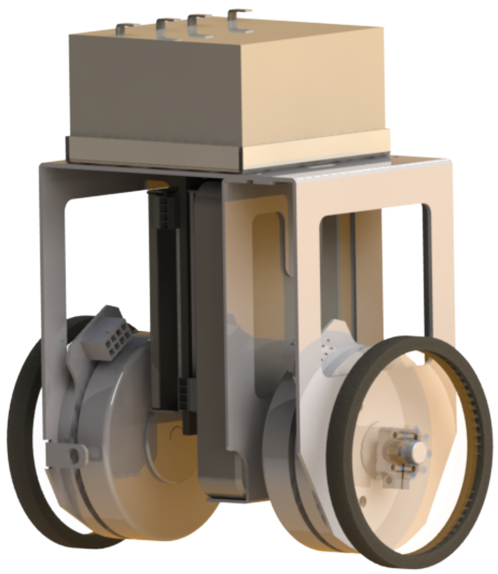
Bachelor Assignment Report
Faculty of Engineering Technology (CTW)
Mechanical Automation and Mechatronics

Realization of Mini Segway Robot Using NI MyRIO



AUTHOR:
Y. X. Mak

SUPERVISORS:
Dr.ir. R.G.K.M. Aarts
Dr.ir. J. van Dijk

EXTERNAL EXAMINER:
Dr.ir. H. Wormeester

February 2015

UNIVERSITEIT TWENTE.

Abstract

This bachelor assignment report presents a realization of a two-wheeled segway robot, controlled using velocity reference inputs via a joystick. The realization involves making a mathematical model and simulation of the system, designing the feedback control loop, designing the physical system mechanically and electrically, and finally implementing the controller to a National Instruments myRIO hardware platform.

The controller used in this realization is based on full state-feedback controller, and the Linear Quadratic Regulator (LQR) is used to tune to control gains. An algorithm is created to calculate tilt angle with respect to gravity vector using the accelerometer sensor outputs from the myRIO. The controller is implemented using LabView myRIO 2013 SP1 development environment. The realization of the two-wheeled robot is successful, however further testing and dynamics analysis are needed in order to control the robot optimally.

CONTENTS

CHAPTER 1

INTRODUCTION AND OVERVIEW

The National Instruments myRIO-1900 is a portable reconfigurable I/O (RIO) device that can be used to implement control for mechatronics systems. The myRIO platform is targeted at students and academic environments who are interested in highly portable IO device to be used in academic projects.

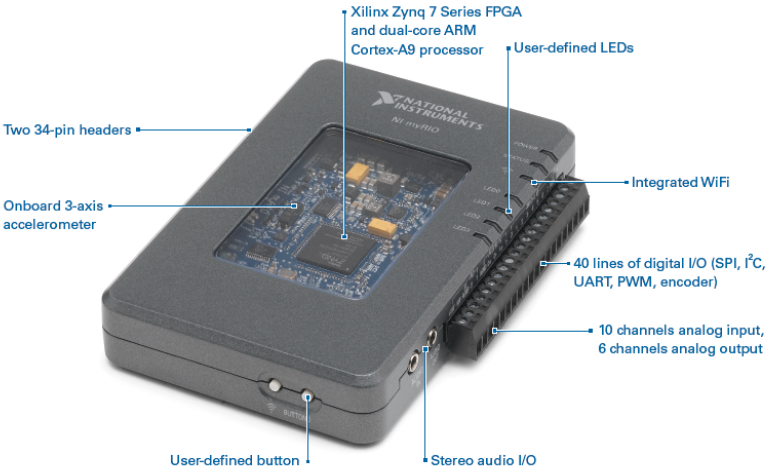


Figure 1.1: NI myRIO-1900

Since NI myRIO was launched at the end of 2013, the platform is relatively new among academic researchers and has not been extensively used for research purposes. Therefore, one aim of this bachelor project is to discover capabilities of the myRIO as hardware platform and LabView with LabviewRT as development environment.

A two-wheeled robot that has the center of mass above its axle, is unstable without any active control mechanism. Building such a system, will give sufficient challenges in control, electrical, and mechanical field to thoroughly assess the ease of use and capabilities of myRIO as a hardware platform. From the scope of this project, myRIO is an excellent choice as hardware platform because of its compact and light characteristics, while still possessing wide variety of functions (such as accelerometer sensors and Wi-Fi).

1.1 Project Framework

The goal of this bachelor project is to set up a feedback controlled small two-wheeled robot using NI myRIO hardware platform as a real-time controller. Apart from control, designing the electrical and mechanical system for the mini segway robot is also part of the assignment. The end product of the project is to have a battery-powered system with velocity reference input that can move around being controlled wirelessly (via Wi-Fi), since myRIO hardware has capabilities to do this (using internal Wi-Fi module and embedded nonvolatile memory [1]).

Realization of mini segway system involves:

1. Modeling the physical system: mathematical model of 3D inverted pendulum with 2 wheels is

needed to create simulation in LabView.

2. Designing the feedback control loop that is suited for the mini-segway model.
3. Making simulation of the physical system in LabView based on the mathematical model.
4. Building the controller in LabView which will be deployed to myRIO.
5. Design and build a battery powered mini-segway robot (mechanically and electrically).
6. Testing controller and implementations with the prototype physical system.
7. Re-iterate the steps in order to tune the controller for the mini-segway robot.

In the following Chapter 2, the background theory used in modeling and controller tuning is briefly explained. In Chapter 3, modeling, simulation, and all plant-side modification will be thoroughly explained, while in Chapter 4, controller-side design and implementation will be discussed. The physical realization and also electrical implementation, together with the test results will be discussed in Chapter 5. Finally, Chapter 6 contains conclusions, remarks, and recommendations for future projects.

CHAPTER 2

THEORY

This chapter introduces some background theory of Hamel's equation, which is used to derive the equation of motion of the 3D inverted pendulum, and the Linear Quadratic Regulator (LQR) tuning method which is used extensively to tune the control gain.

2.1 Hamel's Equation

Using Hamel's equation to derive the model of a segway has several advantages over other approaches, see [2, 3]. The main advantage is reducing the complexity and number of terms involved in the derivation while still preserving the non-linearities in the model. Later, we recognize that these non-linear terms appear from the coupling between forward tilt motion and steering rotation of the robot.

The difficulties of deriving the equation of motion (EoM) for a segway lies on the nonholonomic constraints that are apparent in the motion of the rolling disk wheel. Hamel's equation enables us to appoint some quasi-velocities ω_i at some 'arbitrary' points which tackles this nonholonomic constraints problem elegantly [4].

Hamel's equation of motion [5] can be expressed as:

$$\frac{d}{dt}\frac{\partial T}{\partial \omega_k} - \sum_{j=1}^{n}\left(\frac{\partial T}{\partial q_j} - \frac{\partial U}{\partial q_j}\right)\frac{\partial \dot{q}_j(\omega)}{\partial \omega_k} + \sum_{j=1}^{n}\sum_{l=p+1}^{n}\frac{\partial T}{\partial \omega_j}\gamma_{l,k}^{j}(q)\omega_l = f_k, \tag{2.1}$$

where Hamel-coefficients $\gamma_{l,k}^{j}$ are given by:

$$\gamma_{l,k}^{j} = \sum_{\nu,\sigma=1}^{n}\left(\frac{\partial^2 \omega_j}{\partial \dot{q}_\nu \partial q_\sigma} - \frac{\partial^2 \omega_j}{\partial \dot{q}_\sigma \partial q_\nu}\right)\frac{\partial \dot{q}_\sigma}{\partial \omega_l}\frac{\partial \dot{q}_\nu}{\partial \omega_k} \tag{2.2}$$

Hamel's Equation (2.1), by its structure, is very similar to Langrange-Euler equations of motion, see Neimark and Fufaev [6]. It differs in the differentiation with respect to velocities ω_k instead of time derivates of coordinates $\dot{q}_j$, and an additional term with the Hamel-coefficients. This third term in the equation accounts for the nonholonomic constraints.

2.2 Linear Quadratic Regulator (LQR)

Linear quadratic regulator (LQR) is a linear control system design technique which minimizes the LQR performance index:

$$J_{LQR} = \int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}\right) dt, \tag{2.3}$$

which basically minimizes the weighted states and weighted control effort of the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$
$$\mathbf{z} = \mathbf{C}_1\mathbf{x}.$$

Taking $\mathbf{Q} = \rho\mathbf{C}_1^T\mathbf{C}_1$ and $\mathbf{R} = 1$ [7, p.493], Equation (2.3) reduces to the more simple form in which the role of weighing parameter ρ can be directly seen.

$$J_{LQR} = \int_0^\infty \left(\rho \mathbf{z}^T \mathbf{z} + \mathbf{u}^T \mathbf{u}\right) dt. \tag{2.4}$$

Note that the matrix $\mathbf{C}_1$ does not need to be the same as the plant output matrix. Therefore it can be adjusted to express motion error in the physical system that needs to be kept small, for example, the error in forward velocity can be weighted with regard to the error in tilt angle of the segway.

