ATTITUDE CONTROL USING REACTION WHEELS

MAJOR PROJECT REPORT

Submitted in partial fulfilment of the requirements for the course

Major Project II (ME499)

By

NITHISH K GNANI

HARI SHANKAR S

DENNIS JOSHY

B SURESH

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY
KARNATAKA, SURATHKAL, MANGALORE – 575025

26th April, 2017

ACKNOWLEDGEMENTS

The team would like to thank our project guide Professor Prasad Krishna, Department of Mechanical Engineering NITK, for all the guidance and mentorship provided during the course of the project. His valuable advice helped us formulate the problem statement and pointed us in the right direction. From the initial literature survey to the final design, fabrication testing and procurement of the necessary hardware, Professor Prasad Krishna has been a guiding hand for us at all times.

The team also thanks Professor K V Gangadharan, Head and Coordinator of Center for Systems Design and Dr K R Guruprasad, Assistant professor, Department of Mechanical Engineering for all their valuable inputs and the insights they imparted to us on how to approach the control engineering aspect of the problem statement.

The contributions of Ms Soumya, research scholar of the robotics laboratory, in advising us about the hardware, the intricacies related to the control algorithms and the associated literature is duly acknowledged. We also thank all the research scholars of the tribology laboratory for teaching us how to use their equipment, permitting our project to move forward. We also thank Mr Kathik Samthani for his contribution in tuning our control system.

ABSTRACT

Space systems involve multiple scenarios where system parameters will fluctuate in real time. The objective is to implement a simple adaptive control algorithm, MRAC using the MIT rule for yaw control of a suspended body using a reaction wheel and compare it with PID control. The direct implementation of this technique to such a system was not found in literature.

A realistic model of the reaction wheel system was designed using SolidWorks. The model was exported to SimMechanics. PID and MRAC were implemented in Simulink environment. A physical model, closely resembling the CAD model was fabricated using open source components. PID control was successfully implemented. Further, MRAC implemented in Simulink was wirelessly interfaced in real time to the physical model.

The results depict that the control system works almost as good as PID in terms of system response and simultaneously provides the much needed adaptive capability in such environments.

Keywords: *reaction wheel, PID, MRAC, yaw control*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The attitude control of a satellite is an important part of most missions in the space. Not only a suitable control methodology, but also appropriate actuators should be applied in order to help to achieve the goal of the mission and satisfy the constraints, e.g. solar radiation [K.D. Kumar, et al], reaction wheels [L.H. Geng, et al], magnetic torque rod [M. Lovera, et al], thrusters [M.J. Sidi, et al], control moment gyro [B. Bohari, el al], etc. To stabilize the system with uncertainties, different methods have been proposed over the years. It is a well-known fact that PID controllers have dominated industrial control applications even in aerospace engineering [K.D. Kumar, et al], although there has been considerable interest to research about the implementation of advanced controllers. They are straightforward to use, as almost everyone with some basic knowledge in control engineering may be able to employ it satisfactorily. The fixed gain PID controller cannot perfectly stabilize non- linear systems with uncertainties in terms of the model and parameters. To enhance the performance of the PID controller, a simple adaptive control algorithm, Model Reference Adaptive Control (MRAC) using the MIT rule for yaw control of a suspended body using a reaction wheel was implemented

1.1 Reaction Wheel

It is common knowledge that the tail rotor of a helicopter is used to stabilize the body which would otherwise spin out of control. The conservation of angular momentum requires that if the rotor rotates in one direction with a certain amount of angular momentum, the body should rotate in the opposite direction with the same angular momentum.

A reaction wheel functions on the same principle. An unstable system, like an inverted pendulum for instance, tends to fall under the torque exerted by gravity. If a reaction wheel is correctly positioned on the pendulum, the changes in the angular momentum of the reaction wheel will apply a counter-torque that causes the pendulum to rotate in the opposite direction. Using efficient control strategies, the inverted pendulum can be stabilized.

A reaction wheel, also called as a momentum wheel, is a type of flywheel used for orientation control of bodies. They are common in satellites. Orientation control is accomplished by equipping the body with an electric motor attached to a flywheel which, when its rotation speed is changed, causes the body to begin to counter-rotate proportionately through conservation of angular momentum [Muehlebach, et al, 2012][Mohanarajah, et al, 2012][Muehlebach, et al].

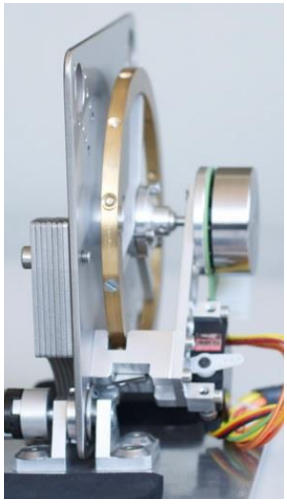


Figure 1.1 Reaction Wheel

There are several parameters to be kept in mind while designing a reaction wheel. Some of these include:

1. Moment of inertia
2. Radius
3. Geometry (number of spokes)

In usual practice, the reaction wheel assumes the shape of a flywheel to concentrate the mass at the rim. The Radius is an important factor when space becomes a constraint in small CubeSats [Nudehi, et al, 2008].

1.2 One dimensional inverted pendulum

The inverted pendulum project is a standard problem statement used to study new control algorithms, control mechanisms etc. For example, the cart model simulates a pendulum that balances itself using a sliding connector. Another technique is to use

gyroscopic moments to stabilize the pendulum. It is a simple platform to observe and verify all such techniques.

Using the concepts explained in the preceding section, the pendulum will be balanced using reaction wheels. The restoring moment of the reaction wheel on the motor will balance the unstable inverted pendulum.

2 SIMULATION OF 1D INVERTED PENDULUM

2.1 Assumptions

Some of the assumptions made for simulating the inverted pendulum include:

1. Rigid body components and state of pure bending
2. Idealized motor and gearbox
3. No air resistance
4. Linearization around 180 degrees
5. Neglect vibrations of the body

2.2 CAD model of the 1D inverted pendulum

A model of the pendulum was designed in SolidWorks. Each part of the model was designed and later assembled with the required constraints. The parts include base, revolute pin, pendulum body, motor and reaction wheel. Coincident and concentric constraints were applied between the base and the revolute pin which holds the pendulum body. The revolute pin, the pendulum body and the motor were rigidly constrained and hence function as a single unit. Coincident and concentric constraints were applied between the reaction wheel and motor. The CAD model of the system was exported using a second generation SimMechanics link – to generate an xml file.

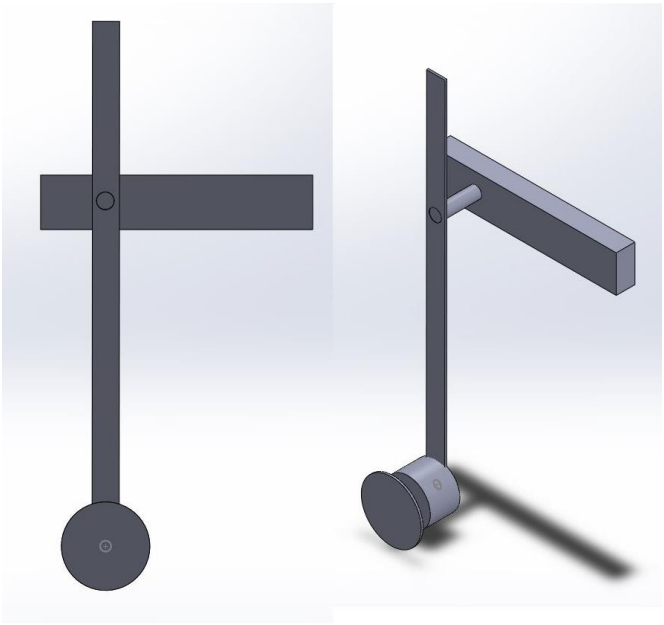


Figure 2.1 CAD model of the 1D inverted pendulum

2.3 1D inverted pendulum system in SimMechanics

On importing the model into Simulink, it made the necessary initializations such as the environment blocks. The motor and the gear box subsystems were obtained from an online source.

Second generation SimMechanics models are highly versatile. It permits assigning internal mechanics such as spring stiffness and damping parameters. It also has options for adding sensing ports and receiving signals as inputs.

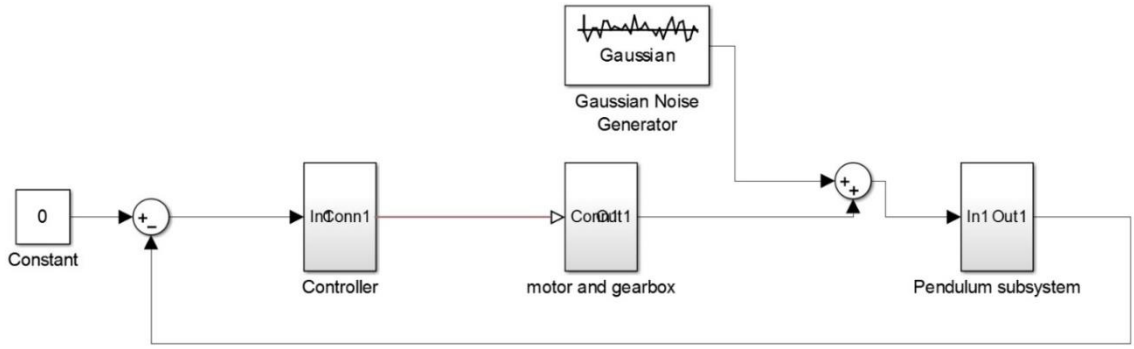


Figure 2.2 SimMechanics Block diagram of the 1D inverted pendulum system

2.4 Motor and Gearbox subsystem

Next, the motor and gearbox were modelled using an online source. The circuit uses a controlled voltage source, a dc motor coupled with a gearbox. Additional components representing inertia and friction were added to make the system more realistic. The motor had options to accept armature resistance, inductance, inertia and the damping. As the motor had not been characterized and this data was not available in the online catalogues, dummy values were used in place of the actual ones.

The blue circuit represents an electrical circuit. The DC motor block transforms the current to torque. The ideal torque sensor reads the torque value and displays it on a scope block.

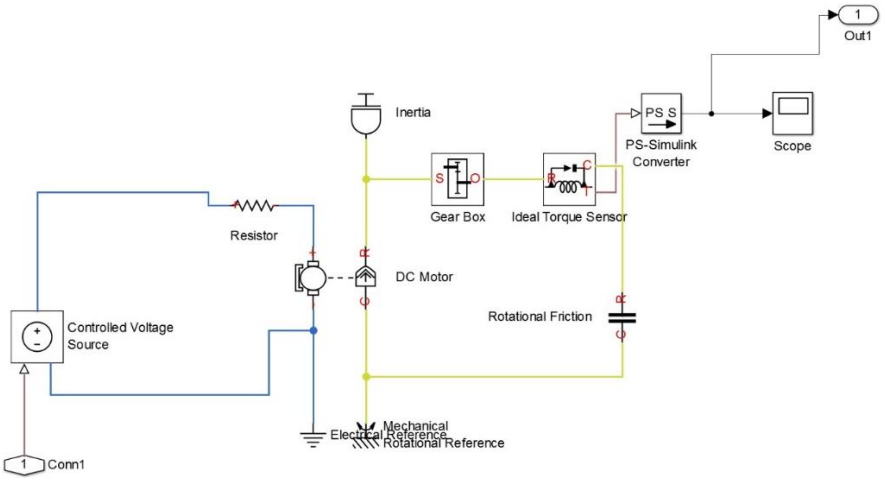


Figure 2.3 SimMechanics block diagram of motor and gear box subsystem

2.5 The pendulum subsystem

The torque from the motor is transmitted to the reaction wheel on top of the pendulum. The moment applied by the reaction wheel results in a case of pure bending. There are no shear forces generated along the length of the pendulum body. This means that the moment acting along the pendulum body is the same regardless of the position of the motor. The moment applied by the reaction wheel is sensed and transferred to the revolute joint at the base of the pendulum.

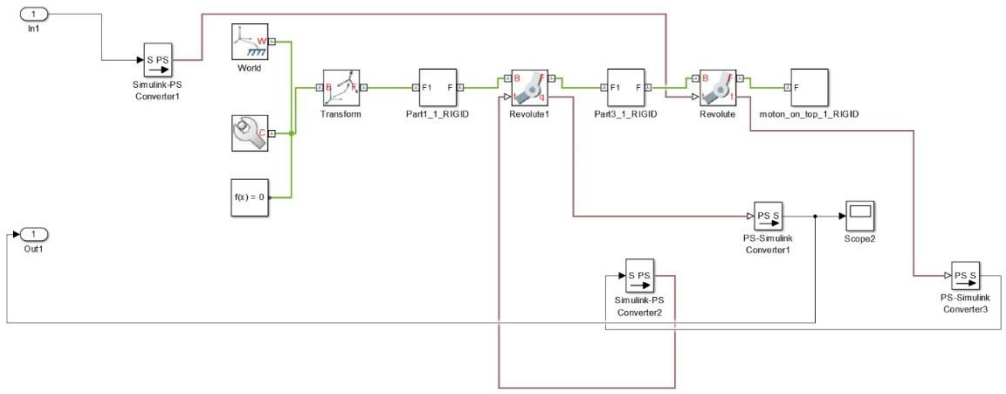


Figure 2.4 SimMechanics block diagram of Pendulum subsystem

2.6 The controller subsystem

The PID controller implemented was also reduced to a subsystem:

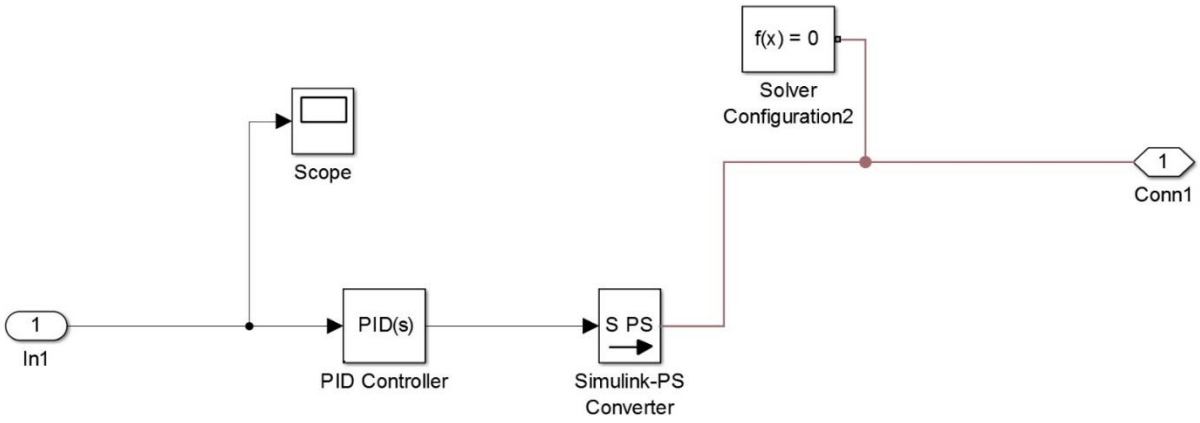


Figure 2.5 SimMechanics block diagram of Controller subsystem

Finally Gaussian noise of 0 mean and 0.001 variance was added to the motor input to factor in ambient and environmental disturbances that influences the inverted pendulum.

