of narrowpassages and the robots run across some obstacles. The mecanum wheeled mobile platform has the advantages over the conventional wheeled mobile robots one in terms of mobility particularly in closely spaced environments, for instance factories, offices, hospitals and similar areas. The specialized wheels and structure are needed for the mobile robot to have the omni-directional maneuverability. Four wheels structure is using ‘Mecanum wheel’ are the examples of the omni-directional platforms. Especially, the omni-directional mobile platform with mecanum wheels are used in the forklift, the wheelchair and so other applications. [1]

Besides, they have some advantages and disadvantages. When they are driven in narrow space, their movement is restricted. In those environments, more effective driving is required so that omnidirectional driving is needed. The omnidirectional driving is said to move any direction. For example ackermann steering mechanism should have more space to turn and with one direction to move. In omni-directional driving it can move to eight directions and in addition it can rotate clockwise and counterclockwise directions. And some applications ıt can be driven any desired angles with different control methods.

There have been some studies to deal with development of mecanum wheeled platform designing and methods of controlling it. However, there has been a small number of research to deal with applications of these kind of projects. In this thesis, as an application of mecanum wheels are introduced. The aim of the project was developing controller wheelchair for the disabled who use it in daily life.

In daily life this kind of mobile robots encounter with external disturbances, consequently to overcome like these kind of detrimental variables overall system should be identificated with some techniques. To control mecanum wheeled robot used dc motors parameters must be known. Because dealing with motor velocites and torque is the main subject.

1.1 Purpose of Thesis

In this thesis, introduction section concerning practical applications for mobile robotic platform based on conventional wheel is presented. Mobile robot equipped with four Mecanum wheels have the omnidirectional property, which means, they have the ability to move instantaneously in any direction, from any configuration. Therefore, compared to conventional platforms, these vehicles possess multiple advantages in terms of their mobility in narrow spaces or crowded environments. They have the ability to easily perform certain tasks in congested environments foreseen with static obstacles, dynamic obstacles or narrow areas. [2] Usually, such environments are found in factory workshops, warehouses, hospitals, etc. Hence the resulting needs to create this kind of robotic platforms to satisfy the requirements of various fields, such as: industrial, military, naval, medical and last but not least, the educational field (as the basis for research). The characteristics of the Mecanum wheel, a short comparison between this type of wheel and a conventional wheel, as well as the constructive and design solutions are described.

Conventional wheels are mechanically simple, have high load capacity and high tolerance to work surface irregularities. However, due to their non-holonomic nature, they are truly omni-directional. Designs have been proposed to achive near omni-directional mobility using conventional wheels. The most common design are those using steering wheels. Vehicles based on this design have at least two active wheels, each of which has both driving and steering actuators. They can move in any direction from any configurations. However, this type of system is not truly omni-directional. Because it needs to stop and re-orient its to desired direction whenever it needs to travel in a trajectory with non continnuous curvatures. [3]

In order to implement control strategies, localisation of system is needed to obtain which consist of real position and velocities. And these are based on robot kinematics. Also system limits should be known before. If the control strategy is designed without considering these limits, system could be fail.

Control of robots related velocities and torques has established the existence to be most common problems. In order to accomplish, it is paid attention to motion control of mecanum wheeled robot. Assumed control contains required limits for developed cartesian velocities. This study includes tracking control through weighted matrix. To achieve desired movements and controller are used to ensure bounded velocities.

1.2 Objectives

The main aim is to develop appropriate control algorithm to omni directional mobile robot. The mecanum wheels are driven by dc motor. For controlling direction also is needed to control speeds of each motor. This project also contains to create whole mathematical system which are dc motor, forward kinematics and backward kinematics.

Defining mathemetical model of mecanum wheels according to forward kinematics and to define weighted jacobian matrix for distibution motor speeds to each motor. To analyze and validate total system according to mathematical modeled system. To determine best controllers types which are model predictive control and linear quadratic control, also maximize control input efficiency.

To analyze and validate the mecanum wheeled robot in terms of response robustness and error.

1.3 A Brief History of Mecanum Wheels

Omnidirectional wheels have been used in robotics, in industry, and in logistics for many years. The main source of omnidirectional wheels are companies which produce them for omnidirectional conveyor systems, for example, for handling packages. Omnidirectional wheels are popular for omnidirectional robots, especially in the Robocup. An omnidirectional robot can drive along a straight line from point A to point B, while rotating along the line in order to arrive with the correct orientation. Omnidrectional wheels have also been used for wheelchairs, for service vehicles in airports, and many other applications. [4]

It is not widely known that the first omnidirectional wheel was patented in 1919 by J. Grabowiecki in the US. Figure 1.1 shows an image from the patent application. The assembly consists of a main wheel and transversal rollers, such as those used by most Robocup teams. As early as 1907, inventors were considering the design of vehicles capable of moving forward and sideways without steering the wheels. [4]

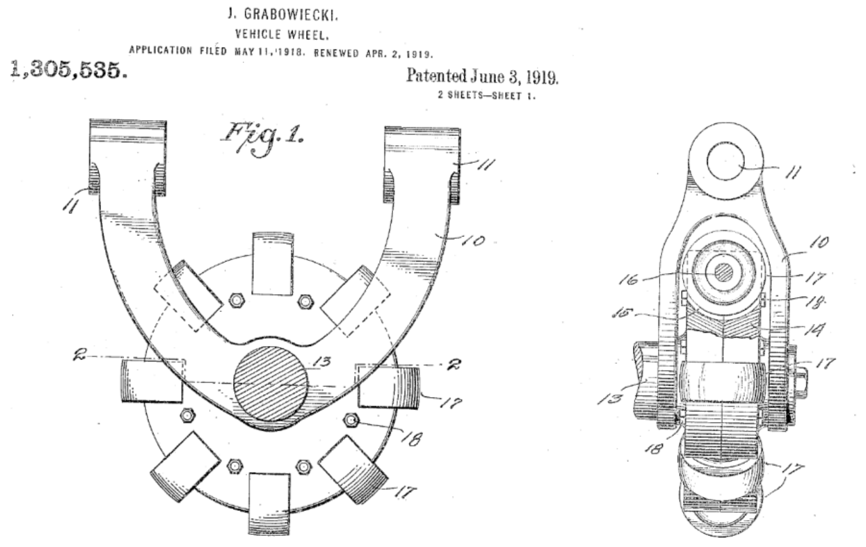


Figure 1.1: Drawing of probably the first omnidirectional wheel, as described in Grabowiecki's US Patent of 1919

One of the first modern omnidirectional wheels was developed by the Swedish inventor Bengt Ilon around 1973. Figure 1.2 shows the design of the Ilon wheel. The profile of the wheel is very nearly circular.

The wheel is omnidirectional but transversal forces produce excessive friction in the axes of the small rollers. A clever alternative are "Killough rollers", which are usually built using two truncated spheres. Such rollers were used by the Cornell Robocup team in 2000, and were still in use until 2004 in Robocup competitions. Although the rollers are named after Killough, the rollers had been actually patented in 1980 by Bradbury.

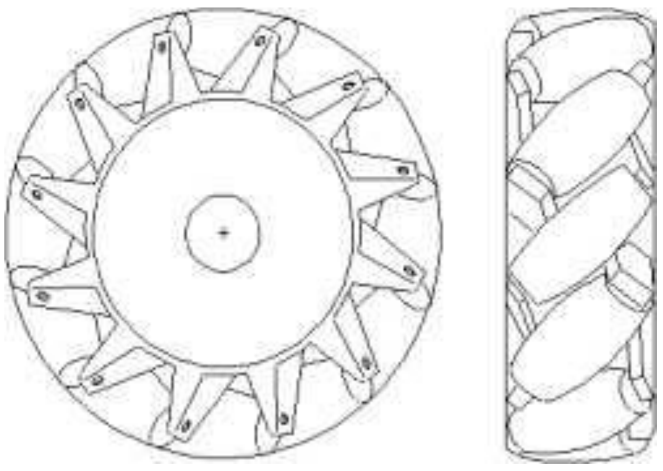


Figure 1.2: Omnidirectional wheel view from side and front

Much effort has been spent on improving the Swedish or Mecanum wheels, as they are sometimes called. The omniwheels can only roll smoothly if the profile of the complete wheel assembly is perfectly round, without gaps. Therefore, some groups have used spheres as a basis for the robot. The spheres can be activated with rollers, as in a mechanical mouse, or a group of spheres can be moved by chains or transversal bars. The spheres provide smooth rolling but the necessary mechanics is rather bulky. Conventional wheels can be also used for omnidirectional robots, if the wheels are rotated by a second steering motor. [5]

2. MODELLING

This is a top view looking down on the drive platform. Wheels in positions 1, 4 should make X-pattern with wheels 2, 3 (Figure 2.1). If not set up like shown, wheels will not operate correctly. Directions can be determined according to wheels rotations. Desired movements of this system and wheel rotations are listed in Table 2.1.

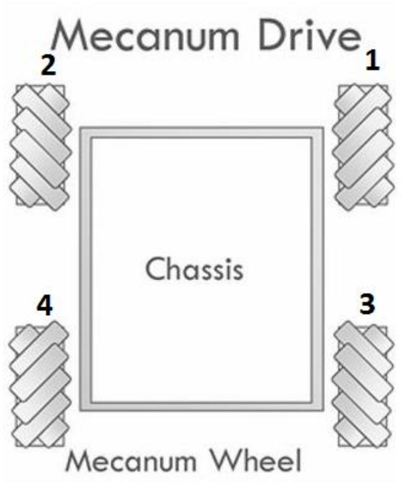


Figure 2.1: Mecanum drive wheels position and enumaration

Direction of movement	Wheel actuation
Forward	All wheels forward same speed
Reverse	All wheels reverse same speed
Right shift	Wheels 1,4 forward 2,3 backward
Left shift	Wheels 2,3 forward 1,4 backward
Right forward	Wheels 2,3 forward 1,4 stop
Left forward	Wheels 1,4 forward 2,3 stop
Right backward	Wheels 1,4 backward 2,3 stop
Left backward	Wheels 2,3 backward 1,4 stop
CW Turn	Wheels 1,3 forward 2,4 backward
CCW Turn	Wheels 2,4 forward 1,3 backward

Table 2.1: List of directions according to wheels rotation

Using four of mecanum wheels provides omni-directional movement for a vehicle without needing a conventional steering system slipping is a common problem in the mecanum wheel as it has only one roller with a single point of ground contact at any one time. Due to the dynamics of the mecanum wheel, it can create force vectors in both the x and y-direction while only being driven in the y-direction. Positioning four mecanum wheels, one at each corner of the chassis (two mirrored pairs), allows net forces to be formed in the x, y and rotational direction. Mecanum wheels direction distrubution according to motor rotations are shown (Figure 2.2).

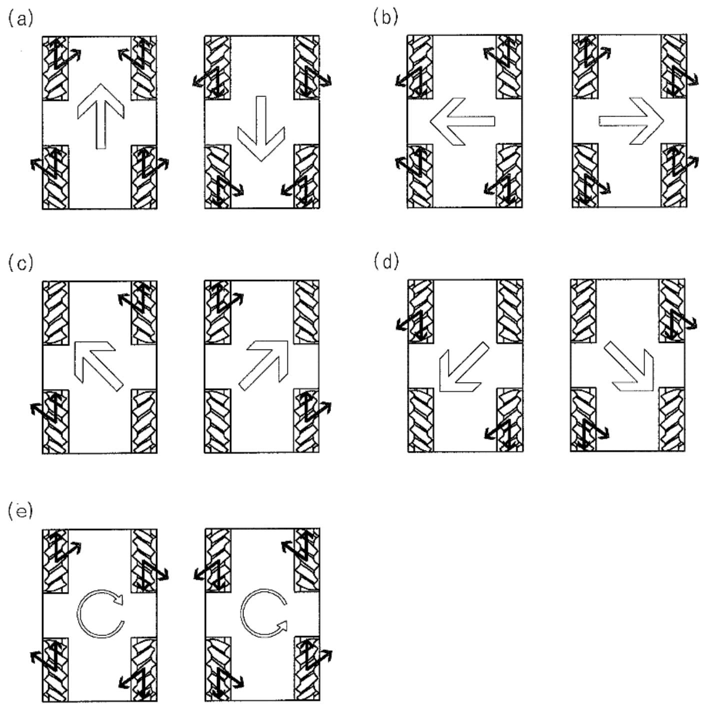


Figure 2.2: Wheels forces according to system directions

Before establishing the kinematic analysis of the moving mechanism, in order to facilitate modeling, we can make the following assumptions according to practical applications:

- The platform conducts regular exercise on a flat surface with four rounded wheels simultaneously functioning;
- The friction force between the Omni-directional wheels and the floor is large enough so there is no slipping wheels;

The platform is rigid so the case of deformation does not to be taken into consideration. [6]

2.1 Modeling Dc Motor

DC motor can be modeled by two sub parts, electrical and mechanical (Figure 2.3). Electrical parts consist of armature inductance, armature resistance and magnetic flux of stator. A second part is mechanical one. It consists of inertia of motor and load. The difference in motor speed is caused by the electromagnetic moment generated by current, load and friction of motor. [7]

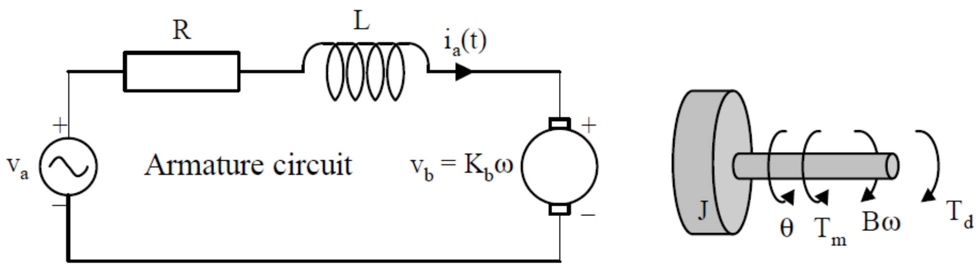


Figure 2.3: Dc motor mathematical model

The advantage of dc motors are easy to control speed and position and adjustable wide range of scale. And it is widely used in industry because of this reasons.

R : armature resistance

L : armature inductance

J : moment of inertia

K_t : motor torque constant

K_e : back emf constant

B : viscous friction

T_m : motor torque

$$T_m(t) = Bw(\mathrm{t}) + \mathrm{J}\frac{dw(t)}{dt}$$

$$L\frac{di_a(t)}{dt} + Ri_a(t) = v_a(t) - v_b(\mathrm{t})$$

$$v_b = K_e w(t)$$

$$T_m = K_t i_a(t)$$

When taking Laplace transform, the relation between voltage and angular speed designed in Simulink (Figure 2.4).

$$\frac{\Omega(s)}{V_a(s)} = \frac{K_t K_e}{(\mathrm{Js+B})(\mathrm{Ls+R}) + K_t K_e}$$

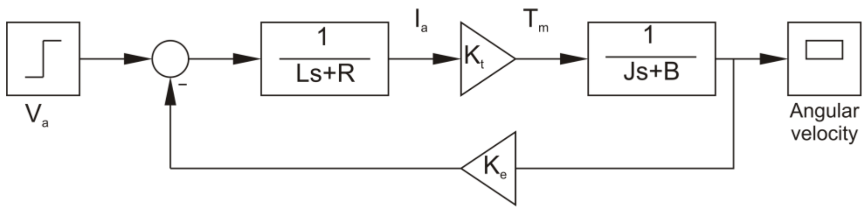


Figure 2.4: Dc motor model in simulink

2.2 Kinematic Analysis of Mecanum Wheels

Omnidirectional movement with the mecanum wheels is realized by appropriately controlling the angular velocity of each wheel separately. Depending on each individual wheel rotation direction and velocity, the resulting combination of the wheels produces a total movement in the desired direction without changing the orientation of the wheels.

To accomplish it kinematics plays an important role to define the position, orientation, velocity and acceleration of robots. When Mecanum wheels are actuated by supplying motion through motors, the angled rollers translate a portion of the force in the rotational direction of the wheel to a force normal to the wheel direction. Depending on each individual wheel direction and velocity, the resulting combination of all these forces produce a total force vector in any desired direction thus allowing the platform to move freely in the direction of the resulting force vector, without changing of the wheels themselves. [8]

The driving force of each wheel can be decomposed into two force components. One component is in the roller direction, and the other is the force in direction of rotation Four-wheel structure on local co-ordinates. That subtracts the sub-wheel direction force from the driving force. The sub-wheel direction force is exhausted by rolling the subwheel.

The addition of the net forces (Figure 2.5) of the four wheels determines the moving direction of the mobile platform. [9]

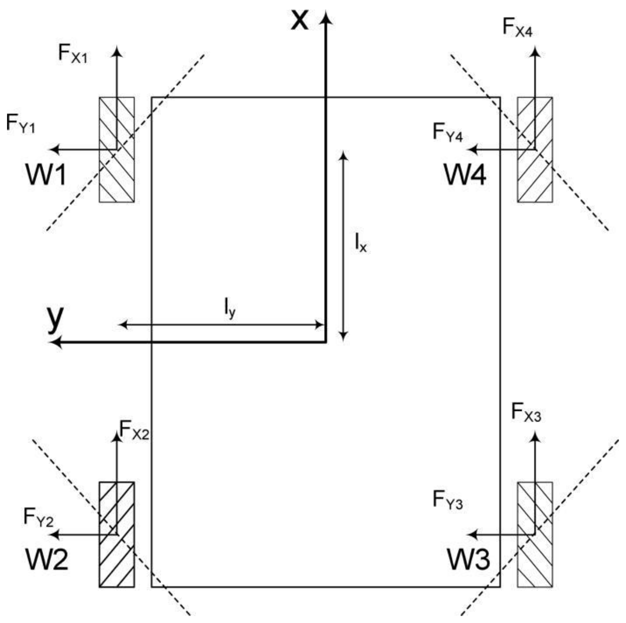


Figure 2.5: Force distrubution of system

$$V_{1x} = V_{1w} + \frac{V_{1r}}{\sqrt{2}}, V_{1r} = \frac{V_{1r}}{\sqrt{2}}$$

$$V_{2x} = V_{2w} + \frac{V_{2r}}{\sqrt{2}}, V_{2r} = -\frac{V_{2r}}{\sqrt{2}}$$

$$V_{3x} = V_{3w} + \frac{V_{3r}}{\sqrt{2}}, V_{3r} = -\frac{V_{3r}}{\sqrt{2}}$$

$$V_{4x} = V_{4w} + \frac{V_{4r}}{\sqrt{2}}, V_{4r} = \frac{V_{4r}}{\sqrt{2}}$$

$$V_{1x} = \upsilon_x - l.\omega_z, V_{1y} = \upsilon_y + L.\omega_z$$

$$V_{2x} = \upsilon_x + l.\omega_z, V_{2y} = \upsilon_y + L.\omega_z$$

$$V_{3x} = \upsilon_x - l.\omega_z, V_{3y} = \upsilon_y - L.\omega_z$$

$$V_{4x} = \upsilon_x + l.\omega_z, V_{4y} = \upsilon_y - L.\omega_z$$

Meanwhile, $\boldsymbol{V_x, V_y, W_z}$ represent the x and y elements of the velocity and angular velocity of the vehicle, respectively. In addition, $\boldsymbol{V_{ix}, V_{iy}}$ are expressed by using, $\boldsymbol{V_x, V_y, W_z}$ above. By comparison with equations and the following equations are obtained:

$$V_{1w} = \upsilon_x - \upsilon_y - (L+l)w_z$$

$$V_{2w} = \upsilon_x + \upsilon_y + (L+l)w_z$$

$$V_{3w} = \upsilon_x + \upsilon_y - (L+l)w_z$$

$$V_{4w} = \upsilon_x - \upsilon_y + (L+l)w_z$$

Combinnig equations ito below equation, which represents the inverse kinematics equation, yields:

$$V_w = J.V_0$$

Where $\boldsymbol{V_o}(\boldsymbol{V_x}, \boldsymbol{V_y}, \boldsymbol{W_z})$ is velocity vector in cartesian coordinates;

$\boldsymbol{V_w} = (\boldsymbol{V_{1w}}, \boldsymbol{V_{2w}}, \boldsymbol{V_{3w}}, \boldsymbol{V_{4w}})$ is the wheel velocity vector corresponding to angular velocity,

$$J = \begin{bmatrix} 1 & -1 & -(\mathrm{L}+\mathrm{l}) \\ 1 & 1 & (L+l) \\ 1 & 1 & -(L+l) \\ 1 & -1 & (L+l) \end{bmatrix}$$

is the transformation matrix,

Oppositely, the vehicle velocity can be obtained from the Wheel velocity using pseudo inverse matrix as equation,

$$V_0 = J^*.\mathrm{V}_w + (I - J^*.\mathrm{J})\,\mathrm{w}$$

The mobile robot is under velocity control. Given the Cartesian space velocity command, the velocity command to each motor is computed using the inverse Jacobian

$$\begin{bmatrix} V_{1w} \\ V_{2w} \\ V_{3w} \\ V_{4w} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -(\mathrm{L}+l) \\ 1 & 1 & (\mathrm{L}+l) \\ 1 & 1 & -(\mathrm{L}+l) \\ 1 & -1 & (\mathrm{L}+l) \end{bmatrix} . \begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix}$$

Where $\boldsymbol{V_x}$ is system speed in x direction, $\boldsymbol{V_x}$ is system speed in y direction and W is system rotational speed according to center point.