The control law that minimizes J_{LQR} is given by the linear state-feedback

$$u = -\mathbf{K}\mathbf{x}. \tag{2.5}$$

If we take G_0 as the open-loop transfer function from u to z

$$\mathbf{G_0} = \mathbf{C}_1 \left(s\mathbf{I} - \mathbf{A}\right)^{-1} \mathbf{B}, \tag{2.6}$$

then the LQR performance index becomes

$$J_{LQR} = \int_0^\infty \left(\rho \mathbf{u}^T \mathbf{G_0}^T \mathbf{G_0} \mathbf{u} + \mathbf{u}^T \mathbf{u}\right) dt,$$

which is minimal for

$$\rho G_0(-s) G_0(s) + 1 = 0, \tag{2.7}$$

Note that this is a root locus problem with respect to the parameter ρ, with the resulting root locus is symmetrical with respect to the imaginary axis, hence called the *symmetric root locus (SRL)* [8]. Therefore the optimal value of $\mathbf{K}$ is such that it places the closed-loop poles at the stable roots of the *symmetric root locus* equation (2.7).

To summarize, steps to tune the controller gain $\mathbf{K}$ using the LQR minimization method is as follows:

1. Choose matrix $\mathbf{C}_1$. This defines the weight of each state variable in the LQR minimization.
2. Choose the value of parameter ρ. The scalar parameter ρ weights the importance of the tracking error $\mathbf{z}^T\mathbf{z}$ to the control effort $\mathbf{u}^T\mathbf{u}$.
3. Calculate the transfer $\mathbf{G_0}$, and plot the corresponding SRL diagram for each transfer $G_0(s)$. The location of closed-loop poles of corresponding ρ in the SRL then can be fed into pole placement algorithm to get gain $\mathbf{K}$. Or using more straightforward way, MATLAB function `lqr` can be used, while giving less visual representation the relation between ρ and location of the poles.

CHAPTER 3

MODELING AND SIMULATION USING LABVIEW

Inverted pendulum problem is a well-known example in the field of control engineering. But in order to apply it to a real physical system, a mathematical model of inverted pendulum in 3D space is needed.

3.1 Deriving Equation of Motion and State-space Model

Consider a two-wheeled robot with two independently driven motors rolling on a horizontal plane, illustrated in Figure 3.1. The body of the robot and the wheels are assumed rigid. The angles θ and α denotes rotation of the robot's body around the vertical axis (steering angle) and axle (tilting angle), respectively. The wheel angles with respect to the robot's body are described by ϕ_1 and ϕ_2. The position of center of the axle can be expressed in global coordinates x and y. The wheel's radius and length of the axle is denoted by constants r and b respectively. The distance between axle and the center of gravity of the robot's body is denoted by l. The gravity acts on the vertical direction with gravity constant g. The principal moments of inertia of the robot's body with regard to its center of mass are J_x, J_y, and J_z, while the total mass of the main body is m_o. The motors have moving parts with moments of inertia J_w, while the mass of each wheel is given by m_w. To control the robot, torques τ_1 and τ_2 are applied between the robot's body and the wheels by each motor.

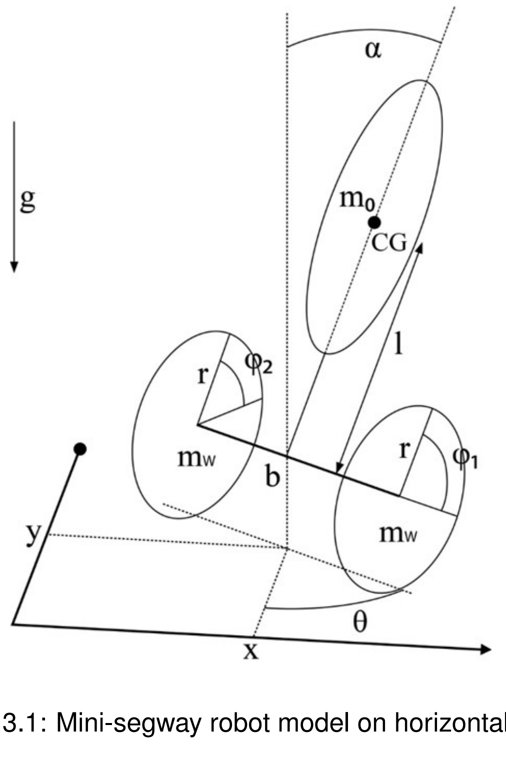


Figure 3.1: Mini-segway robot model on horizontal plane

First, six body frame positional coordinates are defined:

$$q_0 = \theta, \qquad q_1 = x, \qquad q_2 = y, \qquad q_3 = \varphi_1, \qquad q_4 = \varphi_2, \qquad q_5 = \alpha.$$

To use Hamel's Equation to get Equation of Motions (EoM) of two-wheeled robot in 3D space, six quasi-velocities will be introduced [4]. These are chosen in such a way that it makes the application to Hamel's equation easier, this will be made clear a bit later in this section.

$$\begin{aligned}
&\omega_0 = b\dot{\theta} - r(\dot{\varphi}_1 - \dot{\varphi}_2), && \omega_1 = \dot{x}\sin\theta - \dot{y}\cos\theta, && \omega_2 = r\dot{\alpha} + \frac{r}{2}(\dot{\varphi}_1 + \dot{\varphi}_2) - (\dot{x}\cos\theta + \dot{y}\sin\theta), \\
&\omega_3 = \dot{\alpha}, && \omega_4 = \dot{x}\cos\theta + \dot{y}\sin\theta, && \omega_5 = \frac{r}{b}(\dot{\varphi}_1 - \dot{\varphi}_2).
\end{aligned}$$

The velocities ω_1 is the velocity at axle center point parallel to the wheels' axis, while ω_4 is the forward velocity at the same point. Slip of the wheels when it is turning around its vertical axis is given by ω_0, while slip of the wheels when driving forward is ω_2. Further, ω_3 and ω_5 are the angular velocities of the robot's body around the wheels' axis (tilt) and vertical axis (steer) respectively.

The quasi-velocities ω_0, ω_1, and ω_2 are chosen such that it conveniently describes all constraints to the fixed world, while ω_3, ω_4, and ω_5 are chosen in such a way to get inverse transformation to body frame velocities easily.

Since wheel slipping is not permitted, ω_0, ω_1, and ω_2 must equal 0. The constraint $omega_0 = 0$ is holonomic and can be integrated easily resulting

$$\theta = \frac{r}{b}(\varphi_1 - \varphi_2)$$

taken that initial angle $\theta_0 = 0$. While the remaining constraints results the following nonholonomic constraints:

$$\omega_1 = \dot{x}\sin\theta - \dot{y}\cos\theta = 0, \qquad \omega_2 = r\dot{\alpha} + \frac{r}{2}(\dot{\varphi}_1 + \dot{\varphi}_2) - (\dot{x}\cos\theta + \dot{y}\sin\theta) = 0. \tag{3.1}$$

The inverse transformation from the quasi-velocities to body frame velocities are given by:

$$\dot{\alpha} = \omega_3, \qquad v = \omega_4, \qquad \dot{\theta} = \omega_5, \tag{3.2}$$

$$\dot{\varphi}_1 = \frac{1}{r}\omega_2 - \omega_3 + \frac{1}{r}\omega_4 + \frac{b}{2r}\omega_5, \qquad \dot{\varphi}_2 = \frac{1}{r}\omega_2 - \omega_3 + \frac{1}{r}\omega_4 - \frac{b}{2r}\omega_5. \tag{3.3}$$

Then, we need to derive the kinetic and potential energy in terms of each velocity coordinate. The kinetic energy terms are

$$\begin{aligned}
T = {} & \frac{m_o + 2m_w}{2}\omega_1^2 + \frac{m_w}{2}\omega_2^2 + \frac{J_y + m_o l^2}{r}\omega_3^2 \\
& + \left(\frac{m_o + 2m_w}{2} + \frac{J_w}{r^2}\right)\omega_4^2 + \frac{J_5}{2}\omega_5^2 + m_o l(\omega_3\omega_4\cos\alpha - \omega_1\omega_5\sin\alpha) - m_w\omega_2\omega_4,
\end{aligned} \tag{3.4}$$

where J_5 is mass moment of inertia of the whole robot with regard to the vertical axis, defined by:

$$J_5 = (J_x + m_o l^2)\sin^2\alpha + J_z\cos^2\alpha + \frac{3}{4}m_w b^2 + \frac{1}{2}m_w r^2. \tag{3.5}$$

The potential energy is simply:

$$U = -m_o l g\cos\alpha. \tag{3.6}$$

Plugging in all those kinetic and potential energy terms into the Hamel's Equation (Eq. 2.1), taking velocity derivatives with regards to ω_3, ω_4, and ω_5, and then applying inverse transformations back to original body frame velocities, yields the non-linear equation of motions of two-wheeled robot in 3D space:

$$\begin{bmatrix} J_y + m_o l^2 & m_o l\cos(\alpha) & 0 \\ m_o l\cos(\alpha) & m_o + 2m_w + \frac{2J_w}{r^2} & 0 \\ 0 & 0 & J_5 \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \dot{v} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\left(J_z - J_x - m_o l^2\right)\dot{\theta}^2\sin(2\alpha) - m_o lg\sin(\alpha) \\ -m_o l\sin\alpha(\dot{\theta}^2 + \dot{\alpha}^2) \\ \left(J_x - J_z + m_o l^2\right)\dot{\alpha}\dot{\theta}\sin(2\alpha) + m_o lv\dot{\theta}\sin(\alpha) \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ \frac{1}{r} & \frac{1}{r} \\ \frac{b}{2r} & -\frac{b}{2r} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{3.7}$$

The first step to decouple the equation of motion into forward drive motion and steering motion is to decouple the torque inputs into torque for stabilization, and torque that steers/rotates the robot w.r.t. the vertical axis. Hence

$$\begin{aligned} u_1 &= \tau_1 + \tau_2, \\ u_2 &= \tau_1 - \tau_2. \end{aligned} \tag{3.8}$$