3 ONE DIMENSIONAL INVERTED PENDULUM

For slightly more than a century, inverted pendulum systems have been an indispensable part of the controls community. They have been widely used to test, demonstrate and benchmark new control concepts and theories.

Although most inverted pendulums are implemented with a mobile base (a cart and wheel setup), in our model, the pendulum is connected to a base fixed to the ground. It is balances itself using the torque provided by a reaction wheel calculated by a PID implementation on a micro-controller.

In order to demonstrate the ability of the reaction wheel for attitude control applications, the performance of the reaction wheel for an inverted pendulum fixed to the ground via a revolute joint, was studied. A physical model of the CAD model in preceding section was developed and the associated circuitry – including the microcontroller, motor driver, DC power system and the wiring was implemented.

The details of each component used for the assembly of the model are explained in the next few subsections. Although the paper suggests earlier that a PID control was used, this state was arrived at after testing with PD, PI alternatives. The pendulum was able to balance for a short period of time, but further tuning is required to increase its range of operations and stability.

Albeit, PID is the most popular control strategy that is used for CubeSat technology, other control strategies such LQR (Linear Quadratic Regulator) are also found to be effective as demonstrated by [Mohanarajah, et al, 2012] in project Cubli. The study of the reaction performance with more complicated control strategies will be an interesting area of research.

3.1 Structure of the pendulum

Figure 3.1 One dimensional inverted pendulum

The revolute joint of the pendulum was mounted on a sheet of acrylic. The Arduino board and the motor driver is screwed onto the holes made in the acrylic sheet. A roller bearing was press fit into the revolute joint pin of the pendulum. The diameter of the pipe was expanded by heating and the bearing was firmly fit into it. A circular clamp was out around the bearing to ensure that the arrangement remains fastened. A bolt was passed through the hole of the bearing and the end of the bolt was fastened onto the acrylic sheet.

The MPU6050 - six axis IMU (Inertial Measurement Unit) was stuck to the point of the revolute joint. The counter-weights were added to the lower half of the pendulum body and the motor fastened to the top of the upper half using zip ties and tape. The two halves were connected together using a T – junction. Finally, the reaction wheel was attached to the motor shaft.

Table 3.1 - Masses of pendulum components

Sl No.	Element	Mass (g)
1.	Upper half of pendulum body	60
2.	Lower half of pendulum body	50
3.	Motor	186
4.	Counter-weight	246
5.	Reaction wheel	40

Table 3.2 - Dimensions of Pendulum Components

Sl No.	Pendulum part	Dimensions (cm)
1	Upper half	52
2	Lower half	29
3	Pin of revolute joint	6

3.2 Circuitry

The Reaction wheel based inverted pendulum was built in various steps. Several different motors and reaction wheel setups were tried and tested. The image shown above is the first model of the inverted pendulum. The electronic components used were the open source Arduino Uno development board and the MPU6050 – three axis accelerometer and the three axis gyroscope. A simple ball bearing was used as the reaction wheel in this model. Later on, counter-weights were added (as opposed to positioning the Arduino board there as shown in the first model). The dc motor we tried was a BO – Battery Operated motor (found in toy cars and the like). An L293D motor driver was used to drive the motor. The PID control was implemented in the development board with appropriate thresholds and this calculated the required PWM(Pulse Width Modulation) values depending on the values received from the

motion processing unit. Although, the control system was working well in changing directions and trying to stabilize the pendulum: the torque generated was simply not enough to balance the pendulum.

So, a high torque motor that ran at nearly 21000 rpm at no load condition was acquired. However, the motor driver L293D could not handle the amounts of current drawn by this motor. Another option was to use BLDC (BrushLess Direct Control) motors. But we were unable to find to a driver that could handle the required current and at the same time switch directions quickly. The motor driver RKI-1341 was rated to run a peak current up to 20 A for up to one minute.

This was a cost-effective solution to the issue of finding the right motor controller. The motor driver and the motor circuit is powered by a 12 volt DC power supply. Using this motor driver, the motor was able to run without any problems.

Some of the highest values of current were noted during starting and direction reversal from the extreme speeds. The steady state current noticed was between 2.87A and 3 A.

Table 3.3 - Measured Values of Current During Motor Operation

No.	**Motor event**	**Current (A)**
1.	Start up	11
2.	Change of directions	15.5
3.	Steady State	3
4.	Brake condition	0.5

4 PROOF OF CONCEPT OF YAW CONTROL IN SUSPENDED BODY

Figure 4.1 Suspended body for yaw control

Satellites in space perform attitude control using reaction wheels. It uses one reaction wheel for each of the three axes. Unlike an inverted pendulum, this is a neutral equilibrium system.

The yaw motion of a satellite is simulated using a suspended body. A reaction wheel is used to control the yaw. There is a micro controller on board which implements PI control. It takes the yaw of the structure as an input from the magnetometer and the output of the controller goes to the motor driver which in turn drives the motor with reaction wheel attached to the shaft [Meyer, et al, 2009] [Ismail and Varatharajoo].

A Bluetooth module is used to receive information from a cell phone. The cell phone feeds the required reference orientation. The error is calculated as the difference between these two values and fed into the controller. By sending the yaw angle from the phone to the structure, the structure can mirror the smartphone's orientation. With proper tuning this structure can compensate for any external disturbances. If you offset the structure by any angle it will turn back to the reference angle.

The list of components involved in the construction of the model includes:

1. DC (Direct Current) 12-volt motor
2. Reaction wheel
3. L293D motor driver
4. Arduino Uno
5. HMC5883L – 3 axis digital magnetometer
6. HC05 – Bluetooth module
7. Acrylate boards and strings for fabrication
8. 1500mAh Lithium Polymer (Li-Po) 12-volt battery
9. Smartphone with magnetometer and Bluetooth, with Arduino 2.0 application installed

The various subsystems that went to making this model is as follows:

- Mechanical subsystem: The frame was fabricated from acrylic and wood. The above mentioned components were arranged so that the centre of gravity of the system was made close to the line of suspension.
- Electronics subsystem: The battery on board directly powers the L293D motor driver. The voltage regulator on the driver supplies power of 5 volts to the Arduino. The I^2C (Inter Integrated Circuit) communication protocol was used to read the sensor values from the magnetometer to the Arduino. The smartphone sent its orientation with respect to north to the paired Bluetooth module using the Arduino 2.0 application at a sampling time of 100ms. This data was received by the Bluetooth module and sent to the microcontroller using UART (Universal Asynchronous Receiver Transmitter) protocol (serial communication). A voltage divider circuit was used to convert the Bluetooth

logic level from 3.3 volts to 5 volts. PWM signal was sent to the motor driver to drive the motor.

- Control subsystem: PI controller was implemented on the microcontroller. The input to the control algorithm was in degrees of deviation from the set points. The controller output was given to the motor driver as PWM signal. The PI output was constrained to remain in the range from -255 to 255. The Integral term of the PI was also constrained to be in the range of -100 to 100.

5 COMPUTER AIDED MODEL AND DESIGN

The suspended body (satellite) is modelled and designed in SolidWorks 2013. The model was made using a combination of assemblies which in turn were made of subassemblies and parts. It was taken into account that the design could be easily fabricated using the materials and component that are easily available to the team. The parts used in the final assembly are given in the figure below.

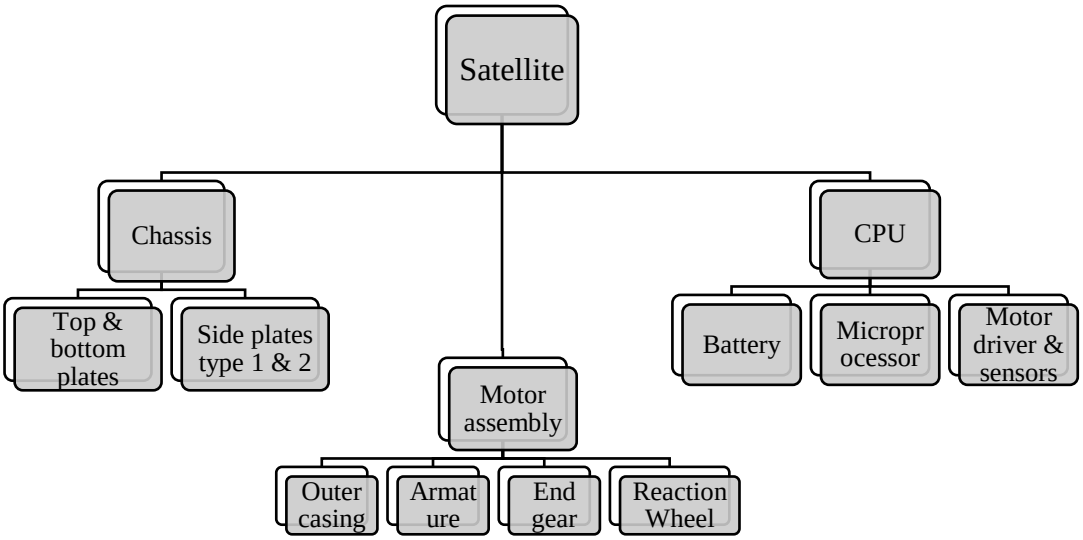


Figure 5.1 Components of the satellite

5.1 Design based on initial crude model

The model shown is the minimalistic one we had visualized initially for the demonstration of attitude control using reaction wheels. This was based on the crude suspended body model that we had built in the previous semester. SolidWorks 2013 was used.

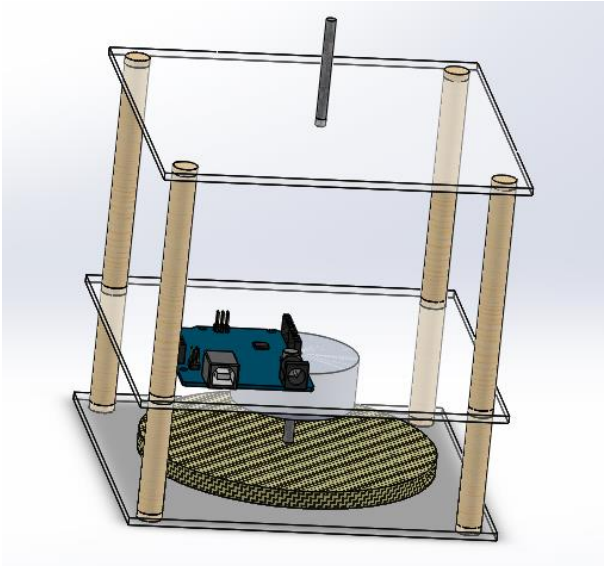


Figure 5.2 First model

5.2 New design

We then decided that the simplistic design was not going to be sufficient to house all the components that we planned to use for the model. Another model was designed in SolidWorks. All the components that we were going to use were included as parts and the final model was assembled as shown.

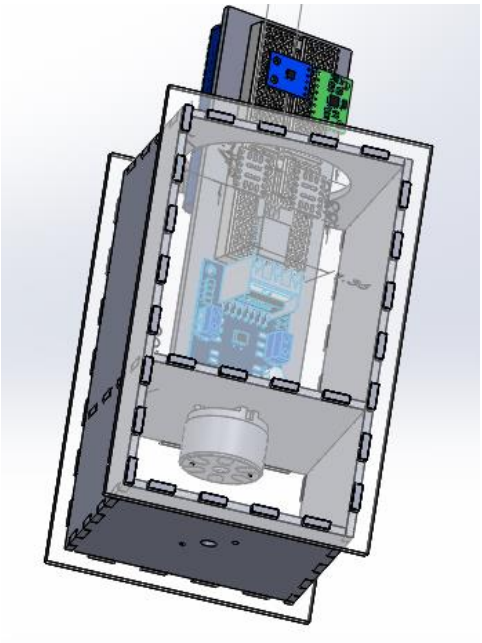


Figure 5.3 New model

5.3 Chassis

In the new model, interlocking type plates were designed for the chassis. This was done because it would be easier to disassemble and reassemble the interior as and when it was required. During the course of our work, the battery would need to be recharged, adjustments may need to be made to the motor or reaction wheel, connections may need to be changed, working code may need to be updated multiple times. These were the reasons the interlocking type plates were used.