As a result, in the global reference coordinates (X, Y, Z), the velocity of the mobile platform is defined by each element is given follows;

R represents mecanum wheel radius.

$$V_x = \frac{R_w}{4} \cdot \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_4 \right)$$

$$V_y = \frac{R_w}{4} \cdot \left(-\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 - \dot{\theta}_4 \right)$$

$$W_z = \frac{R_w}{4(\mathrm{L} + l)} \cdot \left(-\dot{\theta}_1 + \dot{\theta}_2 - \dot{\theta}_3 + \dot{\theta}_4 \right)$$

3. SYSTEM DESIGN

In this section, mechanical and electrical design of the proposed system will be introduced. In the mechanical design sub-section, the following topics are covered: The designed and modelled wheel specifications, dimensional features of the chassis, introduction to actuators used in the mechanical system. In the electrical design sub-section, the following topics are covered: micrcontroller, brushed dc motor, logic level shifter and motor drivers. System elements are assembled as below (Figure 3.1).

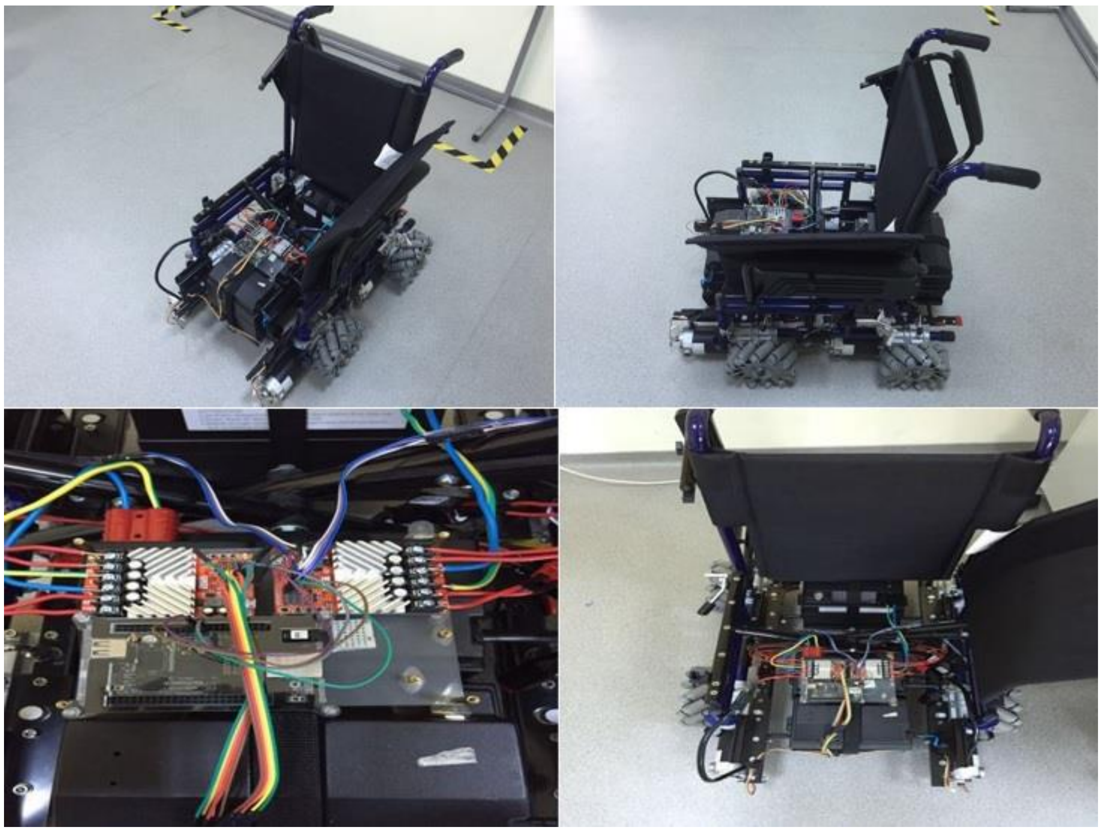


Figure 3.1: Completed view of designed system

3.1 Mechanical Features of Mecanum Wheel

With the help of improving mecanum wheel, it has higher weight capacity and smooth motion. And the higher weight capacity is achieved through the 8" Mecanum Support Spacer. Using this spacer in place of the 1600 Spacer increases the load capacity from about 80lbs/wheel to 500 lbs/wheel(nearly 226kg). This spacer is molded to support each individual tab on the aluminum side plates of the mecanum wheel. This reduces the risk that the side plates bend and the roller axles will become misaligned.

By adding the outer rollers to the 8" mecanum wheels (Figure 3.2) we achieve a smoother rolling wheel. During the transition from one of the 12 inner rollers to the next, the outer roller hits the ground and decreases the amount of up and down motion.

Specifications (for an individual wheel):

- Diameter: 203 mm
- Body Material: Aluminum
- Load Capacity: 226 kg
- Coefficient of Friction, Forward/Backwards: .7
- Coefficient of Friction, Sideways: .6
- Weight: 1,5 kg
- Number of Rollers: 12 Inner Rollers, 24 Outer Rollers

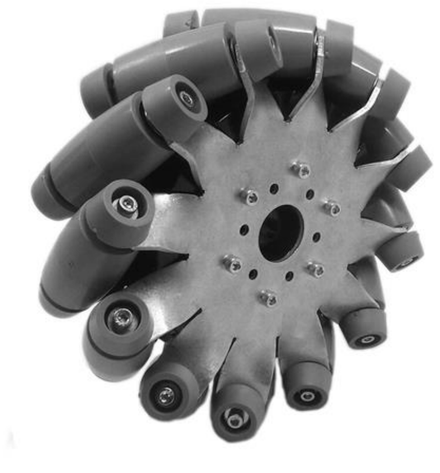


Figure 3.2: Mecanum wheel with 12 rollers

3.2 Dc Motor Driver

The selected Dc motor specifications are 350w and 24v. Therefore a high current flow capacity dc motor driver was selected. Besides it was considered communication interfaces for to communicate with controller card. Because of this needs dual dc motor driver was the possible option.

The RoboClaw motor controllers (Figure 3.3) from Ion Motion Control can control a pair of brushed DC motors. The controller has USB serial, TTL serial, RC, or analog inputs for communication. Main pinout are shown as below (Figue 3.4). Integrated dual quadrature decoders make it easy to create a closed-loop speed control system.

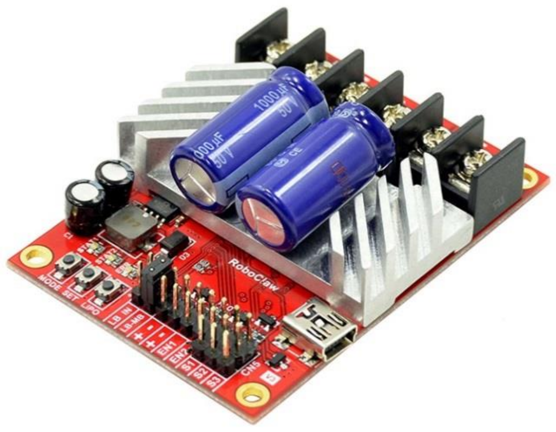


Figure 3.3: Roboclaw 2x30A Motor driver

Key features of motor driver (Table 3.1) as it below;

Motor Channels	2
Operating Voltage	6V ~ 34V
Continuous output current	30A
Peak output current	60A
Communication interfaces	USB Serial, TTL, RC, Analog

Table 3.1: Main features of dual motor driver

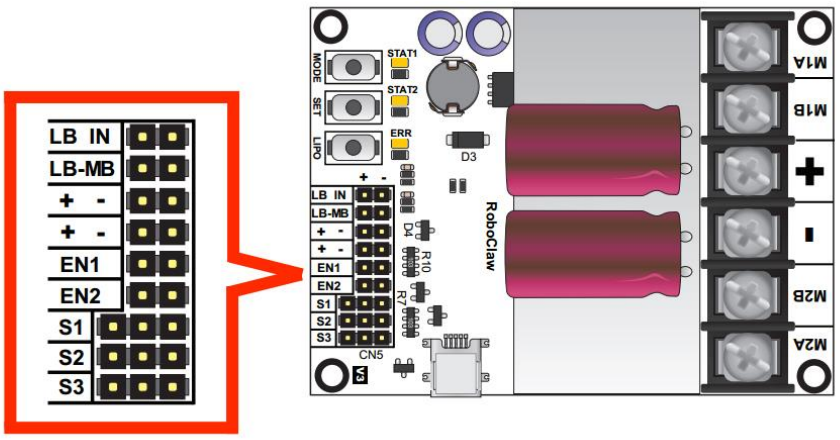
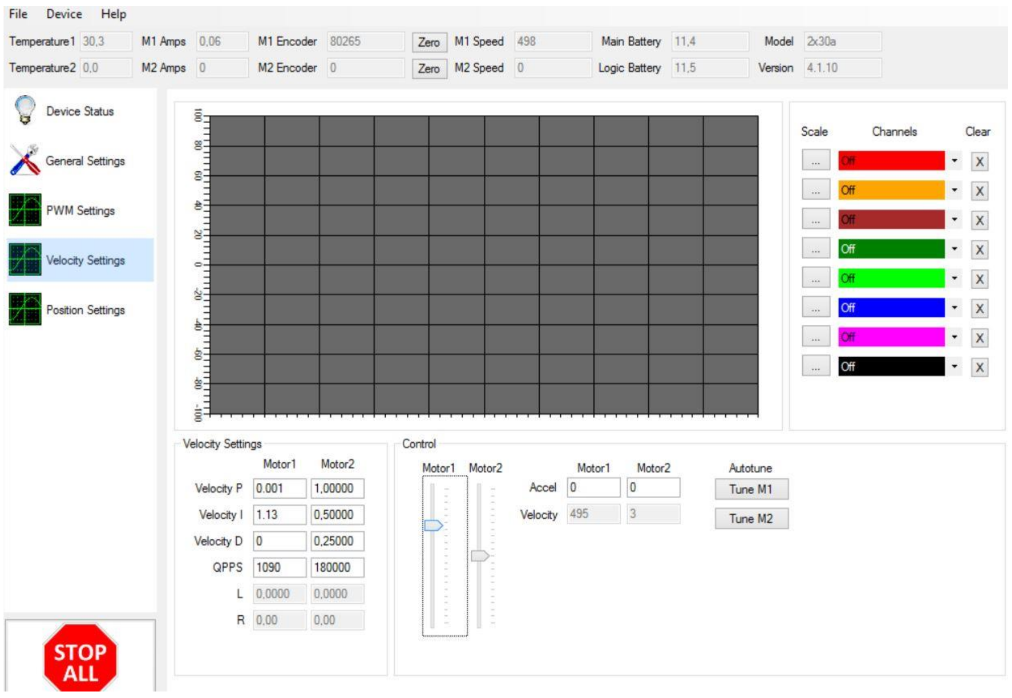


Figure 3.4: Main pinout of Roboclaw 2x30A

It has also dual feedback inputs for PID closed-loop control and speed control with quadrature encoders, up to 19,6 million encoder pulses per second also position control with analog encoders or potentiometers.

Ion Motion provides a GUI (Figure 3.5) to control each motor respectively. Interface of control program over USB serial as shown below;



Temperature1 30,3 | M1 Amps 0,06 | M1 Encoder 80265 | M1 Speed 498 | Main Battery 11,4 | Model 2x30a

Temperature2 0,0 | M2 Amps 0 | M2 Encoder 0 | M2 Speed 0 | Logic Battery 11,5 | Version 4.1.10

Velocity Settings

	Motor1	Motor2
Velocity P	0.001	1,00000
Velocity I	1.13	0,50000
Velocity D	0	0,25000
QPPS	1090	180000
L	0,0000	0,0000
R	0,00	0,00

	Motor1	Motor2
Accel	0	0
Velocity	495	3

Figure 3.5: Ion motion motor driver card interface

It supported by C++ library and Python2x module. With the help of these libraries, another controller card can communicate properly.

3.3 Mini Computer

Motor driver needs to be controlled by another controller. For this purpose a board is needed. It can be used any specific board to do it. When we examine our system requirements, beaglebone black is the good choice. The beaglebone black (Figure 3.6) is credit card sized single board computer which has embedded linux operating system in onboard flash.

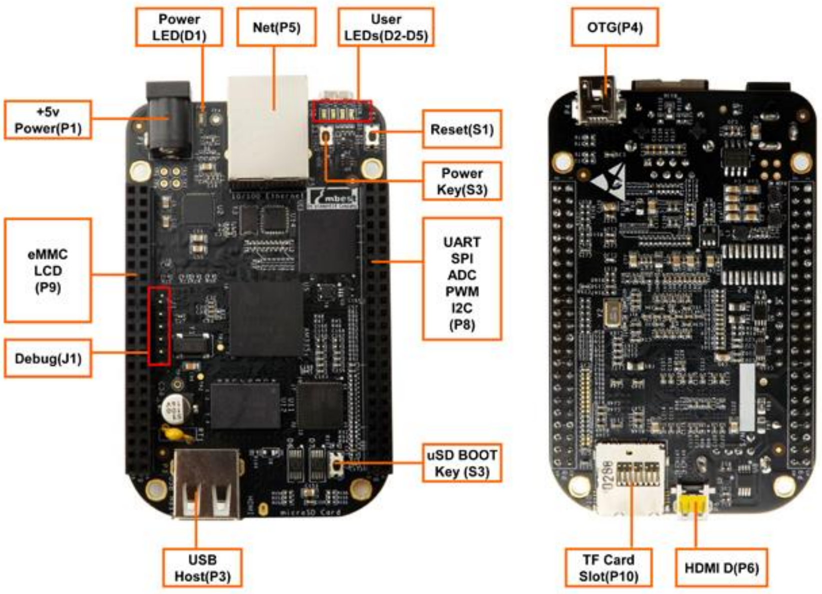


Figure 3.6: Beaglebone Black Rev C

Key features of beaglebone black (Table 3.2) as below;

Processor	Sitara AM3359AZCZ100 1GHz
SDRAM Flash	512MB DDR3
Onboard flash memory	4GB, embedded flash
TTL level	3.3V
Internet connection	10/100 Ethernet
USB Interface	1 USB port
Power input	5V DC
Flash expantion port	MicroSD port
Video output	Micro HDMI port

Table 3.2: Main features of mini controller

And also it has i/o expansion headers which can be controlled by as if client needs.

- Each digital I/O pin has 8 different modes that can be selected, including GPIO, totaly 65 possible digital i/o ports and each port can produce interrupts.
- Up to 8 digital pins can be configured with pulse with modulators to produce to control motors or create analog voltage without taking up any extra CPU cycles.
- For the analog inputs there are 7 analog input with 1.8v level and 12 bit analog to digital converter
- Serial communication is supported by 4 uart ports. Each of them has tx and rx port.
- There are 2 I2C bus is utilized for reading EEPROMS on cape add-on boards, the second I2C bus is available for you to configure to use.
- For shifting data fast, there are 2 SPI ports.

3.4 Logic Level Shifter

TTL levels between motor controller and our computer is slightly different, however the difference makes big data loss between communications. Beaglebone black communicate at 3.3V ttl level which means 3.3V is true value and less than 3.3V is False value for it. It should transfer and receive data inter motor controller. However roboclaw communicate with 5V TTL level. To overcome this problem, it was necessary to use bi-directional logic level shifter (Figure 3.7). With the help of this integrated circuit, motor driver cards and controller card are compatible with each other.