Where in physical understanding, u_1 is the torque exerted by the motors to move forward and backward, while u_2 is the differential torque between the motors, which is used to rotate the robot in the vertical axis. Decoupling the input torque yields

$$\begin{bmatrix} J_y + m_o l^2 & m_o l\cos(\alpha) & 0 \\ m_o l\cos(\alpha) & m_o + 2m_w + \frac{2J_w}{r^2} & 0 \\ 0 & 0 & J_5 \end{bmatrix} \begin{bmatrix} \ddot{\alpha} \\ \dot{v} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\left(J_z - J_x - m_o l^2\right)\dot{\theta}^2\sin(2\alpha) - m_o lg\sin(\alpha) \\ -m_o l\sin\alpha(\dot{\theta}^2 + \dot{\alpha}^2) \\ \left(J_x - J_z + m_o l^2\right)\dot{\alpha}\dot{\theta}\sin(2\alpha) + m_o lv\dot{\theta}\sin(\alpha) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ \frac{1}{r} & 0 \\ 0 & \frac{b}{2r} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.9}$$

Further, in order to get the matrix A and B of the state-space model, the mass matrix must be inverted and multiplied to the non-linear matrix and input matrix respectively.

$$A' = \begin{bmatrix} J_y + m_o l^2 & m_o l\cos(\alpha) & 0 \\ m_o l\cos(\alpha) & m_o + 2m_w + \frac{2J_w}{r^2} & 0 \\ 0 & 0 & J_5 \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{2}\left(J_z - J_x - m_o l^2\right)\dot{\theta}^2\sin(2\alpha) - m_o lg\sin(\alpha) \\ -m_o l\sin\alpha(\dot{\theta}^2 + \dot{\alpha}^2) \\ \left(J_x - J_z + m_o l^2\right)\dot{\alpha}\dot{\theta}\sin(2\alpha) + m_o lv\dot{\theta}\sin(\alpha) \end{bmatrix}, \tag{3.10}$$

$$B' = \begin{bmatrix} J_y + m_o l^2 & m_o l\cos(\alpha) & 0 \\ m_o l\cos(\alpha) & m_o + 2m_w + \frac{2J_w}{r^2} & 0 \\ 0 & 0 & J_5 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 0 \\ \frac{1}{r} & 0 \\ 0 & \frac{b}{2r} \end{bmatrix}. \tag{3.11}$$

Next, to get the linearized the state-space model, some assumptions has to be made to simplify the terms in the model. As long as the real system is operating within these assumptions, the model should be sufficient. First, since we will control the tilt angle α, it can be assumed to stay small, therefore approximation of $\sin(\alpha) \approx \alpha$ and $\cos(\alpha) \approx 1$ for small angle α can be used. And then, it is assumed also the angular velocity term are small since it is also controlled to stay around 0, therefore we can make assumptions that all square terms in angular velocities are 0 ($\dot{\theta}^2 = 0, \dot{\alpha}^2 = 0, \dot{\theta}\dot{\alpha} = 0$).

Finally, the terms in state-space model left are all linear, which means now the state-space model can be used to do frequency analysis or any control design approach needed. The linear state-space model for two-wheeled robot in 3D is:

$$\begin{bmatrix} \ddot{\alpha} \\ \dot{\alpha} \\ \dot{v} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & \frac{\left(m_o\, r^2+2\, m_w\, r^2+2\, J_w\right) m_o\, lg}{M} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -\frac{{m_o}^2 l^2 r^2 g}{M} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \alpha \\ v \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} -\frac{lm_o\, r+m_o\, r^2+2\, m_w\, r^2+2\, J_w}{M} & 0 \\ 0 & 0 \\ \frac{r\left(l^2\, m_o+lm_o\, r+J_y\right)}{M} & 0 \\ 0 & \frac{b}{2\, J5\, r} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{3.12}$$

where M is

$$M = 2m_o m_w l^2 r^2 + 2J_w m_o l^2 + J_y m_o r^2 + 2J_y m_w r2 + 2J_w J_y, \tag{3.13}$$

and the plant outputs are the states of the system, meaning the C matrix of the state space model is identity matrix,

$$\mathbf{y} = \mathbf{I}\,\mathbf{x}. \tag{3.14}$$

3.2 RHP Zero Problem and Virtual Sensor Location

From the linearized state-space model obtained in previous section, the model is used in LabView for analysis, controller design and creating transient response simulation of the closed loop. The steps on how the controller is designed will be expanded more in Chapter 4, while in this section, modifications needed to the plant output will be explained.

First, consider only the part of the model for stabilization and forward driving motion for now. This model has 1 input and 3 outputs, the outputs are angular velocity $\dot{\alpha}$, angle α, and forward velocity v.

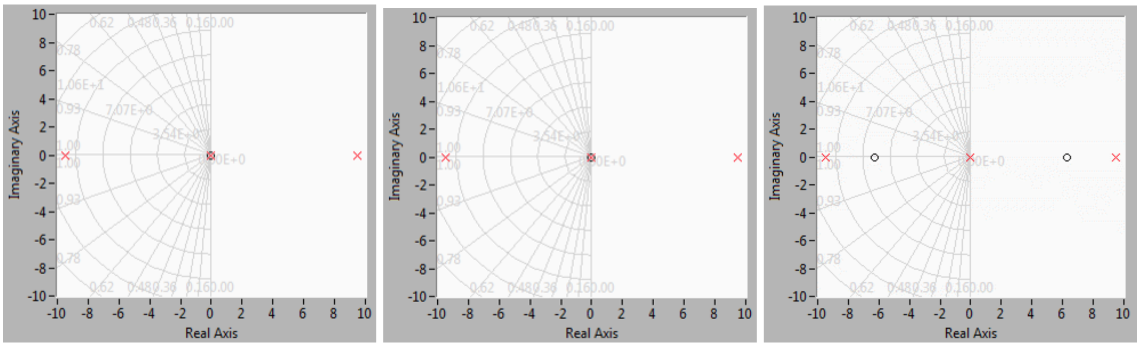


Figure 3.2: The zero-pole plot of the linear model. Transfer from $u_1 \rightarrow \dot{\alpha}$ (left), $u_1 \rightarrow \alpha$ (center), and $u_1 \rightarrow v$ (right)

As seen in Figure 3.2 above, in the transfer between $u_1 \rightarrow v$, there is a zero exist in the right hand plane of the imaginary axis. The RHP zero will cause a problem since it limits the bandwidth of the controller. This zero cannot be moved simply by the controller, therefore the plant itself has to be modified.

The plant outputs can be modified by combining the existing sensor outputs, or in physical understanding is to move the sensor to a virtual point. In this case, the input to velocity transfer needs to be combined with the transfer of angular velocity, so that the roots of the nominator of new velocity transfer is moved to the imaginary axis. This virtual velocity located at distance $l_{virtual}$ from axle is given by

$$v_{out} = v - \dot{\alpha} l_{virtual} \tag{3.15}$$

Therefore the output matrix from the state-space model is no longer identity matrix:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -l_{virtual} & 0 & 1 \end{bmatrix} \mathbf{x}. \tag{3.16}$$

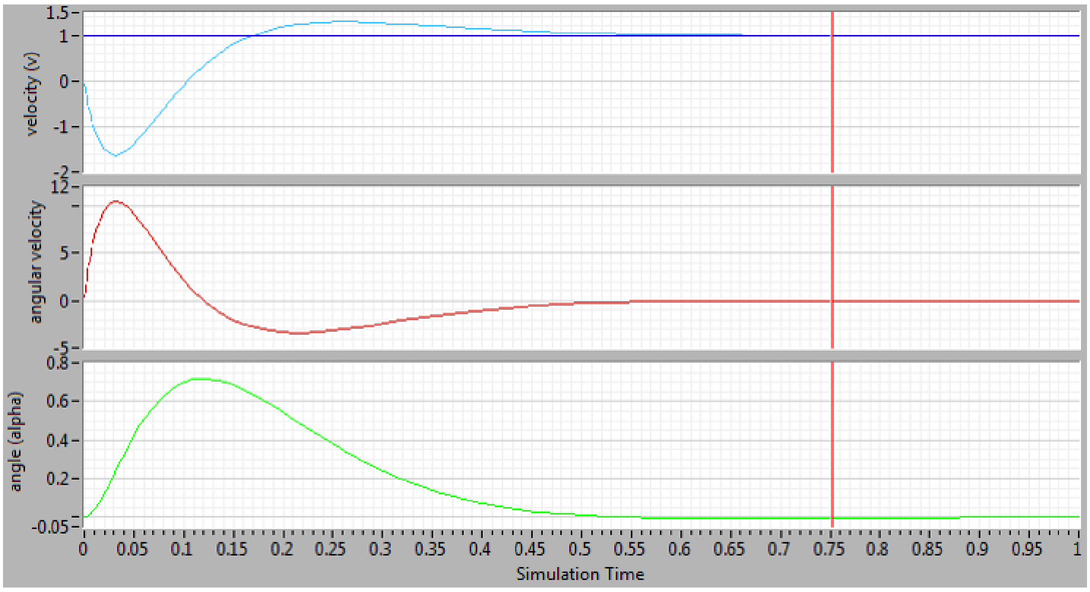


Figure 3.3: Typical behavior of a system with a RHP zero. The transient moves the other way before approaching the final value due to the RHP zero.