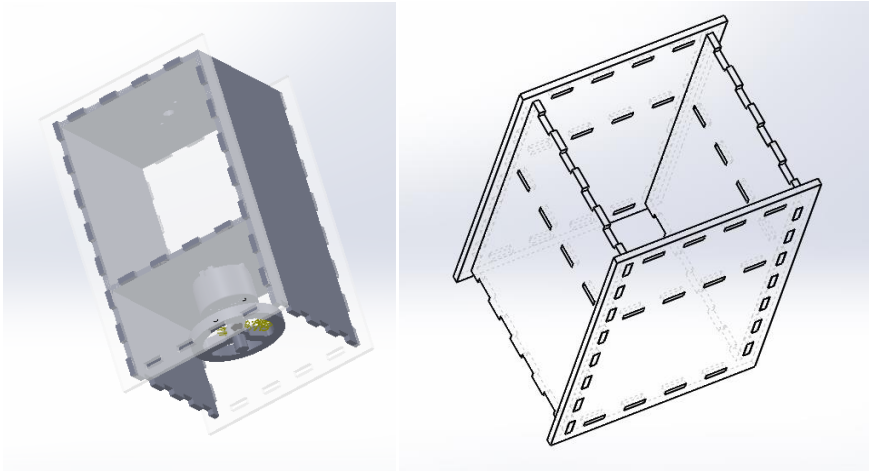


Figure 5.4 The Chassis

5.4 Motor assembly

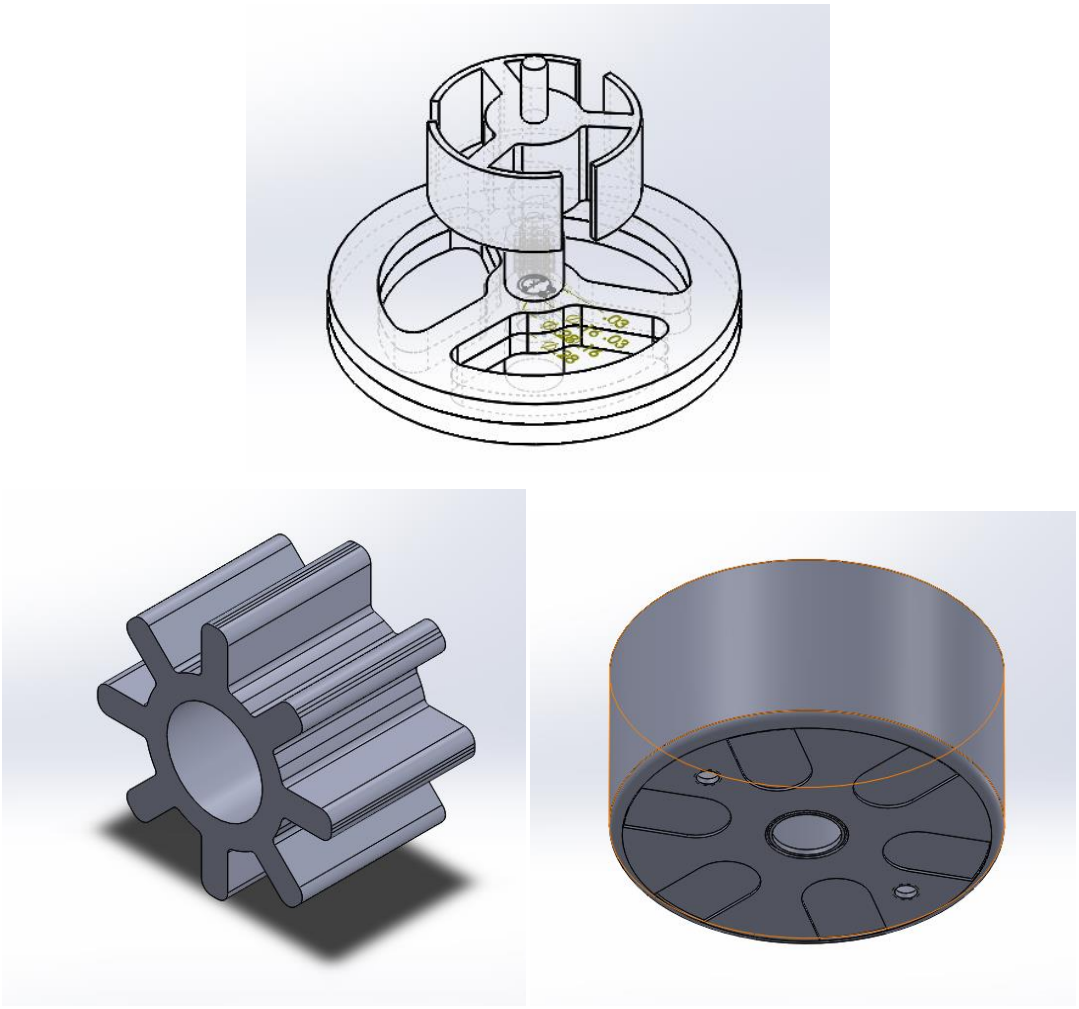


Figure 5.5 Components of the motor

5.5 CPU

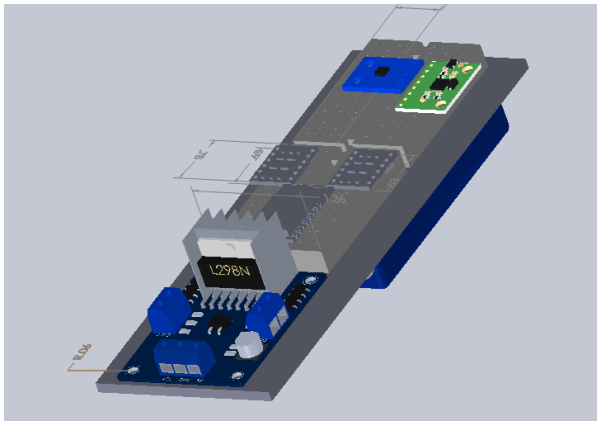


Figure 5.6 The CPU Arrangement

The battery, ESP, motor driver, the two sensors were all modelled individually if the original source was not available officially and were included in the CAD model. Various arrangements were tried such that the centre of gravity was coinciding with the geometric axis.

6 ADAPTIVE CONTROL

Fixed gain controllers are designed to function within a certain range of operations of the plant. The constants that appear in the control law are tuned manually depending on desired response. For instance, in a Proportional-Integral-Derivative (PID) control, the integral constant is tuned to minimize steady state error.

After world war two and at the advent of the space race, there was significant interest in the development of sturdy and reliable autopilot systems. This was particularly important in dynamical systems where the system parameters (especially mass) continuously changes such as a rocket, space-vehicles that need to eject components often or even damaged aircrafts.

Adaptive Control was designed with this goal in mind. [Link1] shows how a model remote controlled (RC) aircraft continues to maintain its trajectory by varying the output of its motor. The Model Reference Adaptive Control was one such control algorithm that was designed to accomplish this. The control law of this algorithm contains constants that are tuned automatically depending on the solution to an optimization problem []. A desired reference model is used and the output of the physical system is made to emulate the reference model`s response by tuning the gains in a manner described above.

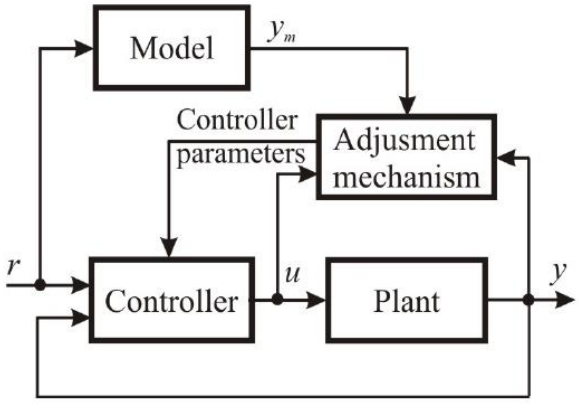


Figure 6.1 Block diagram of MRAC

Apart from autopilot control systems, Adaptive control is currently being applied to a wide spectrum of applications including automotive, aerospace, process industries, bio-

medical applications etc. [Kilic et al. 2016], explains the use of MRAC for the speed of induction motors, where a radial basis function neural. [Sami et al. 2016], presents the use of an L1 adaptive control design process for automatic tuning of control parameters for the desired performance and robustness. [Kyaw and Gavrilov 2016] describes a fault tolerant sliding mode attitude control for flexible space-crafts, while [Han et. al. 2015] explains the use of adaptive control for attitude control during actuator failure. [Harmonie et al. 2017] speaks of using robust adaptive attitude control during payload deployment for microsatellites. The Luzi adaptive control algorithm is also described in this work.

MRAC has been applied for unconventional applications like impedance control for human robot interaction [Bakur et. al. 2016]. Further, [Jaeyoung et al. 2017] emphasizes the applicability of MRAC in thermal management of automotive fuel cells. Vector controlled induction motor applications using MRAC with rotor flux and back EMF methods have explained in [Munshi and Choudhuri 2016]. Taking a step further, in [Wei and Wang 2015] the application of MRAC for fractional order linear systems have been explained. Another interesting study in 'road following applications' between vehicles have been studied in [Hassan and Sudhin 2014] using MRAC. MRAC is also finding application in cancer treatment as explained in [Salami and Salamci 2016].

6.1 Methods of implementation of Model Reference Adaptive Control (MRAC)

There are multiple ways to implement the Model Reference Adaptive Control. The most basic method being the MIT rule [Adrian et al. 2008] [Ranjan and Rai 2012] which was developed jointly by '' and '', who were both professors from the '' department of MIT in the 1960s. [Coman and Boldisor 2014 and 2013] demonstrates the application of this algorithm for accomplishing MRAC. Another interesting method for implementing this control is called design by Lyapunov approach. [Ampsefidis et al. 1993] and [Black et al. 2014] provide details on the design of MRAC depending upon a-priori information. A comparative study between design by MIT rule and Lyapunov approach is provided in [Pankaj et al. 2011] for the reader`s reference. The application of the Lyapunov approach for other adaptive control systems are discussed in [Misbawu et al. 2014].

6.1.1 The MIT rule

The MIT rule essentially implements a mathematical condition that reduces the error between the reference model and the physical model.

If J represents the error between the reference and the physical model:

$$J(\theta_1, \theta_2 \dots \theta_N) = \left| \frac{e^2}{2} \right|$$

Where $e(t) = y_m(t) - y(t)$ and $\theta_1, \theta_2 \dots \theta_N$ represent the gains in the control law

In order to achieve the condition

$$\frac{dJ}{dt} = \frac{\partial J}{\partial t} + \frac{\partial J}{\partial \theta}\frac{d\theta}{dt}$$

If the second term resulting from the chain rule is sufficiently large, the total time derivative of the model error will be made negative. For this,

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta}$$

So that

$$\frac{dJ}{dt} = \frac{\partial J}{\partial t} - \gamma \left(\frac{\partial J}{\partial \theta}\right)^2$$

$$\frac{\partial J}{\partial \theta} = 2e\frac{\partial e}{\partial \theta}$$

$$\frac{d\theta}{dt} = -2\gamma e \frac{\partial e}{\partial \theta}$$

Assuming that the transfer function in the reference model is G_{ref} and the plant to be controlled is G_p, the error can be expressed as:

$$e(t) = kG_p\big(\theta(t).u_c(t)\big) - k_0 G_{ref}\big(\theta(t).u_c(t)\big)$$

$$\frac{\partial e}{\partial \theta} = k.G_p\big(u_c(t)\big) = \frac{k}{k_0}.y_m(t)$$

$$\therefore \frac{d\theta}{dt} = -\gamma_n . y_m(t) . e(t, \theta)$$

Where $\gamma_n = 2\,\gamma . \left(\frac{k}{k_0}\right)$ is called the adaptation gain.

Another modification that is used is normalization of the MIT rule using the following formula [Jain and Nigam 2013]:

$$\frac{d\theta}{dt} = -\frac{\gamma e \frac{\partial e}{\partial \theta}}{\alpha + \left(\frac{\partial e}{\partial \theta}\right)^2}$$

Substituting for $\frac{\partial e}{\partial \theta}$

$$\frac{d\theta}{dt} = -\frac{\gamma_n e y_m}{\alpha + (y_m)^2}$$

Where, y_m is the reference model output and $\gamma_n$and α are constants. This modification is done to avoid division by zero in the final integration step of the adaptive control algorithm and is found to work satisfactorily. This is also referred to as the normalized version of the MIT rule.

The derivative value of the gain is computed using simple the system outputs in Simulink MATLAB and is integrated to obtain the gain values in real time. The gain values converge to a specific value depending on the response of the reference model.

Now that the diversity of the applications of MRAC has been established, and the simplicity with which it can be implemented, it can be concluded that MRAC is an extremely reliable technique for dynamical systems that needs to function adaptively. Hence, the current study is focused on implementing the MRAC law using the MIT rule for the attitude control system of a suspended body oriented by a single reaction wheel.

7 SIMULATION OF ATTITUDE CONTROL

Simulink in MATLAB is a data-flow type programming language that was originally built with control systems in mind. Simulink offers simple arithmetic operations as well as complex domain specific functions in digital signal processing, control system design and design optimization function. The easiness of setting up the control system and calculating system responses prompted the use of this programming environment

7.1 The Control System Diagram

The main control system diagram consists of the following blocks as shown below:

i. PID controlled reference model
ii. Controlled satellite model
iii. Adaptive controller
iv. Input conditioning
v. Summation blocks and scopes
vi. Set-point input block

The constituent blocks and operations of each of the blocks in the main control system will be explained.

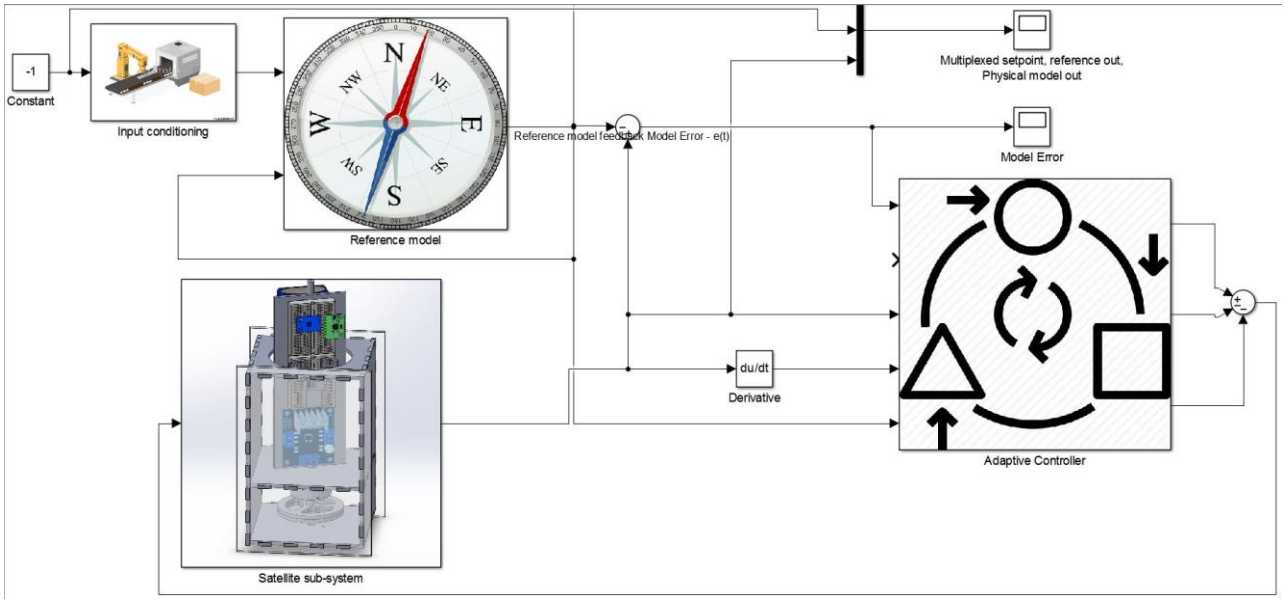


Figure 7.1 Complete Simulink Control System Diagram

7.2 Input conditioning block

The input conditioning block is a simple operation used to find the remainder of the setpoint input or the desired angle input when it is divided by 2π. The block diagram for the block is given by:

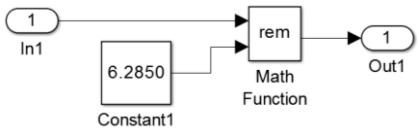


Figure 7.2 Input conditioning block

7.3 Controlled Reference model

The controlled reference model is the desired system model with a Proportional-Integral-Derivative(PID) control implemented. Depending on the power of the actuator (motor), the reference model will maintain at the set-point angle given by the user. The reference transfer function was obtained after dynamic modelling of the system using free-body diagrams assuming dynamic equilibrium

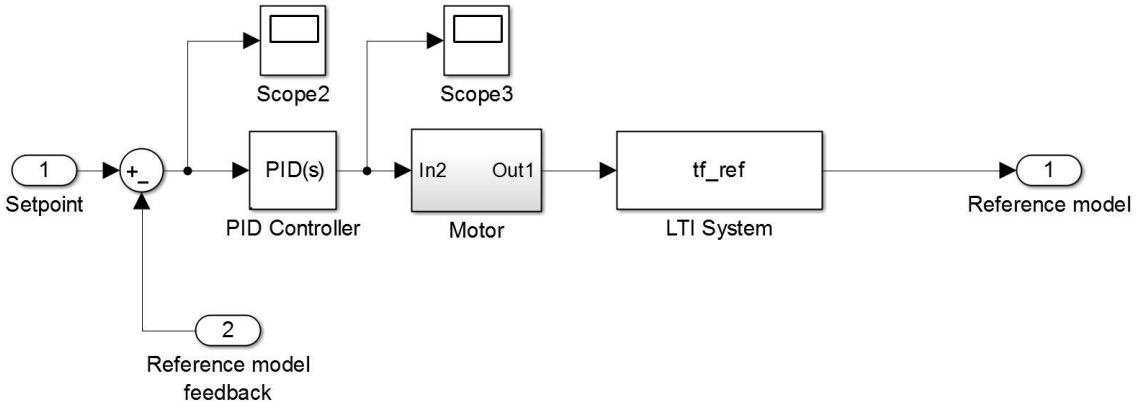


Figure 7.3 PID controlled reference system

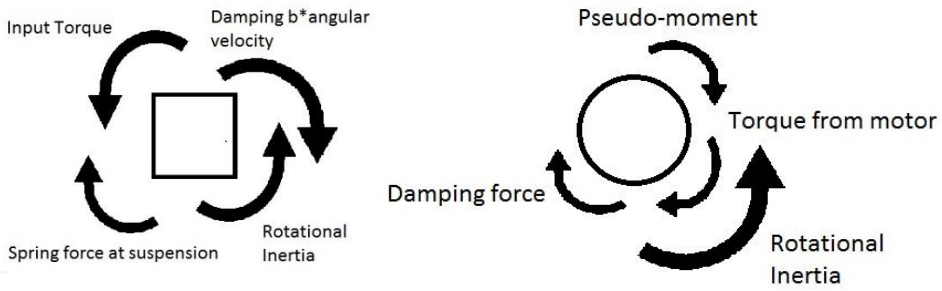


Figure 7.4 Free body diagram of chassis and reaction wheel in dynamic equilibrium