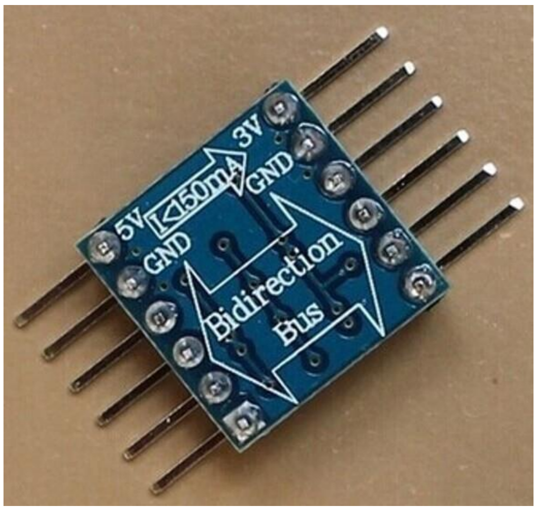


Figure 3.7: Bidirectional logic level shifter

3.5 Brushed Dc Motor

A Brush dc motor provides precision control of speed, driven by a direct current. Noted for a particularly high ratio of torque to inertia, the brush dc Motor has the potential to supply three to four times more torque than it is rated torque. If needed, it can even provide up to five times more, without stalling. The brush dc Motor consists of six different components: the axle, armature/rotor, commutator, stator, magnets, and brushes. The brush dc motor offers stable and continuous current, using rings to power a magnetic drive that operates of the motor armature. Perhaps one of the earliest used motors, the brush dc motor (Figure 3.8) is commonly used because of the ability to vary the speed-torque ratio in almost any way. There proposed system requires 4 brushed dc motor. The mechanism should be carry own weight and user weight, are able to hold resistance torques. The dc motor specifications (Table 3.3) are;

Operating voltage	24V
Amper	14A
RPM	4200
Watt	350Watt
Ratio	32:1
Output shaft diameter	17 mm
Output shaft speed	135 RPM
Input shaft speed	4320 RPM

Table 3.3: Features of Brushed DC motor

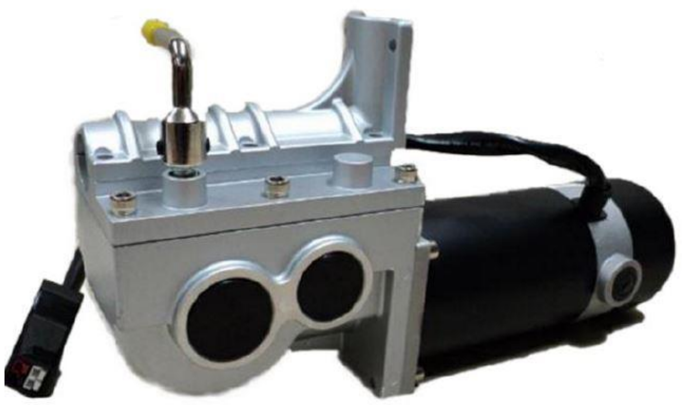


Figure 3.8: 24v DC motor with gearbox

4. SYSTEM IDENTIFICATION

The design of a control system requires a mathematical model of the dynamics of the process often a dynamical model can be difficult to obtain due to the complexity of the process, whose dynamics may be even (partially or completely) unknown. Even if we have a mathematical model, sometimes this is too complex to base a controller design on it (large state dimensions, nonlinearities, etc.) Model reduction is a way to go, but requires a (linear) model to start with. System identification is a procedure to build a mathematical model of the dynamics of a system from measured data. There are some approches to identify system (Figure 4.1) data from measured data. For instance, White-box identification is based on estimating parameters of a physical model from data, Grey-box identification is based on giving generic structure estimate parameters from data, Black-box model determining model structure and estimate parameters from data. There are some methods to take datas from system for example impulse response and step response can be use. [10] The process of identification shown as below,

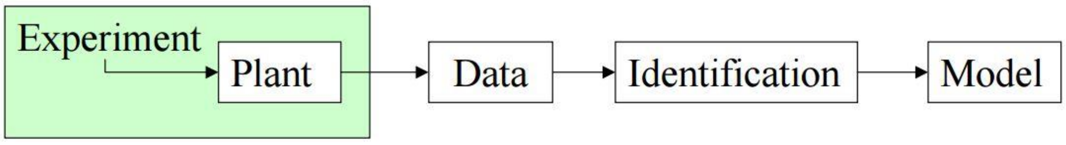


Figure 4.1: System identification process

For mecanum wheelchair dc motor constant values should be determined cause there is no way to take these values from manufacturer. The linear model is created by system designer which is a mathematical approch to physical system. After taking datas from system if we have knowledge about mathematical model system poles and zeros can be implemented. Then model identification algorithm can determine exact values according to measured data. Then the rest of system linear models are added. Most important thing about finding right model of system is measured data, poles and zeros. If there is no clue about system mathemcatical model in other words its hard to

define it cause of complexity, the approch is estimate and error. Despite our system elements is mathematicaly modeled before, model paramters are unknown.
Simplified model and plant relation (Figure 4.2) as below;

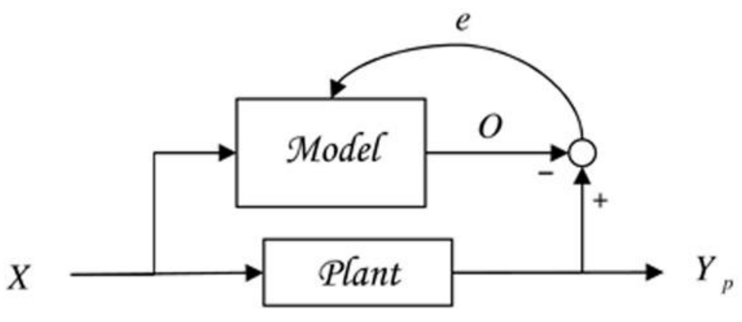
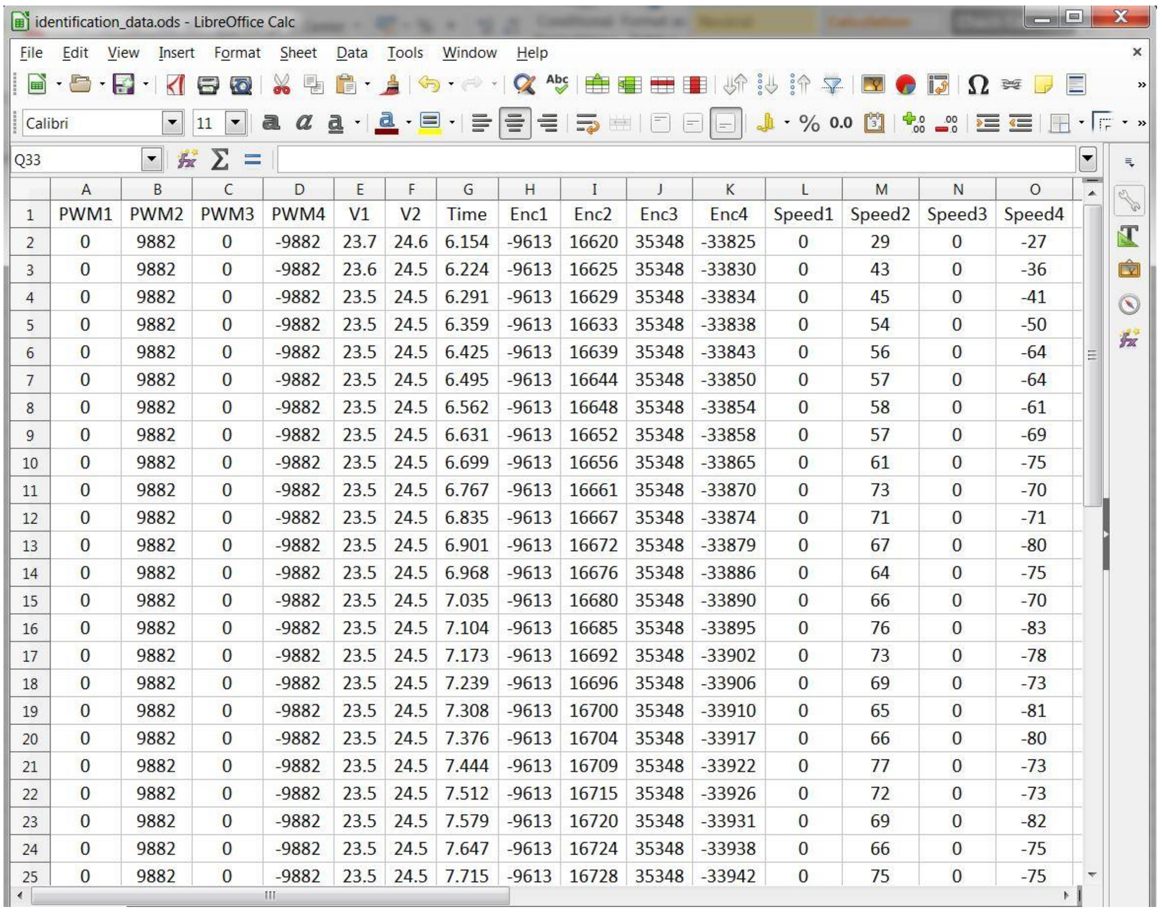


Figure 4.2: Plant and Model relationship

All datas (Figure 4.3) from system are taken with Beaglebone black rev c and xls format used for possible to open in excel and Matlab as it figure. Pwm values, time, voltage, encoder and speed are taken from motors for identification.



identification_data.ods - LibreOffice Calc
File Edit View Insert Format Sheet Data Tools Window Help
Calibri
11
Q33

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	PWM1	PWM2	PWM3	PWM4	V1	V2	Time	Enc1	Enc2	Enc3	Enc4	Speed1	Speed2	Speed3	Speed4
2	0	9882	0	-9882	23.7	24.6	6.154	-9613	16620	35348	-33825	0	29	0	-27
3	0	9882	0	-9882	23.6	24.5	6.224	-9613	16625	35348	-33830	0	43	0	-36
4	0	9882	0	-9882	23.5	24.5	6.291	-9613	16629	35348	-33834	0	45	0	-41
5	0	9882	0	-9882	23.5	24.5	6.359	-9613	16633	35348	-33838	0	54	0	-50
6	0	9882	0	-9882	23.5	24.5	6.425	-9613	16639	35348	-33843	0	56	0	-64
7	0	9882	0	-9882	23.5	24.5	6.495	-9613	16644	35348	-33850	0	57	0	-64
8	0	9882	0	-9882	23.5	24.5	6.562	-9613	16648	35348	-33854	0	58	0	-61
9	0	9882	0	-9882	23.5	24.5	6.631	-9613	16652	35348	-33858	0	57	0	-69
10	0	9882	0	-9882	23.5	24.5	6.699	-9613	16656	35348	-33865	0	61	0	-75
11	0	9882	0	-9882	23.5	24.5	6.767	-9613	16661	35348	-33870	0	73	0	-70
12	0	9882	0	-9882	23.5	24.5	6.835	-9613	16667	35348	-33874	0	71	0	-71
13	0	9882	0	-9882	23.5	24.5	6.901	-9613	16672	35348	-33879	0	67	0	-80
14	0	9882	0	-9882	23.5	24.5	6.968	-9613	16676	35348	-33886	0	64	0	-75
15	0	9882	0	-9882	23.5	24.5	7.035	-9613	16680	35348	-33890	0	66	0	-70
16	0	9882	0	-9882	23.5	24.5	7.104	-9613	16685	35348	-33895	0	76	0	-83
17	0	9882	0	-9882	23.5	24.5	7.173	-9613	16692	35348	-33902	0	73	0	-78
18	0	9882	0	-9882	23.5	24.5	7.239	-9613	16696	35348	-33906	0	69	0	-73
19	0	9882	0	-9882	23.5	24.5	7.308	-9613	16700	35348	-33910	0	65	0	-81
20	0	9882	0	-9882	23.5	24.5	7.376	-9613	16704	35348	-33917	0	66	0	-80
21	0	9882	0	-9882	23.5	24.5	7.444	-9613	16709	35348	-33922	0	77	0	-73
22	0	9882	0	-9882	23.5	24.5	7.512	-9613	16715	35348	-33926	0	72	0	-73
23	0	9882	0	-9882	23.5	24.5	7.579	-9613	16720	35348	-33931	0	69	0	-82
24	0	9882	0	-9882	23.5	24.5	7.647	-9613	16724	35348	-33938	0	66	0	-75
25	0	9882	0	-9882	23.5	24.5	7.715	-9613	16728	35348	-33942	0	75	0	-75

Figure 4.3: Data log example

To get best results from simulations, dc motor transfer function should be determined. Therefore Matlab system identification toolbox (Figure 4.4) is used. To increase confidence level motor have tested under different scenarios. As an input driven voltage, as an output shaft velocity are taken. Different scenarios are tested which are full duty cycle, half duty cycle and increasing duty cycle 10% in every 10 seconds. This process repeated for each motor.

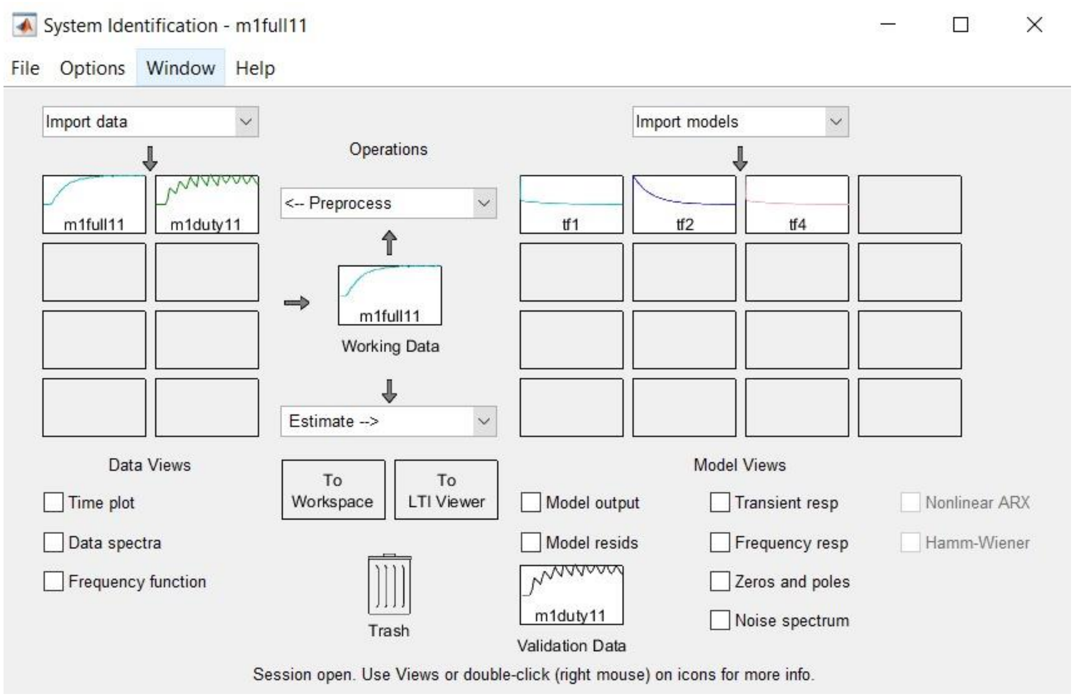


Figure 4.4: General view of system identification toolbox

After taking measured datas from system, these values are simulated and system parameters are found. Step response of mathematical modelled system and identified transfer function are shown as below (Figure 4.5 – Figure 4.6 –Figure 4.7 – Figure 4.8).

