Project Report

MSE 483 – MODERN CONTROL SYSTEMS, SPRING 2016, SFU

Submitted To:	Dr. Mehrdad Moallem
Submitted by:	
Juzer Antria	301199594
Refayet Siam	301210102
Sohail Sangha	301186636
Submitted On:	19th , April, 2016

1 Abstract

In modern control problems, state space representation wrapped with ideas from classical frequency theory are extensively used to design controllers for plants to produce desired outputs. The project reported in this report aims at taking the concepts from theory out into the world of simulation and application. Students go through the process of analyzing the kinematics of a two wheeled robot (Segway), simulating and studying it through concepts of linear state space control, develop and fine tune servo mechanism controller, and take it further by building a low-cost miniature version of the robot and attempt to program it with the results obtained from the simulation. The report is sectioned in two parts, 1) simulation: contains the background on the kinematics, analysis of controllability, observability and other aspects, controller design, and analysis of the results. 2) prototype: contains the ideology and design behind the hardware prototype, embedded code structure, and the results. While the students were able to design a controller for the simulation, implementation on the physical prototype fell through due to time limitations. The work on physical prototype was halted during the controller fine-tuning process, but the substantial work done to reach that stage is detailed in the report to minimize efforts spent in future attempts.

Table of Contents

List of Tables

List of Figures

2 System Simulation

The process of developing controller for a plant starts with the building of a mathematical model. The model is used to apply classical concepts such as controllability, observability, and stability to the system, which give the designer ideas about the applicable controllers. Further work usually needs to be done on the controller in fine-tuning and producing system behavior meeting certain requirements. The following sections go through the process step by step.

2.1 Simulation Requirements

The objective is to produce a controller for a two-wheeled inverted pendulum as in the figure below. The problem is twofold as it involves the control of the motion of the robot (through back and forth, and yaw motion), while controlling the instability of the system (through converging the body pitch to the upright position). Thus, the requirements from the system are directly applicable to the performance delivered across these variables.

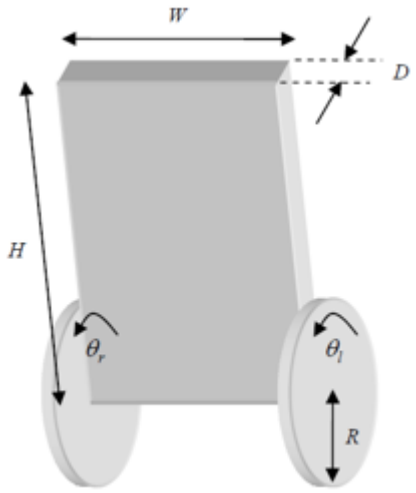

Figure 1: Two-wheeled inverted pendulum

The state space model of the system will be detailed in the following section, which will introduce readers to: wheel angle θ, body pitch ψ, and body yaw ϕ. The body pitch directly indicates the stability of the system, further, linearization of the model dictates the bounds on the value. The wheel angle and body pitch are linked to motion, thus the requirements deal with concepts such as speed of motion, and overshoot.

Table 1: Requirements for different variables

Variable	Property	Requirement
Body Pitch	Range	$-15^0 < \psi < 15^0$
Wheel Angle	Movement Speed	$Wheel\ Radius * \lvert\dot{\theta}\rvert \geq 2\ cm/s$
	Overshoot	$Wheel\ Radius * \theta_{os} \leq 5\ cm$
Body Yaw	Movement Speed	$\lvert\dot{\phi}\rvert \geq 45\ degrees/s$
	Overshoot	$\phi_{os} \leq 10\ degrees$

2.2 State Space Representation

The major variables defining the motion of the robot are as follows:

1. θ_l, rotational position of left wheel
2. θ_r, rotational position of right wheel
3. θ, wheel position, $\theta = (\theta_r + \theta_l)/2$
4. ϕ, wheel axle yaw, $\phi = (\theta_r - \theta_l)/2$
5. ψ, body pitch, zeroing at upright position of inverted pendulum

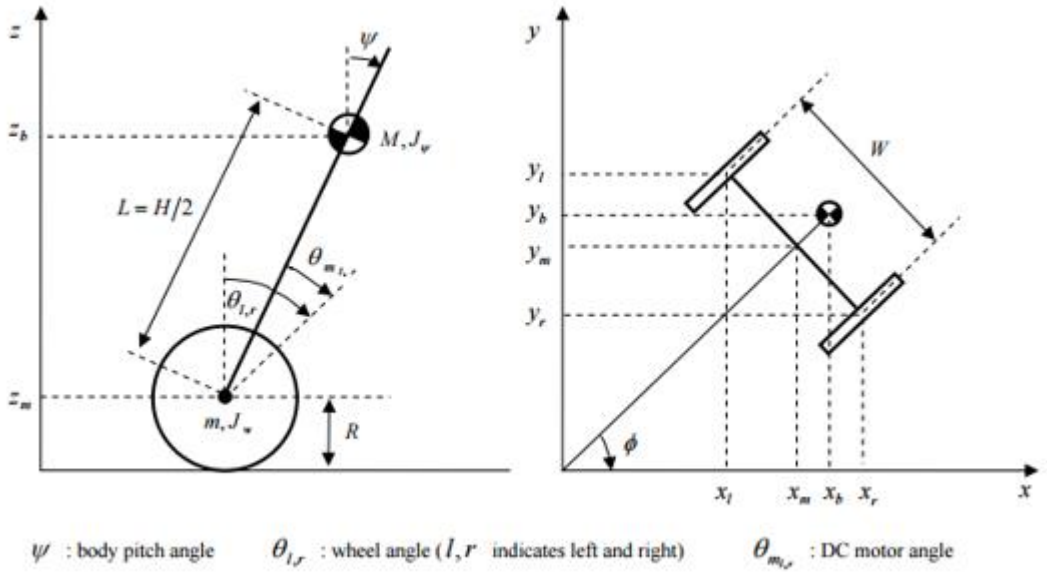

Figure 2: Side and Plane View of Two-wheeled inverted pendulum

The state space model was derived using equations of motions of a two-wheeled inverted pendulum [1]. Linearizing the motion equations about the balance point gives the following state space equation for the system:

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A(3,2) & A(3,3) & A(3,4) \\ 0 & A(4,2) & A(4,3) & A(4,4) \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ B_1(3) & B_1(3) \\ B_1(4) & B_1(4) \end{bmatrix}$$

$$\boldsymbol{x_1} = \left[\boldsymbol{\theta}, \boldsymbol{\Psi}, \dot{\boldsymbol{\theta}}, \dot{\boldsymbol{\Psi}}\right]^T, \boldsymbol{u} = [\boldsymbol{v_1}, \boldsymbol{v_r}]^T$$

$$C = [1 \quad 0 \quad 0 \quad 0], D = [0]$$

The final state space model looks like:

$$\dot{\boldsymbol{x}} = A_1 \boldsymbol{x_1} + B_1 \boldsymbol{u}$$

Output equation looks like:

$$y = C\boldsymbol{x_1} + D\boldsymbol{u}$$

Where:

$$A_1(3,2) = -\frac{MgLE(1,2)}{\det(E)}$$

$$A_1(4,2) = \frac{MgLE(1,1)}{\det(E)}$$

$$A_1(3,3) = -\frac{[(\beta + f_w)E(2,2) + 2\beta E(1,2)]}{\det(E)}$$

$$A_1(4,3) = \frac{[(\beta + f_w)E(1,2) + 2\beta E(1,1)]}{\det(E)}$$

$$A_1(3,4) = \frac{\beta(E(2,2)) + 2E(1,2)}{\det(E)}$$

$$A_1(4,4) = -\frac{\beta(E(1,2)) + 2E(1,1)}{\det(E)}$$

$$B_1(3) = \frac{\alpha(\frac{E(2,2)}{2}) + E(1,2)}{\det(E)}$$

$$B_1(4) = -\frac{\alpha(\frac{E(1,2)}{2}) + E(1,1)}{\det(E)}$$

$$E = \begin{bmatrix} (2m+M)R^2 + 2J_w + 2n^2J_m & MLR - 2n^2J_m \\ MLR - 2n^2J_m & ML^2 + J_\Psi + 2n^2J_m \end{bmatrix}$$

$$\det(E) = E(1,1)E(2,2) - E(1,2)^2$$

$$\alpha = \frac{nK_t}{R_M}, \qquad \beta = \frac{nK_tK_b}{R_m} + f_m$$

Table 2: Parameters used for the Representation

Parameters			Units
Gravity acceleration	:	$g = 9.81$	$[\frac{m}{sec^2}]$
Wheel weight	:	$m = 0.0113$	$[kg]$
Wheel radius	:	$R = 0.03$	$[m]$
Wheel inertia moment	:	$J_w = \frac{mR^2}{2}$	$[kgm^2]$
Body weight	:	$M = 0.2$	$[kg]$
Body width	:	$W = 0.075$	$[m]$
Body depth	:	$D = 0.04$	$[m]$
Body height	:	$H = 0.15$	$[m]$
Distance of the center of mass from the wheel axle	:	$L = \frac{H}{2}$	$[m]$
Body pitch inertia moment	:	$J_\Psi = \frac{ML^2}{3}$	$[kgm^2]$

DC motor inertia moment	:	$J_m = 0.00001$	$[kgm^2]$
DC motor resistance	:	$R_m = 3.75$	$[\Omega]$
DC motor back EMF constant	:	K_b =0.573	$[V\frac{sec}{rad}]$
DC motor torque constant	:	K_t=0.309	$[Nm/A]$
Gear ratio	:	$n = 1$	
Friction coefficient between body and DC motor	:	$f_m = 0.0022$	
Friction coefficient between wheel and floor	:	$f_W = 0$	

2.3 Controllability

The interaction of the inputs to the state space can be observed by deriving conditions under which the state trajectory is driven to the origin. This can be used to determine if a system can be controlled or not by comparing the controllability matrix of the system with the system's rank. A system's said to be controllable if the row rank of the controllability matrix is equal to the number of rows of the nonsingular A matrix of the state space model. The controllability matrix for the given state space model is:

$$P = [\ B_1(:,1) \quad B_1(:,2) \mid A_1B_1(:,1) \quad A_1B_1(:,2) \mid A_1^2B_1(:,1) \quad A_1^2B_1(:,2) \mid A_1^3B_1(:,1) \quad A_1^3B_1(:,2)\ ]$$

Using MATLAB

A_1 =

0	0	1	0
0	0	0	1
0	-382.38	-730.94	710.73
0	204.98	271.80	-266.08

B_1 =

0	0
0	0
592.57	592.57
-221.85	-221.85

C_1 =

1	0	0	0

P = 1.0e+11 *

0	0	0	0	-0	-0	0.0059	0.0059
0	0	-0	-0	0	0	-0.0022	-0.0022
0	0	-0	-0	0.0059	0.0059	-5.86	-5.86
-0	-0	0	0	-0.0022	-0.0022	2.18	2.18

Rank(P) = n = 4

This confirms the system is controllable. But we are also interested in the controllability of the system for servo mechanism purposes. Thus we want to assess the controllability of the following pair:

$$\left(\begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}, \begin{bmatrix} B \\ 0 \end{bmatrix}\right)$$

For controller design purpose matrix B can be converted into a column vector as both columns of the matrix are similar and outputs for both θ_l, θ_r are the same. For above pair

P = 1.0e+14 *

0	0	-0	0	-0.0059
0	-0	0	-0	0.0022
0	-0	0	-0.0059	5.83
-0	0	-0	0.0022	-2.17
0	0	0	0	-0

Rank(P) = n = 4

This shows that a servomechanism design for the system is not controllable and the stabilizability of the system will need to be investigated, which will be discussed in later sections.

2.4 Observability

The behavior of the entire state space model can be observed by using the system's output. Observability and controllability of the system are mathematical duals, however, a system can be entirely controllable but only be observable for certain outputs and not others. A system's said to be observable if the row rank of the observability matrix is equal to the number of rows of the nonsingular A matrix of the state space model.

$$Q = \begin{bmatrix} C \\ CA_1 \\ CA_1^2 \\ CA_1^3 \end{bmatrix}$$

For output as θ_1 in x_1,

Q = 1.0e+05 *

0	0	0	0
0	0	0	-
0	-0.0038	-0.0073	0.0071
0	4.2518	7.2745	-7.0899

Rank(Q) = 4, meaning that system is observable. For any other variable in x_1 selected as output, the rank of Q matrix comes as 3 meaning that we can build observers for the system by just using the theta as output, although this may prove useful in some situations, for a robotic system could be problematic as the unstable system could suffer from degradation very quickly, further as the system has been linearized for study in simulation, the model only holds true for assumed condition. To combat any unknown scenarios, and given the limited type of variables needed to define the system, the system is assumed to have sensors for all state variables. Further details on how this can be accomplished are given in the System Implementation section.

2.5 Stability

The stability of the system was determined by using the poles of the transfer function. The transfer function of the state space model is:

$$H(s) = \frac{168.2s^2 - 1.434e^{-11}s - 1.04e^4}{s^4 + 329.6s^3 + 245.5s^2 - 1.303e^4 s}$$

The poles found for the transfer function are:

$$s_1 = 0 \quad s_2 = -328.7531 \quad s_3 = -6.7445 \quad s_4 = 5.8771$$

From the poles, it can be determined that the system is unstable ($s_4 = 5.8771$).

2.6 Controller

The system to be controlled has multi-input multi-output aspects, because of which it is imperative to outline the state variable of interest to be controlled. Among the six state variables we are mainly in two

$$\theta, culmulative\ wheel\ angle$$

$$\phi, body\ yaw\ angle$$

Two variables which represent the velocities of the above two components will not be controlled directly as following the ideology of an actual system, the differentiated values will be highly error prone and would contribute towards the instability of the system thus it pays off to minimize their contribution towards stability.

The last two variables deal with $\psi\ (body\ pitch\ angle)$ and its differential, but we are interested in converging the value to zero regardless of the condition of other state variables.

The above requirements outline the need for a servo controller which acts as an inner loop state space feedback controller responsible for driving the reference signal equivalent to $[\theta_{ref}, 0, 0, 0]^T$.