Hence, the equations obtained from these diagrams are:

$$I_2(\ddot{\theta_2} + \ddot{\theta_1}) + b_2\dot{\theta_1} + T_c = 0$$

$$(I_1 + I_2)\ddot{\theta_1} + I_2\ddot{\theta_2} + (b_1 + b_2)\dot{\theta_1} + G\theta_1 = T$$

The state space equations can be derived using the substitution

$$u_1 = \dot{\theta_1}$$

$$\text{And } u_2 = \dot{\theta_2}$$

Making the substitutions as mentioned above:

$$I_2(\dot{u_1} + \dot{u_2}) + b_2u_1 = -T$$

$$(I_1 + I_2)\dot{u_1} + I_2\dot{u_2} + (b_1 + b_2)u_1 + G\theta_1 = T$$

The system can be written in the form:

$$\begin{bmatrix} 0 & I_2 & 0 & I_2 \\ 1 & 0 & 0 & 0 \\ 0 & I_1 + I_2 & 0 & I_2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta_1} \\ \dot{u_1} \\ \dot{\theta_2} \\ \dot{u_2} \end{bmatrix} + \begin{bmatrix} 0 & b_2 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ G & b_1 + b_2 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ u_1 \\ \theta_2 \\ u_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} \dot{\theta_1} \\ \dot{u_1} \\ \dot{\theta_2} \\ \dot{u_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -17.42 & -17.42 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 17.42 & 4.922 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ u_1 \\ \theta_2 \\ u_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 3484 \\ 0 \\ -4734 \end{bmatrix} T$$

$$y = [1\ 0\ 0\ 0]\begin{bmatrix}\theta_1\\ u_1\\ \theta_2\\ u_2\end{bmatrix} + 0.T$$

This is the state space form of the governing differential equations. Using the function tf() in MATLAB, the transfer function for this system between the torque T and the angular displacement of the chassis θ_1 is given by;

$$\frac{\theta_1(s)}{T(s)} = \frac{3484}{s^2 + 17.42s + 17.42}$$

This is the transfer function that is controlled using the PID controller and is referred to as tf_ref in the block diagram depicted earlier.

7.4 Adaptive Controller

The adaptive controller block consists of two smaller blocks. The first block calculates the factor $y_m(t)$ and the remaining block computes the value of $k_1$and k_2 after the integration and after calculating the control law sends it as feedback to the satellite sub-system.

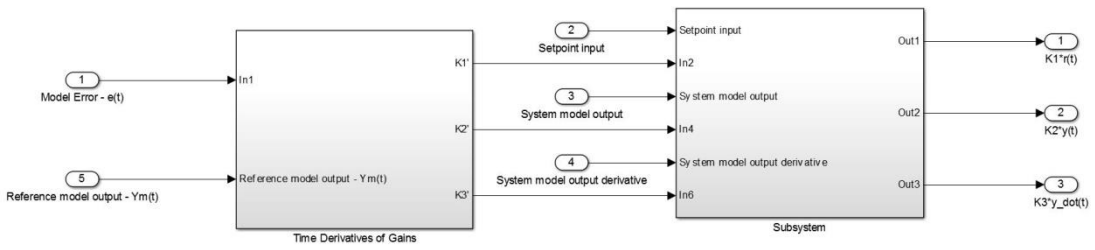


Figure 7.5 First layer of sub-system

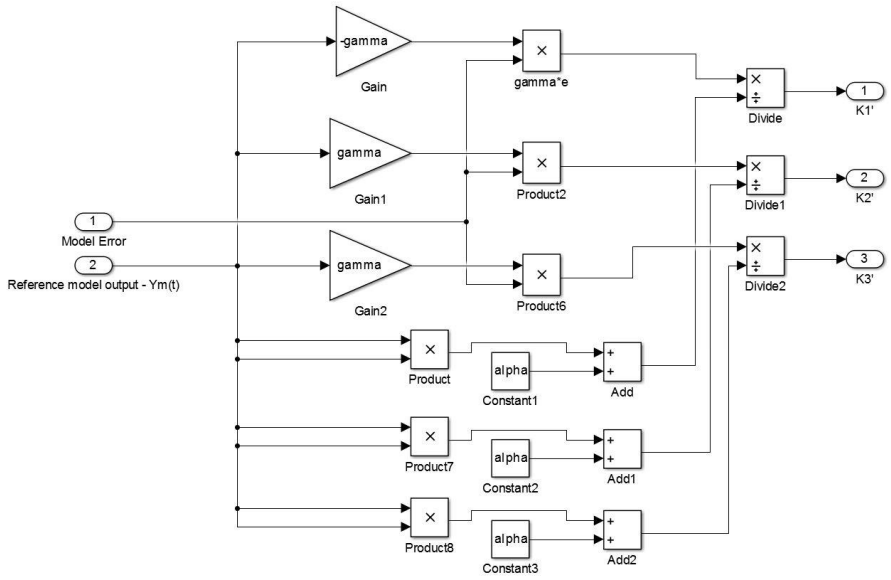


Figure 7.6 Sub-system for the time-derivative of gains

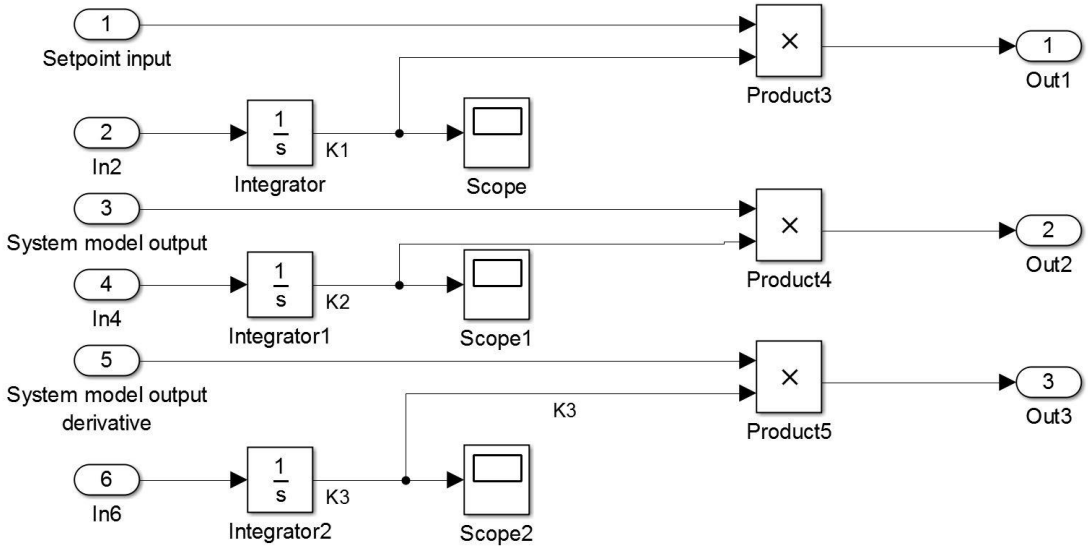


Figure 7.7 Subsystem for the integration of time derivatives of gains

7.5 Controlled Satellite model

The Controlled satellite model is the block representing the physical system that was modelled in SOLIDWORKS 2013. The 3D model contains the mass and inertia properties of all the mechanical (chassis) and the electronics sub-systems (CPU) that were idealized in the dynamic model described in section b). This model receives a single input – a torque input at the reaction wheel revolute element and the chassis

position obtained from a sensor reading obtained from the chassis block of the model as shown below. For more details about the model, please refer to the CAD modelling sub-section of the report. The SOLIDWORKS file was exported as a second generation SimMechanics file, which was imported into Simulink using the command smimport('filename').

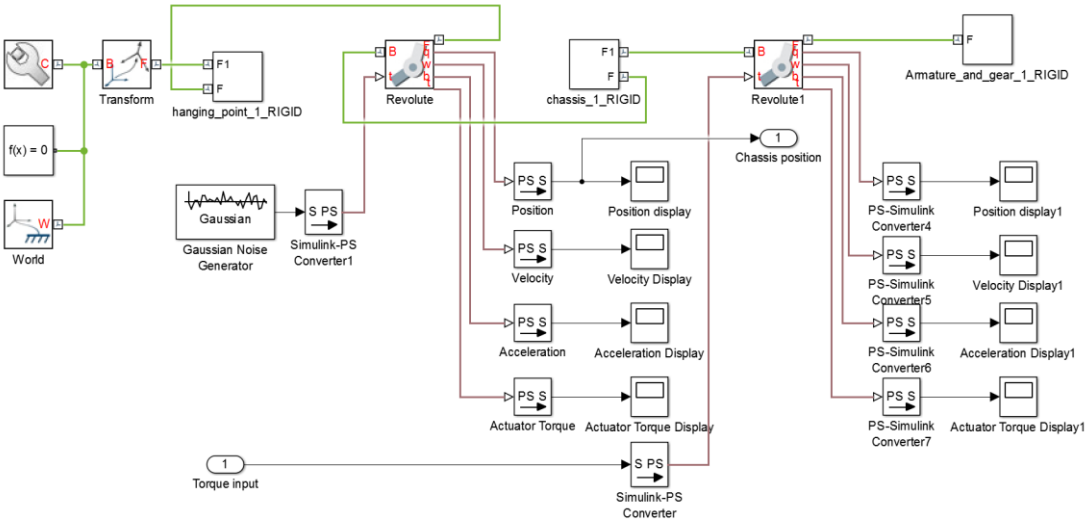


Figure 7.8 Satellite sub-system

7.6 The Motor Sub-system

The motor sub-system is meant to simulate the features of a 12V DC motor that is to be used in the physical model. The model was available in MATLAB as an example model and the remaining work was to obtain the motor parameters by performing a motor characterization. Although a data-sheet could have been procured from the internet, as these values indicated the as-purchased values of the motor parameters. Hence, to determine the current parameters, the motor characterization was done which has been explained in section 8.

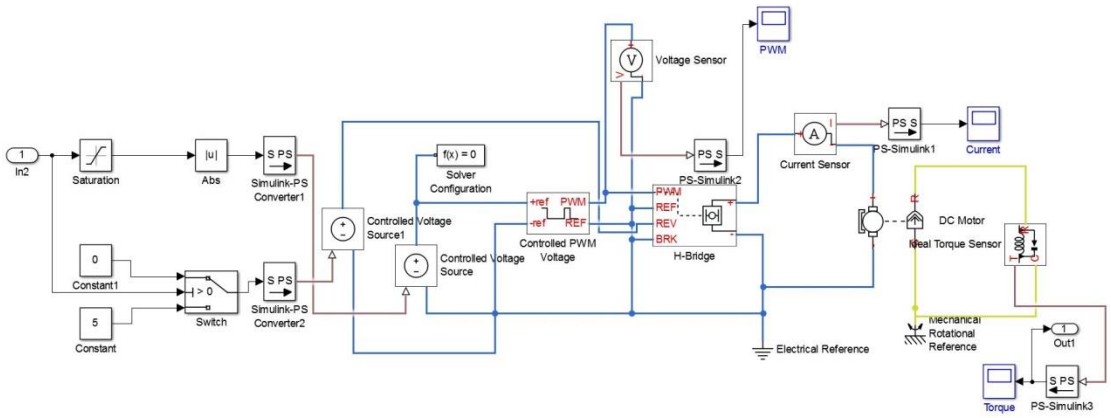


Figure 7.7 Controlled DC motor sub-system

8 MOTOR CHARACTERIZATION

Motors form an essential part of many electro-mechanical systems. They are one of the most common actuators available that convert electrical energy into rotational energy in the form of angular velocity and torque. When the system under study has to be subjected to a control system, perhaps to be simulated in a program, a system-level transfer function that characterizes the motor is convenient [Seaton]. Apart from this advantage from a control systems perspective, certain motor parameters such as efficiency, point of stall may need to be recalculated to decide if the motor can perform the expected task[Harington and Kroninger 2013]. The term 'characterization' can also be used in contexts such as propeller characterization.

The mathematical model of the motor in the form a transfer function is pivotal in studies were efficient compensators have to be designed [Anguluri 2014], [Rose et al. 2014] where the moment of inertia and friction co-efficient were experimentally discovered, [Niekerk et. al 2015] for UAVs and analyzing the effects of motor characterization on hybrid vehicles as in [Mehdrad 2003].

The complete system-level transfer function [Seaton] of the DC motor is of the form:

$$\frac{\omega}{V_a} = \frac{K}{(J.L_a)s^2 + (J.R_a + \ b.L)s + b.R_a + K^2}$$

Where ω is the angular velocity, K is the motor constant, J is the inertia of the motor, $L_a$is the inductance of the motor, R_a is the armature resistance and b is the friction co-efficient. Within the scope of this project, the friction co-efficient is considered to be negligible []. Hence, the modified transfer function is of the form:

$$\frac{\omega}{V_a} = \frac{K}{(J.L_a)s^2 + (J.R_a + \ L)s + K^2}$$

The following methods as per [Seaton] was used to measure the armature resistance and the inductance of the dc motor:

Keeping aside the dynamics of the motor, the motor is essentially an electrical circuit. Hence, clamping the motor shaft and measuring the current flowing through the system for an applied voltage can be used to measure the resistance of the motor using Ohm`s law:

$$V_a = I_a R_{arm}$$

The resistance was measured for six different voltage values for six different armatures. The voltage was applied using a DC power supply.