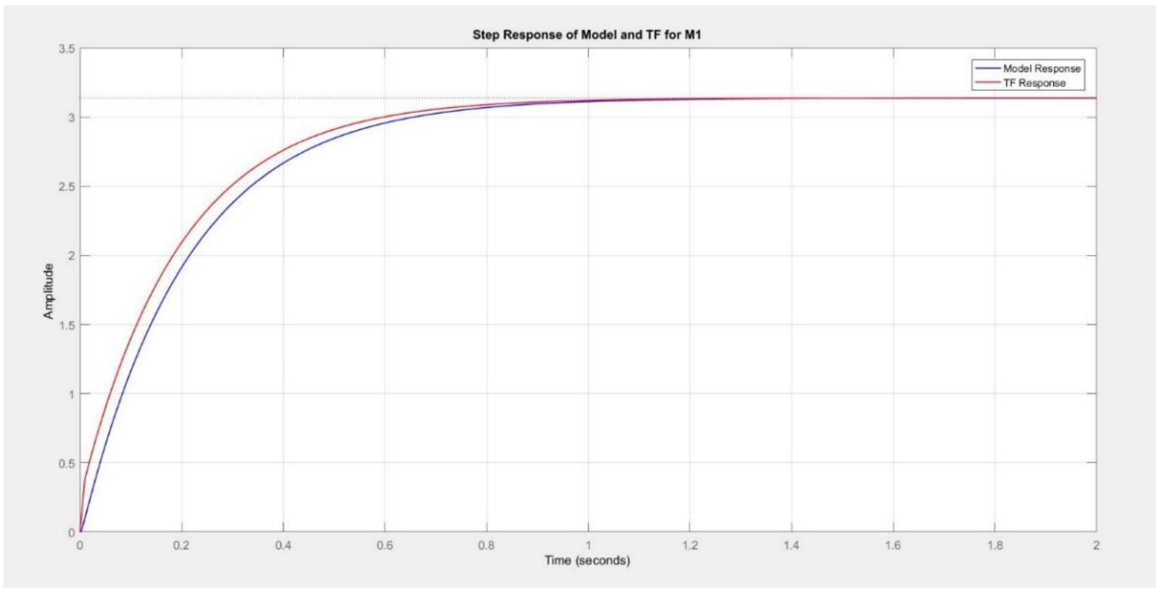


Figure 4.5: Step response of dc motor model and transfer function for motor 1

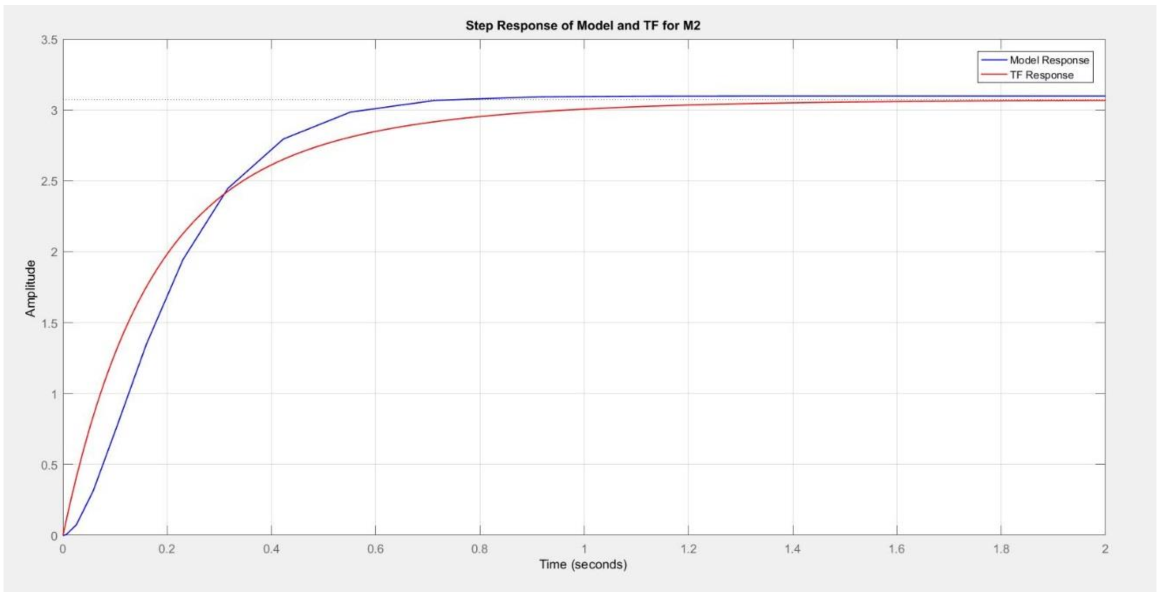


Figure 4.6: Step response of dc motor model and transfer function for motor 2

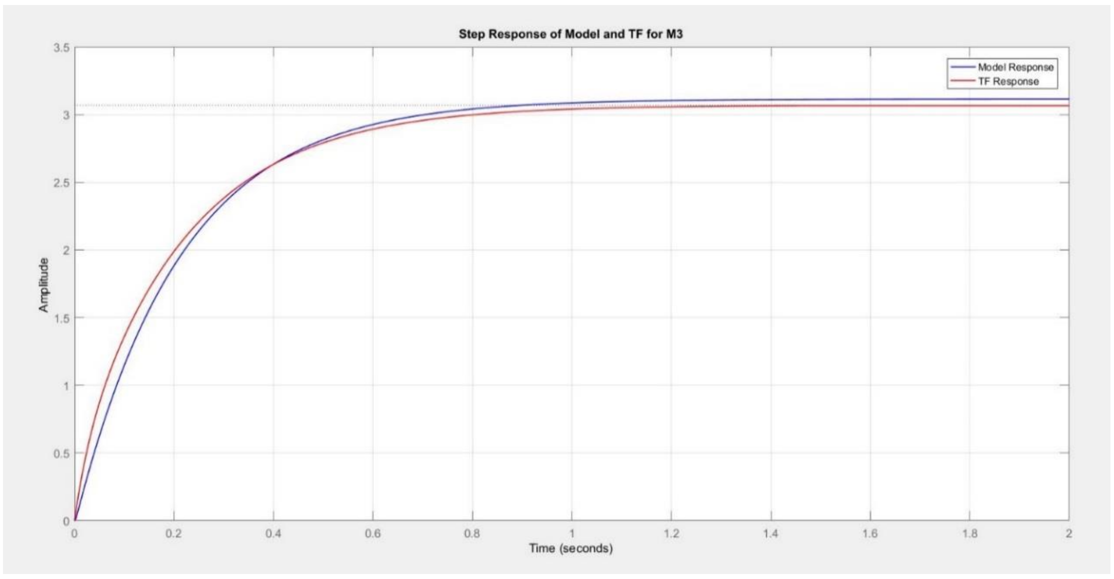


Figure 4.7: Step response of dc motor model and transfer function for motor 3

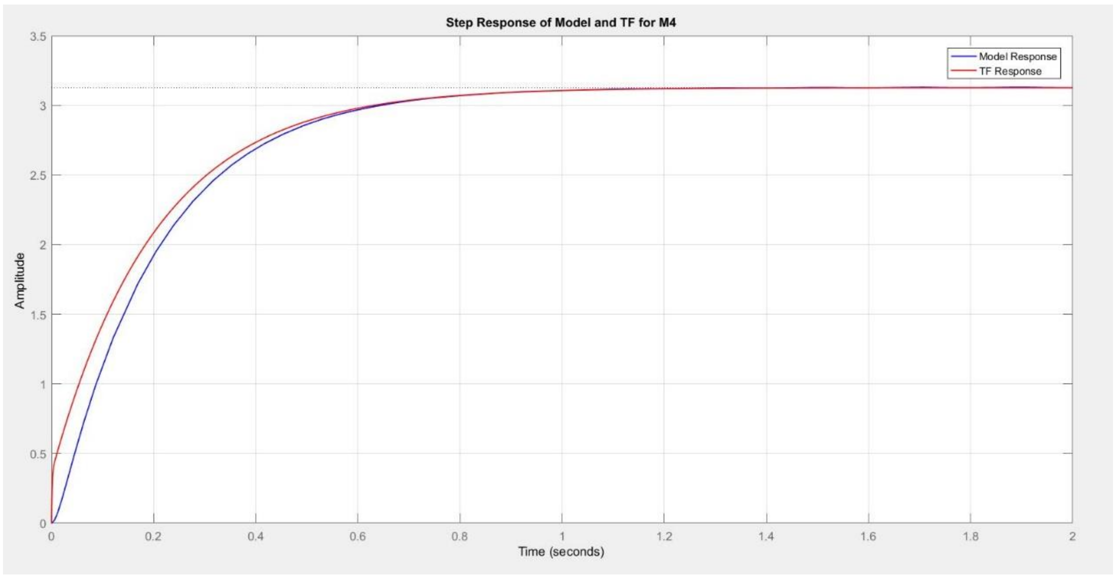


Figure 4.8: Step response of dc motor model and transfer function for motor 4

After parameter estimation (Table 4.1), dc motor parameters are implemented to the mathematical model.

	Bm	Jp	Kp	Ra	Ta
M1	0,29475	0,20424	0,22149	0,3297	0,0028326
M2	1,0861	0,17899	0,038898	0,26134	0,081072
M3	0,25556	0,15695	0,20891	0,43873	0,0020515
M4	0,3368	0,034265	0,25939	0,018088	1,0961

Table 4.1: Estimated motor simulation parameters

With the identification toolbox different transfer functions are obtained. When all variables considered suitable identified functions (Table 4.2) are chosen.

M1	$\frac{111s + 6113}{s^2 + 383.3s + 1948}$
M2	$\frac{17.27s + 59.55}{s^2 + 9.785s + 19.39}$
M3	$\frac{26.04s + 482.6}{s^2 + 39.29s + 157.2}$
M4	$\frac{317.1s + 1287}{s^2 + 847.9s + 4116}$

Table 4.2: Identified transfer function for each motor

Identified transfer functions of each motors are compared to each other. In order to establish the confidence level the cross correlation tests for each identified motor transfer function are realized. According to residual level test results shown as follow figures (Figure 4.9 – Figure 4.10 –Figure 4.11 – Figure 4.12). The all obtained residue of the transfer functions are located in confidence level.

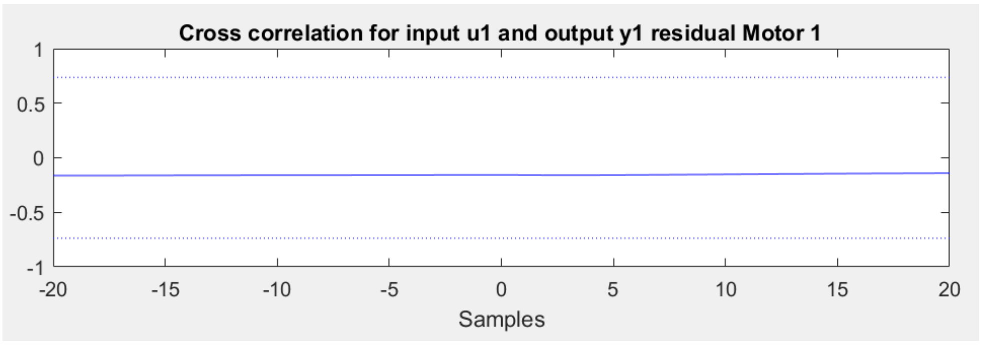


Figure 4.9: Cross correlation results for motor 1

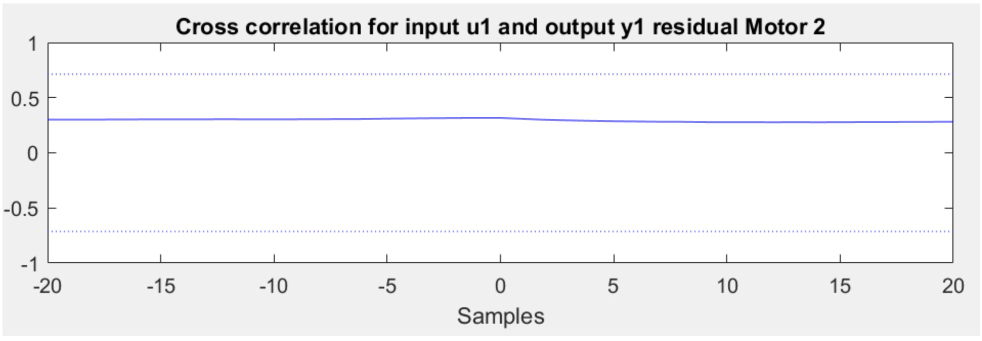


Figure 4.10: Cross correlation results for motor 2

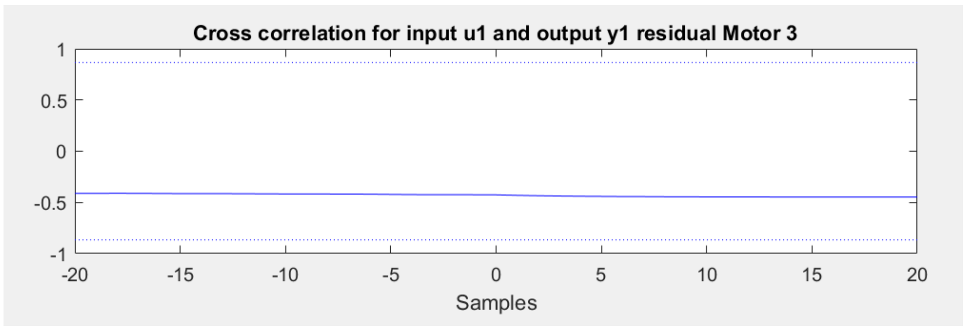


Figure 4.11: Cross correlation results for motor 3

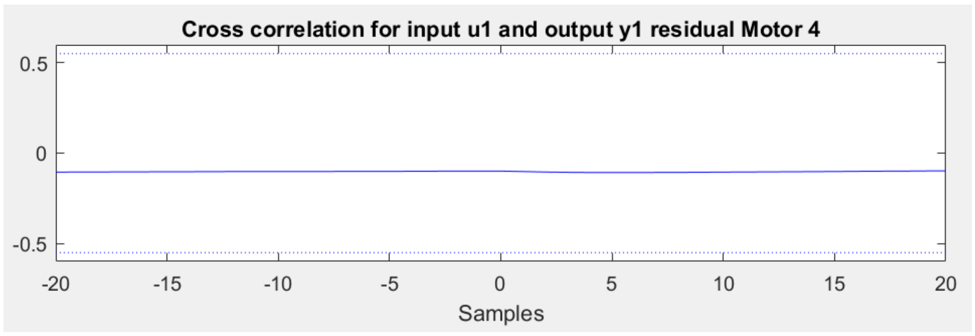


Figure 4.12: Cross correlation results for motor 4

5. CONTOL OF SYSTEM

There are several algorithms for motor control. For instance PID, LQR and MPC can be used as a controller. Classical pid control that generates the control input using the error between desired velocity and instant velocity. To improve performance of algorithm the position error at the previous control time step when it produces the current control input. LQR is based on state feedback control and MPC is based on known plant model and has an algorithm inside which calculates control inputs according to referance value.

5.1 Linear Quadratic Regulator

Optimal control refers to a class of methods that can be used to synthesize a control policy which results in best possible behavior with respect to the prescribed criterion (i.e. control policy which leads to maximization of performance). The main objective of optimal control is to determine control signals that will cause a process (plant) to satisfy some physical constraints and at the same time extremize (maximize or minimize) a chosen performance criterion (performance index (PI) or cost function). The optimal control problem is to find a control which causes the dynamical system to reach a target or follow a state variable (or trajectory) and at the same time extremize a PI which may take several forms. [11]

Linear quadratic regulator (LQR) is one of the optimal control techniques, which takes into account the states of the dynamical system and control input to make the optimal control decisions. This is simple as well as robust. After linearization of nonlinear system equations about the equilibrium position having initial conditions, the linear state-space equations is obtained as,

$$\dot{X} = AX + Bu$$

The state feedback control

$$u = -KX$$

leads to

$$\dot{X} = (\mathrm{A} - \mathrm{BK})\,\mathrm{X}$$

where, K is derived from minimization of the cost function

$$J = \int_{0}^{\infty} (X^T QX + u^T Ru)\, dt$$

where Q is a positive-definite Hermitian or real symmetric matrix and R is a positive-definite Hermitian or real symmetric matrix. The second term on the righthand side of equation accounts for the expenditure of the energy of the control signals. The matrices Q and R determine the relative importance of the error and the expenditure of this energy. In this problem, we assume that the control vector u(t) is unconstrained.
Therefore, if the unknown elements of the matrix K are determined so as to minimize the performance index, then u(t)=–Kx(t) is optimal for any initial state x(0). The LQR gain vector K is given by,

$$K = R^{-1} B^T P$$

where, P is a positive definite symmetric constant matrix obtained from the solution of matrix algebraic reccatti equation is,

$$A^T P + PA - PBR^{-1} B^T P + Q = 0$$

As shown below in basic block diagram of LQR controller (Figure 5.1) K feedback constant determined according to cost function which is related to Q an R constant matrices.

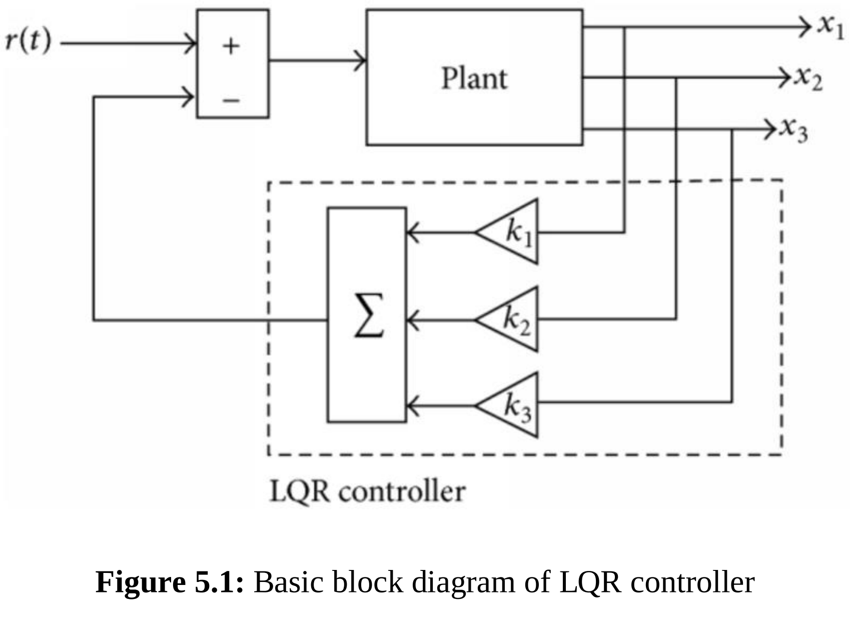


Figure 5.1: Basic block diagram of LQR controller

The main aim in LQR control design is to minimize the quadratic cost function of J. And then it turns out values of Q and R, the cost function has a unique minimum that can be obtained by solving the algebric ricatti equation as showed above. The parameter can be Q and R can be used as design parameters to penalize the state variables and the control signals. The larger these values are, the more penalized signals. Basically, choosing a large value for R means trying to stabilize system with less energy. This usually called expensive control strategy. On the other hand choosing small value for R means, called cheap control strategy because control input is not penalized. Similarly large values of Q means trying to stabilize system with least possible changes in the states and large Q implies less concern about the changes in states.

But there are some methods to identify Q and R. Q matrix basicaly selected according to states as mentioned above. The diagonal of Q matrix as below and matrix should be symetric. q is related to system states and is positioned diagonally in Q matrix.

$$Q = \begin{bmatrix} q_1 & & & \\ & . & & \\ & & . & \\ & & & q_n \end{bmatrix}$$

And R matrix related to input. r represents inputs of system.

$$R = \begin{bmatrix} r_1 & & & \\ & . & & \\ & & . & \\ & & & r_n \end{bmatrix}$$

5.2 Model Predictive Control

Model Predictive Control (MPC) is an optimal control theory based on numerical optimization. Future control inputs and future responses are predicted according to identificated or modelled system and optimized at regular intervals with respect to a cost function. From its origins as a computational technique for improving control performance in applications within the process and petrochemical industries, predictive control has become arguably the most widespread advanced control methodology currently in use in industry. [12] MPC has a sound theoretical basis and its stability, optimality, and robustness properties are well understood. Despite being very simple to design and implement, MPC algorithms can control large scale systems with many control variables, and, most importantly, MPC provides a systematic method of dealing with constraints on inputs and states. Such constraints are present in all control engineering applications and represent limitations on actuators and plant states arising from physical, economic, or safety constraints. In MPC these constraints are accounted for explicitly by solving a constrained optimization problem in real-time to determine the optimal predicted inputs. Nonlinear plant dynamics can be similarly incorporated in the prediction model.

5.3 History of Model Predictive Control

Towards the end of the 1970 there was published mant articles over model predictive control methods related to the implementation of the industry.
Richalet 1978 Model Algorithmic Control algorithm, and Cutler and Ramakt 1980
In the dynamic matrix control algorithm came up with the first principles. In both algorithms there was used a dynamic process to foresee how to input effects system output for future output.
In model algorithmic control is used impulse response coefficients, despite that in dynamic matrix control algorithm is used coefficients obtained from the step response. In these studies are tried to select predicted control signals to minimize error.
In 1980 Garcia gathered MPC many different algorithms in the petrochemical sector under the study. Many of these applications were used multivariate systems.
Clarke brought out from her works generalized predictive control algorithm in 1987. This algorithm is based on generalized minimum variance method. Also Richalet brought out the predictive functional contol after these articles in the future. [12]
In 1994, work of Morari there are study on model predictive control algorithm in forms of state space. This study is pionered to use state space theories in this algorithm, in addition like nonlinear systems in many complex systems this control algorithm is played major role.
Although there are some studies on robustness theorems on generalized predictive control algorithm, the absentess of generel stability results still was in there. In 1991 Clarke and Scattolini developed bounded predictive control algorithm to overcome stability problem. [12]

5.4 Advantages and Disadvantages of Model Predicitive Control

Model predictive control does not comprise of single control method. At first it contains MAC(model algoritmic control), DMC(dynamic matrix control) and

GPC(generalized predictive control) and many control methods. Including first recommended methods, the main feature of these methods includes intermal model and control signal is calculated based on sliding horizin principle and predicted system responses. The difference between MPC control methods is internal models and cost functions to calculate control signal. [13]

At the process control effect of model predictive control is great. At first like robot manipulators, cement industry, distialation column many applications have been developed by being predictive control. These systems successfully control gives a good idea about the capacity of MPC.

When MPC compared other control methods, the main features are reported by Camacho and Bordons (1999) as follows;

- It can be easily understandable and easy to adjust by people who have limited knowladge about control
- It is used for complex systems, dynamicly simple systems, had long response time systems and unstable systems
- It can be used in MIMO systems easily.
- Its nature is configured to compansate for dead time delays.
- It is useful when future reference values known.
- It produces easily applicable a linear control law.
- It has suitable format to develop and regulations.

However MPC has some disadvantages,

- Although easy to implement control algorithm and low calculation performance, the acquisition is more complex than conventional PID controller.
- If the process does not change and there is no restriction, control law can be calculated independent from offline, however in adaptive case all calculations must be done every sampling time. If technologic developments in last ten years considered and central process unit capacity increased according to moore law process control can be done.
- Other disadvantage of MPC is strictly dependent to mathematical model. If model is not precise MPC would not work properly. Some unmodelled system components cause to wrong MPC parameters. If overcome these problems theories can be applied on real system.

5.5 MPC Algorithms

MPC has different control sub metodogy. Most used methods are DMC, MAC and GPC. Dynamic matrix control (DMC) uses finite step response and easy to apply. Also adaption by workers in industries quickly. It does not need any information about system order. Besides DMC does not fit for open loop unstable systems. It is especially used in petrochemistry industry. Second method MAC is similar to DMC however it use finite impulse response model to work. The number of adjustable parameter is less. Control horizon N_c is selected equal to prediction horizon N_p. Prediction calculation starts at the first step. Generalized predictive control (GPC) was proposed by Clarke(1987). It is based on CARIMA (controlled auto regressive integrated moving average), however it was organized for state space model. Them most important advantage of this controller is to control unstable open loop systems. Also it has similar characterisctics with quadratic method. All basic MPC algorithms do not guarantee stability. [13]

5.6 Receding Horizon Approach

Only the first element of the optimal predicted input sequence u*(k) is input to the plant.