While an outer loop ties together the ϕ through differential feed to the two wheels. The reason for excluding ϕ from state space equation is to simplify the requirements from the state controller. The state controller is primarily used for pitch control, whereas including ϕ to it can result in the voltages to the motors oversaturating and resulting in instability.

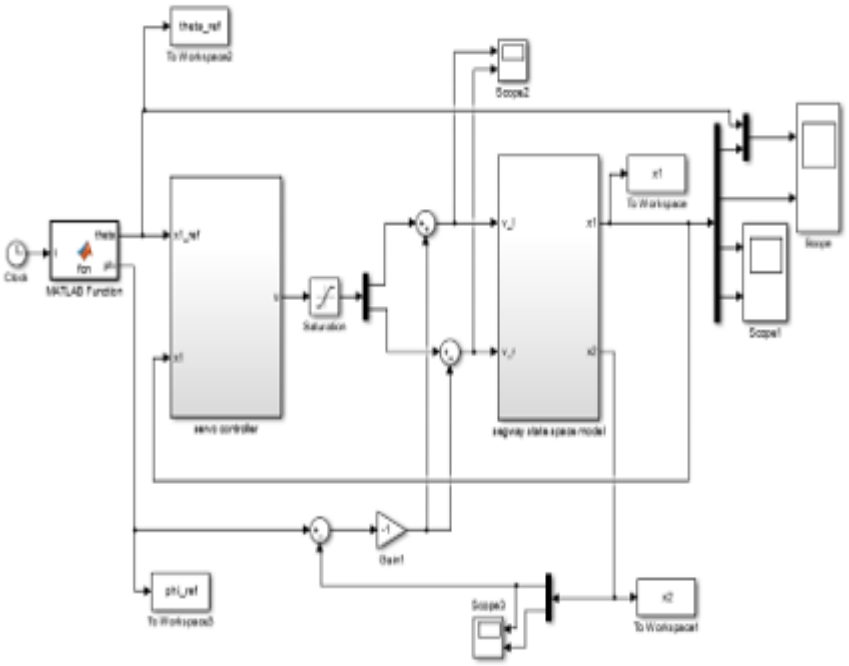

Figure 3: Simulink Model

As seen in the Simulink model above, the servo controller produces the required signals for the motors, which are fed through a differential controlling the feedback from ϕ loop. The feedback from ϕ loop is maintained by a gain controller to have insignificant effect on the stability of the system; the gain value is selected through trial and error.

2.6.1 Segway State Space Model

The Segway state space model is a numerical representation of the mathematical state space equations discussed above. The two major blocks represent the computation of $\overrightarrow{x_1}$ and$\overrightarrow{x_2}$. Initial conditions are provided but mostly set to vectors of zeros to denote a standing Segway. Numerical integration of state vector velocity is used to feed the system, as such, using variable step capability and proper solver in Simulink is important to achieve reasonable results.

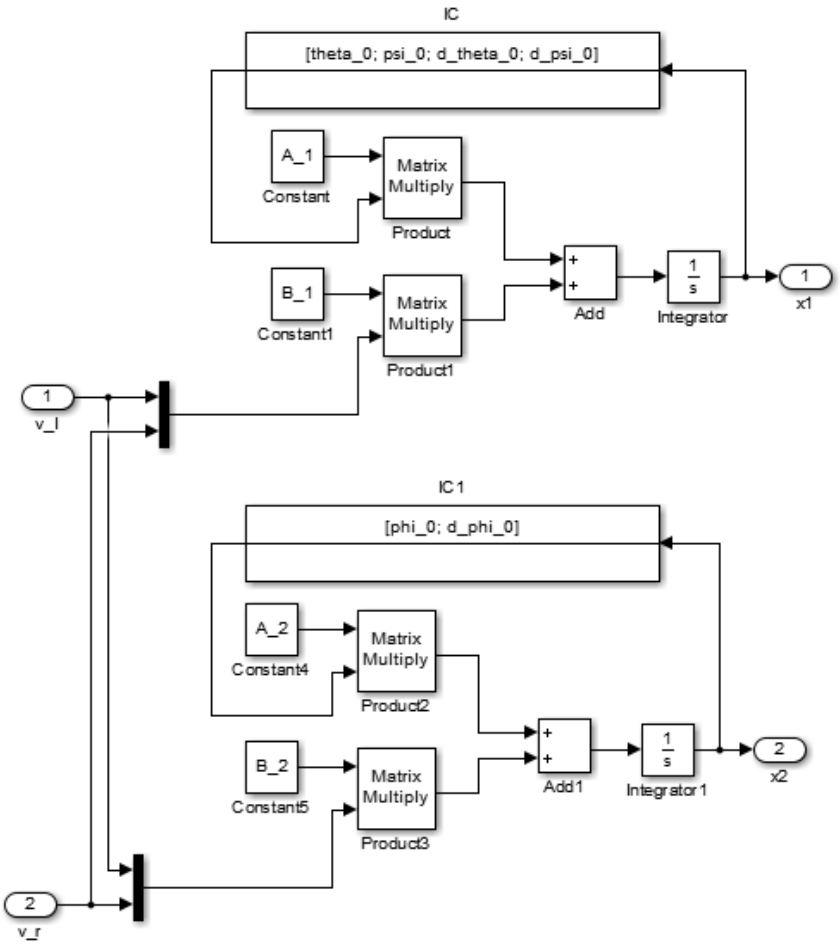

Figure 4: State Space Model

The state space model above is fully replaceable by hardware elements on the Segway, where data from motor encoders and gyroscope pitch form $\overrightarrow{x_1}$, while gyroscope yaw form$\overrightarrow{x_2}$. Further, as the gyroscope is synchronized with an accelerometer and magnetometer using a complementary filter, it reduces the need of an observer for sanity checking.

2.6.2 Servo Controller

The design of the servo controller is borrowed directly from [1] with certain changes to the reference input. The servo controller is a state space feedback controller with the dominant gain acting on the vehicle pitch. As such, the integrator loop contributes towards making the θ controllable but adds the possibility of instability. The servo controller is augments the defining matrices as follows:

$$\begin{bmatrix} A - BK & Bk_1 \\ -C & 0 \end{bmatrix}$$

As discussed in previous sections, the new matrix formed by the control law is not controllable, thus we use the Linear-Quadratic Regulator designer in the MATLAB Control toolbox. The lqr function [2] is fed the system and the weights to influence the augmented matrix defined under the servo controller. Following the strategy from [1] the gain matrix is defined as follows:

Q =

0	0	0	0	0
0	600000	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	400

As it can be seen, the heaviest emphasis is to control the body pitch, while the second largest weight is being used for the integral part of the servo controller. The resulting gain matrix is defined as follows:

K =

-0.98	-28.45	-0.84	-2.28	0.45
-0.98	-28.45	-0.84	-2.28	0.45

The two rows in the gain matrix are similar because of the nature of B matrix of the system. The resulted augmented A matrix for the system is as follows:

A = 1.0e+04 *

0	0	0.0001	0	0
0	0	0	0.0001	0
0.12	3.33	0.0264	0.3414	-0.053
-0.04	-1.24	-0.0101	-0.1278	0.0198
-0.0001	0	0	0	0

With eigenvalues:

Eig(A) = 1.0e+02 *

-9.9623 + 0000i	-0.1115 + 0000i	-0.0052 + 0.0052i	-0.0052 – 0.0052i	-0.0537 + 0.0000i

As it can be seen above, all the eigenvalues of the augmented matrix lie in the negative real plane thus the system has been stabilized. The weights supplied to lqr function can be changed to impact the location of dominant poles to achieve a faster or sluggish system as per need.