After increasing $l_{virtual}$ to a certain threshold, the zeroes switched to imaginary axis. This threshold point is known as the *point of percussion*, where at this point when a torque is applied at the axle, the translation and rotation component at this point cancel each other, resulting in zero displacement. Putting $l_{virtual}$ above the *point of percussion* results in new zero-pole plot of $u_1 \rightarrow v_{out}$:

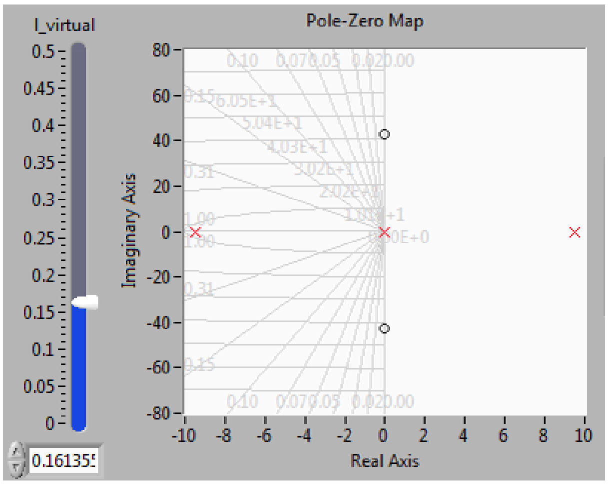


Figure 3.4: The zeroes of the velocity transfer of the plant moves to the imaginary axis after using virtual velocity sensor location.

In practice, the RHP zero behavior is noticed due to when the robot is stationary, the robot's main body is jiggling. This can be explained in control point of view by the control effort of the controller trying to make both velocity v and tilt angle α both to zero. Intuitively, when trying to stabilize α to 0, movement at the axle of the robot has to be present. By moving the controlled velocity to a virtual point, this problem is solved.

3.3 Angle Measurement Algorithm with Accelerometer Sensor

In order to get forward tilt angle measurement signal from accelerometer, an algorithm needs to be devised. The basic idea behind the angle measurement algorithm is to measure the angle of gravity vector which is always perpendicular to the ground plane. A problem arises when the robot is moving, the acceleration measured by the accelerometer is not only measuring gravitational acceleration, but also body acceleration and centripetal acceleration.

When the sensor is mounted at the axle, the accelerometer does not measure centripetal acceleration component anymore. Still, the body acceleration component remains and needs to be compensated. This algorithm is devised to calculate forward tilt angle from accelerometer signal (3 Cartesian axes acceleration) and forward acceleration at the axle, as seen in Figure 3.5.

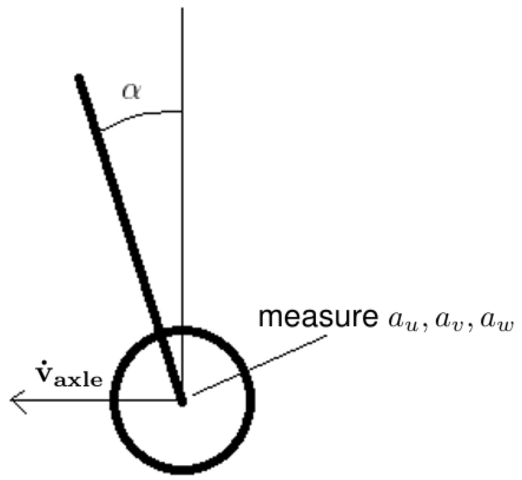


Figure 3.5: Placement of myRIO accelerometer sensor on the body of the robot. The accelerometer frame changes along with local body coordinate frame u, v, and w.

Consider forward acceleration $\dot{\mathbf{v}}_{\mathbf{axle}}$ is in same direction as the tilt angle α, and gravity vector $\mathbf{g}$ is pointing downwards in z-axis in global coordinates frame.

$$\dot{\mathbf{v}}_{\mathbf{axle}} = \begin{bmatrix} 0 \\ -\dot{v} \\ 0 \end{bmatrix}, \qquad \mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}. \tag{3.17}$$

The rotation matrix for forward tilt angle α (constrained in x-axis) is defined as

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}. \tag{3.18}$$

Applying this rotation matrix to forward acceleration $\dot{\mathbf{v}}_{\mathbf{axle}}$ and gravitational acceleration $\mathbf{g}$ results

$$\mathbf{a_{meas.}} = R_\alpha \mathbf{g} + R_\alpha \dot{\mathbf{v}}_{\mathbf{axle}}, \tag{3.19}$$

$$\mathbf{a_{meas.}} = \begin{bmatrix} a_u \\ a_v \\ a_w \end{bmatrix} = \begin{bmatrix} 0 \\ -\sin(\alpha)\, g - \cos(\alpha)\, \dot{v} \\ -\cos(\alpha)\, g + \sin(\alpha)\, \dot{v} \end{bmatrix}. \tag{3.20}$$

Taking the second and third row of the matrix equation and solving it for α yields

$$\alpha = \arctan\left(\frac{a_v g - a_w \dot{v}}{a_v \dot{v} + a_w g}\right) \tag{3.21}$$

Can be observed from Equation (3.21), α is the arctan of some function in accelerometer output signal and forward acceleration signal. This means for small angle, an approximation of $\arctan x \approx x$ can be made. This approximation can be helpful for reducing the computation load on myRIO processor, therefore increasing the loop-rate bandwidth.

The only missing thing that is needed in Eq. (3.21) is the forward acceleration at the sensor location. There are 2 main ways to obtain this signal: feedbacked from other sensors, or feedforwarded from reference input. The first method relies on the encoder output from the motors, which directly gives us the position signals at the axle. The problem with this method is the time-differentiation process needed to get to the acceleration signal, as differentiation process amplifies the noise floor and strongly discouraged in practice.

The latter method, relies on the velocity reference input to the robot. The time-differentiated signal from the velocity reference is easily obtained since the reference itself is a prescribed third-order step curve. That means since the third-order reference itself is made by integrating a step-like function 3 times, any derivatives of the reference signal up to 3 times can be obtained. Using this feedforward method, the forward acceleration at the accelerometer sensor is

$$\dot{v} = \dot{v}_{ref} - \ddot{\alpha} l_{virtual} - b_{sensor}\dot{\theta}. \tag{3.22}$$

Since the reference velocity is located at the virtual point, the second term in Eq. (3.22) is added to revert the location back to rotation point. The third term has to do with placement of myRIO that is not exactly centered with regard to the vertical axis. The parameter b_{sensor} is the horizontal distance parallel to the axle between the center of rotation (for steering) and the sensor location.

Again, the problem with Eq. (3.22) is obtaining $\ddot{\alpha}$, but this term can be substituted if some assumptions holds true (see Figure 3.6). Within the assumption that the robot tracks the velocity reference, and all assumptions made for the linearised model used in the simulation, can be observed that there are proportionality between angle α and acceleration reference at virtual point $\dot{v}_{ref}$, also between $\dot{\alpha}$ and $\ddot{v}_{ref}$. Both α and $\dot{\alpha}$ are linearly related to $\dot{v}_{ref}$ and $\ddot{v}_{ref}$.

$$\begin{aligned}\alpha &\approx k\dot{v}_{ref} \\ \dot{\alpha} &\approx k\ddot{v}_{ref} \\ \ddot{\alpha} &\approx k\dddot{v}_{ref}\end{aligned} \tag{3.23}$$

Therefore it can be concluded that Eq. (3.22) can be substituted as

$$\dot{v} = \dot{v}_{ref} - k\dddot{v}_{ref}\, l_{virtual} - b_{sensor}\dot{\theta}. \tag{3.24}$$

This proportionality constant k is the same for all 3 approximations. The value of k is only dependent on physical parameters of the plant, mainly the masses and inertias of the system. Using physical understanding, this observation makes sense since the controller side parameter is only causing the robot track the reference input better, and not changing the relation between states (e.g. amount of tilt angle needed to make certain amount of acceleration). While the distribution of masses in the system highly influence the relation between these states.

To conclude this chapter, it is important to note that the model used in the simulation is linearised with small angle and small angular velocities assumption, which neglects the coupling terms between