$u(\mathrm{k}) = \mathrm{u}^*(\mathrm{k} \mid \mathrm{k})$

The process (Figure 5.2) of computing u*(k) by minimizing the predicted cost and implementing the first element of u* is then repeated at each sampling instant k = 0, 1,.... For this reason the optimization defining u* is known as an online optimization. The prediction horizon remains the same length despite the repetition of the optimization at future time instants, and the approach is therefore known as a receding horizon strategy. Since the state predictions x and hence the optimal input sequence u* depend on the current state measurement x(k), this procedure introduces feedback into the MPC law, thus providing a degree of robustness to modelling errors. [14]

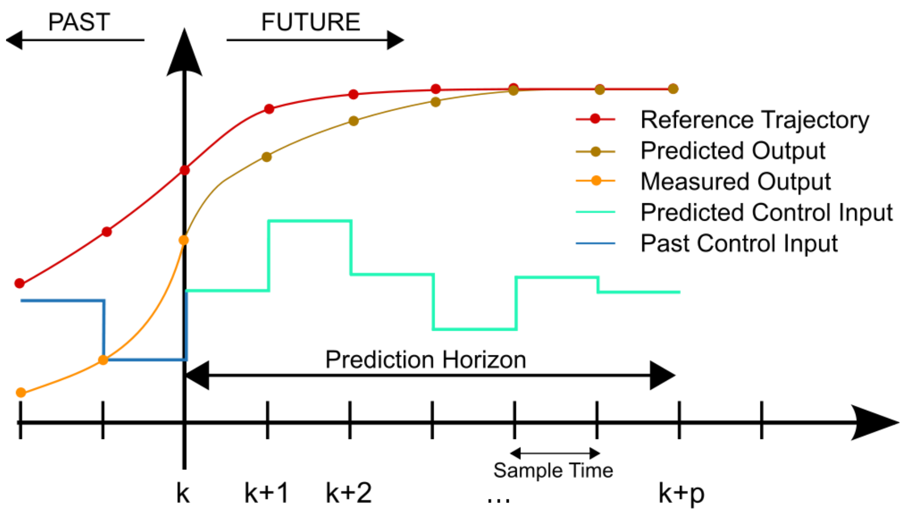


Figure 5.2: Discrete MPC receding horizion scheme

5.7 Model Types Used in Model Predictive Control

Prediction models are cornerstones of model predictive control. There are many model types when the MPC design. Some of them are clarified. Used prediction model should catch up process dynamics and calculate predicitons. Modeled wrong or deficient system models causes to unseccessful controlled system. Different strategies of MPC, interaction between system output and measurable inputs can be obtained by different models. In measurable inputs contain disturbances, consequently disturbance model should be considered. It is a great deal of advantage usage of noise model unmeasured inputs for capturing noise and modelling errors. Different algorithms use different process models. Because of this reason it should be considered. [12]

5.7.1 Step response model

It is used by DMC and similar algorithms. It is easy to obtain. After applying step input to the system the parameters obtain from depending on system response (Figure 5.3). The response obtained from a finite period of time, divided by step according to the sampling times. There is a widespread use in industry. The biggest advantage is that it does not require any prior knowledge of desired system model. The disadvantage of that is suitable for stable system however it can not be used in unstable systems. Besides, obtainin a large number of parameters of system is the model deficiency. [14]

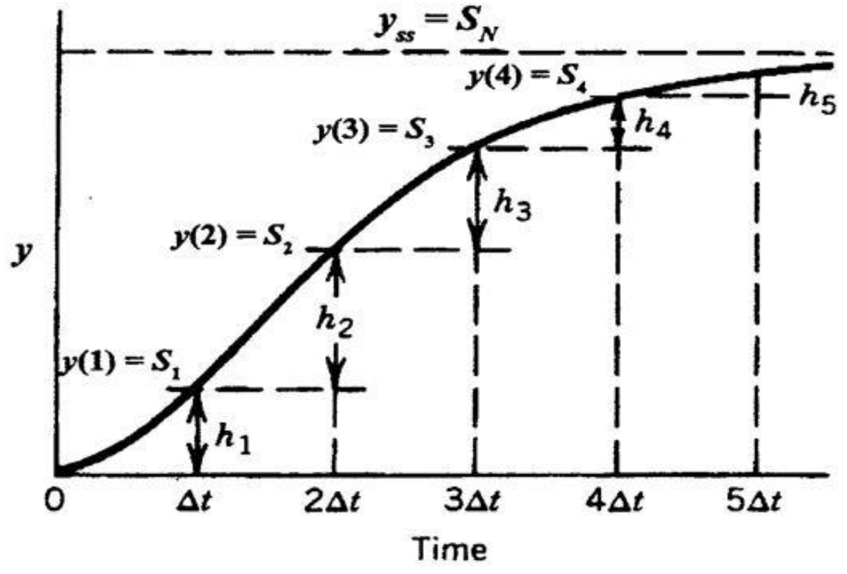


Figure 5.3: MPC step response model scheme

Consider a single input single output system, where u and y deviation variables. The response of step input over Δt period $\mathbf{h_1}$, $\mathbf{h_2}$….,$\mathbf{h_t}$. Δt can be selected settling time. Donate sampled values as $\mathbf{y_1}, \mathbf{y_2} \ldots \mathbf{y_n}$ and $\mathbf{u}, \mathbf{u_2} \ldots \mathbf{u_n}$ and predicted outputs . And incremental change in u will be donated as;

$$y_k = \sum_{i=1}^{N} h_{k-i} u_{k-i}$$

And u will be,

$$\Delta u_k = u_k - u_{k-1}$$

5.7.2 Impulse response model

MAC and similar algorithms use this model. Advantages and disadvantages are similar to step response model, the only difference is to apply dirac function as an input (Figure 5.4). Input output relation as follow,

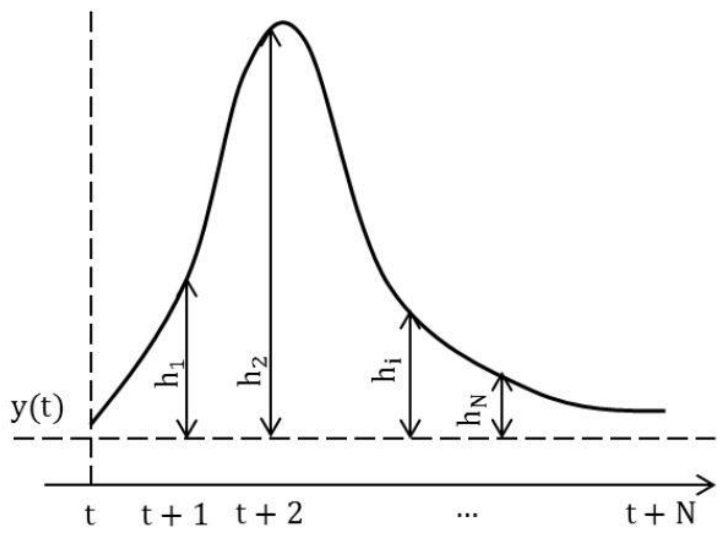


Figure 5.4: MPC step response model scheme

$$y(t)=\sum_{i=0}^{T} h_i u(t-i) = H(z^{-1})\,u(t)$$

After using this model prediction model equation is,

$$\hat{y}(t+k/t)=\sum_{i=0}^{T} h_i u(t+k-i) = H(z^{-1})\,u(t+k/t)$$

5.7.3 Transfer function model

GPC algoritihm uses this model. u(t) represents input, y(t) represents output.

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_{na} z^{-na}$$

$$B(z^{-1}) = 1 + b_1 z^{-1} + b_2 z^{-2} + ... + b_{nb} z^{-nb}$$

$$A(z^{-1})\,y(t) = B(z^{-1})\,u(t-1)$$

And prediction equation as follow,

$$\hat{y}(t+k/t) = \frac{B(z^{-1})}{A(z^{-1})} u(t+k/t)$$

5.7.4 State space based model predictive control

If the system model linearized and given in discrete time,

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = C_y x(k)$$

$$z(k) = C_z x(k)$$

In these equations x is an n-dimentional state vector, u is l-dimentional input vector, y is a measurable my dimentional output vector and z is a mz-dimentional can be controllable output vector. y and z vectors in the long run overlap, thus they can be accepted as same vector. On the other hand they can be measurable. According to this y=z and for C_y, C_z matrices mutual m dimentianla matrix will be use. For this reason m matrix will be used for common matrix.

The use of standard formats, such as state space model given above reason that is because has a direct connection with the theory of linear systems and control.

k is given as a moment of needs to be done as follows,

- Measure of y(k)
- Calculation of desired system input u(k)
- Applying u(k) to the system

As seen above, there is a delay between each implementation of measuring y(k) and applying u(k)

5.8 Base Parameters of MPC

In this section the most important parameters of designing MPC and selection criterias of these parameters will be discussed. These parameters directly effects of system outputs and effects on robustness and controller performance dramaticaly.

5.8.1 Prediction horizon

It is defined as $\boldsymbol{N_p}$. When receding horizon calculation is made, it will determine how long each horizon. Multiplied by the sampling time of the prediction horizon should be long enough to go through at least steady state of the closed loop system. Typical values are changes however between 20-30. However it may be longer or shorter depending on the model structure. MPC's processing load increase a very large selections. In case of long selections the sytem will reach the softer and slowly to the reference value. If the short selection of this values controller will work more aggressively and may be cause instability in some cases. If is there any time delay in system, it is not significant to select prediction horizon value as 1, because the output values will not change over delayed time.

5.8.2 Control horizon

When calculating how many controller signal can be change over the period of sampled time and defined by N_c. As described N_p and N_p input signal isconsidered constant. When the control horizion is increased controller behave more aggressively and calculation load increases, system react more quickly and more sensitive to the disturbances, thus its durability decreases. Relatively small selections are recommended. Generally at the beginning of design, can be selected quarter of prediction horizon. It recommends the product of the control sampling time and control horizon, it should include of 60% of steady state response. Control horizon should be obtained with some experiments on system. At each sampling control signal changes. However it can be changed by blocking at every sample time and it can be kept constant at specific ranges.

5.8.3 Weight matrix

One of the most important points to be considered in MPC design is the choice of the weight matrix. The most common form a weight change of input values punished by a matrix. Like in LQR controller, in which case the change is desired to be smaller, than it values is taken largest. If the change is less important than its weight is selected smaller. By changing values of these matrices, depending on system output and performance of system these values can be selected.

5.8.4 Reference trajectory

Another advantage of MPC, in case of previously known reference value of system, the controller operates in a more efficient manner by making calculations accordingly. Especially in robotic, servo systems and bacth reference change previously known often. Even when the reference value is constant and the reference change known before it provides an improvement in controller response.

5.8.5 Cost function

Designed controller to obtain controller law tries to make smaller the cost function. Different MPC algorithms uses different functions. MPC type controller in the state space model structure, the output reference deviation, the change of input signal and input signal deviation creates a cost function by multiplying pre determined weight matrix. The optimization problem is solved by determining the smallest cost fuction along with prediction horizon. The cost function to be considered for this as follows,

$$J(\mathrm{N}_1,\mathrm{N}_2,\mathrm{N}_3)=\sum_{J=\mathrm{N}_1}^{N_2}\delta(j)[\hat{\mathrm{y}}(\mathrm{t}+\mathrm{j}\,|\,\mathrm{t})-\mathrm{w}(\mathrm{t}+\mathrm{j})]^2+\sum_{J=1}^{N_u}\lambda(j)[\Delta(\mathrm{t}+\mathrm{j}-\mathrm{t})]^2$$

$$J_{mpc}=\sum_{i=0}^{N_p-1}\left(\begin{array}{c}\sum_{j=1}^{ny}|\,\mathrm{w}_{i+1}^{y}(\mathrm{y}_j(\mathrm{k}+\mathrm{i}+1\,|\,\mathrm{k})-\mathrm{r}_j(\mathrm{k}+\mathrm{i}+1))\,|^2+\ldots\\ \sum_{j=1}^{nu}|\,\mathrm{w}_{i.\mathrm{j}}^{\Delta u}\,\Delta\,\mathrm{u}_j(\mathrm{k}+\mathrm{i}\,|\,\mathrm{k})\,|^2+\ldots\\ \sum_{j=1}^{nu}|\,\mathrm{w}_{i.\mathrm{j}}^{\Delta u}(\mathrm{u}_j(\mathrm{k}+\mathrm{i}\,|\,\mathrm{k})-\mathrm{u}_{Tj}(\mathrm{k}+\mathrm{i})\,|^2+\ldots\end{array}\right)+\rho_{\varepsilon}\varepsilon^2$$

6. SIMULATION RESULTS AND COMPARISON

6.1 Simulation of Linear Quadratic Regulator

Acoording to LQR control full system block diagram is created with reverse kinematic, forward kinematic, dc motor models, state feedback constants. And full system block diagram (Figure 6.1) as shown below;

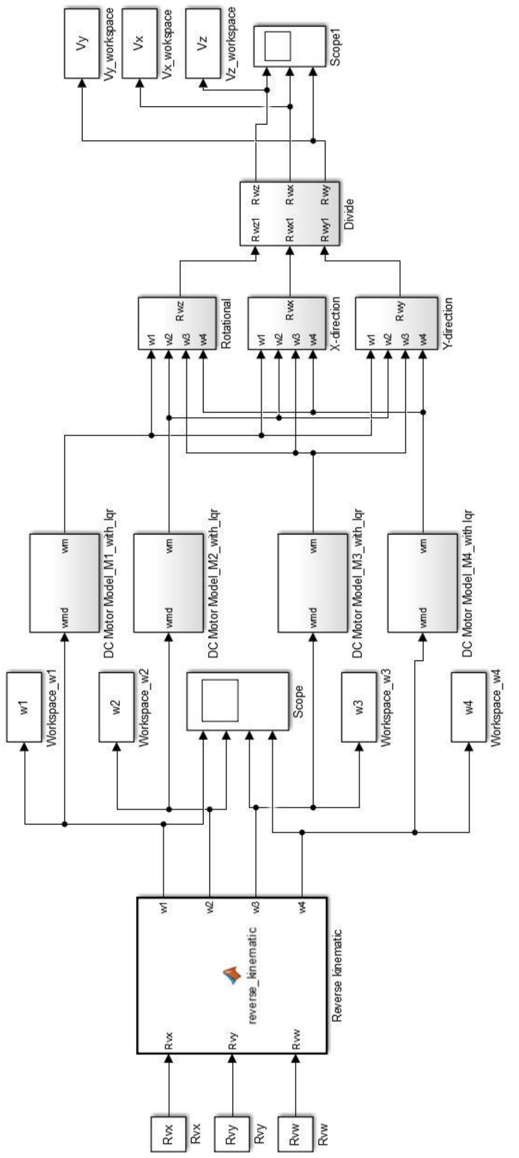


Figure 6.1: Simulation block diagram of full system with state feedback

Dc motor state space matrix is created according to identificated transfer functions and implemented in motor models.

After defining dc motor model (Figure 6.2) lqr state feedback matrix should be defined as user needs, and dc motor simulink model as below,

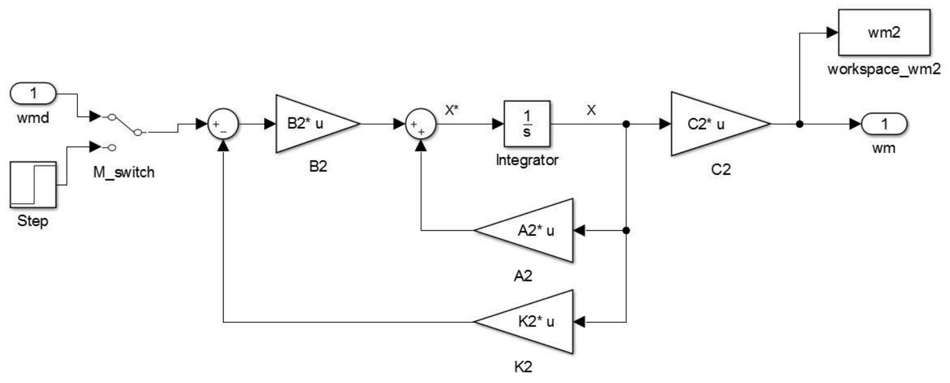


Figure 6.2: LQR control of dc motor

The state feedback matrix K is defined by Q and R matrixes. These two matrixes defined by user. After defining of Q and R matrix, K feedback matrix should be calculated by matlab. There is a lqr() function lqr(A,B,Q,R) and with the help of this function K matrix can be obtain.

R value is selected 1.5 and user defined input signal performed to the system. After simulation output responses (Figure 6.3) as below,

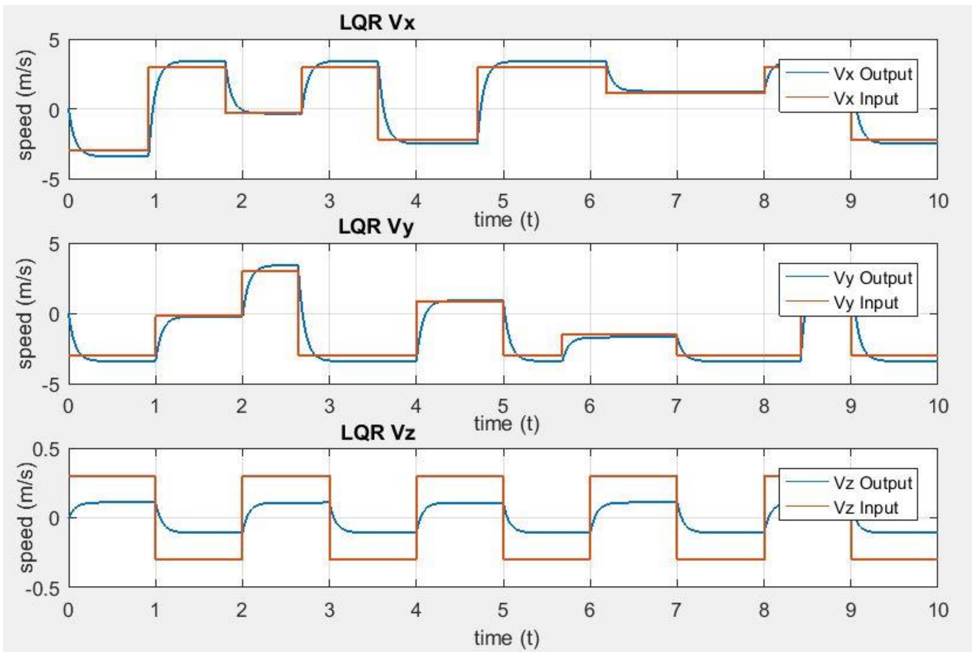


Figure 6.3: LQR control increased R (1.5) value

Increased R value effects target value. As seen from the simulation outputs system has overshooted in terms of user inputs. Decreasing R value enhances system output response.