Figure 8.1 Setup for measurement of armature resistance

Table 8.1 Observations for measurement of armature resistance

Voltage	Current	Resistance (Ohm`s law)
1	0.14	7.142
2	0.23	8.695
3	0.33	9.090
4	0.41	9.756
5	0.48	10.416
6	0.55	10.909
Average Resistance		9.335

The resistances were measured for different rotor positions and the resistance was averaged over all the values:

$$R_a = \overline{R_{arm}} = 9.685\Omega$$

Next, the inductance of the motor armature is measured. As the motor windings consist of both resistive and inductive elements, there is a time constant associated with the rise and decay of current within the circuit. The circuit used was of the following form:

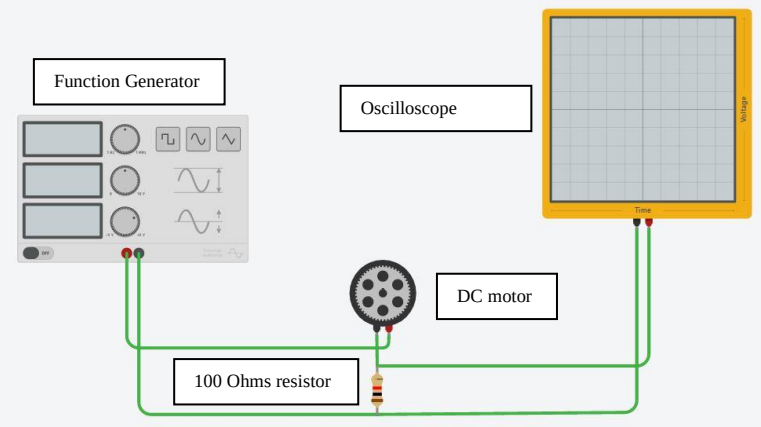


Figure 8.2 Circuit for measurement of inductance

The equipment used were Agilent 13220A, 20 MHz function/arbitrary waveform generator and Agilent technologies, InfiniVision DSO-X 2002A oscilloscope.

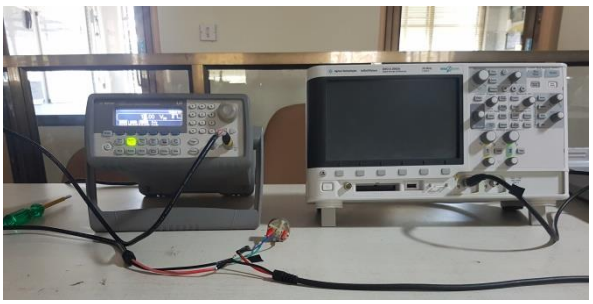


Figure 8.3 Setup fot measurement of inductance

The waveform obtained on the oscilloscope:

Figure 8.4 Waveform obtained

Assuming that the total time of decay was approximately equal to 5 time constants:

$$\frac{5L}{R_{eq}} = 500\mu s$$

$$R_{eq} = R_{arm} + 100$$

$$\therefore L = 10.9685mH$$

Further, the no-load characteristics of the dc motor were measured. This was done using the same DC power supply used to measure the armature resistance. Simultaneously, a laser tachometer was used to measure the RPM of the motor at each data point. The resulting voltage-RPM characteristics are plotted below. Using the formula:

$$P = \frac{2\pi NT}{60}$$

The values of torque at each of the data points were also calculated

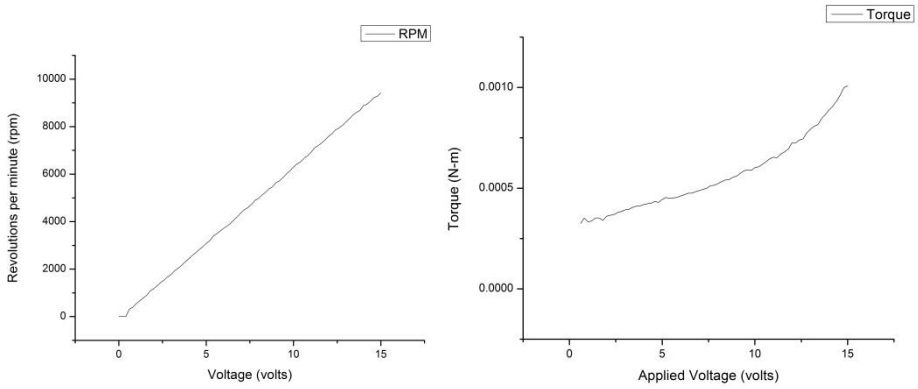


Figure 8.5 RPM and Torque Curves

Finally, the motor constant was calculated. The motor constant is given by:

$$k = \frac{V - V_{BEMF}}{\omega}$$

The back EMF $V_{BEMF}$is calculated by multiplying no-load current and armature resistance and dividing by the angular velocity at that point. This operation was carried out at all the data points and the average value was calculated.

$$K = 0.00158\ \frac{volts - s}{rad}$$

Therefore, the transfer function obtained is:

$$\frac{\omega}{V_a} = \frac{0.00158}{(0.0000057)s^2 + (0.01566)s + 0.00000249}$$

9 PHYSICAL MODEL

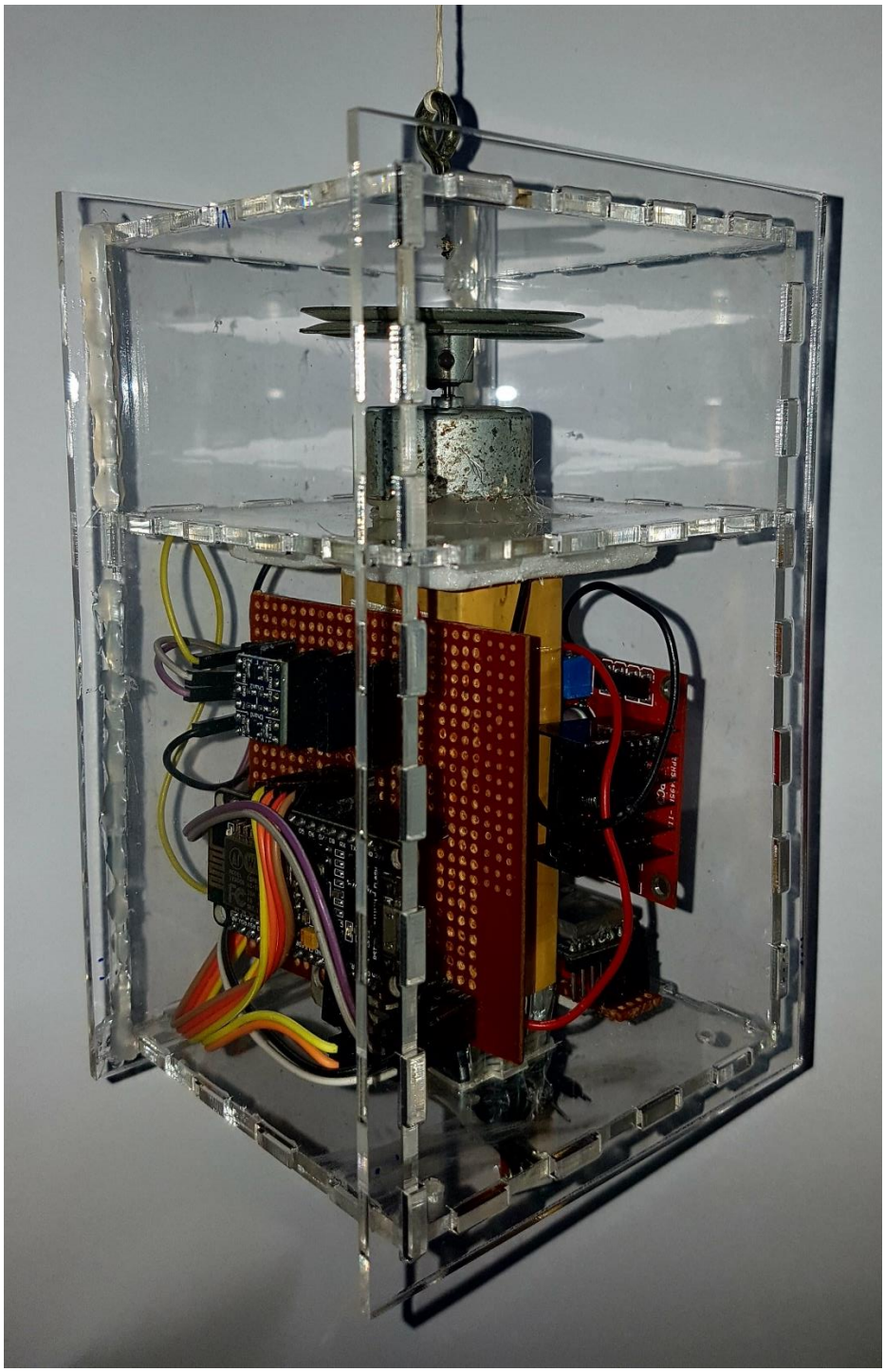


Figure 9.1 Working model

9.1 Fabrication of the chassis

Based on the CAD model, the suspended body is made using cast acrylic sheets. An interlocking type walls is to be manufactured. Hence, the CAD file is converted to a CoralDRAW file and fed into a CNC laser cutter.

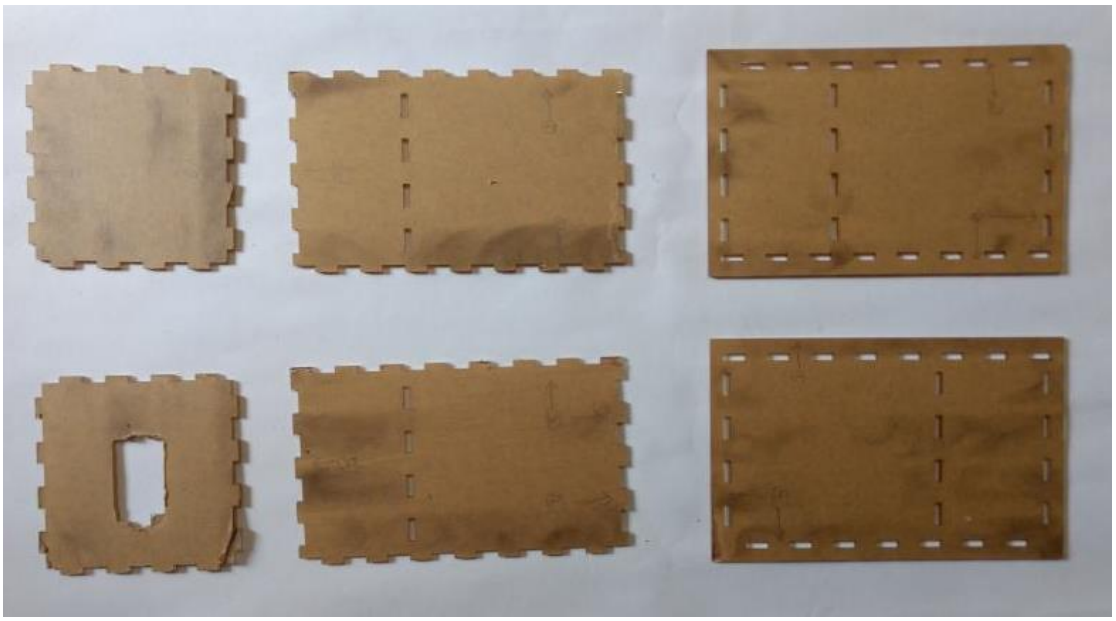


Figure 9.2 Individual components of the chassis

These toothed walls can be easily snapped on to each other to complete the chassis of the suspended body.

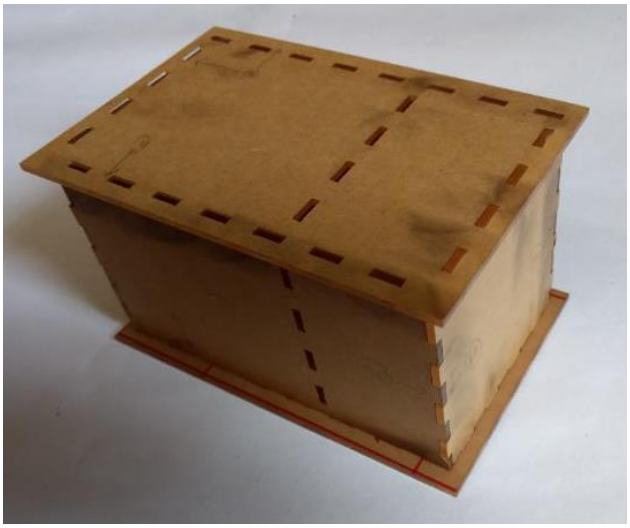


Figure 9.3 Assembled chassis

9.2 Circuitry

The electronic components used in the satellite are connected with one another using a circuit board. The circuit board is made by soldering berg strips onto a copper clad general purpose PCB. Hence the required electronic components can simply be snapped onto the circuit. Provision has been given to connect jumper wires using additional berg strips.

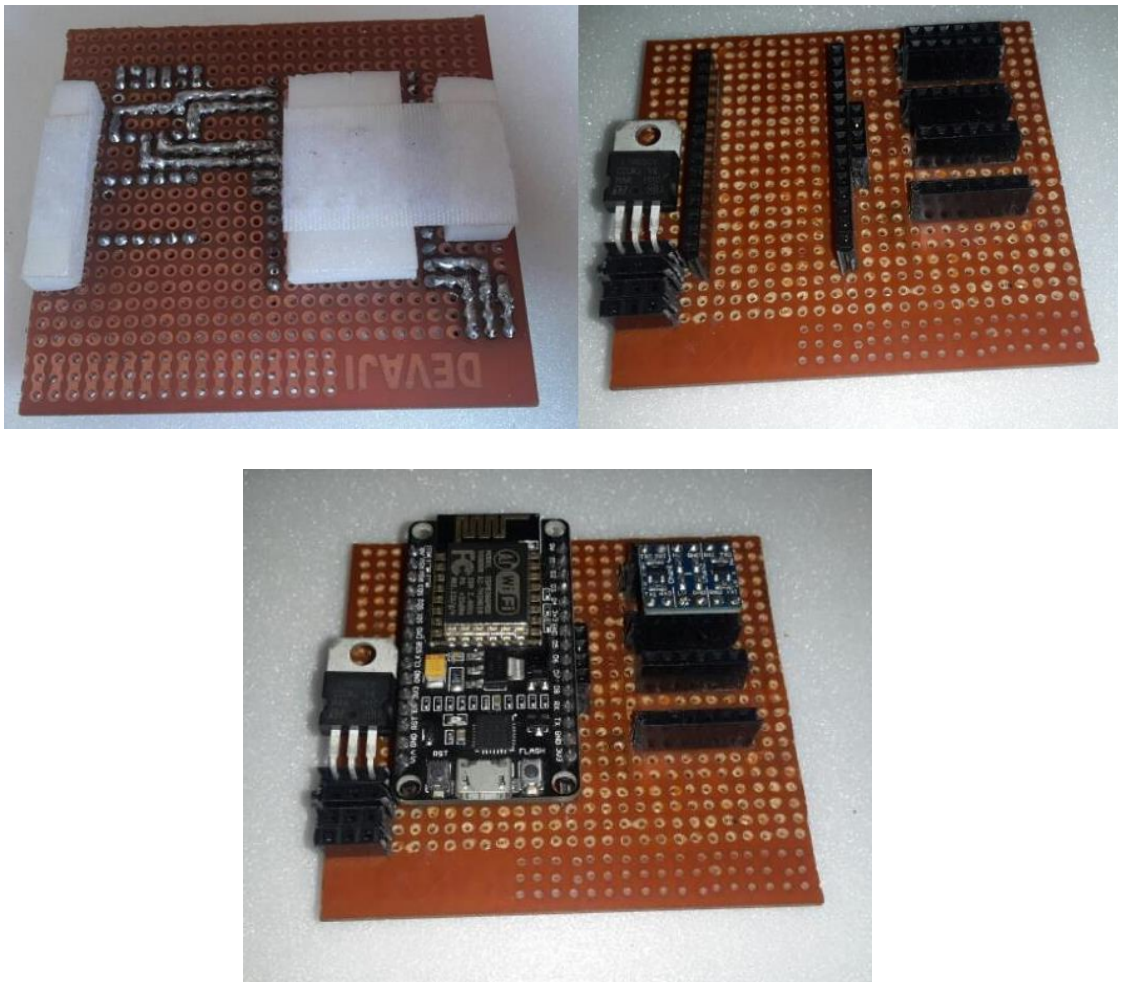


Figure 9.4 Circuit Board

9.2.1 L29810 Motor Driver

The LN298 is a high voltage, high current, dual full-bridge motor driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. The unit used in this project has an inbuilt heat sink and hence is equipped to handle heating due to high current situations like direction change.