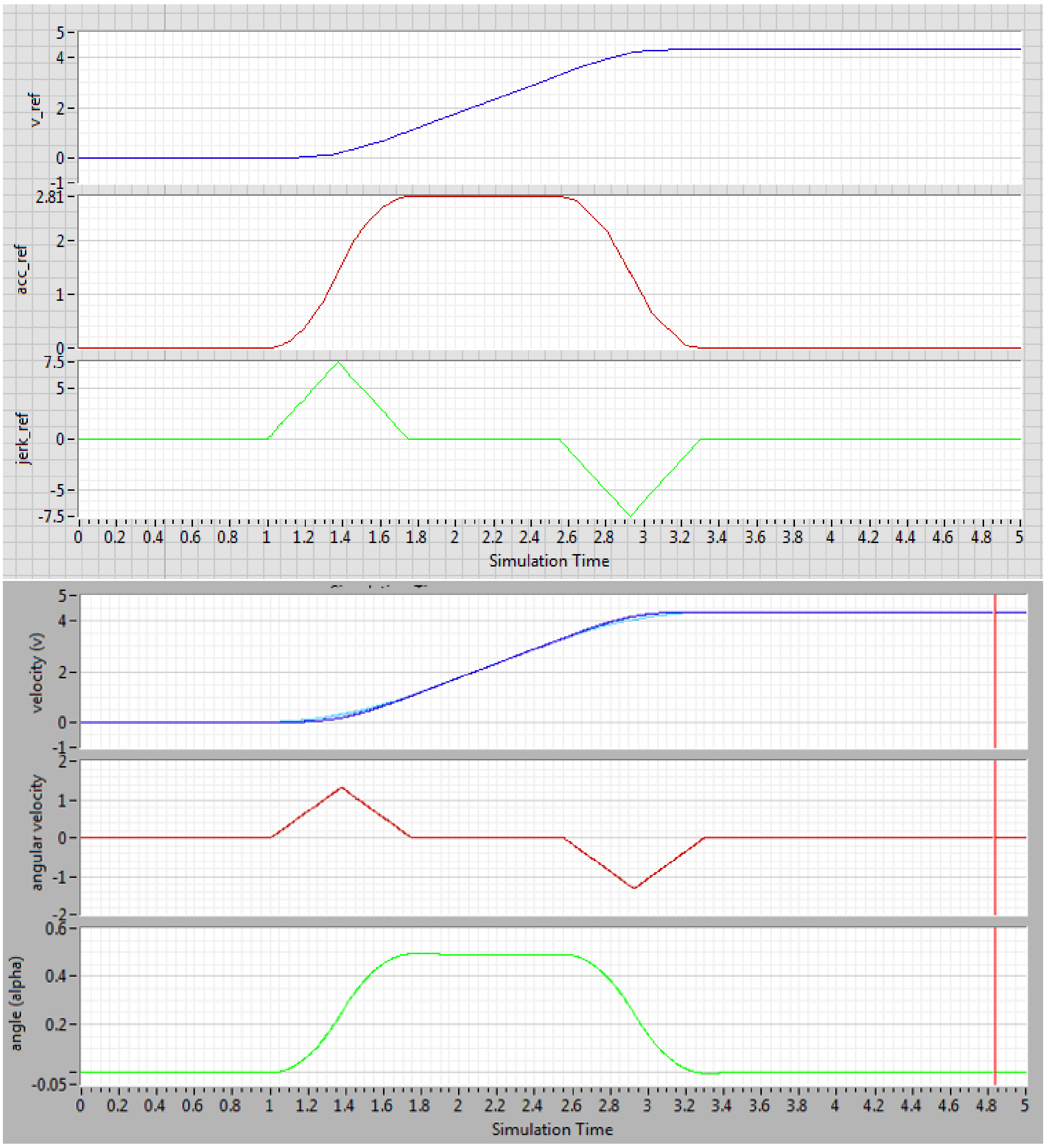


Figure 3.6: Compare the 3rd order velocity reference signal and its derivatives (top) to state variables in the simulation (bottom). As the robot tracks the reference value, it is observed that the forward tilt angle α is proportional to forward acceleration at the virtual point, and angular velocity $\dot{\alpha}$ is proportional to the jerk at virtual point.

tilt angle α and steering angle θ. The behavior caused by RHP zero in the system, can be removed by placing the velocity plant output at some virtual point (slightly above the point of percussion). Finally, angle α can be measured using algorithm indicated in Eq. (3.21) and Eq. (3.22), with assumptions that the system tracks the reference input, and both angles and angular velocities are kept small.

CHAPTER 4

CONTROLLER DESIGN AND TUNING

Controller design and implementation for the two-wheeled segway robot will be addressed in this chapter.

4.1 General Overview of the Controller

Using Eq. (3.8), the plant input is decoupled into stabilization/forward velocity and steering. Consequently, the controller can also be separated respectively into 2 parts. The reference block used to generate a 3rd polynomial reference signal and it's derivatives will be expanded later in Section 4.3.

The stabilization controller with output u_1 is a state-feedback controller with an added feed-forward term to counteract the inertial dynamics of the plant (see Fig. 4.1 below).

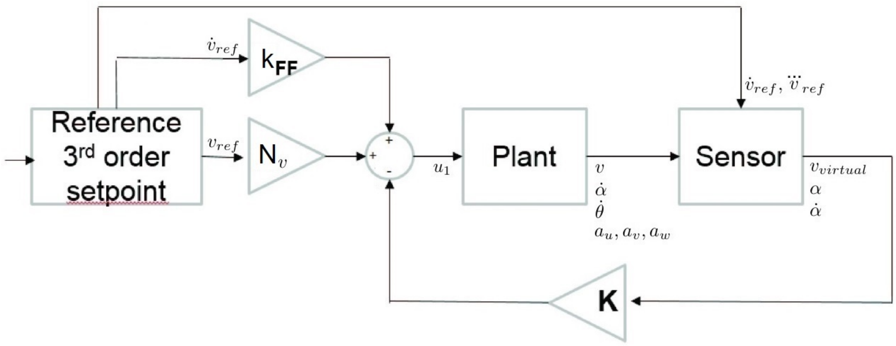


Figure 4.1: Control loop used for velocity control and stabilization of the robot

The values of v, $\dot{\alpha}$, and accelerometer outputs (a_u, a_v, and a_w) are measured, and then angle α is calculated using the angle measurement algorithm from Section 3.3. Choosing

$$u_1 = N_v v_{ref} + k_{FF}\dot{v}_{ref} - k_v v_{virtual} - k_\alpha \alpha - k_{\dot{\alpha}}\dot{\alpha} \tag{4.1}$$

with mass feed-forward gain

$$k_{FF} = r\left(m_o + 2m_w + \frac{2J_w}{r^2}\right)$$

and setting the velocity input reference value normalized with respect to the states,

$$N_v = k_v$$

the closed-loop system poles location can be modified by changing the gains k_v, k_α, and $k_{\dot{\alpha}}$. The method of tuning these control gains using LQR will be expanded later in Section 4.2.

The steering controller with output u_2 is also a state-feedback controller using the plant output $\dot{\theta}$, with added non-linear terms to compensate the compensate the coupling between the stabilization dynamics to the steering motion (see Fig. 4.2).

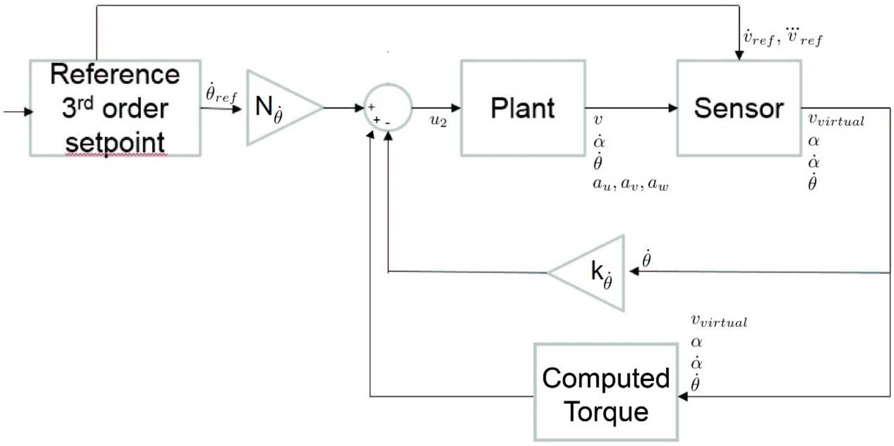


Figure 4.2: Control loop used for steering the two-wheeled robot

The controller output u_2 is given as

$$u_2 = N_{\dot{\theta}}\dot{\theta}_{ref} + \frac{2r}{b}\left(\left(J_x - J_z + m_o l^2\right)\dot{\alpha}\dot{\theta}\sin(2\alpha) + m_o l v\dot{\theta}\sin(\alpha)\right) - k_{\dot{\theta}}\dot{\theta} \tag{4.2}$$

where reference input gain is

$$N_{\dot{\theta}} = k_{\dot{\theta}}.$$

Using this non-linear controller, the remaining dynamics left for the steering motion given from Eq. 3.9 are only the inertial term and the feedback term from the controller. With the supplied reference input signal $\dot{\theta}_{ref}$ is assumed not changing very fast, the inertial term $J_5\ddot{\theta}$ is considered very small, hence mass feed-forward term in the controller is not needed.

With regard to the relations between the plant outputs and the sensors used in the physical system, only 1 gyroscope is used to measure angular velocity $\dot{\alpha}$, and 2 velocity outputs from the wheels are used to calculate the forward velocity v and rotational velocity $\dot{\theta}$. The velocities are given by

$$\begin{aligned} v &= \frac{r}{2}\left(\dot{\varphi}_1 + \dot{\varphi}_2\right), \\ \dot{\theta} &= \frac{r}{b}\left(\dot{\varphi}_1 - \dot{\varphi}_2\right). \end{aligned} \tag{4.3}$$

Lastly, the angle α is calculated from the accelerometer outputs using the algorithm described in Section 3.3.

4.2 Control Gain Tuning with LQR

The next step in implementing the controller is to tune the control gain using Linear Quadratic Regulator method. The LQR tuning method gives physical insight and the flexibility to weight the outputs tracking error with respect to each other. The steps to get the control gain k_v, k_α, $k_{\dot{\alpha}}$, and $k_{\dot{\theta}}$ using LQR is outlined in Section 2.2.