Then different Q matrixes are tested. Large values of Q means trying to stabilize system with least possible changes in the states and large Q implies less concern about the changes in states. By taking into these considerations large valued Q simulation result (Figure 6.4) as below,

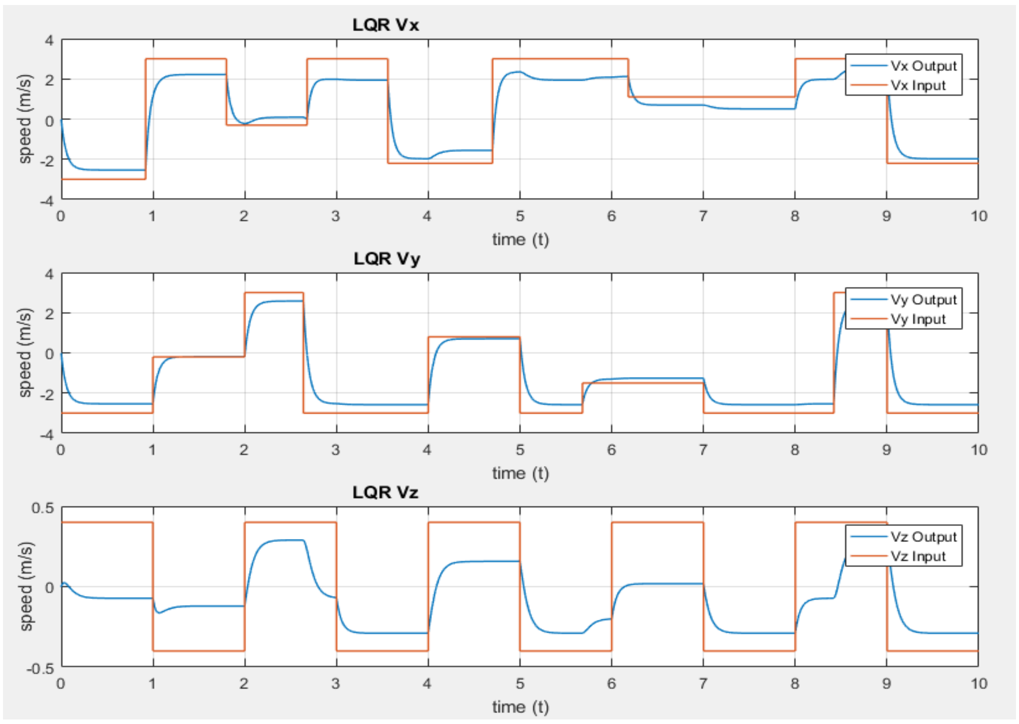


Figure 6.4: LQR control increased Q matrix value

As seen from the simulation results Q matrix penalizes system input and as a result system output (Figure 6.5) does not go to preferred value. Because of this reason Q and R values should be selected minimun values in terms of cheap cost function.

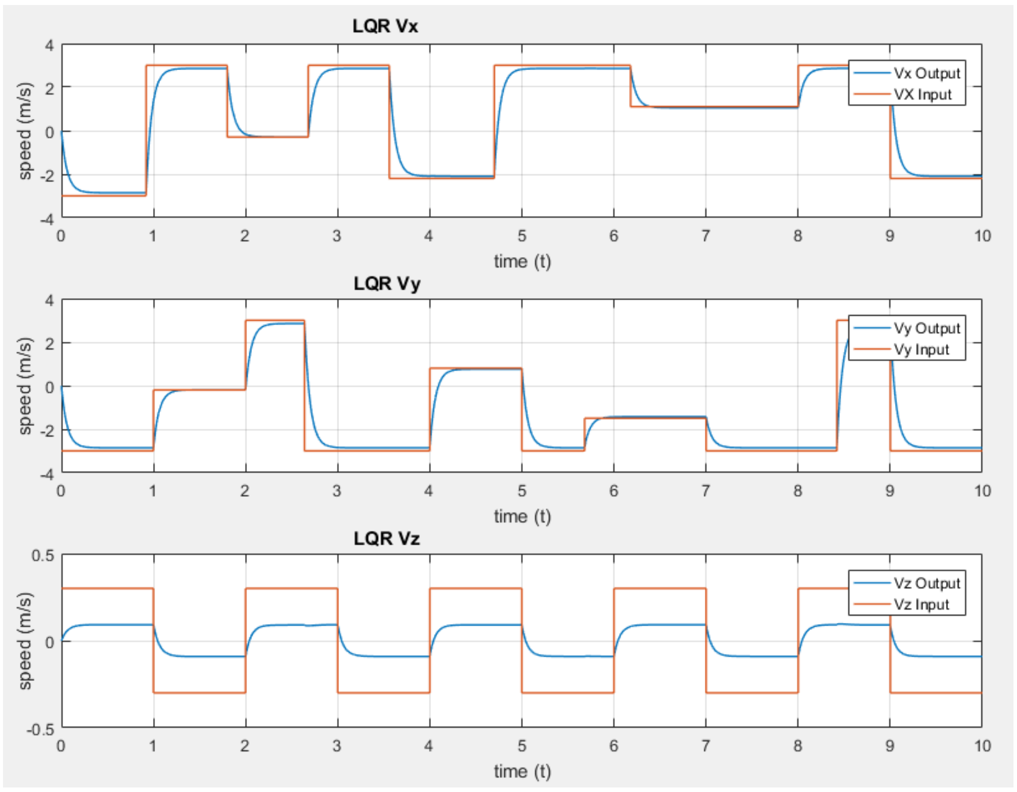


Figure 6.5: LQR control optimum Q and R values

According to simulation results (Figure 6.6) lqr response of each motor are shown as below. These input and outputs are selected optimum values of lqr.

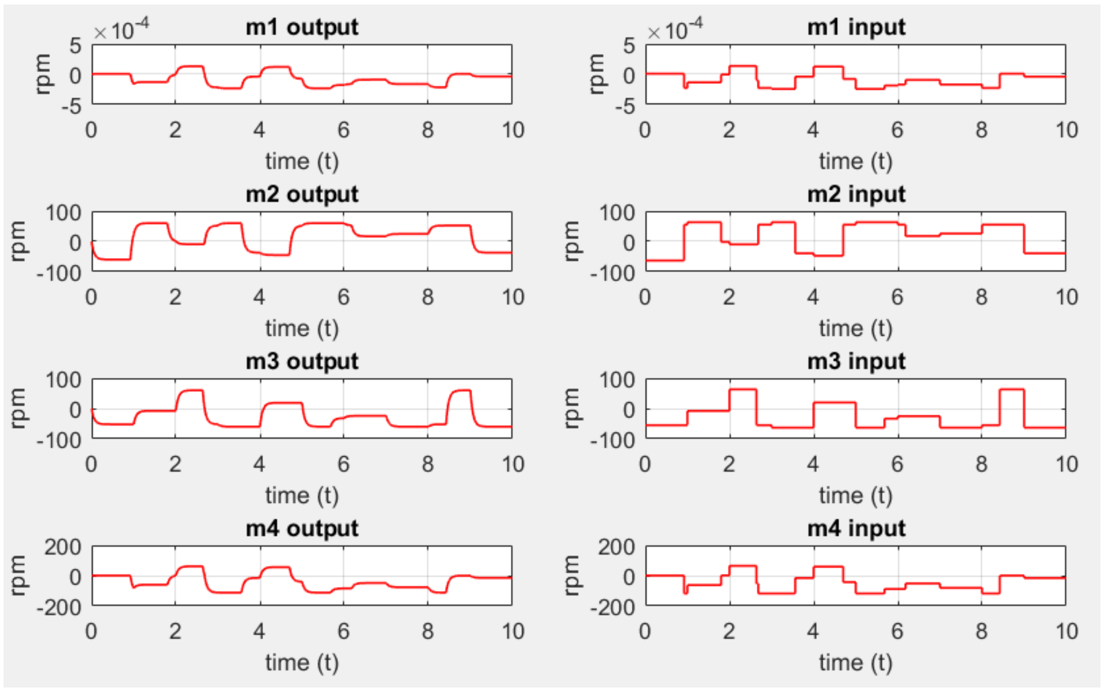


Figure 6.6: LQR motor inputs and outputs respect to optimum Q and R values

After selected optimum Q and R values system response time, overshoot and other criterias are enhanced. According to these values K state feedback matrixes for each motor are listed,

$$K1 = \begin{bmatrix} 26.8 & 4467.9 \end{bmatrix}$$

$$K2 = \begin{bmatrix} 12.1247 & 43.2373 \end{bmatrix}$$

$$K3 = \begin{bmatrix} 14.7702 & 350.3575 \end{bmatrix}$$

$$K4 = \begin{bmatrix} 57.5722 & 196.5196 \end{bmatrix}$$

6.2 Simulation of Model Predictive Control

MPC controller basically runs on all plant. All system block diagram (Figure 6.7) consist of mpc controller, motor model, reverse kinematics, forward kinematics and reference input signal.

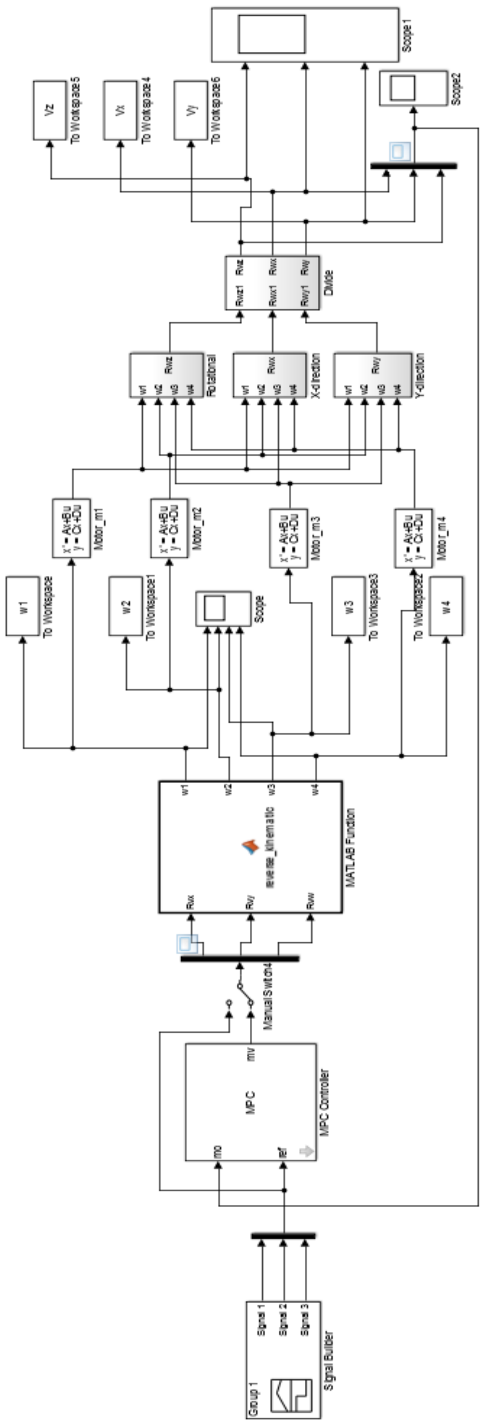
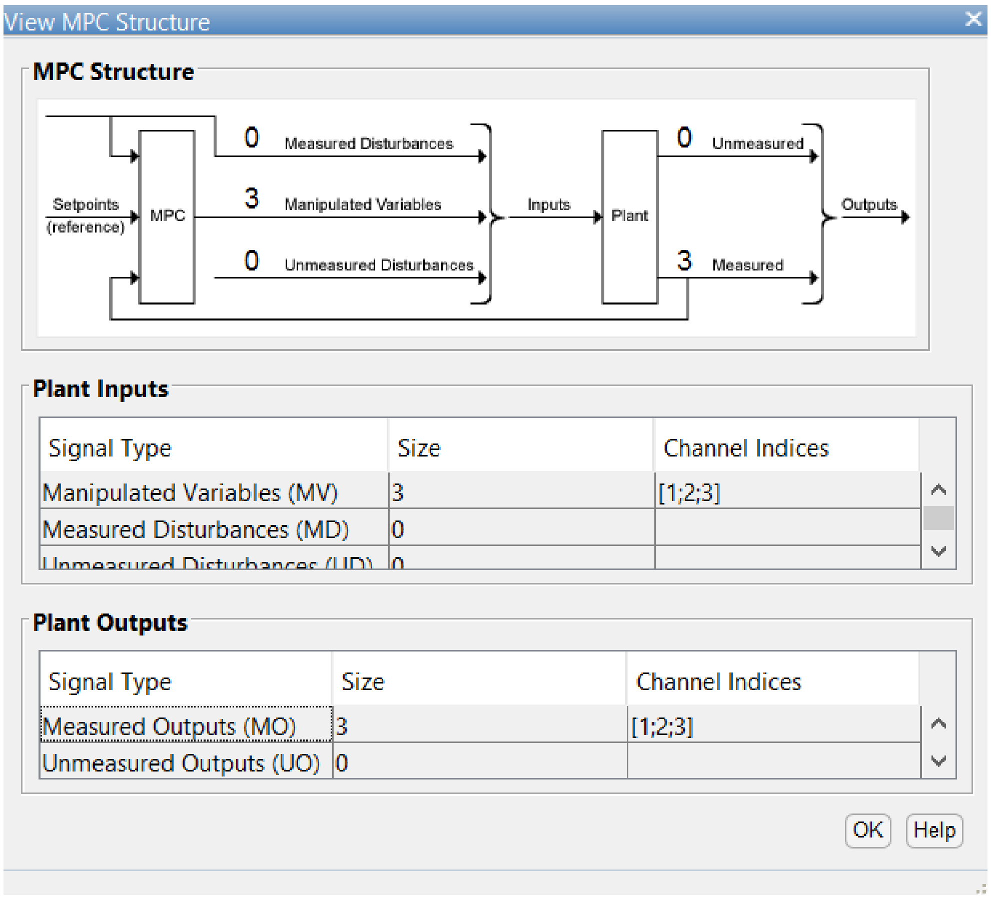


Figure 6.7: Simulation block diagram of MPC controlled system

Before start to define mpc controller, in simulink mpc controller toolbox (Figure 6.8) needs to a linerized plant model. At the first approach system was designed and identified with another identification toolboxes. And then plant model and MPC controller is implemented in simulink. With the help of this toolbox number of inputs and outputs defined before run simulation.



Plant Inputs

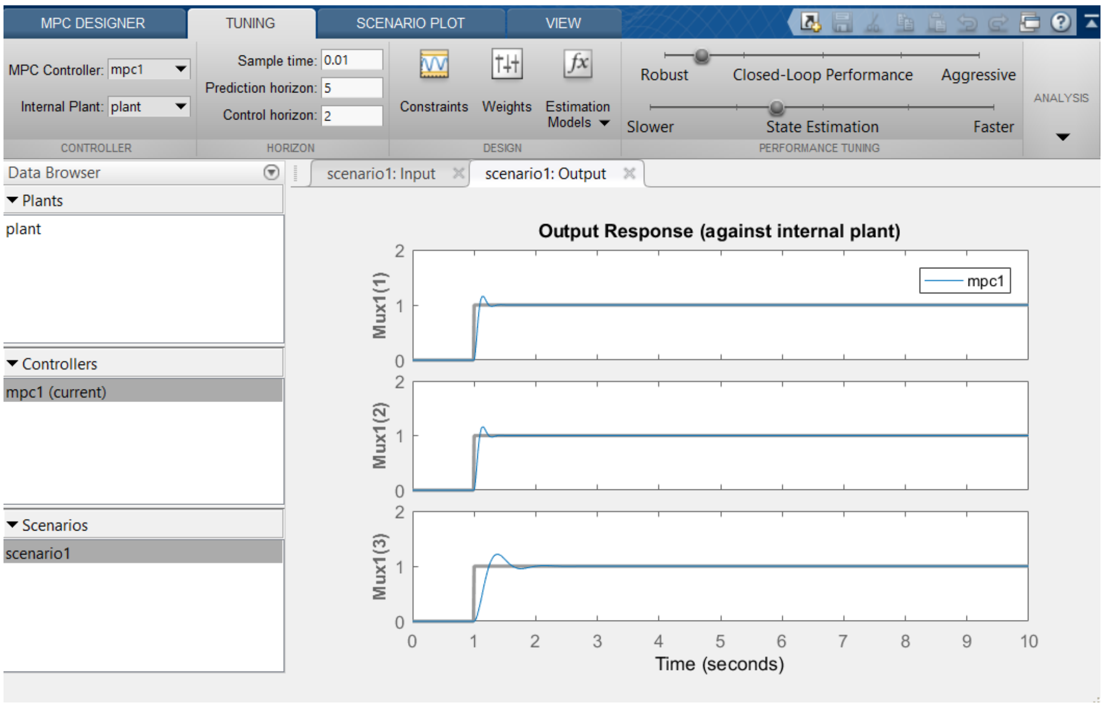
Signal Type	Size	Channel Indices
Manipulated Variables (MV)	3	[1;2;3]
Measured Disturbances (MD)	0	
Unmeasured Disturbances (UD)	0	

Plant Outputs

Signal Type	Size	Channel Indices
Measured Outputs (MO)	3	[1;2;3]
Unmeasured Outputs (UO)	0	

Figure 6.8: MPC controller structure in Simulink

After defining three inputs and three outputs system signals which are manipulated variables and measured outputs were defined. After defining these variables the other system constraints defined according to system limits and response time etc. In addition input and output specifications (Figure 6.9) were defined.


MPC Controller: mpc1

Internal Plant: plant

Sample time: 0.01

Prediction horizon: 5

Control horizon: 2

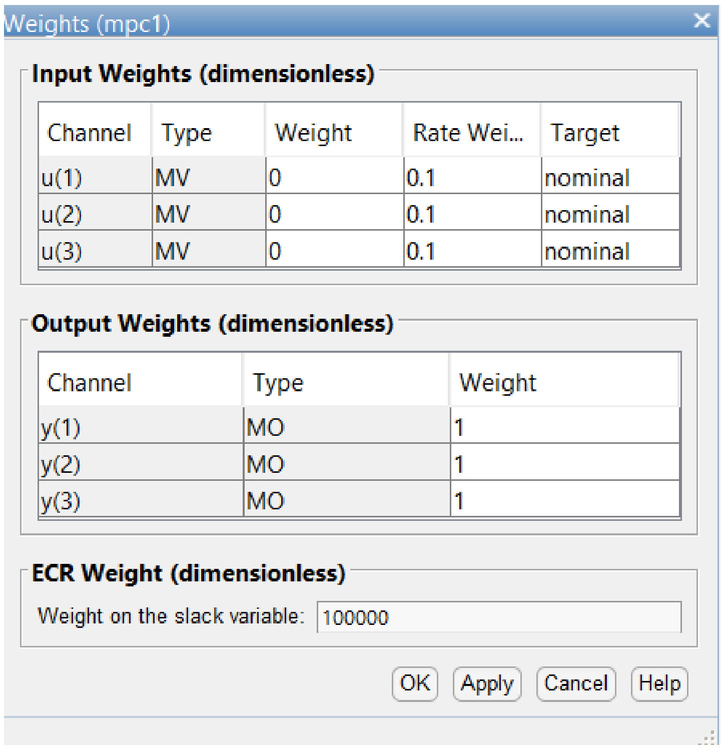
Figure 6.9: MPC controller tuning interface

Sample time, predicition horizon, control horizon are selected as controller performance. Prediction horizon determines the number of future samples to be predicted by plant model. Besides control horizon is used for how many control action to be calculated in the future. If control horozion increase control calculation increases too thus control effort increase.

In the constraints section (Figure 6.10), system inputs and outputs limits can be penelized by user. Also input rate limits (Table 6.1) can be selected as pre-defined priority constraints.

Factor constant	Priority
0.2	Low priority: Large tracking error acceptable
0.5	Below-average priority
1	Average priority – the default.
5	Above average priority
20	High priority: Small tracking error desired

Table 6.1: Weighting factor constant relationship


Weights (mpc1)

Input Weights (dimensionless)

Channel	Type	Weight	Rate Wei...	Target
u(1)	MV	0	0.1	nominal
u(2)	MV	0	0.1	nominal
u(3)	MV	0	0.1	nominal

Output Weights (dimensionless)

Channel	Type	Weight
y(1)	MO	1
y(2)	MO	1
y(3)	MO	1

ECR Weight (dimensionless)

Weight on the slack variable: 100000

Figure 6.10: MPC weights inputs and outputs

After defining all parameters of mpc controller simulation can be run as user inputs.