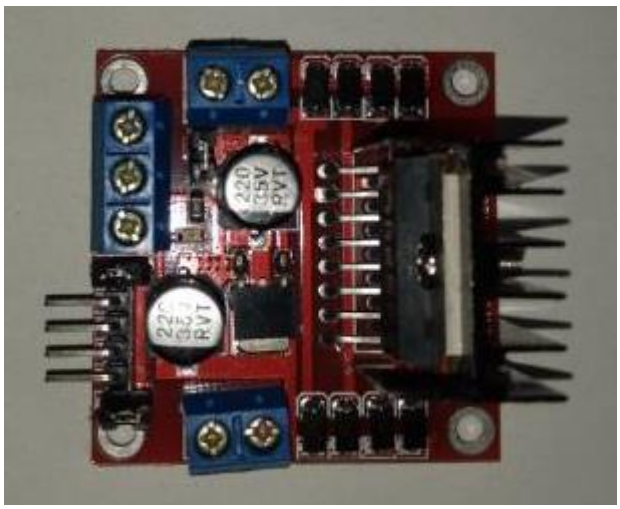


Figure 9.5 L29810 Motor Driver

9.2.2 DC Motor

A 12V D motor is used. Refer to Section 8 for specification of the motor.

Figure 9.6 DC Motor

9.2.3 11.1V Li-Po Battery

A Lithium-Polymer (Li-Po) Battery is used as the power source for motor and microcontroller. It is capable of giving instantaneous discharge current up to 55A. It has a very light weight and is small in size compared to Ni-Cd and Lead acid batteries. It has a long life without losing charging capacity. It weighs 167 g.

Figure 9.7 Battery

9.2.4 Sensors

MPU 6050

The InvenSense MPU-6050 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefor it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the NodeMCU micro.

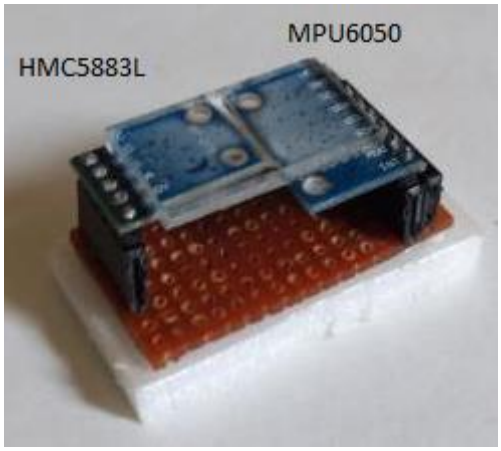


Figure 9.8 Sensors Used

HMC 5883L

This is a breakout board for Honeywell's HMC5883L, a 3-axis digital compass. Communication with the HMC5883L is simple and all done through an I2C interface. There is no on-board regulator, so a regulated voltage of 2.16-3.6VDC should be supplied.

9.2.5 ESP 8266 – Node MCU

An Arduino Uno development board was used in the inverted pendulum and the initial model of satellite. In this model, a NodeMCU was used as the controller.

The NodeMCU has an ESP8266 Wi-Fi SoC from Espressif Systems. The firmware uses the Lua scripting language. It can also be flashed with firmware that supports interfacing with the Arduino IDE.

The NodeMCU has several advantages over the Arduino Uno. Firstly, it is inexpensive. It has the ESP8266 which carries an inbuilt WiFi module which can be used to wirelessly communicate between the NodeMCU and devices connected to the same network like smartphones or computers for uploading code, and sending data. It has 16 General Purpose Input/Output (GPIO) pins. The primary advantage is the high frequency clock (80MHz) and the large storage space (4 MB). [igrr, et al, 2017]

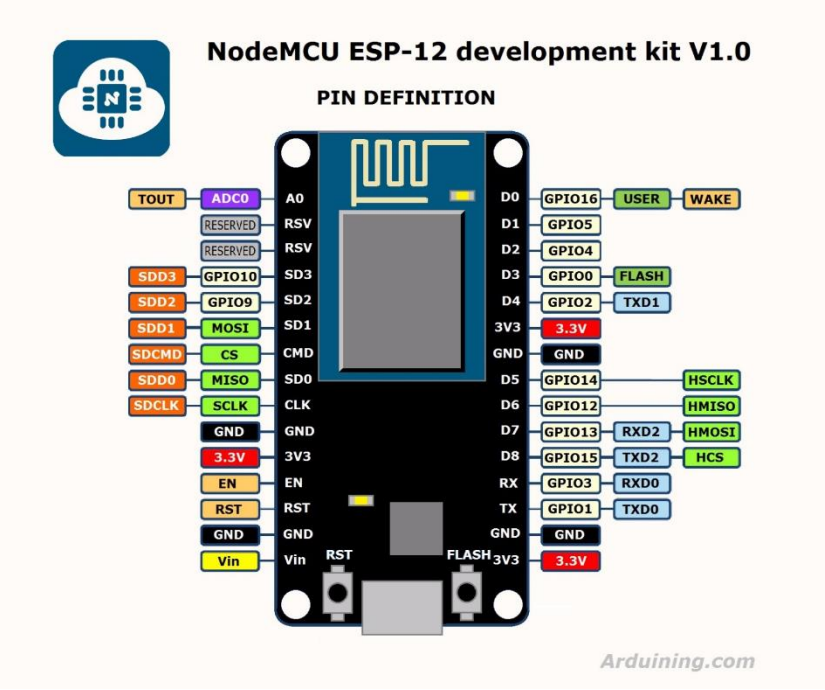


Figure 9.9 NodeMCU pin definition diagram

10 IMPLEMENTATION ON HARDWARE

10.1 Implementation of PID

$$Output = K_P e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t)$$

Equation 10.1 Standard PID Equation

Where, $e = Setpoint - Input$

```
error = angle - refAngle;
error = ((int(error*100)+18000+36000)%36000-18000)/100.0;
```

10.1.1 Calculation for the input to PID

The error needed to be mapped between -180 degrees and 180 degrees. This is required to avoid the 0 to 360 discontinuity when it reaches the Setpoint (refAngle). This ensures that the error is centered around zero at the vicinity of the Setpoint and is mathematically consistent for PID calculation at all values of the setpoint. The error is multiplied and divided with 100 because the modulo function only works with the datatype 'int'.

```
  errsum += error; \\Discrete Integration
  derror = error - preverror;\\ Discrete Differentiation
  preverror = error;
```

10.1.2 Calculating integral and derivative terms

Due to constant time interval between successive PID calculations (~10 milliseconds), the discrete integration and differentiation can be reduced to summation and backward differencing.

```
  if(errsum>1500){
    errsum=1500;
  }
  else if(errsum<-1500){
    errsum=-1500;
}
```

Capping of integral term

This is required so that the integral term does not drastically increase when debugging.

```
float pid=kp*error + ki*errsum + kd*derror;
```

Calculating PID

```
  if(pid>0){
    if(pid>1023){
      pid=1023;\\Saturating PID
    }
    analogWrite(4,pid);
    analogWrite(5,0);
  }
  else{
    if(pid<-1023){
      pid=-1023;\\Saturating PID
    }
    analogWrite(5,-1*pid);
    analogWrite(4,0);
  }
```

This snippet of code Saturates PID output at maximum and minimum values of PWM and direct mapping of PID output to the PWM to drive the motors.

10.2 Reading from Magnetometer

A HMC883L magnetometer is used to obtain the absolute orientation of the system with respect to magnetic north. The sensor gives raw magnetic field strength in its own axis. The sensor has a full-scale range of ±8 Guass and a resolution of up to 5 milli-Gauss. Communication with the HMC5883L is done through an I2C interface using the Wire library of Arduino.

The yaw angle of the system is calculated as arctan of the ratio between x axis and y axis strengths where the magnetometer's xy-plane is kept horizontal . Due to low magnetic declination of the testing location, the calculation of declination correction is omitted. The yaw angle is in the range of 0 to 360 degrees from due north. For the complete program code, please refer Appendix C.

10.3 Online Tuning of PID Parameters Using Blynk

Blynk is an Internet of Things platform with a mobile application builder that allows to visualize sensor data and control electronics remotely within the same network. In our case, blynk was used to remotely connect to the onboard microcontroller via a custom TCP-IP protocol. The PID parameters are tuned to from the app with the assistance of real-time plot of the orientation. The Blynk support package for ESP8266 is used on the microcontroller. An interface is made on the app for tuning using sliders. For the complete program code, please refer Appendix C.

Figure 10.1 Interface on the Blynk App

10.4 WiFi Communication With Smartphone

The Setpoint can be changed by remotely sending it via wifi to the microcontroller on the cube. To accomplish this, User Datagram Protocol(UDP) was chosen because of its high speed. UDP can deliver packets faster than TCP with less delay. In this case, the microcontroller is the UDP server. The Wifi client that transmitted the ‘refAngle’ was a smartphone app-HyperIMU. This app transmitted the phone orientation via UDP protocol and this was used as a Setpoint for the system hence, mirroring the smartphone. For the complete program code, please refer Appendix C.

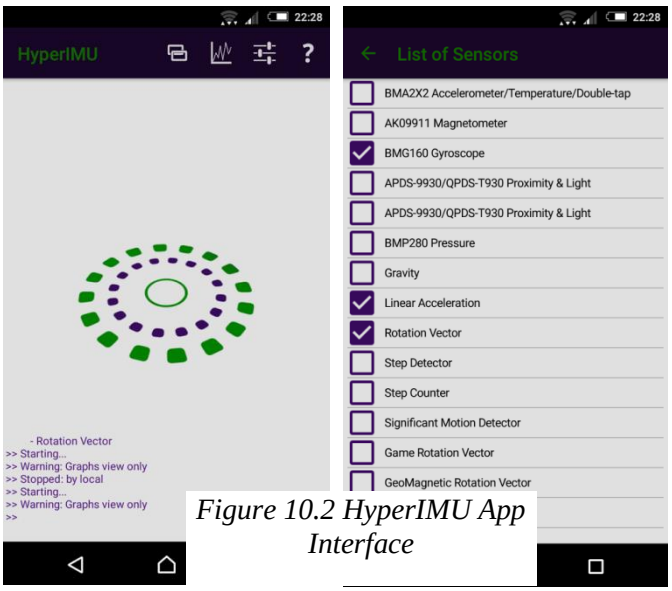


Figure 10.2 HyperIMU App Interface

10.5 Wifi Communication with Simulink

In order to implement adaptive control by processing in Simulink itself, a communication channel needed to be setup between the microcontroller and Simulink.

For this, Simulink Desktop Real-Time toolbox was used. 'Stream Input' block was used to receive current orientation of the physical model. 'Stream Output' block was used to send PWM signals to drive the motor.

These blocks used UDP protocol to communicate. In Simulink, there is no access to buffers of the UDP stream and are automatically taken care of.

The code receives the input buffer, typecasts it to 'int' datatype and assigns it to 'pwm' variable.

```
    char toSend[255];
    String(String(angle)+"\n").toCharArray(toSend,255);
    Serial.printf("%s\n", toSend);
    Udp.beginPacket(Udp.remoteIP(), 8820);
    Udp.write(toSend);
    Udp.endPacket();
```

The above snippet of code converts ‘angle’ into a ‘string’ datatype and adds it to the output buffer. For the complete program, refer Appendix D.

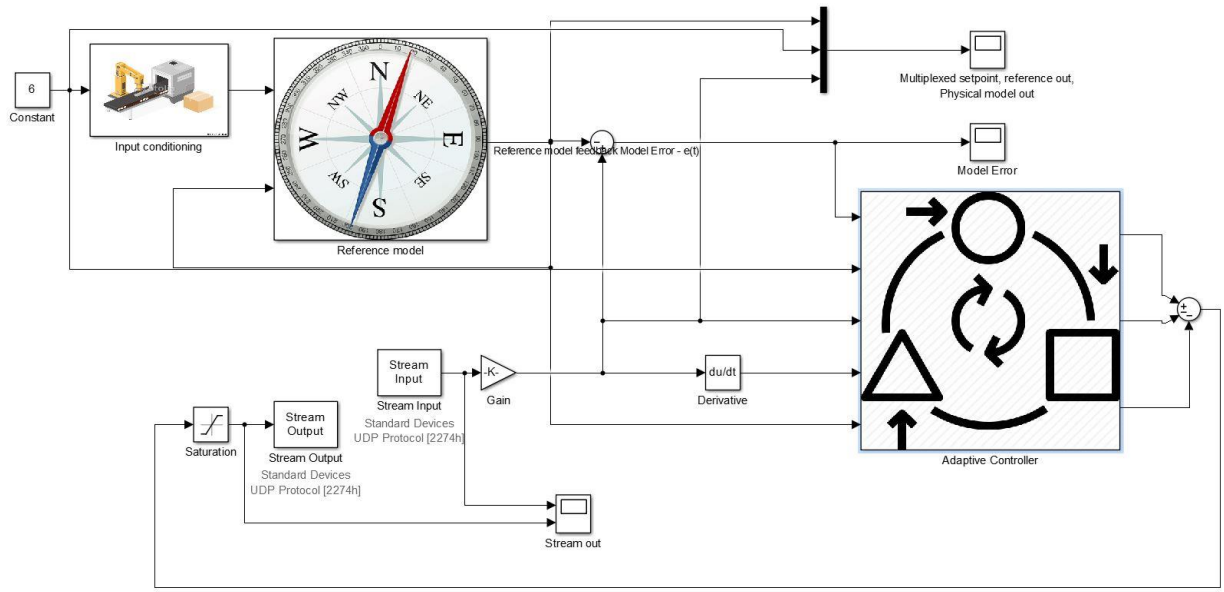


Figure 10.3 Simulink block diagram

11 RESULTS

Some of the most important performance metrics of any control systems are the rise time, the settling time and the steady state error. The variation of these three metrics is observed with respect to the adaptation gain. The rise time is defined as the time taken to reach 90% of the set-point from 10% of the set-point. The settling time is the time taken by the system to reach between 2% and 5% of the set-point value. Finally, the steady state error is the amplitude of oscillations sustained by the system as time tends to infinity.

The settling times and rise times were plotted for adaptation gains ranging from 0 to 6 with steps of 0.25. In both cases, the values saturated towards a specific value on increasing the adaptation gain.

However, the changes in adaptation gain did not have any effect on the steady state error as represented by the erratic changes observed in Figure.