First the values in matrix $\mathbf{C}_1$ needs to be determined. A common method to weight the values in the matrix is by taking one divided by the expected operating range of each output. This way, each outputs are similarly important over the full range they are operating at, instead of weighting every 1 unit of the outputs. Taking the expected maximum values from the simulation, $\alpha_{max} = 0.23\,rad$, $\dot{\alpha}_{max} = 0.98\,rad/s$, and $v_{max} = 4\,m/s$, $\mathbf{C}_1$ is chosen as

$$\mathbf{C}_1 = \begin{bmatrix} 1.02 & 0 & 0 \\ 0 & 4.34 & 0 \\ 0.25\,l_{virtual} & 0 & 0.25 \end{bmatrix} \tag{4.4}$$

Then, solving the symmetric root locus problem

$$\rho G_0(-s)G_0(s) + 1 = 0,$$

for each transfer, produces symmetric root locus diagrams as seen in Figure 4.3

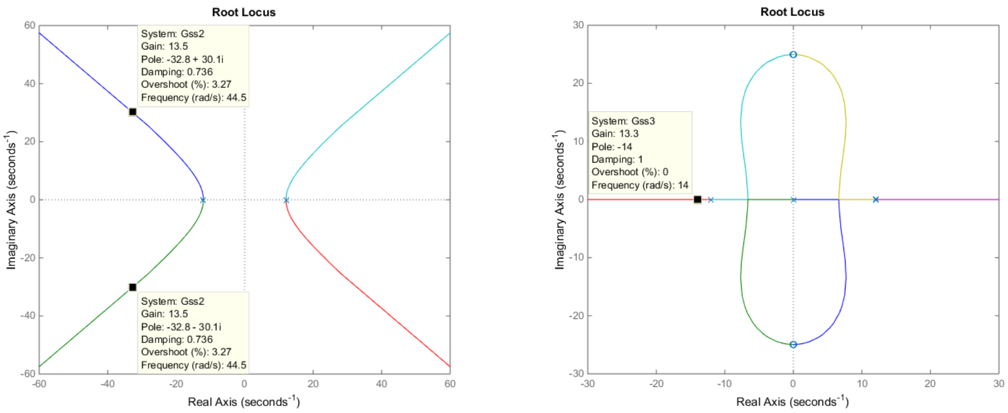


Figure 4.3: The symmetric root locus diagram for the transfer between input u_1 to angle α (left) and input u_1 to velocity v (right)

Choosing $\mathbf{C}_1$ as in Eq. (4.4) and $\rho = 13.5$ will place the closed-loop system poles at $-32.8 \pm 30.1i$, and -14. The two poles in the left diagram are responsible for keeping the robot stays upright, while the remaining pole in the right diagram is responsible for tracking the reference velocity input. This results in system's bandwidth for stabilization to be $7.1\,Hz$ and for velocity tracking to be $2.2\,Hz$. The poles' locations corresponds to gains $k_v = 46.2$, $k_\alpha = 24.59$, and $k_{\dot{\alpha}} = 4.44$

Apparent problem that can be seen while tuning the stabilization controller is the unexpected vibration caused by a pole around $70\,Hz$ that is present when the gain on the angle and velocity gain are high (typically above 50). This pole may be caused by internal pole of one of the sensors, or from the flexible eigenmode of the physical plant itself.

Next, tuning the steering angular velocity $\dot{\theta}$ is very straightforward since it is assumed to be already decoupled from other state variables. Taking bandwidth of $20\,Hz$ puts the gain $k_{\dot{\theta}} = 1.29$.

4.3 Introducing Velocity Reference Input

The main purposes of the reference block is to create 3rd order polynomial reference velocities signal and it's derivatives based on reference setpoints (sampled at low rate), also to limit the acceleration of the robot to a prescribed value. First, jounce signal (third derivative of velocity) is prescribed and the maximum value is kept constant, and then integrated several times until finally velocity signal is obtained. By changing the integration time, the acceleration and velocity signal can be modified (which is useful to limit the maximum acceleration prescribed into the robot and setting the final value of the velocity).

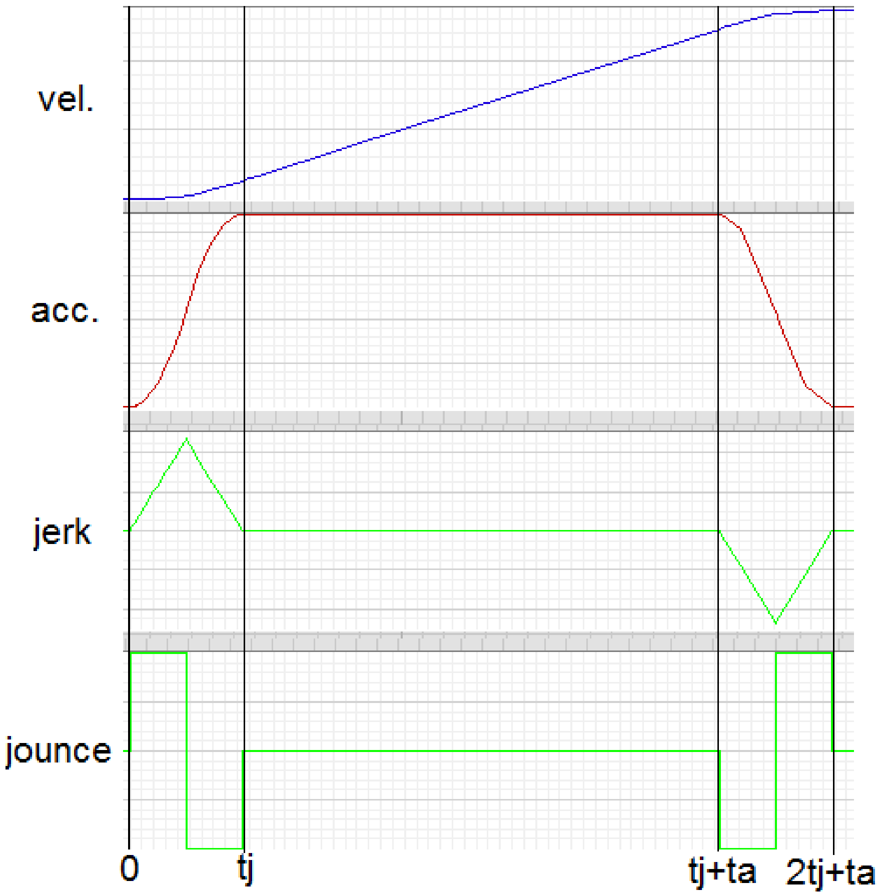


Figure 4.4: From the top to bottom: velocity signal, acceleration, jerk, and prescribed jounce with a constant maximum. t_j is the time used to reach the maximum acceleration, and t_a is the time needed to accelerate/ time used to reach final velocity value.

Looking at Fig. 4.4 above, taking the maximum jounce constant s_{max}, calculating the maximum value for each signal using finite integration over each respective time span results:

$$\begin{aligned} j_{max} &= \frac{s_{max}}{2} t_j, \\ a_{max} &= \frac{s_{max}}{4} t_j^2, \\ v_{max} &= \frac{s_{max}}{4} t_j^2 \left(t_a + t_j\right). \end{aligned} \tag{4.5}$$

Using this 3 equations, we can set the acceleration amount and the final velocity value just by adjusting t_j and t_a. The prescribed maximum permissible acceleration is set at $1.1\,m/s$ (see Section 5.1), therefore the final value given by the setpoint reference cannot be reach everytime. To tackle this problem, few cases are defined by if-else-then algorithm made in the implementation. First case is when the final velocity value is below the limit

$$|v_{end}| \leq 1.1 \left(T - \sqrt{\frac{4 * 1.1}{s_{max}}}\right)$$

where T is the sampling time period of the reference block, then final velocity value can be reached by adjusting t_j and t_a within the maximum limit. Second case, is when the final value is over-limit, then the reference block is told to keep accelerating at maximum acceleration. From the second case, a third and fourth case needs to be added for continuing the second case, whether it needs to keep accelerating at maximum acceleration until the next time period, or whether it can reach the maximum value at that time period.

4.4 Controller Implementation in LabView

The controller is implemented in LabView using Timed Loop and Flat Sequence Structure. The Flat Sequence structure is used to initialize, execute the Timed Loop, and then end the program when stop button is pressed. While Timed Loop is used since it is a deterministic loop, meaning that the loop will synchronize to the internal clock of the myRIO, and will try to finish the calculation before each time period ends. Control Design and Simulation Loop can also be used instead of Timed Loop, but with the disadvantages that the maximum loop-rate will be limited to 1 kHz only. With the Timed Loop, the implemented loop-rate for the controller is 1.667 kHz.

First communication port is initialized, and then the loop is commenced. The implemented controller in LabView inside the Timed Loop is given in Figure 4.5 below.

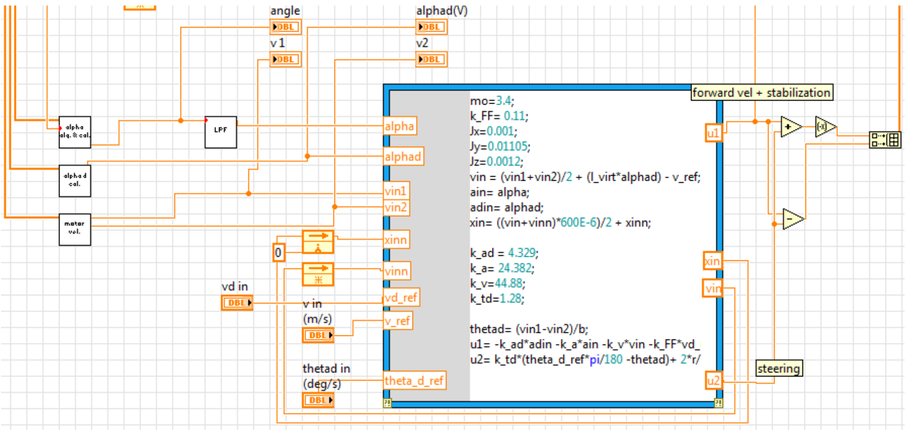


Figure 4.5: The controller is implemented using MathScript structure, while the subsystems on the left is used to calibrate and adjusting the gains/sensitivities of the sensors, also implementing the angle measurement algorithm.