Prediciton horizion = a

Control horizon = b

Sample time = c

Weights input = d

All the simulation results are perfomed according to MPC variables. Such as prediction horizon, control horizon, sample time and weight input. As a result system outputs graphs as below in terms of velocity of x direction, velocity of y direction and velocity of z direction respectively.

As seen from simulation results (Figure 6.11) control horizon, prediction horizion, sample time and weight inputs are selected in the system limits. Response of all 3 direction catches reference value and response time quite low.

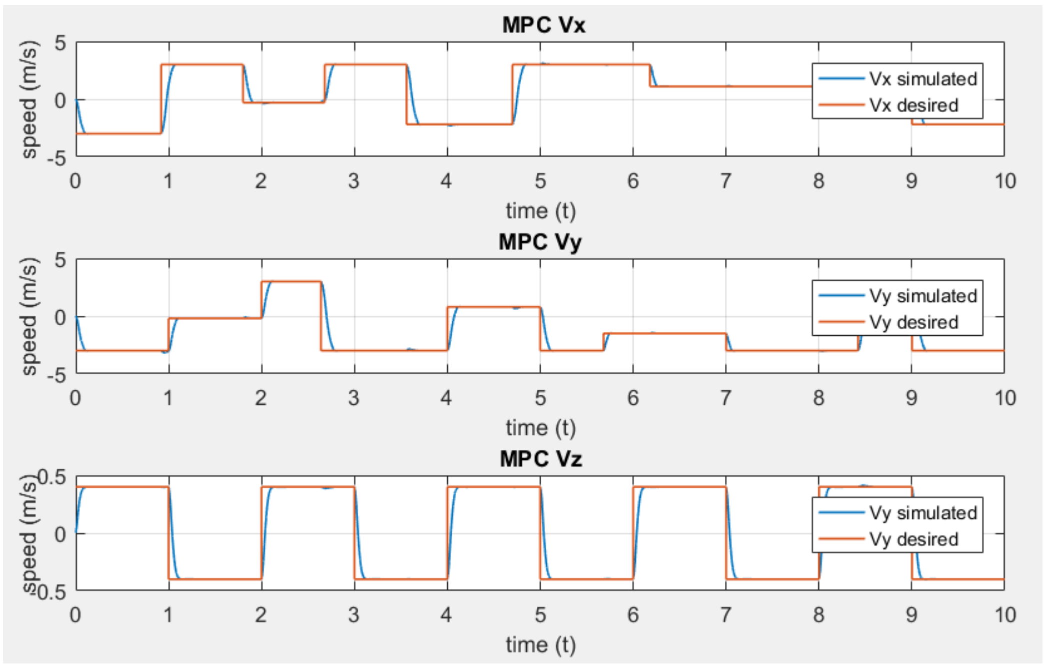


Figure 6.11: MPC control simulation results a=10, b=2, c=0.01, d=0.1

According to response (Figure 6.12), system sample time was changed to reduce accuracy of system. As shown response time was rised perceptibly. Other variables are remained constant.

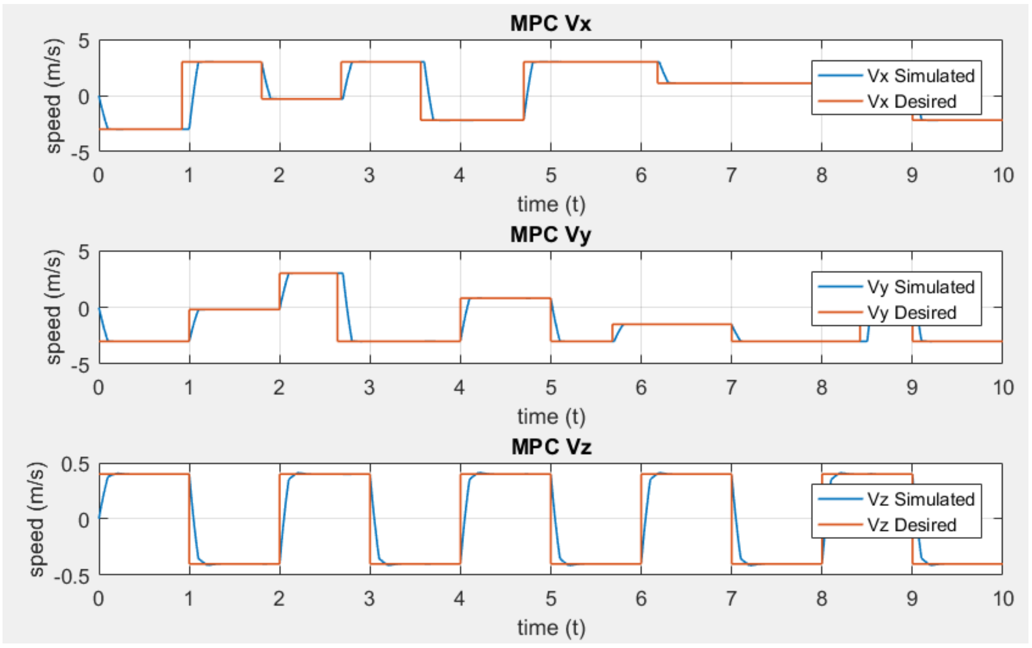


Figure 6.12: MPC control simulation results a=10, b=2, c=0.1, d=0.1

In simulation results (Figure 6.13) control horizon and prediction horizon were reduced due to see how effects these variables on system behavior. As is seen in figure responses can catch desired values, however response time was increased and behavior of system became more unassertive.

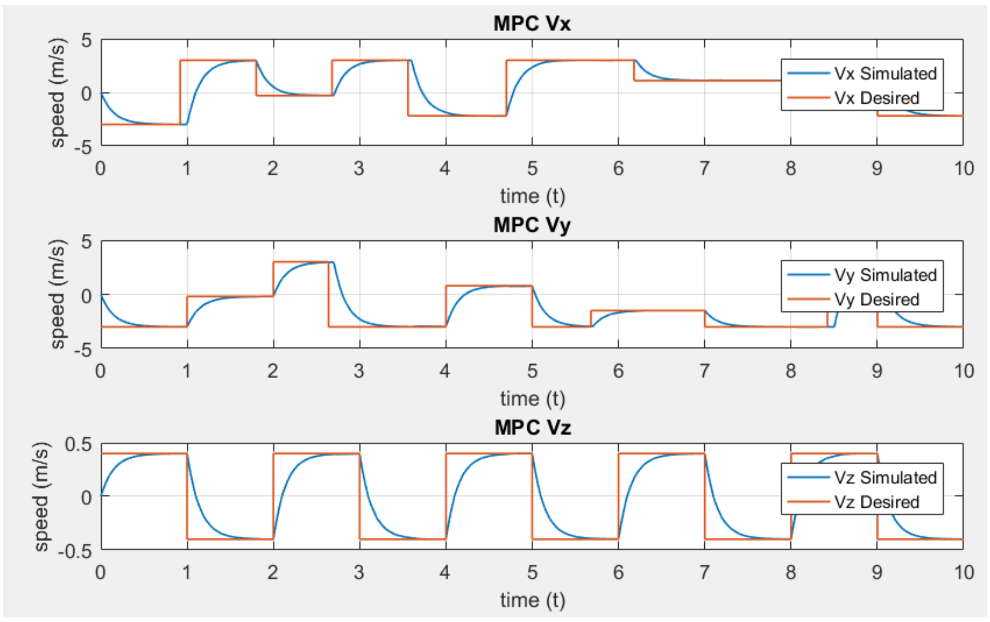


Figure 6.13: MPC control simulation results a=6, b=1, c=0.1, d=0.1

In Figure 6.14 control horizon are increased in comparison with Figure 6.13. When we compare of these two figures it can be seen clearly increasing control horizon decreases response time and bring out to system behave more aggressive.

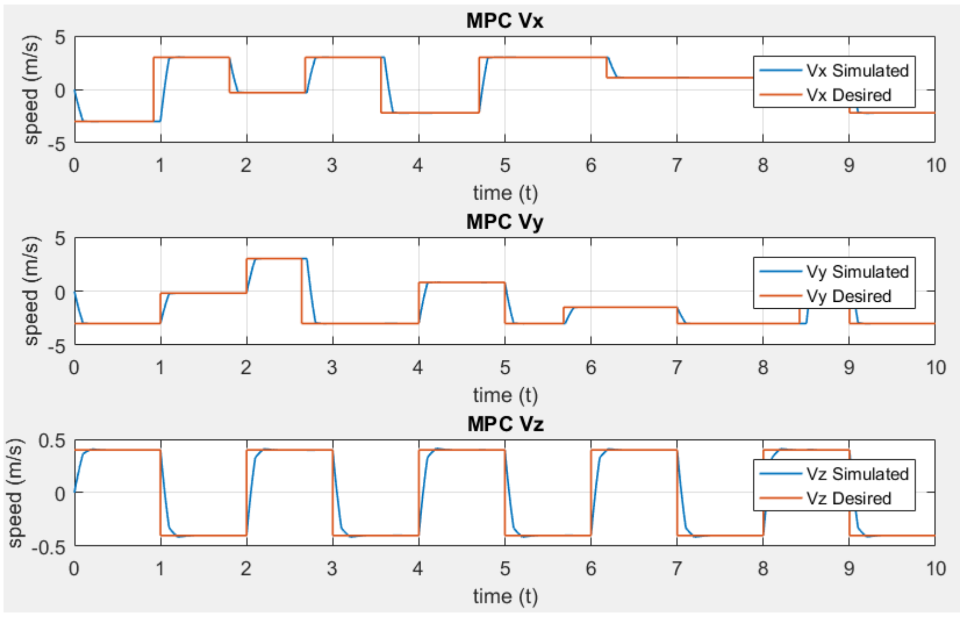


Figure 6.14: MPC control simulation results a=6, b=5, c=0.1, d=0.1

As one can see from Figure 6.15 input weights are increased. It effects system behave more unassertive and put down reaction time.

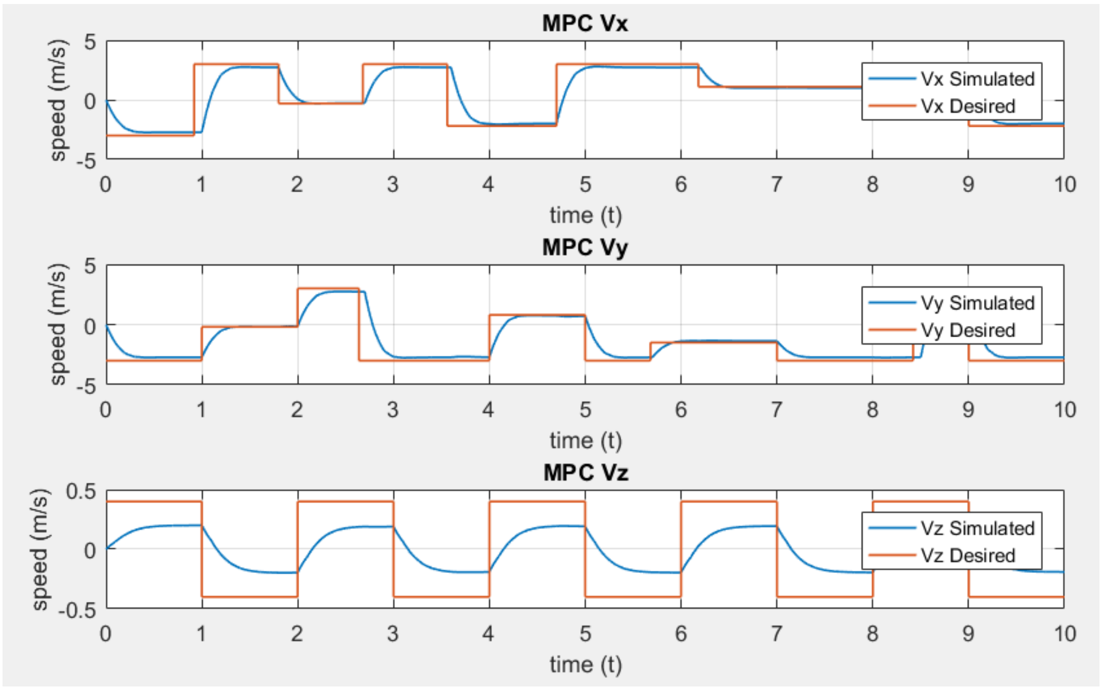


Figure 6.15: MPC control simulation results a=6, b=5, c=0.1, d=1

As one can see from Figure 6.16 input weights are more increased in comparison with Figure 6.15. It effects system behave more unassertive and can not catch desired reference value.

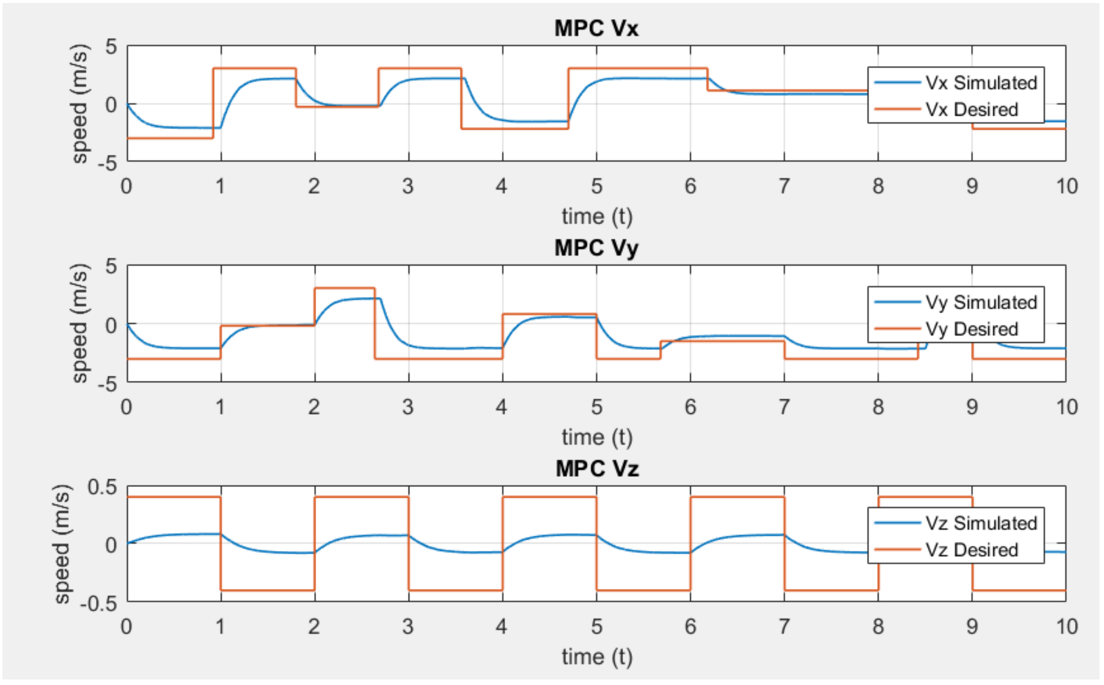


Figure 6.16: MPC control simulation results a=6, b=5, c=0.1, d=2

Defining all optimum parameters for MPC controller control inputs and outputs regulate by mpc controller as user input. Controller signals in three direction are shown (Figure 6.17). if MPC controller knows the path before, it creates a controller signal. In other words MPC send own regulated signals before waiting system reference input come.

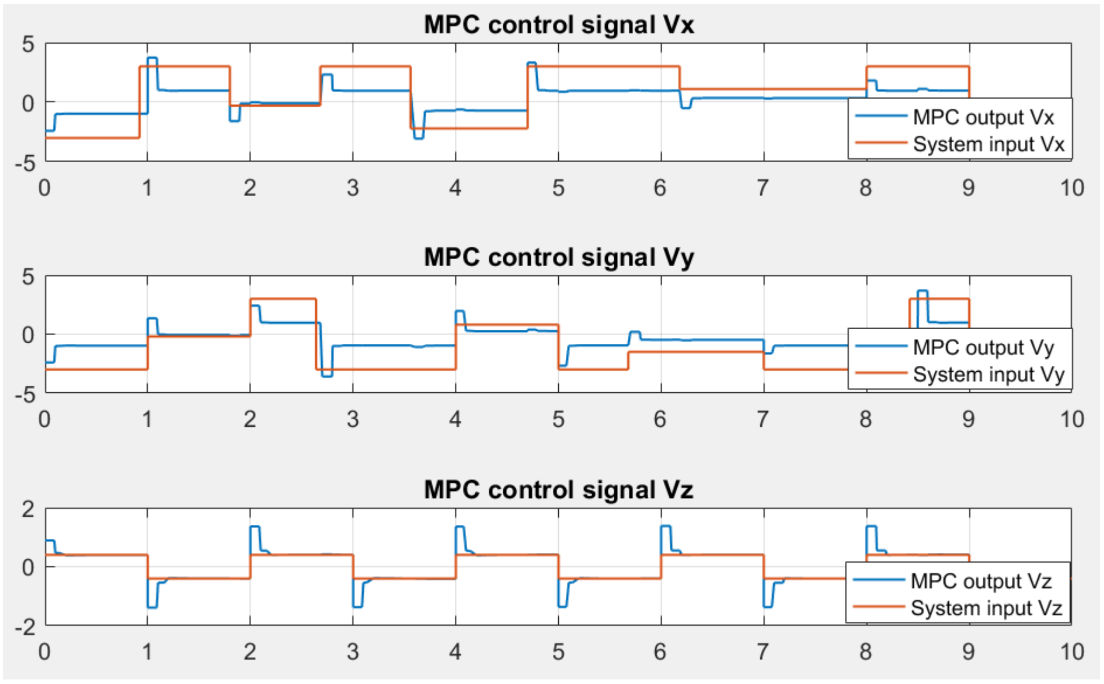


Figure 6.17: MPC control signal comparing with user input

6.3 Comparison of MPC and LQR

Simulation in different MPC values showed that how can be effected. Also in LQR different Q and R matrixes are proved these values should be selected precisely. After taking into consideration for optimum values both LQR and MPC controller, which are compared. All results are implemented in one figure as below (Figure 6.18). Control horizon is selected as 5, prediction horizon is selected as 2, sample time is selected as 0.05 and input weight is selected as 1. For the LQR controller, Q matrix and R matrix are selected at low values. As below MPC controller give best response then designed LQR controller (Table 6.2).