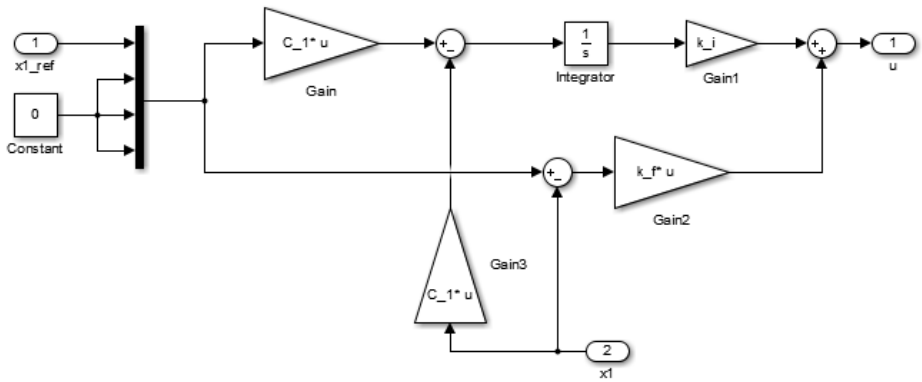

Figure 5: Servo Controller

2.7 Analysis

The controller discussed above was supplied with various reference signals and the outputs were observed. The objective was to confirm that the system remains reasonably within the assumptions of the linearized system. As no collisions detection was implemented in the system, any fall of the Segway was characterized by exponential instability of any output. Any such behavior led to outright rejection of the results of the model and further tweaking was done to clamp the outputs within reasonable bounds. All the systems are fed with signal of different levels or behavior but each one evokes a different response at specific time intervals, which can be summarized as such:

Table 3:Reference signals for θ and ϕ at different time intervals

Time [seconds]	$\boldsymbol{\theta_{ref}}$ [radians]	$\boldsymbol{\phi_{ref}}$ [radians]
0 - 5	Step/Move Forward	0
5 - 10	Hold position	Step to π
10 - 15	Step/Move Backward	π
15 - ∞	Hold position	π

2.7.1 Small Step Input

Table 4: Small step reference signal

Time [seconds]	$\boldsymbol{\theta_{ref}}$ [radians]	$\boldsymbol{\phi_{ref}}$ [radians]
0 - 5	Step to 5	0
5 - 10	5	Step to π
10 - 15	Step to 0	π
15 - ∞	0	π

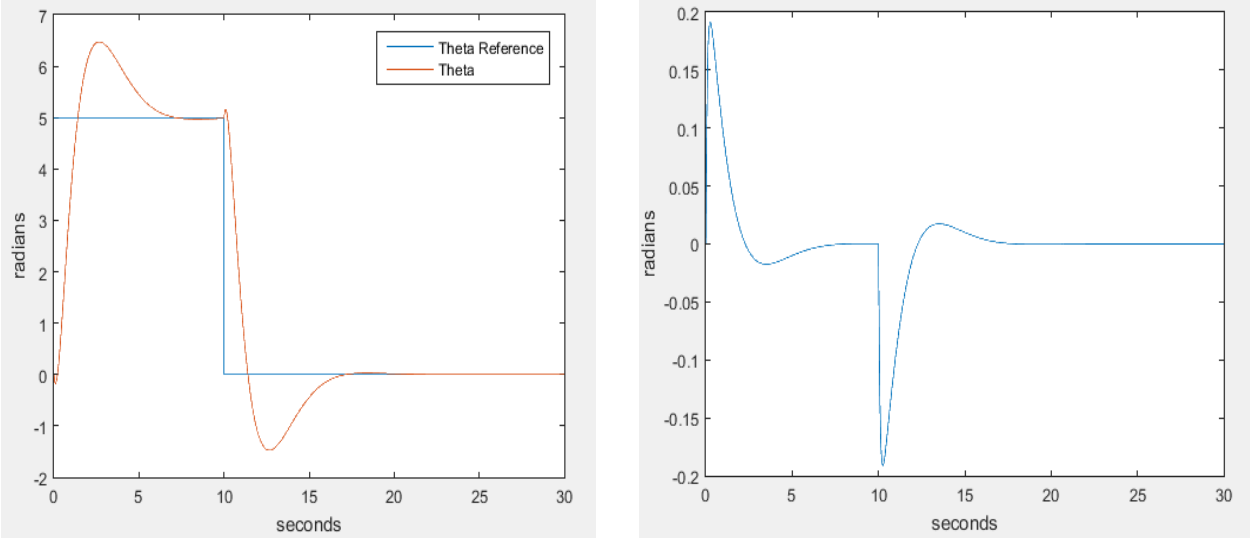

Figure 6: Output showing θ on left and ψ on right for small step input

In the plot above, it can be seen that the system suffers from overshoot which is direct consequence of the integrator loop being used for θ, but the benefit is seen in the next plot which shows that the highly stable with respect to the pitch angle.

The yaw of the system suffers no adverse effects and contribute little towards the saturation of the actuators, while the response time of the system with respect to yaw is slow, it is kept so as to avoid making the actuators saturated. As the simulated system does not suffer from miss-alignment between the wheels no correction factor is needed, but experimental system would need a gain factor comparing the differential between the wheels and synchronizing them while moving.