The angle measurement algorithm is implemented in the first subsystem on the top left (see the implementation on Fig. 4.6 below).

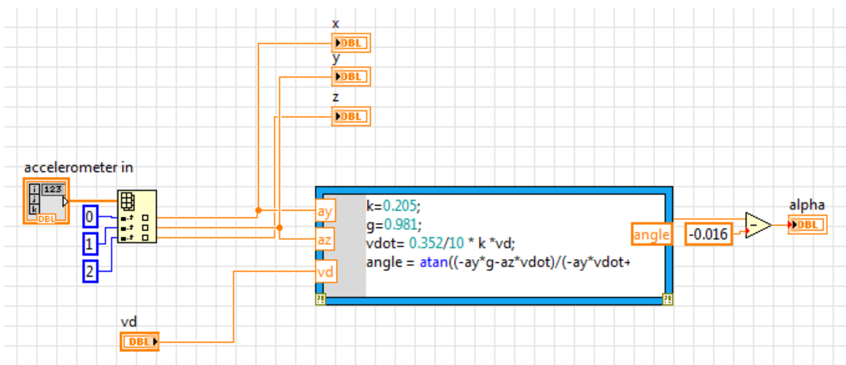


Figure 4.6: The angle measurement algorithm is also implemented using MathScript structure inside a subsystem.

CHAPTER 5

VEHICLE DESIGN AND RESULTS

One of the main challenges in realizing a small two-wheeled robot is to design and build the physical system itself. In this chapter, all aspects of mechanical assembly, power design, design choices and requirements will be discussed.

5.1 Design Choices and Requirements

An important part of building a feedback controlled system is choosing the sensors and actuators to be used in the system. But in this case, we will prioritize the available sensors and motors to be used in the system. Here is a list containing available sensors that will be used in the final assembly:

- Built-in 3 axes accelerometer of the myRIO (± 8g range, 12 bits resolution)
- Encoder MILE 6400 counts per turn, embedded in the motors
- Analog Devices ADXRS623 $\pm 150°$/sec yaw rate gyroscope

The encoder sensors from the motor are position sensors, but the motor driver has a function to output the averaged velocity of the motor. Meaning that the position signal from the encoder are differentiated, and then averaged over few samples. This process will introduce delay into the velocity signal, but it is negligible due to the fact that the sample rate of the encoder is much higher than the control bandwidth of the feedback system.

The motors used in the system are 90 Watt brushless Maxon EC 90 flat with ESCON 50/5 servo controller, also chosen due to their availability. The motors are able to exert $0.444\,Nm$ continuous torque, but are limited to $0.352\,Nm$ maximum torque due to the maximum current limitation of the ESCON 50/5 motor drivers. With simple Newtonian calculation, using $5.1\,cm$ wheel radius and $4.04\,kg$ total mass (see Section 5.2), the system can accelerate up to $1.71\,m/s^2$, excluding the torque needed for stabilization and counter-acting external disturbances. Therefore, a maximum acceleration of $1.1\,m/s^2$ is prescribed to the reference input in order to put a margin for counter-acting disturbances while accelerating and also for stabilization.

However, the acceleration of the the two-wheeled robot is also dependent on the forward tilt angle of the robot. As it has been previously explained in Section 3.3, there is a linear relation between tilt angle α and acceleration at virtual point v_{out}. Therefore the bigger the prescribed acceleration to the robot, the bigger the tilt angle is needed. With the prescribed $1.1\,m/s^2$ of acceleration, and masses/inertias obtained from Solidworks (Section 5.2), we have a maximum of $0.226\,rad$ or $12.9\,°$ of tilt angle α, which is still within the assumptions of the model.

Making the robots overall build relatively small is one of the requirements. Smaller robots typically also means lighter in mass, hence able to do bigger acceleration and more agile. Other requirement for the design is the myRIO's accelerometer sensor needs to be mounted at the rotation axis, in-line with the wheel's axle.

5.2 Mechanical Assembly

In order to assemble all the parts together, a chassis and mounting platforms for the myRIO and the batteries are needed. The wheels used in the system are Actobotics 4" Precision Wheels, with outer rubber diameter of $10.2\,cm$. Two stainless steel hub are made to connect the wheels to the axle of the motors.

The mechanical design and assembly are done using Solidworks 2014 (see Figure 5.1).

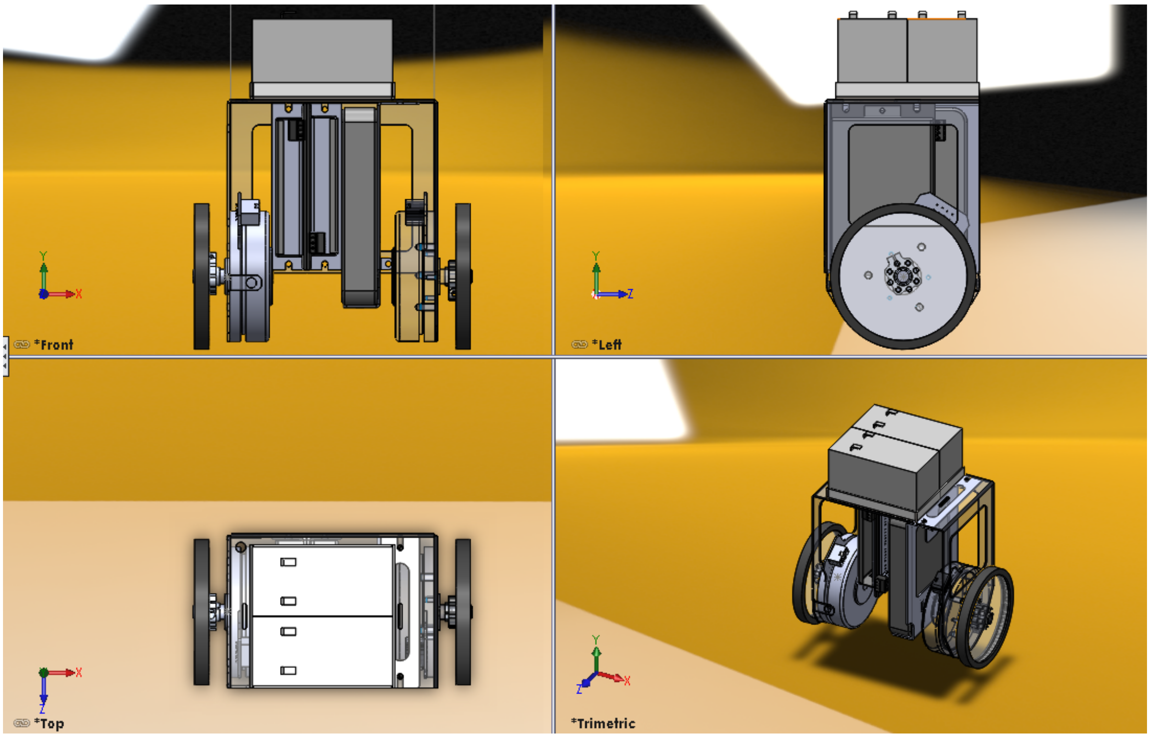


Figure 5.1: Front, top, side, and isometric view of the design made with Solidworks

The mounting platform of the myRIO and the chassis are made with 1.5 mm thick stainless steel sheet, while the battery platform is made with 1 mm stainless steel sheet. All parts are made by laser-cutting and bending the after-cut parts. Then, the chassis is welded at the back to ensure the structural rigidity of the robot.

As the final assembly is done with Solidworks, mass properties calculation can also be done in Solidworks, this will give us the estimates of the masses and inertias of the robot with sufficient accuracy. The total mass of the robot weighs $4.04\,kg$, and the principal moments of inertias at the center of mass are $11050.05\,kg\,mm^2$ w.r.t. the wheel's axle and $6528.70\,kg\,mm^2$ w.r.t. the vertical axis. The center of mass of the robot is located at $6.96\,cm$ above the wheel's axle.

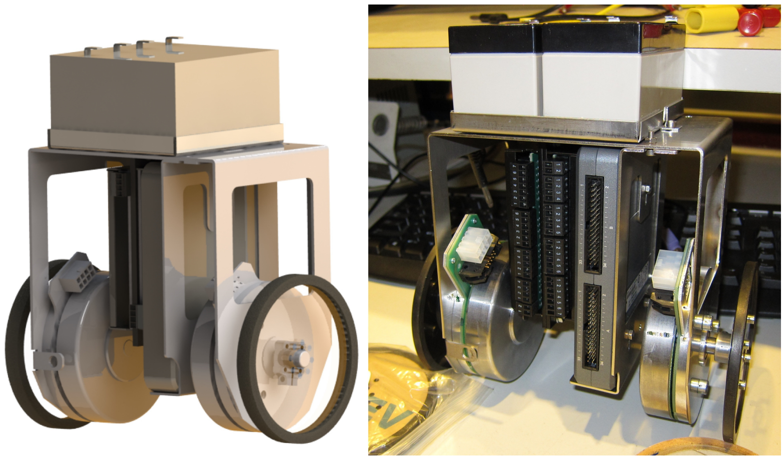


Figure 5.2: The rendered assembly in Solidworks (left) and the actual finished assembly without cable connections (right)

5.3 Power and Electrical Design

Concerning power delivery, a battery needs to be chosen as the power source for the robot. The motors need 5A max current at 24V each, and the myRIO needs about 1.2A at 12V. Looking at these requirements, the maximum current discharge of the battery has to be in the range of 11A, which is rather big. Therefore lead-acid battery type is chosen as the battery for the robot because of this current discharge requirement, and also has relatively low price compared to other types of batteries. The drawbacks of using lead-acid battery are the size and the weight of the battery, due to lead-acid battery has low energy density (amount of energy can be stored per volume of the battery).