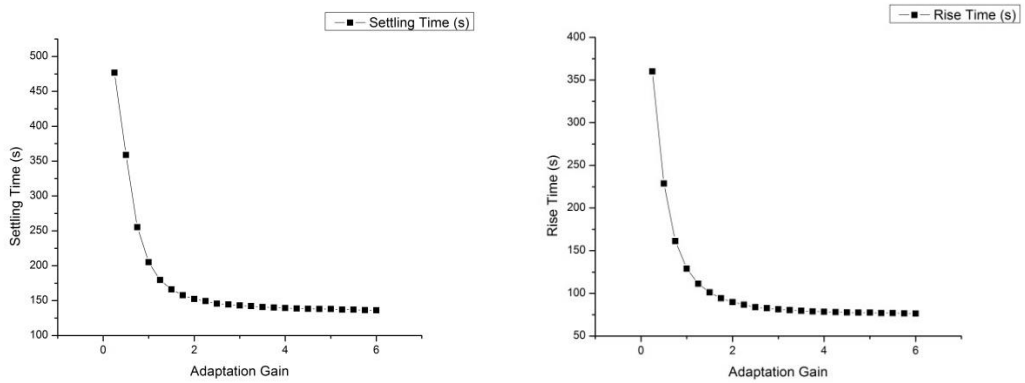


Figure 11.1 Graphs of Setting Time and Rise Time vs Adaptation Gain

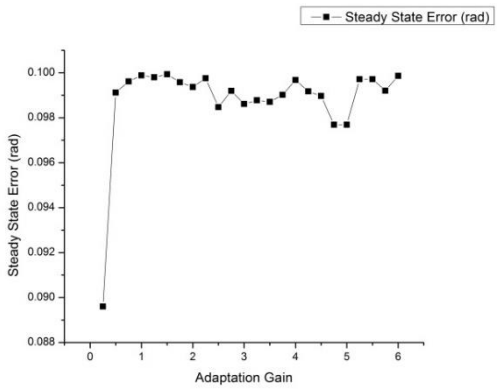


Figure 11.2 Steady State Error vs Adaptation Gain

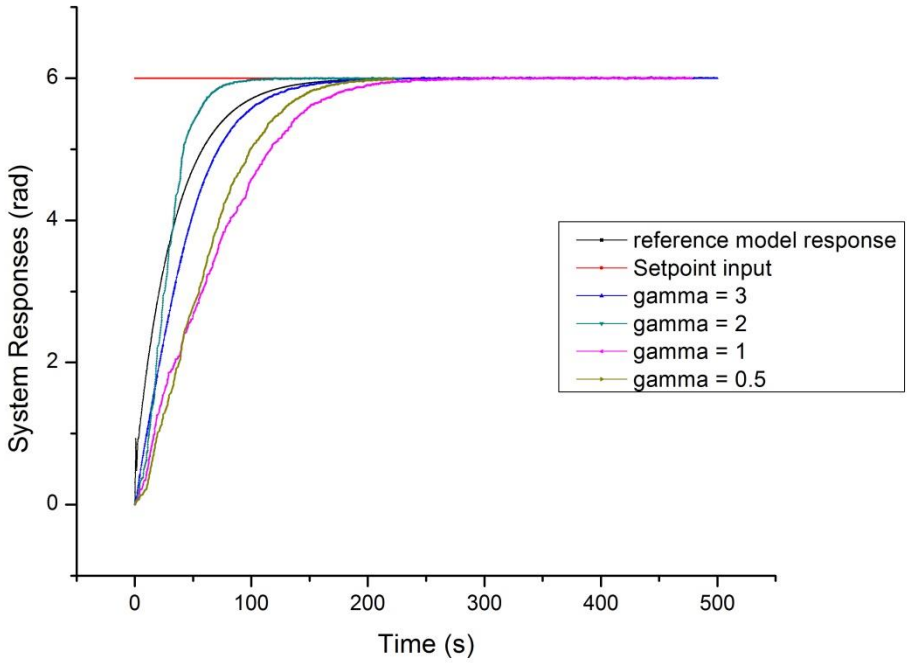


Figure 11.3 Response of MRAC for different Gamma values

Further, the response of the MRAC implemented physical model was measured for different values of adaptation gains. It is observed that for low values of adaptation gain(γ), the response is noisy and has longer rise times. The value γ=3, seems to most accurately follow the reference system`s response. This suggests that for a given system, there might be a specific value of adaptation gain at which it best resembles the controlled reference model.

Finally, the MIT implementation of MRAC was compared with the Lyapunov method of implementing the same control system. The Lyapunov method relies on the use of the output of the system in the gain updation parameters unlike the MIT rule. It is widely acknowledged that the Lyapunov approach is better than the MIT rule, however, in the simulations completed, the model implementing the MIT rule appears to settle faster and have lesser off-shoot.

The following images depict the responses for the reference system model, MRAC-MIT and MRAC-Lyapunov rule for comparison.

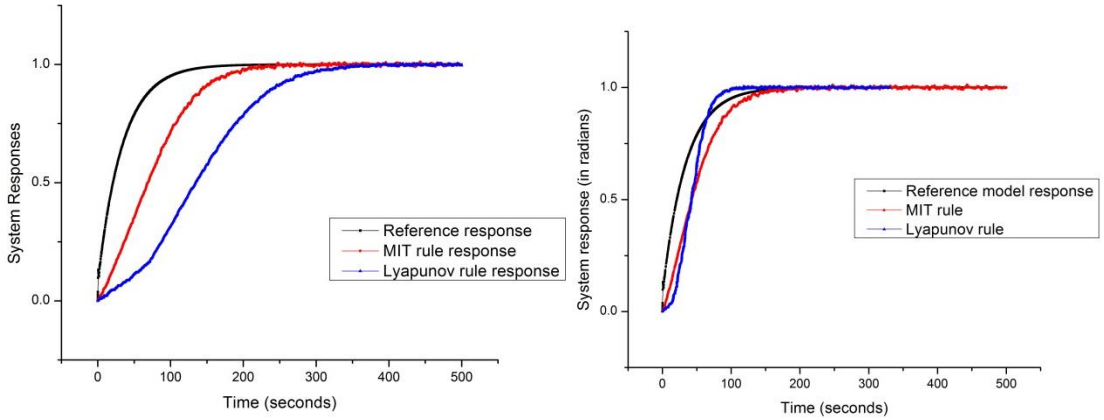


Figure 11.4 System responses for Gamma = 1, gamma = 2

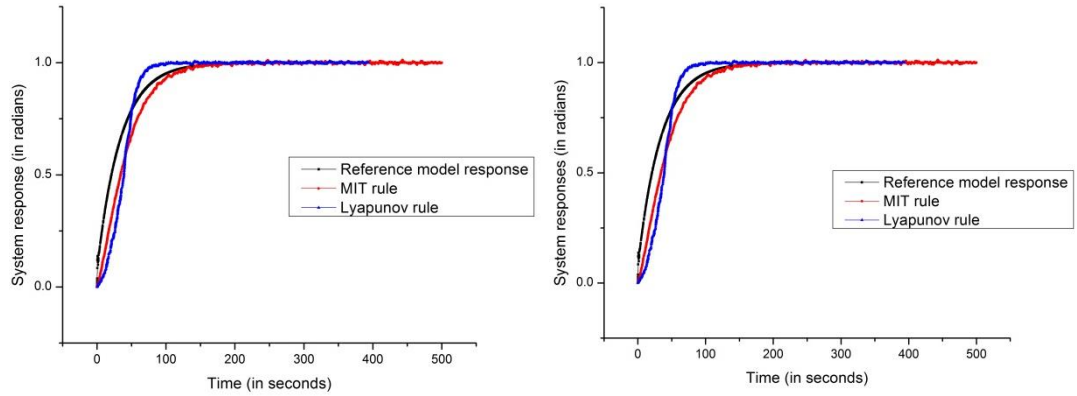


Figure 11.5 System Responses for Gamma = 3, Gamma = 4

12 FUTURE SCOPE OF WORK

There are several different kinds of performance metrics. From a computational perspective, the CPU-time used for controlling the system is relevant. Another important metric is the actuator energy that can also have a dependence on the adaptation gain. Both of these would be highly relevant where extremely spartan and energy-efficient designs such as in aerospace.

An interesting study would be to program a neural network with the performance metrics as inputs to the first layer and the final year calculating the adaptation gain. The simulated experiment performed above already provides several data-sets.

MRAC should also be compared with more generic evolutionary optimization algorithms like Particle Swarm Optimization, Genetic Algorithms etc. along the same metrics.

It is envisioned to derive a composite index function by assigning weights to different performance metrics specified above. The adaptive control has to be implemented in such fashion that the error will be derived from this index function for each response in real time and the adaptive tuning will arrive at the most ‘appropriate combination of gains’. This way, the weights of the index function can be altered to make the system more instinctive or sensitive towards a specific performance metric.

As depicted in the results section, the MIT rule appears to perform better than the Lyapunov rule contrary to popular opinion. Hence, a study about the specificity of MRAC towards the system to favour one particular kind of implementation would prove relevant.

Also, it was noticed that during the hardware implementation, as the PID output saturates, there is less chance for the system recovering from that state. This provides an interesting area for investigation.

LIST OF APPENDICES

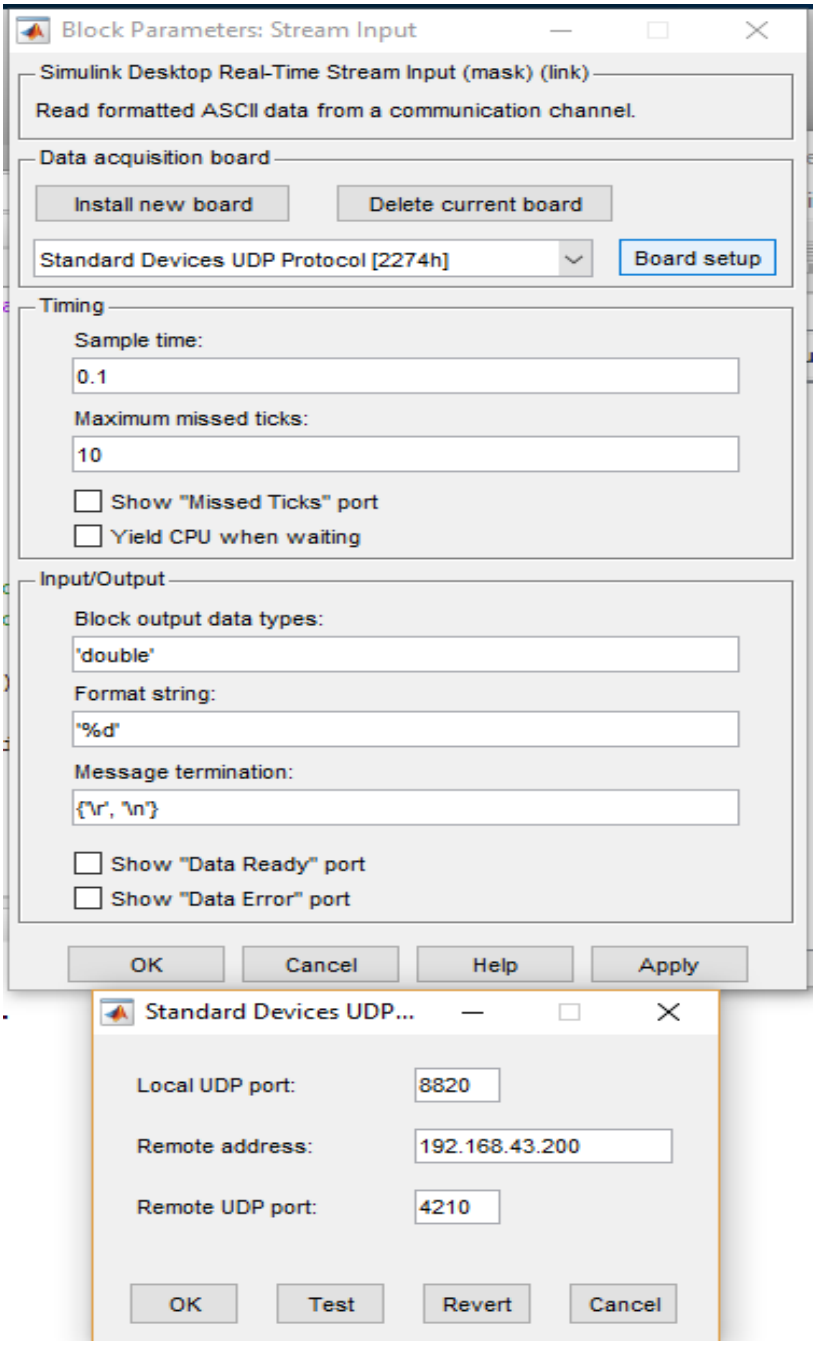
Appendix – A

DC Motor Specifications:

No.	Property	Value
1.	Dimensions	$\phi 42.3 \times 67mm$
2.	Shaft diameter	$\phi 5.005\ mm$
3.	Input voltage	18 volts
4.	No load speed	20950 rpm
5.	No load current	2.9 A
6.	Weight	186 g

Appendix - B

Stream input block parameters:



Appendix – C

Program MPtest2

```
#define BLYNK_PRINT Serial
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <WiFiUdp.h>
#define Addr 0x1E               // 7-bit address of HMC5883 compass

const char* ssid = "Dendb";
const char* password = "csnitk123";

char auth[] = "4906ef7023834762b123af1bf52f975e";

WiFiUDP Udp;
unsigned int localUdpPort = 4210;  // local port to listen on
char incomingPacket[255];  // buffer for incoming packets
char  replyPacekt[] = "Hi there! Got the message :-)";  // a reply
string to send back

//String inputString = "";     // a string to hold incoming data
float refAngle=180;
double angle=180.0, error=0,errsum=0, preverror=0, derror = 0;
float kp=0, ki=0.0, kd = 0;

BLYNK_WRITE(V1)
{
  kp = param.asFloat()/10; // assigning incoming value from pin V1 to
a variable
  Blynk.virtualWrite(5, kp);
}
BLYNK_WRITE(V2)
{
  ki = param.asFloat()/100; // assigning incoming value from pin V1 to
a variable
  Blynk.virtualWrite(6, ki);
}
BLYNK_WRITE(V3)
{
  kd = param.asFloat()/10; // assigning incoming value from pin V1 to
a variable
  Blynk.virtualWrite(7, kd);
}

void setup() {
  Blynk.begin(auth, ssid, password);
//  Serial.printf("Connecting to %s ", ssid);
//  WiFi.begin(ssid, password);
//  while (WiFi.status() != WL_CONNECTED)
//  {
//    delay(500);
//    Serial.print(".");
//  }
//  Serial.println(" connected");

  Udp.begin(localUdpPort);
```

```
  Serial.printf("Now listening at IP %s, UDP port %d\n",
WiFi.localIP().toString().c_str(), localUdpPort);

  // put your setup code here, to run once:
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  Serial.begin(9600);
  Wire.begin(12,14);

  // Set operating mode to continuous
  Wire.beginTransmission(Addr);
  Wire.write(byte(0x02));
  Wire.write(byte(0x00));
  Wire.endTransmission();
}

void loop() {

  int time = micros();
  Blynk.run();

  int packetSize = Udp.parsePacket();
  if (packetSize)
  {
    // receive incoming UDP packets
    Serial.printf("Received %d bytes from %s, port %d\n", packetSize,
Udp.remoteIP().toString().c_str(), Udp.remotePort());
    int len = Udp.read(incomingPacket, 255);
    if (len > 0)
    {
      incomingPacket[len] = 0;
    }
    Serial.printf("UDP packet contents: %s\n", incomingPacket);
  Udp.beginPacket("192.168.43.196", 5555);
  Udp.write(incomingPacket);
  Udp.endPacket();
  int i=0;
  String inputString = "";
  while(incomingPacket[i]!=',')
  {
    inputString += (char)incomingPacket[i++];
  }
  refAngle = inputString.toFloat();

  Serial.println(refAngle);
  }

  //Compass
  int x, y, z;

  // Initiate communications with compass
  Wire.beginTransmission(Addr);
  Wire.write(byte(0x03));       // Send request to X MSB register
  Wire.endTransmission();

  Wire.requestFrom(Addr, 6);    // Request 6 bytes; 2 bytes per axis
  if(Wire.available() <=6) {    // If 6 bytes available
    x = Wire.read() << 8 | Wire.read();
```

```
    z = Wire.read() << 8 | Wire.read();
    y = Wire.read() << 8 | Wire.read();
  }
  // If compass module lies flat on the ground with no tilt,
  // just x and y are needed for calculation
  if(x > 10000) x-=65536;
  if(y > 10000) y-=65536;
  if(z > 10000) z-=65536;
  float  heading=atan2(x, y)/0.0174532925;
  if(heading < 0) heading+=360;
  //heading=360-heading; // N=0/360, E=90, S=180, W=270


  angle = heading;
  Serial.print(angle);
  Serial.print('\t');
  error = angle - refAngle;
  error = ((int(error*100)+18000+36000)%36000-18000)/100.0;
  errsum += error;
  derror = error - preverror;
  preverror = error;
  if(errsum>1500){
    errsum=1500;
  }
  else if(errsum<-1500){
    errsum=-1500;
  }

  float pid=kp*error + ki*errsum + kd*derror;

  if(pid>0){
    if(pid>900){
      pid=1023;
    }
    analogWrite(4,pid);
    analogWrite(5,0);
  }
  else{
    if(pid<-900){
      pid=-1023;
    }
    analogWrite(5,-1*pid);
    analogWrite(4,0);
  }
  Serial.print(pid);
  Serial.print('\t');
  Serial.println(error);

  delay(8);

  Blynk.virtualWrite(8, error);
  //Blynk.virtualWrite(9, pid);
 }


#define BLYNK_PRINT Serial
#include <Wire.h>
#include <ESP8266WiFi.h>
```

```
#include <BlynkSimpleEsp8266.h>
#include <WiFiUdp.h>
#define Addr 0x1E               // 7-bit address of HMC5883 compass

const char* ssid = "Dendb";
const char* password = "csnitk123";

char auth[] = "4906ef7023834762b123af1bf52f975e";

WiFiUDP Udp;
unsigned int localUdpPort = 4210;  // local port to listen on
char incomingPacket[255];  // buffer for incoming packets
char  replyPacekt[] = "Hi there! Got the message :-)";  // a reply
string to send back

//String inputString = "";     // a string to hold incoming data
float refAngle=180;
double angle=180.0, error=0,errsum=0, preverror=0, derror = 0;
float kp=0, ki=0.0, kd = 0;

BLYNK_WRITE(V1)
{
  kp = param.asFloat()/10; // assigning incoming value from pin V1 to
a variable
  Blynk.virtualWrite(5, kp);
}
BLYNK_WRITE(V2)
{
  ki = param.asFloat()/100; // assigning incoming value from pin V1 to
a variable
  Blynk.virtualWrite(6, ki);
}
BLYNK_WRITE(V3)
{
  kd = param.asFloat()/10; // assigning incoming value from pin V1 to
a variable
  Blynk.virtualWrite(7, kd);
}
```

Appendix - D

Program simu

```
void setup() {
  Blynk.begin(auth, ssid, password);
//  Serial.printf("Connecting to %s ", ssid);
//  WiFi.begin(ssid, password);
//  while (WiFi.status() != WL_CONNECTED)
//  {
//    delay(500);
//    Serial.print(".");
//  }
//  Serial.println(" connected");

  Udp.begin(localUdpPort);
  Serial.printf("Now listening at IP %s, UDP port %d\n",
WiFi.localIP().toString().c_str(), localUdpPort);

  // put your setup code here, to run once:
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  Serial.begin(9600);
  Wire.begin(12,14);

  // Set operating mode to continuous
  Wire.beginTransmission(Addr);
  Wire.write(byte(0x02));
  Wire.write(byte(0x00));
  Wire.endTransmission();
}

void loop() {

  int time = micros();
  Blynk.run();

  int packetSize = Udp.parsePacket();
  if (packetSize)
  {
    // receive incoming UDP packets
    Serial.printf("Received %d bytes from %s, port %d\n", packetSize,
Udp.remoteIP().toString().c_str(), Udp.remotePort());
    int len = Udp.read(incomingPacket, 255);
    if (len > 0)
    {
      incomingPacket[len] = 0;
    }
    Serial.printf("UDP packet contents: %s\n", incomingPacket);
  Udp.beginPacket("192.168.43.196", 5555);
  Udp.write(incomingPacket);
  Udp.endPacket();
  int i=0;
  String inputString = "";
  while(incomingPacket[i]!=',')
```

```
{
  inputString += (char)incomingPacket[i++];
}
refAngle = inputString.toFloat();

Serial.println(refAngle);
}

//Compass
int x, y, z;

// Initiate communications with compass
Wire.beginTransmission(Addr);
Wire.write(byte(0x03));       // Send request to X MSB register
Wire.endTransmission();

Wire.requestFrom(Addr, 6);    // Request 6 bytes; 2 bytes per axis
if(Wire.available() <=6) {    // If 6 bytes available
  x = Wire.read() << 8 | Wire.read();
  z = Wire.read() << 8 | Wire.read();
  y = Wire.read() << 8 | Wire.read();
}
// If compass module lies flat on the ground with no tilt,
// just x and y are needed for calculation
if(x > 10000) x-=65536;
if(y > 10000) y-=65536;
if(z > 10000) z-=65536;
float  heading=atan2(x, y)/0.0174532925;
if(heading < 0) heading+=360;
//heading=360-heading; // N=0/360, E=90, S=180, W=270


angle = heading;
Serial.print(angle);
Serial.print('\t');
error = angle - refAngle;
error = ((int(error*100)+18000+36000)%36000-18000)/100.0;
errsum += error;
derror = error - preverror;
preverror = error;
if(errsum>1500){
  errsum=1500;
}
else if(errsum<-1500){
  errsum=-1500;
}

float pid=kp*error + ki*errsum + kd*derror;

if(pid>0){
  if(pid>900){
    pid=1023;
  }
  analogWrite(4,pid);
  analogWrite(5,0);
}
else{
  if(pid<-900){
```

```
    pid=-1023;
   }
   analogWrite(5,-1*pid);
   analogWrite(4,0);
 }
 Serial.print(pid);
 Serial.print('\t');
 Serial.println(error);

 delay(8);

 Blynk.virtualWrite(8, error);
 //Blynk.virtualWrite(9, pid);
}
```

REFERENCES

- Abraham, A., Jatoth, R. K., & Rajasekhar, A. (2014). Design of intelligent PID/(PID mu)-D-lambda speed controller for chopper fed DC motor drive using opposition based artificial bee colony algorithm.
- Adam M, Adjei-Saforo KE, Ebrahimpanah S, Adaptive Control Strategy using Lyapunov Stability Theory, International Journal of Engineering Research & Technology Vol. 3 - Issue 9 (September - 2014)
- Adrian, C., Corneliu, A., & Mircea, B. (2008, May). The simulation of the adaptive systems using the MIT rule. In *International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering* (pp. 301-305).
- Alqaudi, B., Modares, H., Ranatunga, I., Tousif, S. M., Lewis, F. L., & Popa, D. O. (2016). Model reference adaptive impedance control for physical human-robot interaction. *Control Theory and Technology*, *14*(1), 68-82.
- Ampsefidis, A. J., Białasiewicz, J. T., & Wall, E. T. (1993). Lyapunov design of a new model reference adaptive control system using partial a priori information. *Kybernetika*, *29*(4), 339-350.
- B. Bohari, L.N.M. Ismail, R. Varatharajoo, Spacecraft pitch and roll/yaw actuations using control momentum gyroscopes for attitude stabilization, Journal Mekanikal 23 (2007) 1–14.
- Black, W. S., Haghi, P., & Ariyur, K. B. (2014). Adaptive systems: History, techniques, problems, and perspectives. *Systems*, *2*(4), 606-660.s
- Coman, S., & Boldisor, C. (2014). Adaptive PI controller design to control a mass-damper-spring process. *Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I*, *7*(2), 69.
- Coman, S., & Boldisor, C. (2013). Model reference adaptive control for a DC electrical drive. *Bull. Transilvanic Univ. Brasov Ser. I: Eng. Sci*, *6*(2), 33-38.
- L.H. Geng, D.Y. Xiao, Q. Wang, T. Zhang, J.Y. Song, Attitude-control model identification of on-orbit satellites actuated by reaction wheels, Acta Astronautica 66 (2010) 714–721.

- Ehsani, M., Gao, Y., & Gay, S. (2003, November). Characterization of electric motor drives for traction applications. In *Industrial Electronics Society, 2003. IECON'03. The 29th Annual Conference of the IEEE* (Vol. 1, pp. 891-896). IEEE.
- Hassan, A. U., & Sudinb, S. (2014). Road Vehicle Following Control Strategy Using Model Reference Adaptive Control Method Stability Approach.
- Harrington, A. M., & Kroninger, C. (2013). *Characterization of small dc brushed and brushless motors* (No. ARL-TR-6389). ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD VEHICLE TECHNOLOGY DIRECTORATE.
- Han, Y., Biggs, J. D., & Cui, N. (2015). Adaptive Fault-Tolerant Control of Spacecraft Attitude Dynamics with Actuator Failures. *Journal of Guidance, Control, and Dynamics*, *38*(10), 2033-2042.
- Ismail, Zuliana and Renuganth, Varatharajoo, A study of reaction wheel configurations for a 3-axis satellite attitude control – Advances in Space Research 45 (2010) 750 – 759
- Jaeyoung H, Sangseok Y, Sun Y, "Advanced thermal management of automotive fuel cells using a model reference adaptive control algorithm" (February 2017). International Journal of Hydrogen Energy, volume 42, issue 7, pages 4328-4341
- Jain, P., & Nigam, M. J. (2013). Design of a model reference adaptive controller using modified MIT rule for a second order system. *Advance in Electronic and Electric Engineering*, *3*(4), 477-484.
- K.D. Kumar, M.J. Tahk, H.C. Bang, Satellite attitude stabilization using solar radiation pressure and magnetotorquer, Control Engineering Practice 17 (2009) 267–279.
- Kilic, E., Ozcalik, H.R. and Yilmaz, S., 2016. Efficient speed control of induction motor using RBF based model reference adaptive control method. *automatika*, *57*(3), pp.714-723.
- M. Lovera, A. Astolfi, Global magnetic attitude control of spacecraft, in: 43rd IEEE Conference on Decision and Control, 2004, pp. 267–272.

- Leduc, H., Pittet, C., & Peaucelle, D. (2017). Adaptive attitude control of a microsatellite during payload deployment.
- Meyer, J., N. Delson, and R. de Callafon, Design, modeling and stabilization of a moment exchange based inverted. Preprints of the 15th IFAC Symposium on System Identification, Saint-Malo, France, July 6-8, 2009
- Mohanarajah, Gajamohan, Michael Merz, Igor Thommen and Raffaello D'Andrea, The Cubli: A Cube that can Jump Up and Balance, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. October 7 – 12, 2012. Vilamoura. Algarve, Portugal
- Muehlebach, Michael, Gajamohan Mohanarajah, and Raffaello D'Andrea, Nonlinear Analysis and Control of a Reaction Wheel-based 3D Inverted Pendulum
- Muehlebach , Michael, Tobias Widmer and Raffaello D'Andrea, The Cubli: A Reaction Wheel Based 3D Inverted Pendulum, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. October 7 – 12, 2012. Vilamoura. Algarve, Portugal
- M. Munshi and S. G. Choudhuri, "Model Reference Adaptive System using Rotor Flux and Back Emf techniques for speed estimation of an Induction Motor operated in Vector Control mode: A comparative study," *2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON)*, Varanasi, India, 2016, pp. 44-49. doi: 10.1109/UPCON.2016.7894622
- D. van Niekerk, M. Case and D. V. Nicolae, "Brushless direct current motor efficiency characterization," *2015 Intl Aegean Conference on Electrical Machines & Power Electronics (ACEMP), 2015 Intl Conference on Optimization of Electrical & Electronic Equipment (OPTIM) & 2015 Intl Symposium on Advanced Electromechanical Motion Systems (ELECTROMOTION)*, Side, 2015, pp. 226-231. doi: 10.1109/OPTIM.2015.7426988

- Nudehi, Shahin S., Umar Farooq, Aria Alasty and Jimmy Issa, Satellite attitude control using three reaction wheels, 2008 American Control Conference, Westin Seattle, Washington, USA, June 11-13, 2008
- Pankaj, S., Kumar, J. S., & Nema, R. K. (2011). Comparative analysis of MIT rule and Lyapunov rule in model reference adaptive control scheme. *Innovative Systems Design and Engineering*, *2*(4), 154-162.
- Sahamı, F., & Salamci, M. U. (2016, May). Decentralized model reference adaptive control design for nonlinear systems; application to cancer treatment. In *Carpathian Control Conference (ICCC), 2016 17th International* (pp. 635-640). IEEE.
- Seaton, D. Brushed DC Motor Control: Parameter Characterization, open loop and PI controller simulation.
- Şit, Sami, Erdal Kılıç, Hasan Rıza Özçalık, Mahmut Altun, and Ahmet Gani. "Model Reference Adaptive Control based on RBFNN for Speed Control of Induction Motors." (2016).
- M.J. Sidi, Spacecraft Dynamic and Control, Cambridge University Press, 1997.
- Rajiv R, Pankaj R, Performance Analysis Of A Second Order System Using MRAC, International Journal Of Electrical Engineering & Technology (IJEET), Volume 3, Issue 3, October - December (2012), pp. 110-120
- Rose, C. G., French, J. A., & O'Malley, M. K. (2014, February). Design and characterization of a haptic paddle for dynamics education. In *Haptics Symposium (HAPTICS), 2014 IEEE* (pp. 265-270). IEEE.
- Thu, K. M., & Igorevich, G. A. (2016, October). Modeling and design optimization for quadcopter control system using L1 adaptive control. In *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual* (pp. 1-5). IEEE.
- Wei, Y., Sun, Z., Hu, Y., & Wang, Y. (2016). On fractional order composite model reference adaptive control. *International Journal of Systems Science*, *47*(11), 2521-2531.
- Igrr, Pgollor, Raimohanska, Reaper7, Plinioseniore, Hallard, Kalonk, Lnxbil, Wemos, http://esp8266.github.io/Arduino/versions/2.0.0/doc/reference.html

- Tansel Yucelen, https://www.youtube.com/watch?v=pU5vq6rjmKk
- Beauregard, B. (2017). Improving the Beginner's PID – Introduction. Retrieved February 19, 2017, from brettbeauregard Project Blog: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pidintroduction/