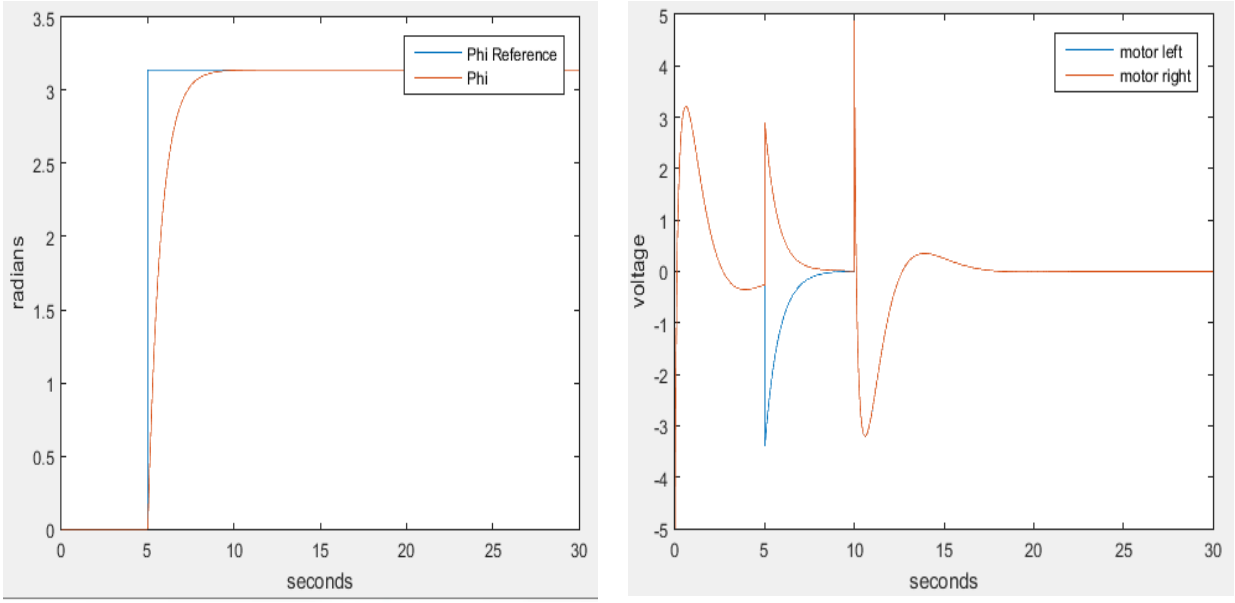

Figure 7: Output showing ϕ on left and voltage input to motors on right for small step input

Comparing the data to the requirements:

Table 5: Comparison (Results for small step input vs Requirements)

Variable	Requirement	Value
ψ	$\lvert range \rvert \leq 15\ degrees$	$\lvert range \rvert = 11.45\ degrees$
θ	$wheel\ radius * \lvert overshoot \rvert \leq\ 5\ cm$	$wheel\ radius * \lvert overshoot \rvert = 4.5\ cm$
$\dot{\theta}$	$wheel\ radius * average(\dot{\theta}) \geq 2cm/s$	$wheel\ radius * average(\dot{\theta}) \approx 9\ cm/s$
ϕ	$\lvert overshoot \rvert \leq 10\ degrees$	$\lvert overshoot \rvert \approx 0\ degrees$
$\dot{\phi}$	$average(\dot{\phi}) \geq 45\ degrees/s$	$average(\dot{\phi}) \approx 45\ degrees/s$

Although the results from the small step input look reasonable, the large overshoot contributed by the integrator on θ illuminates the possibility of instability which is analyzed in the next section.

2.7.2 Large Step Input

Table 6: Large step reference signal

Time [seconds]	θ_{ref} [radians]	ϕ_{ref} [radians]
0 - 5	Step to 10	0
5 - 10	10	Step to π
10 - 15	Step to 0	π
15 - ∞	0	π

As it can be seen in the plots below, θ and ψ go out of bounds towards infinity indicating the Segway has fallen, for the integrator loop for θ information can be obtained by looking at the actuator signals.

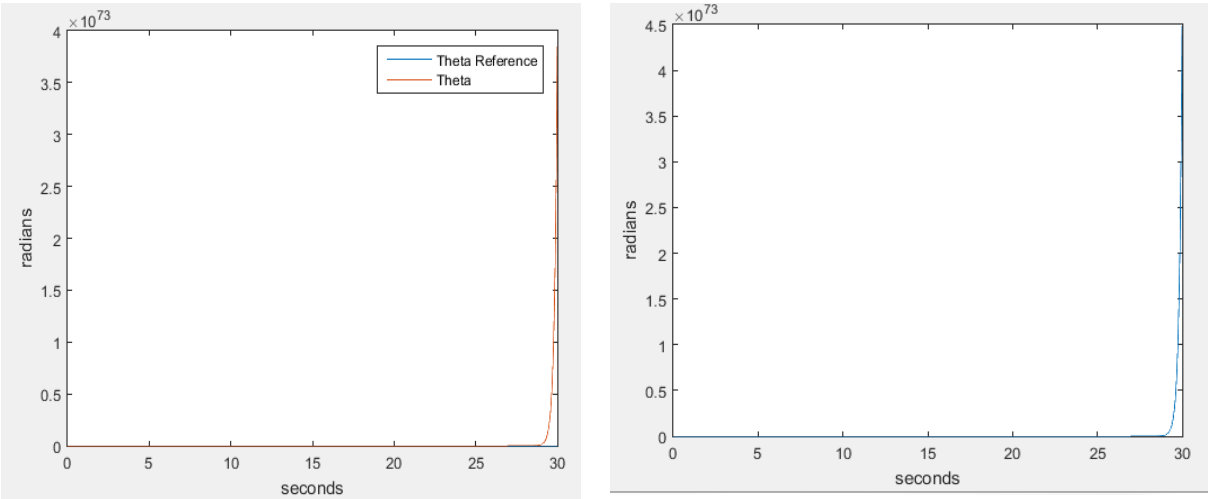

Figure 8: Plots showing results for θ on left and ψ on right for large step input

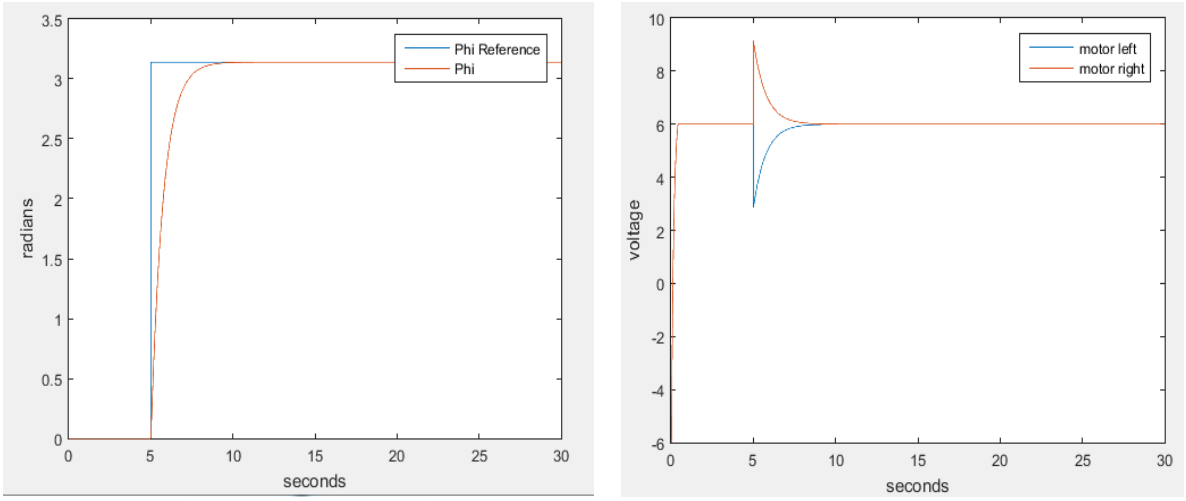

Figure 9: Plots showing results for ϕ and Actuator inputs for large step input

As it can be seen from the actuator signals, both the motors are saturated and hence cannot counteract the effect of initial dominance of θ on the input signal. The spikes seen in the middle are the effect of the ϕ feedback loop but are unreasonable as well as the system reached saturation long ago.

2.7.3 Ramp Input

To make the system reasonably stable for large movements, we have to realize that the integrator terms leads to saturation of actuators, the effect can be minimized by generating reference signal as ramps rather than steps. As such, θ error would stay minimal while effect from ψ stays dominant leading to system stability.