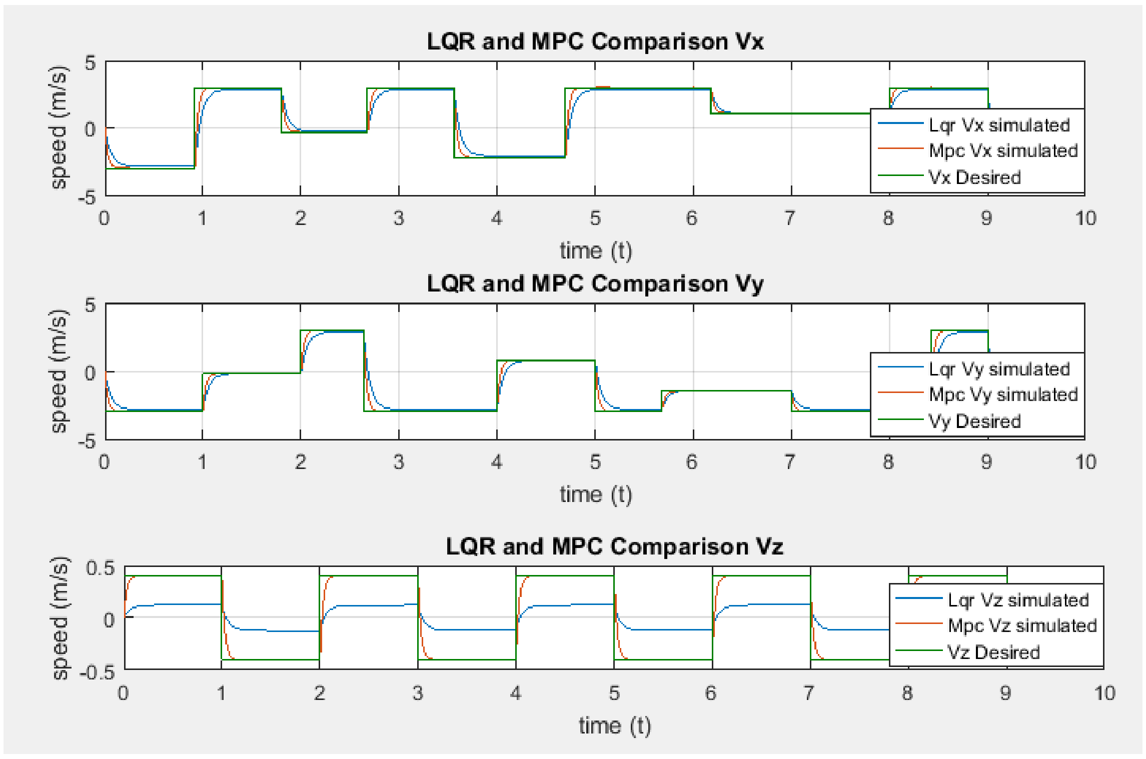


Figure 6.18: LQR and MPC comparison a = 5, b = 2, c=0.05, d =1

	Vx	Vy	Vz
MPC Control Effort	6.3964e+03	3.2351e+03	711.2136
LQR Control Effort	2.0417e+04	2.0348e+04	117.7671
MPC Error	1.10773	1.14637	1.86373
LQR Error	2.28915	2.65215	6.27938
MPC Response Time	0.2835	0.1725	0.4143
LQR Response Time	0.2835	0.4298	0.4784

Table 6.2: Comparison of control effort, response time, error for MPC and LQR

7. CONCLUSION AND RECOMMENDATIONS

In this study mecanum wheed mobile robot designed and for the speed control of system LQR and MPC are used. These two types of controller performance are compared. For two of them system limits are applied to take best results. In this situation LQR system response changes according to Q and R matrix. And for this type of controller state feedback is needed to observe actual speed from system. Because LQR based on state feedback controller. The other type of controller which is MPC needs system input and output. With the help of MPC toolbox system, behaviour can be adjusted according to user demand. The most important about this are sample time, weight ratios, control horizon and prediction horizon and they should be select in system limits. If the system limits well known these variables can be selected at optimum values. Besides generally MPC response time and overshoot are less then LQR.

As seen from LQR simulation graphs Q and R values are very sensetive on system response. Especially system output matrix which is Q, states should be chosen at small values. Increasing state values in Q matrix heads system towards unstable bands. Increasing control amplitute R matrix, it provides more free to act of the control signal. In other words, the larger these values are, the more penalized signals. Basically, choosing a large value for R means trying to stabilize system with less energy. This usually called expensive control strategy. On the other hand choosing small value for R means, called cheap control strategy because control input is not penalized. Similarly large values of Q means trying to stabilize system with least possible changes in the states and large Q implies less concern about the changes in states.

As seen from MPC simulation graphs, they show that how these variables effects system output. For instance decreasing prediction horizon results in more aggressive control action, because of system tries to reach trajectory point quickly. In addition increasing control horizon makes the system more aggressive and increases computational effort. Increasing the values of weights tend to make the MPC controller more effective by reducing the magnitude of input moves. And typically increasing sample time results to system consume more energy. For getting best results from

simulation input and output values are bounded via help of toolbox. Every selection of these values should be considered taken into account processor capabilities.
As it seen from table which is in previous chapter shows that the difference between MPC and LQR performance on whole system. In three reference system, it is clearly seen control effort of MPC is less than LQR. Beside this response time of MPC is less than LQR. The main aim is to reduce error in system and in comparison figure illustrates error in the MPC is less than LQR. These three comparison criterias show MPC is better than LQR.
Also it is important to consider MPC algorithms in toolboxes. Some sort of mpc methods in toolboxes such as DMC, MAC, GPC etc. Their effects can be seen by implementing these algorithms to the mpc solver. DMC uses step response model for open loop BIBO stable process. It is robust however performance is poor for like disturbances and multivariable systems. MAC is easy to implement and better for multivariable systems. It uses impulse response model. It limits it is use for slow referance trajectory. GPC provides offset free response. It has wide application area compared wih other approches. It tracks both varying and constant future set points. GPC is most easy and suitable and have good scope to improve its performance and increase to use of it.
Model predictive contol is the most widely used controller in industy applications, besides due to robustness of theoritical bases are used in academic researches. The popularity comes from the behaviour of the system can be optimized by less variables. The dominant reseach is actually use of model predictive control, which copes with multi input multi output systems. As it shown in figures previous chaptes, MPC gave best controller output in terms of response time and control effort. Model predictive control requires quite less time than calibrating and implementing other control strategies. Moreover its accuracy better than others.

REFERENCES

[1] **Siegwart,F. and Nourbakhsh,I.R.** (2004). Introduction to Autonomous Mobile Robots

[2] **Adascalitei,F., Doroftei,I.** (2011). Practical Applications for Mobile Robots based on Mecanum Wheels a Systematic Survey, MECAHITECH'11, vol. 3

[3] **Yu,H., Dubowsky,S., Skwersky,A.** (2004). Omni-Directional Mobility Using Active Split Offset Castors, Journal of Mechanical design vol 126 issue 5, doi:10.1115/1.1767181

[4] **Rojas,R.** A short history of omnidirectional wheels

[5]**Chang,P., Hebert,M.** (2000) Omni-directional Structure From Motion, Conference of Omnidirectional Vision, DOI: 10.1109/OMNVIS.2000.853819

[6] **Soni,S., Mistry,T, Hanath,J.,** (2014). Experimental Analysis of Mecanum wheel and Omni wheel, International Journal of Innovative Science engineering and Technology, vol 1 Issue 3

[7**] Zaccarian,L.,** DC motors: dynamic model control techniques

[8]**Wakchaure,K.N.,** Bhaskar,S.V., Thakur,A.G., Modak,G.S., Kinematics modelling of mecanum Wheeled mobile platform

[9] **Muir,P.F. and NeumanCharles P.** (1987). Kinematic Modelling for Feedback Control of an Omnidirectional Wheeled Mobile Robot, CH2413-3/87/0000/1772

[10] **Raol,J.R., Girija,G., Singh,J.,** (2004). Modelling and Parameter Estimation of Dynamic Systems

[11] **Prasad,L.B., Tyagi,B., Gupta,H.O.,** (2014). Optimal Control of Nonlinear Inverted Pendulum System Using PID Controller and LQR: Performance Analysis Without and With Disturbance Input, International Journal of Automation and Computing, DOI: 10.1007/s11633-014-0818-1

[12**] Qin,S.J., Badgwell,T.A.,** (2002). A survey of industrial model predictive control technology

[13] **Holkar,K.S., Waghmare,L.M.,** (2010). An Overview of Model Predictive Control, International Journal of Control and Automation Vol 3 No 4

[14] **Murray,M.R.,** (2010). Optimization-Based Control

APPENDIX

APPENDIX A : Matlab m file codes
APPENDIX B : Python2x codes

APPENDIX A

MATLAB Reverse Function

```
function [w1,w2,w3,w4] = reverse_kinematic(Rvx,Rvy,Rvw)
Direct=[Rvx;Rvy;Rvw];
%syms W1 W2 W3 W4;

 W1=1;      %
 W2=1;      %
 W3=1;      % define the main diagonal elements of weigted matrix W
 W4=1;      %

a=0.30;        % half distance left-right side by side (meter)
b=0.225;      % half distance between the center of M2&M4 (meter)
R=0.1016;   % Radius of mecanum wheel (meter)

J=[-R/4 R/4 R/4 -R/4;R/4 R/4 R/4 R/4;R/(4*(a+b)) -R/(4*(a+b)) R/(4*(a+b)) -
R/(4*(a+b))]; % Jacobian Matrix

Wd=[W1,W2,W3,W4];  % Diagonal elements of weighted matrix W
W=diag(Wd);                % Weighted matrix W

J_pseudo=inv(W)*J.'*inv(J*inv(W)*J.');
wa=J_pseudo*Direct;

w1=wa(1);
w2=wa(2);
w3=wa(3);
w4=wa(4);
```

Matlab LQR state feedback matrix

```
TF1_num = [0 111 6113];
TF1_den = [1 383.3 1948];
[A1 B1 C1 D1] = tf2ss(TF1_num,TF1_den);
Q1 = C1'*C1;
R1 = 1;
[K1] = lqr(A1,B1,Q1,R1);

TF2_num = [0 17.27 59.55];
TF2_den = [1 9.785 19.39];
[A2 B2 C2 D2] = tf2ss(TF2_num,TF2_den);
Q2 = C2'*C2;
R2 = 1;
[K2] = lqr(A2,B2,Q2,R2);

TF3_num = [0 26.04 482.6];
TF3_den = [1 39.29 157.2];
[A3 B3 C3 D3] = tf2ss(TF3_num,TF3_den);
```

```
Q3 = C3'*C3;
R3 = 1;
[K3] = lqr(A3,B3,Q3,R3);

TF4_num = [0 317.1 1287];
TF4_den = [1 847.9 4116];
[A4 B4 C4 D4] = tf2ss(TF4_num,TF4_den);
Q4 = C4'*C4;
R4 = 1;
[K4] = lqr(A4,B4,Q4,R4);
```

Matlab set system parameters values

```
a=0.30;
b=0.225;
R=0.1016;
A=[25 25 25 25];
w1=A(1);w2=A(2);
w3=A(3);w4=A(4);

Ad=[5 0 0];
Rvw=Ad(3);
Rvx=Ad(1);
Rvy=Ad(2);
```

APPENDIX B

```
# -*- coding: cp1254 -*-
#--------------------PIN TABLE FOR UART--------------------#
# UART     RX      TX      CTS     RTS      DEVICE        #
#-------------------------------------------------------           #
# UART1    P9_26   P9_24   P9_20   P9_19   /dev/ttyO1  #
# UART2    P9_22   P9_21                              /dev/ttyO2 #
# UART3            P9_42   P8_36   P8_34        /dev/ttyO3 #
# UART4    P9_11   P9_13   P8_35   P8_33   /dev/ttyO4  #
# UART5    P8_38   P8_37   P8_31   P8_32   /dev/ttyO5  #
#-------------------------------------------------------#
#*******************************************************#
#---12bits(0~4095) 0-18V Use P9_32 VDD_ADC P9_34 GNDA_ADC--#
#-------------------------ADC-PINS--------------------------------------------#
#------AIN0---AIN1---AIN2---AIN3---AIN4---AIN5---AIN6---------------#
#------P9_39--P9_40--P9_37--P9_38--P9_33--P9_36--P9_35---------------#
#-------------------------------------------------------------------------------#

#Library functions
import roboclaw
import random
import serial
import struct
import time
import math
import sys

import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.PWM as PWM
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.UART as UART

addressF = 0x80
addressB = 0x81

UART_1 = "/dev/ttyO1"
UART_2 = "/dev/ttyO2"

_trystimeout = 2

#UART.setup("UART4")
#UART.setup("UART2")

#desired speed, angle, z rotation
#VD = ThetaD = VZTheta = ThetaD45 = 0

#speeds of each motor ttyO1
WheelSpeeds = [0, 0, 0, 0, 0]
```

```
#bluetooth data container
#responseBt = "    "

#Command Enums

def Open(comport, rate):
        global port
        port = serial.Serial(comport, baudrate=rate, timeout=0.01,
interCharTimeout=0.01)
        return

def displayspeed():
        enc1 = ReadEncM1(addressF)
        #enc2 = ReadEncM2(addressF)
        speed1 = ReadSpeedM1(addressF)
        #speed2 = ReadSpeedM2(addressF)

        print "Encoder1:"
        print enc1
        if(enc1[0]==1):
                print enc1[1]
                print format(enc1[2],'02x')
        else:
                print "failed"
        print "Encoder2:"
        if(enc2[0]==1):
                print enc2[1]
                print format(enc2[2],'02x')
        else:
                print "failed "

        print "Speed1:"

        if(speed1[0]):
                print speed1[1]
        else:
                print "failed"
        print "Speed2:"

        if(speed2[0]):
                print speed2[1]
        else:
                print "failed "

        #file.write("Speed1 = ")
        #file.write(speed1[1])
        #file.write("\n")
     #file.write("Speed2 = ")
        #file.write(speed2[1])
```

```
#mapping function

def normalize(x, in_min, in_max, out_min, out_max):
   if x == 0:
      return num_const.STOPVALUE
   return int(math.floor((x - in_min) * (out_max - out_min) / (in_max - in_min) +
out_min))


#mecanumdrive(VD, ThetaD, VZTheta)
#VD = 0~10 ThetaD = 0~10 VZTheta 0~10

def drive_func(headingR, magnitudeR, headingL, magnitudeL):
         if (0 <= headingL <= 18):
         magnitudeL = magnitudeL
      elif (18 < headingL <= 35):
            magnitudeL = -magnitudeL

      WheelSpeeds[1] = (-math.sin(math.radians(headingR * 10)) +
math.cos(math.radians(headingR * 10))) * magnitudeR - magnitudeL
         WheelSpeeds[2] = (math.sin(math.radians(headingR * 10)) +
math.cos(math.radians(headingR * 10))) * magnitudeR + magnitudeL
      WheelSpeeds[3] = (math.sin(math.radians(headingR * 10)) +
math.cos(math.radians(headingR * 10))) * magnitudeR - magnitudeL
      WheelSpeeds[4] = (-math.sin(math.radians(headingR * 10)) +
math.cos(math.radians(headingR * 10))) * magnitudeR + magnitudeL
         print WheelSpeeds[1],"   ", WheelSpeeds[2],"   ", WheelSpeeds[3],"   ",
int(WheelSpeeds[4]),"    ","  \n "
      for limit in range(1,5):
            if WheelSpeeds[limit] > (magnitudeR + magnitudeL)* 1.0:
               WheelSpeeds[limit] = (magnitudeR + magnitudeL)
            elif WheelSpeeds[limit] < -(magnitudeR + magnitudeL) * 1.0:
               WheelSpeeds[limit] = -(magnitudeR + magnitudeL)
            elif WheelSpeeds[limit] < 0:
                  WheelSpeeds[limit] = int(WheelSpeeds[limit] - 0.01)
            elif WheelSpeeds[limit] > 0:
                  WheelSpeeds[limit] = int(WheelSpeeds[limit] + 0.01)
            else:
                  WheelSpeeds[limit] = int(WheelSpeeds[limit])

         if magnitudeR == 0 or magnitudeL == 0:
            limiter = 10
         else:
            limiter = 20
         WheelSpeeds[1] = normalize(WheelSpeeds[1], -limiter, limiter, 0, 127)
      WheelSpeeds[2] = normalize(WheelSpeeds[2], -limiter, limiter, 0, 127)
      WheelSpeeds[3] = normalize(WheelSpeeds[3], -limiter, limiter, 0, 127)
      WheelSpeeds[4] = normalize(WheelSpeeds[4], -limiter, limiter, 0, 127)
```

```
        ForwardBackwardM1(addressF, WheelSpeeds[1])
        ForwardBackwardM2(addressF, WheelSpeeds[2])
        ForwardBackwardM1(addressB, WheelSpeeds[3])
        ForwardBackwardM2(addressB, WheelSpeeds[4])


        if int(WheelSpeeds[1]) > 0:
            a1 = "ileri"
        elif int(WheelSpeeds[1]) < 0:
            a1 = "geri"
        elif int(WheelSpeeds[1]) == 0:
                a1 = "---"
        if int(WheelSpeeds[2]) > 0:
            a2 = "ileri"
        elif int(WheelSpeeds[2]) < 0:
            a2 = "geri"
        elif int(WheelSpeeds[2]) == 0:
                a2 = "---"
            if int(WheelSpeeds[3]) > 0:
            a3 = "ileri"
        elif int(WheelSpeeds[3]) < 0:
            a3 = "geri"
        elif int(WheelSpeeds[3]) == 0:
                a3 = "---"
            if int(WheelSpeeds[4]) > 0:
            a4 = "ileri"
        elif int(WheelSpeeds[4]) < 0:
            a4 = "geri"
        elif int(WheelSpeeds[4]) == 0:
                a4 = "---"

      # print "sol ön = ", a2, "   sağ ön = ", a1, "\n"
        print WheelSpeeds[2], "     ", WheelSpeeds[1], "\n"
       #print "sol arka = ", a4, "   sağ arka = ", a3, "\n"
        print WheelSpeeds[4], "     ", WheelSpeeds[3], "\n"

        print "--------------------------------------------------------------------------------\n"


#UART setups
UART.setup("UART4") #BBB TX -> P9_13,  BBB RX -> P9_11
BtJoystick = serial.Serial(port = "/dev/ttyO4", baudrate = 9600)
BtJoystick.close()
BtJoystick.open()
#-----------------------------------------------------------------
UART.setup("UART1") #BBB TX -> P9_24,  BBB RX -> P9_26
Open("/dev/ttyO1", 9600)


while True:
	#global ThetaD, VD, VZTheta, ThetaD45
	responseBT = ""
	responseBTList = [0, 0, 0, 0, 0]

	#data format: [radiusL : angleL = rotation : radiusR = VD : angleR = direction]
	#data range : [radiusL = (0~10), angleL = (0~35), radiusR = (0~10), angleR =
(0~35)]
	if BtJoystick. isOpen():
		responseBT = BtJoystick.read(4)
		for character in range(0,4):
			responseBTList[character + 1] =  ord(responseBT[character])

	ThetaD = headingR = (35 - responseBTList[4]) % 35
	VD = magnitudeR = responseBTList[3]
	headingL = (35 - responseBTList[2]) % 35
	magnitudeL = responseBTList[1]


	if port.isOpen():
	    print "port open"
	    drive_func(headingR, magnitudeR, headingL, magnitudeL)
	    print displayspeed()
	    print "heading L = ", headingL, "magnitude L = ", magnitudeL, "heading R =
", headingR, "magnitude R = ", magnitudeR, "\n"
```

CURRICULUM VITAE

Name Surname	**:** Doğukan Taha TAYFUR
Place and Date of Birth	**:** Çankaya / 1989
E-Mail	**:** dttayfur@hotmail.com
EDUCATION	**:**
High School	**:** Kuleli Military High School, 2007
B.Sc.	**:** Sakarya University, Mechanical Engineering, 2012

PROFESSIONAL EXPERIENCE AND REWARDS:

11.2014 - R&D Engineer Mechatronic, Mercedes-Benz Türk A.Ş., Istanbul

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Baykar A.Ö., Tayfur D.D., Kural A. *Minimizing heading error of omnidirectional wheelchair based on mecanum wheels by applying control algorithm* International Conference on Engineering and Natural Science, ISBN: 978-605-83575-1-8, May 24-28 , 2016 Sarajevo, Bosnia and Herzegovina