Two of Yuasa NP1.2-12 batteries in series are used as the power source of the robot, totaling 1.2 Ah of battery capacity at 24V. With these batteries, at maximum load, 8.9 minutes of operational time can be obtained. The realistic operational time might be far higher due to the fact that the motors do not need to exert much torque when the robot is stationary.

One problem arises when the myRIO and the motor drivers are connected together to same power source. Since the impedance between the myRIO and the motor drivers is low, there will be voltage ripple at the power input of the myRIO due to the big current spikes drawn to the motor drivers. This will cause an inaccurate voltage reference for the myRIO, therefore also influencing the voltage outputs. To solve this problem, a voltage regulator circuitry is used to power the myRIO, creating a voltage buffer between the motors and the myRIO, and also balancing the use of the 2 batteries.

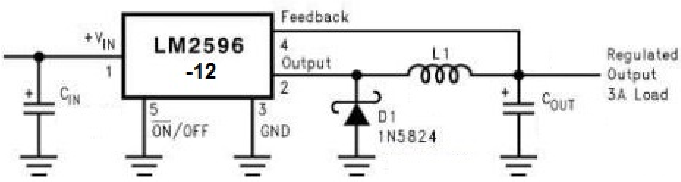


Figure 5.3: Switching 24V/12V step-down voltage regulator circuit using LM2596-12 IC

Using a linear regulator in this case will cause heat problem without a proper heatsink, since half of the supplied power are wasted into heat. Using LM2596 IC, stable voltage can be delivered to the myRIO.

With regard to signal connection, the velocity signals are connected to analog inputs on MSP Connector C of the myRIO. The angular velocity signal from the gyroscope is connected to analog input on MXP Connector A of the myRIO. The motor drivers input are connected to the analog outputs on MSP C port, giving range between -10V to 10V to control the torque of each motor. There is also enable signals at the motor driver that are connected to digital output of myRIO, as a safety precaution if the motors spin out of control.

5.4 Final Results

The final segway robot physical assembly is successfully built, with the total mass of $4.04\,kg$, and it's center of mass $0.6965\,m$ above the rotation axis. The controller is working as expected, able to stabilize the system when a forward velocity or steering angular velocity reference is given to the closed system. While the performance is not tested with a proper methodology due to limitation of time, the robot is sufficiently able to deliver the expected performance judged by empirical observation.

The angle measurement algorithm is working as expected, able to compensate the body acceleration on the system while there is no disturbance applied to the robot. The algorithm is far from perfect and has some flaws, which will be expanded in the Discussion. The reference generator block working, but has perceiveable drift due to the accuracy of the ODE45 solver in a Timed Loop environment (also will be expanded later in the Discussion).

CHAPTER 6

DISCUSSION AND CONCLUSION

In the discussion section, some notes/remarks throughout the process of realizing the two-wheeled segway robot will be reviewed, and recommendations for similar future projects will be noted. And finally in the conclusion, the results of this bachelor project are summarized.

6.1 Discussion

Along the process of working on this bachelor project, some remarks can be made:

1. With regard to the angle measurement algorithm, it is realized that the algorithm has a weakness due to no compensation for the robot's acceleration in z-axis. For example, when the robot tries to move across a ramp, it makes the gravitational acceleration appears smaller/larger to the angle measurement algorithm, therefore produces inaccurate angle measurement result. This problem can be partially solved by simply limiting the maximum acceleration in z-axis to $1\,g$, but it still does not solve the problem when the robot moves down through the ramp, or when the robot is accelerating on a ramp.
2. The reference input used in the setup is not an ideal 3rd order polynomial step input function. Creating a 3rd order polynomial function from a referenced set-point in real time in LabView, is a very difficult task. The difficulties is caused by the 1kHz loop-rate limit of the Timed Loop used to generate the reference signal. The ODE solver for the integrator cannot use a step size small enough because of this 1kHz loop-rate limit to get a sufficiently accurate reference signal. Both discrete integrator block and rate-limiter block (used to limit the prescribed acceleration), needs to be run either in a subsystem inside a Timed Loop, or using Control Design & Simulation Loop, and both are limited to 1 kHz loop-rate for the myRIO.
3. Concerning the batteries, the better subtitute for the lead-acid batteries are using lithium-polymer batteries. This type of batteries have much higher energy density compared to the lead-acid type, while having similar performance for the maximum current that it can discharge. If the batteries are substituted with lithium-polymer, the overall robot build would be lighter in weight, and would save more space on the mechanical design of the robot. The only drawback of lithium-polymer battery is the high price per capacity, comparative to the energy density offered by this battery type.
4. Using LabView to get measurement data from instruments, such as myRIO, is very straightforward as all the drivers are embedded into the LabView software, however, using LabView as a complete software development environment for myRIO is unnecessarily troublesome. Most of the problems are caused by the Control Design & Simulation (CD&Sim) Loop limits and difficulty to communicate deterministically with other type of loop (e.g. Timed Loop).
   The CD&Sim Loop, in author's opinion, is not yet well-developed and quite buggy. While it is quite useful and fast to implement a control system with loop-rates below 1 kHz, using Timed Loop and more low-level LabView programming is proven to be more robust and efficient in terms of computational load on the myRIO.
   Also, using LabView as modeling and analysis tool is definitely not a good idea. Compared to MATLAB for example, it has a very powerful linearization tool (`linmod` function) that can directly gives linearized transfer from a Simulink model, while in LabView, a model created with CD&Sim

Loop can only simulate the transient response and cannot be used to do frequency analysis on the system. In other words, to do frequency analysis, a mathematical model needs to be plugged in into LabView, and then transient simulation using CD&Sim Loop needs to be created. Thus, it is a two step process.

5. The time available to do this bachelor project is very limited, a proper final testing could not be done within the time frame. In order to thoroughly assess the dynamic behavior of the final system, a system identification must be done to the final system. Therefore the apparent resonant poles observed while doing the controller tuning can be compensated exactly (if the exact frequency is known). Final tracking performance test must also be done to assess the controller performance on decoupling the stabilization and steering dynamics (e.g. tracking a circle).

For future works, a solution or a completely new algorithm for measuring angle from accelerometer outputs can be investigated. Other than that, there are still aspects on the design that be improved on, such as using a more simple motor driver (e.g. using an H-bridge circuit) so it does not add much additional space and weight into the system. And then, some follow-up projects can be considered, such as using a real-time video feed from a webcam as position guidance to the robot. The myRIO FPGA has capabilities to do the calculation needed, and the serial port of the myRIO itself provides an easy implemetation of a camera using a standard USB connection.

6.2 Conclusion

From this project, can be concluded that NI myRIO is a great platform to build a small, compact robotic system. However, the software LabView used to develop the program for the myRIO RT target, while it is quite easy to learn, it is very hard to get the program done right. Getting measurement data through LabView is very straightforward, but when trying to implement a filter or a integrator in the controller, the software seems still under-developed.

On the technical side of two-wheeled robot realization, the physical assembly of the robot was done succesfully. The angle measurement algorithm is working as expected, while has flaws as pointed in the Discussion. The reference generator is not working ideally, but the accuracy compared to the supplied joystick signal is sufficient. The stabilization controller and the steering controller is working as expected. While the final system is not tested through a proper methodology, the controller is able to make the robot stabilize itself and sufficiently able to follow the reference input given.

APPENDIX A
LABVIEW DOCUMENTATION

All codes made using LabView is thoroughly documented. Any inquiries regarding all the VIs used on the myRIO RT Target, 3rd polynomial generator block, or VIs used for modeling and simulation can be done by e-mail to `y.x.mak@student.utwente.nl`.

BIBLIOGRAPHY

[1] *User Guide and Specifications NI myRIO-1900*. National Instruments, August 2013. URL http://www.ni.com/pdf/manuals/376047a.pdf.

[2] Michael Baloh and Michael Parent. Modeling and Model Verification of an Intelligent Self-Balancing Two-Wheeled Vehicle for an Autonomous Urban Transportation System. In *The Conference on Computational Intelligence, Robotics, and Autonomous System*, Singapore, December 2003.

[3] Arnoldo Castro. Modeling and Dynamic Analysis of a Two-Wheeled Inverted Pendulum. Master thesis, George W. Woodruff School of Mechanical Engineering, August 2012.

[4] T. Zaiczek and M. Franke. Tracking Control of a Balancing Robot. *Archive of Mechanical Engineering*, 61(2):331–346, August 2014.

[5] Georg Hamel. ber die virtuellen Verschiebungen in der Mechanik. *Mathematische Annalen*, 59: 416–434, 1904.

[6] J.I. Neimark and N.A. Fufaev. *Dynamics of Nonholonomic Systems*. Translations of mathematical monographs. American Mathematical Soc. ISBN 9780821886601. URL http://books.google.nl/books?id=RZ7ir1Lf7TAC.

[7] Gene Franklin, J. D. Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems (5th Edition)*. Prentice Hall, 5th edition, November 2005. ISBN 0131499300. URL http://www.worldcat.org/isbn/0131499300.

[8] T. Kailath. *Linear Systems*. Information and System Sciences Series. Prentice-Hall, 1980. ISBN 9780135369616. URL http://books.google.nl/books?id=ggYqAQAAMAAJ.