Table 7: Ramp Reference Signal

Time [seconds]	**θ_{ref} [radians]**	**ϕ_{ref} [radians]**
0 - 5	Ramp to 10 with slope of 2	0
5 - 10	10	Step to π
10 - 15	Ramp to 0 with slope of -2	π
15 - ∞	0	π

As it can be in the θ and ψ signals below that the system stays stable with the new reference signal. The ϕ behaves similar to the small step signal input and is not impacted by the behavior of pitch and forward and backward movement.

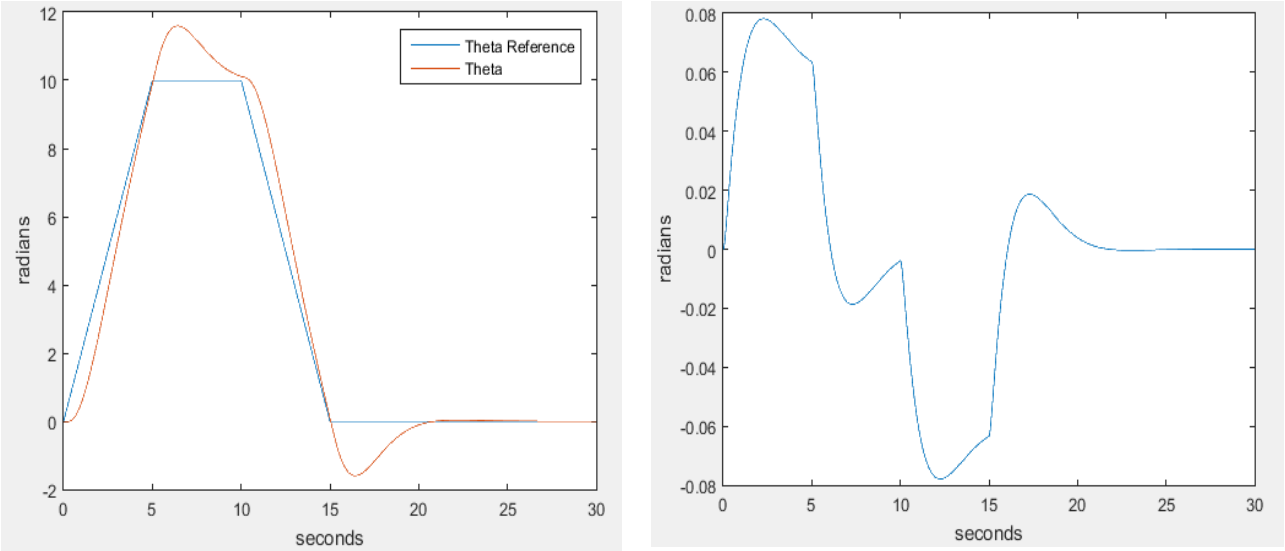

Figure 10: Plots showing results for θ on left and ψ on right with new reference signal

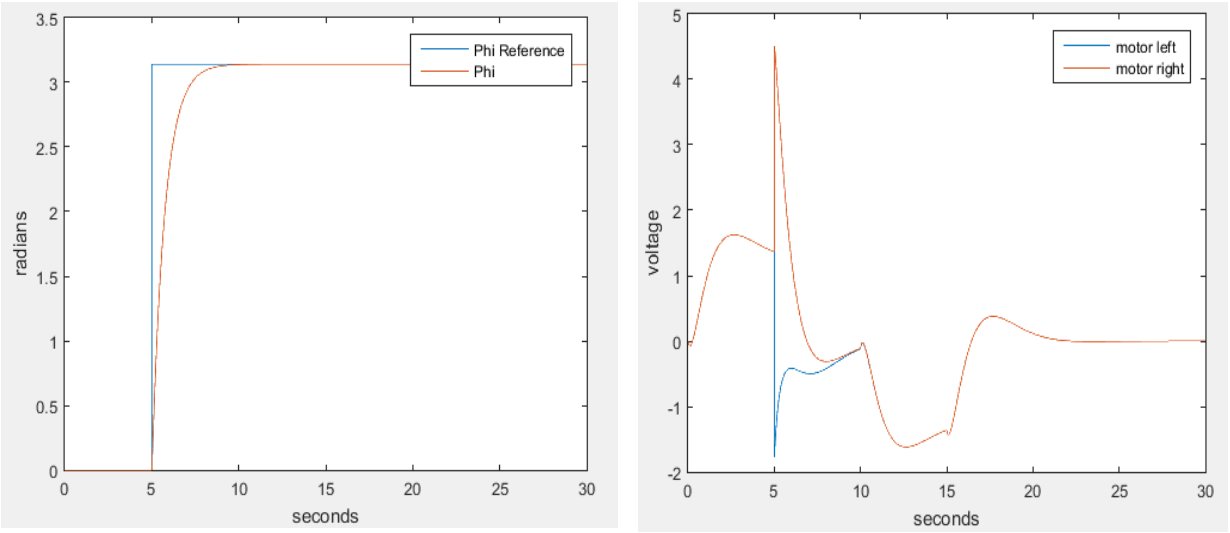

Figure 11: Plots showing results for ϕ on left and actuator inputs with new reference signal

The results from the above simulation are shown in the table below and compared against the earlier requirements.

Table 8: Comparison between Simulation Result Vs Requirements

Variable	Requirement	Value
ψ	$\lvert range \rvert \leq 15\ degrees$	$\lvert range \rvert = 4.58\ degrees$
θ	$wheel\ radius * \lvert overshoot \rvert \leq\ 5\ cm$	$wheel\ radius * \lvert overshoot \rvert = 4.5\ cm$
$\dot{\theta}$	$wheel\ radius * average(\dot{\theta}) \geq 2cm/s$	$wheel\ radius * average(\dot{\theta}) \approx 6\ cm/s$
ϕ	$\lvert overshoot \rvert \leq 10\ degrees$	$\lvert overshoot \rvert \approx 0\ degrees$
$\dot{\phi}$	$average(\dot{\phi}) \geq 45\ degrees/s$	$average(\dot{\phi}) \approx 45\ degrees/s$

By comparing the values in above table to the ones obtained for small step input, it is seen that the ramp input increases the controllability of body pitch considerably even when the distance traversed is much larger. Second change is to the speed which can be seen has been reduced. The reference signal is able to control the contributing factor to the instability of the system earlier, the second benefit is that the controller can manage the slope of the supplied reference signal to control the system, this provides the added benefit of selection between slow and controlled pace or even finding the speed after which the system will become unstable. For the above simulation the value was found to be closer to 2 radians/second. The other significant effect of changing the signal to a ramp is seen in the actuators signals. As it can be seen that the actuator signals do not saturate.

Although the impact of the ϕ is significant and has to considered during the coding of the experimental system. Software checks can make sure that the feedback from the ϕ does not saturate the motor signals and is clamped down if so.

3 System Implementation

The students attempted to take the project from simulation to implementation using a prototype robot. The robot was constructed using rudimentary parts available. An onboard microcontroller acted as the brain, sensors such as incremental encoders, gyroscope acted as the sensory backbone, and a battery pack provided modularity. Software was developed to process the data from the gyroscope and other sensors to eliminate the need of observers, and to implement a faithful representation of controller developed in simulation earlier. The challenge taken by the students was in addition to the requirements of the project, further limitation with time meant that while a hardware prototype and complete software backbone was developed, work had to be halted during the fine-tuning process of the controller from taking it from a simulation to real world.

3.1 Hardware and Electronics

The uniqueness of the inverted pendulum has drawn interest for many researchers due to the unstable nature of the system. We decided to build a two wheeled inverted pendulum model, using the following mechanical components:

- Arduino board (Microcontroller)
- Pololu DC motors
- Battery
- Two wheels
- Piece of cardboard
- And a Styrofoam piece for safety

As the model is mechanically unstable, it becomes necessary to implement a control system to keep the system in equilibrium. As the robot will be moving about on a surface, a PID controller is implemented to control the trajectory of the robot. A gyroscope is used to measure the tilt of the robot and the encoders on the motors, to measure wheel's rotation. A linear state space controller utilizing sensory information from a gyroscope and motor encoders is used to stabilize this system.

The robot's body was designed based on a piece of cardboard attaching Arduino and the circuit chip on one side and battery pack at the back. The motors were fitted on the same side as the Arduino board and the circuit chip, using a micro metal gearmotor mounting bracket to avoid misalignment of the motors.

Trajectory Control

Differential drive robots have proven to be one of the least complicated locomotion systems. The differential scheme consists of two wheels on a common axis, each wheel driving independently. This arrangement gives the ability to drive straight by applying equal amount of power to the motors, but if the system is ideal. Otherwise, these robots cannot drive straight without a proper control system because of the errors like difference in wheel sizes and friction, rough paths; these errors will cause the robot to lose its balance.

Majority of the hardware components were resourced by the students themselves, due to limitation of time, minimum effort was invested to produce the prototype. Previous experience with various components was critical in reducing the time needed to interface the parts on hardware as well as software level.

1. Microcontroller (web link)

Specification	Value
Memory	512 KB
Clock Speed	84 MHz
PWM Outputs	12
Digital Inputs	42
Weight	36 grams
Input Voltage	7-12 V

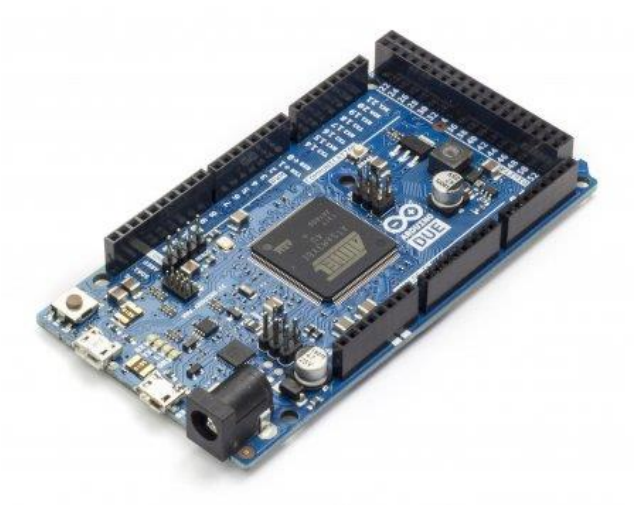

Figure 12: Microcontroller

Actuators:

Figure 13: Pololu DC Motors

The gearmotor is a miniature high-power brushed DC motor with a 297.92:1 metal gearbox. It has a cross section of 10 x 12mm, and the D-shaped gearbox output shaft is 9mm long and 3mm in diameter. These tiny brushed DC geramotors are intended for use at 6 V, having the capability of running at voltages above and below this nominal voltage, so they can comfortable operate in the 3 – 9 V range (rotation can start at voltages as low at 0.5 V.

$$Gear\ ratio: \frac{25 \times 34 \times 37 \times 35 \times 38}{12 \times 9 \times 10 \times 13 \times 10} \approx 297.92:1$$

2. Motors (web link)

Specification	Value
Free Run Speed	100 RPM
Stall Torque	0.4 Nm
Stall Current	1.6 A
Encoder Pulses	3576 per rev
Input Voltage	3-9 V

Figure 14: DC Motor with Magnetic Encoder

3. Magnetic Encoder

The encoders we used, uses a magnetic disc and hall effect sensors to provide 12 count per revolution of the motor shaft. The sensors operate from 2.7 V to 18 V and provide digital outputs that can be connected directly to a microcontroller or other digital circuit.

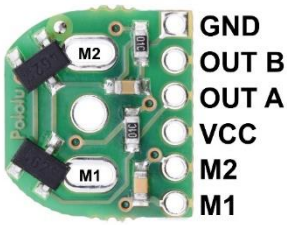

Figure 15: Magnetic Encoder

The sensors are powered through the V_{cc} and GND pins. V_{cc} can be 2.7 V to 18 V, and the quadrature outputs A and B are digital signals that are either driven low (0 V) by the sensors or pulled to V_{cc} through the 10 kΩ pull-up resistors, depending on the applied magnetic field. The sensors' comparators have built-in hysteresis, which prevents spurious signals in cases where the motor stops near a transition point.

4. Micro metal gearmotor Mounting bracket

Figure 16: Micro metal gearbox mounting bracket

Mounting bracket used in our design are specifically designed to securely mount the gearmotor while enclosing the exposed gears.

5. Inertial Measurement Unit (web link)

Specification	Value
Gyroscope	3 axis
Accelerometer	3 axis
Magnetometer	3 axis
Input Voltage	2.5 – 5.5 V

Figure 17: Inertial Measurement Unit

6. Motor Driver (web link)

Specification	Value
Motors Supported	2
Average Current	1.2 A per channel
Peak Current	2 A per channel
Input Voltage	2.7 – 10.2 V

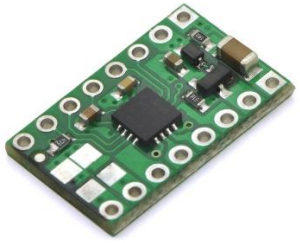

Figure 18: Motor Driver

7. Battery (web link)

Specification	Value
Nominal Voltage	7.4 V
Capacity	1000 mAh
Weight	85 grams

Figure 19: Battery

The electronic components were permanently connected together on a perforated circuit board to eliminate broken connection which could otherwise result from sudden motion or falls. The parts were secured to a compressed cardboard chassis using brackets and zip ties to realize a cost effective chassis. The electronics backbone and competed robot can be seen in the following pictures.

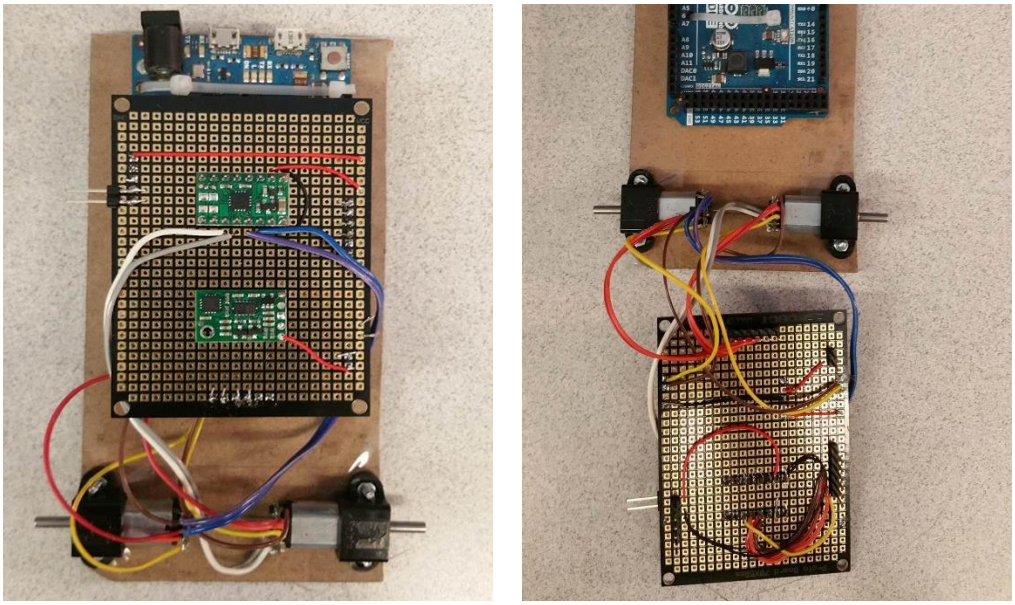

Figure 20: Physical Model Design and Components used

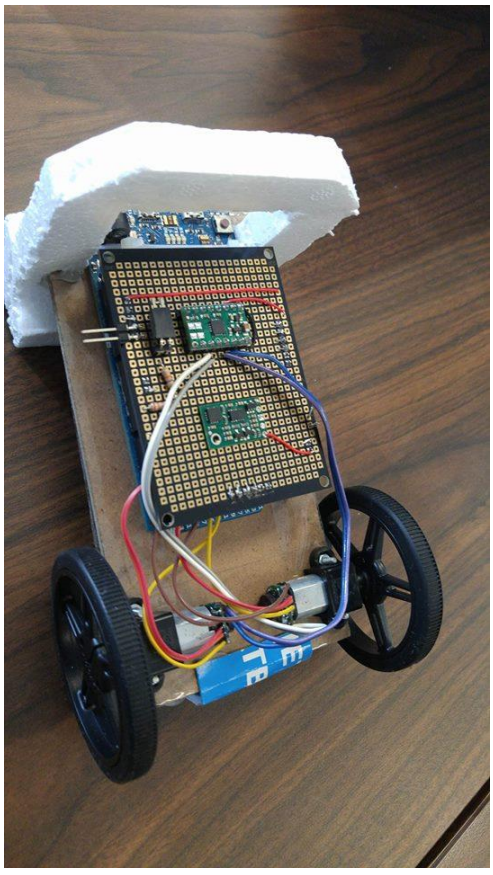

Figure 21 Assembled Protoype

3.2 Software

The embedded software was programmed using Arduino framework to aid effortless development, but still required significant patching and development for the various components. The system was developed as a state machine managed using asynchronous software timers. Only the major sections of the code and their functionality is discussed here:

1. **Sensors**

An onboard 9 degrees of freedom sensors provided body pitch and yaw information. A complimentary filter [3] is used to fuse the data from the gyroscope with data from accelerometer and magnetometer. As such, the robot is sensitive to sudden movements, while not suffering from drifting problems seen in MEMES gyroscopes. Other sensors included incremental encoders for the motors, which were serviced using digital pin interrupts. In future, the robot could be mounted with a proximity sensor to provide it with depth perception.

2. **Motors**

A software section interacts directly with the motor driver board using four PWM channels, the sequence and the on state of the line dictate the voltage supplied to the two motors.

3. **Controller**

The controller was coded so as to faithfully mirror the one designed in simulation but with additional checks in place which monitor the motors for oversaturation and provide compensation by measuring the battery voltage, which drops over the runtime of robot, to provide equal performance throughout the run of the robot.

4. **Utilities**

Various software utilities were coded, such as software timers, UART logging, etc, to facilitate easier development and debugging of problems. The system lacked any remote debugging capability as the students did not have a wireless module at hand, but it could be easily achieved by using modules such as UART-to-Bluetooth chipsets.

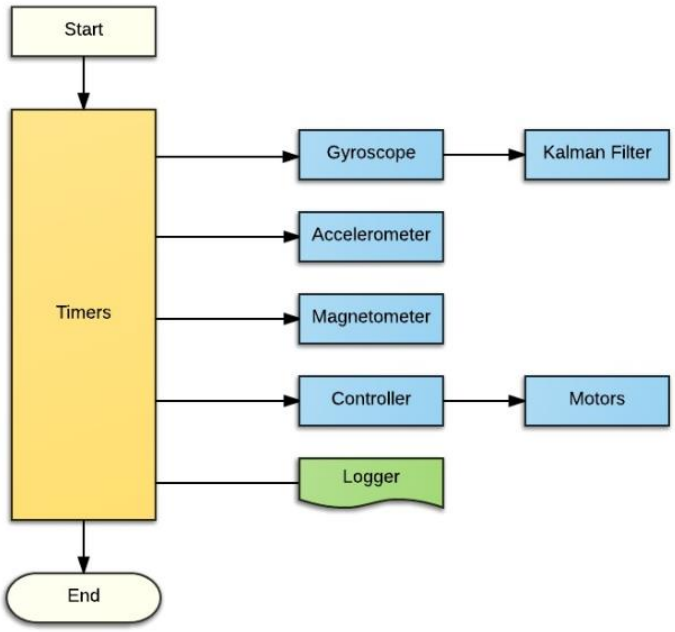

Figure 22: Flowchart detailing Software (Overview of Call Hierarchy)

The timer's library manages all the asynchronous timers, where all the underlying components are running at various frequencies. The Kalman filter used for computing attitude information is attached to timer managing data read for the gyroscope. While the motors are set at the very moment the controller is run. At latest, the inertial measurement unit was serviced at 95 Hz, while the controller ran at 100 Hz. Due to the microcontroller used, the execution speed could be much faster but was limited largely due to the slow nature of the I2C protocol used with the various sensors comprising the IMU.

4 Conclusion

The implementation of the balance control algorithm was based on the detailed theory and simulation as discussed above. Controller gains were obtained from simulation results and applied to the system. Since the dynamic model of a system can never be accurate, the gains found usually acts as foundation for further fine tuning of the controller to achieve the desired response.

This project was on the verge of being successful but unfortunately our main programming laptop's hard drive got burnt and we lost all data and therefore, we couldn't do some final touch ups. But still I would put it as our project was successful in achieving its aims to balance a two-wheeled robot based on the inverted pendulum model.

5 References

[1] Y. Yamamoto, "NXTWay," [Online]. Available: http://www.pages.drexel.edu/~dml46/Tutorials/BalancingBot/files/NXTway-GS%20Model-Based_Design.pdf.

[2] mathworks, "lqr," [Online]. Available: http://www.mathworks.com/help/control/ref/lqr.html.

[3] T. Corinne, "razor IMU," [Online]. Available: https://github.com/sparkfun/9DOF_Razor_IMU.