SELF BALANCING MONOPOD

Ritesh Nayak
3rd Year, ECE Department
IIIT Naya Raipur
ritesh16101@iiitnr.edu.in

Anirudha Pratap singh
3rd Year, ECE Department
IIIT Naya Raipur
anirudha16101@iiitnr.edu.in

Swaraj Paikara
3rd Year, ECE Department
IIIT Naya Raipur
swaraj16101@iiitnr.edu.in

***Abstract*— This paper addresses the hardware implementation of the self-balancing monopod with the use of the PID controller (DCM algorithm) and simulation in MATLAB. Frequency and Time Domain analysis is done in MATLAB with proper PID Tuning.**

Keywords- PID, Frequency response, Inverted Pendulum

I. INTRODUCTION

Basically, the self-balancing platform consists of a platform which is balanced by the movement of three motors in opposite direction to the movement of the platform. Arduino Mega treat the tilt angles taken from IMU and give instruction to the respective servo motors to rotate by some angle depending on its previous position to balance or control the platform. IMU consists of ADXL345 Accelerometer and ITG3200 Gyroscope whose outputs are calibrated accurately by using FILTER to give the accurate angle. This angle is sent to PID or DCM algorithm which covers the error i.e. how far the current position of the platform is from the wanted set point (balancing point). The algorithm tries to reduce the error by altering the process control inputs. That's unusual speak for the technology that allows balancing. With advent of self-balancing devices, be it Segway, DIY, or TIPI, we five were fascinated with the futuristic scope that self-balancing devices hold, be it flying cars or compact car modules on two rollers, be it self-stabilized and Bluetooth controlled cameras clicking in courteous moves of Hollywood stars or be it a simple self-stabilizing skateboard, controlled by your gestures, the idea of self-stabilizing skateboard controlled by our leg movements did take rounds in our interested team. As of delving deep into huge knowledge pool of self-controlled and stabilized devices, the team felt to get firsthand information of different control mechanisms, IMUs, filters, robust mechanical system, and henceforth, concluded to engineer a manually controlled-cum-self stabilizing platform with three axes of freedom

II BLOCK DIAGRAM

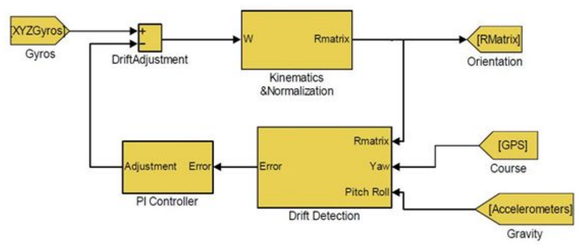
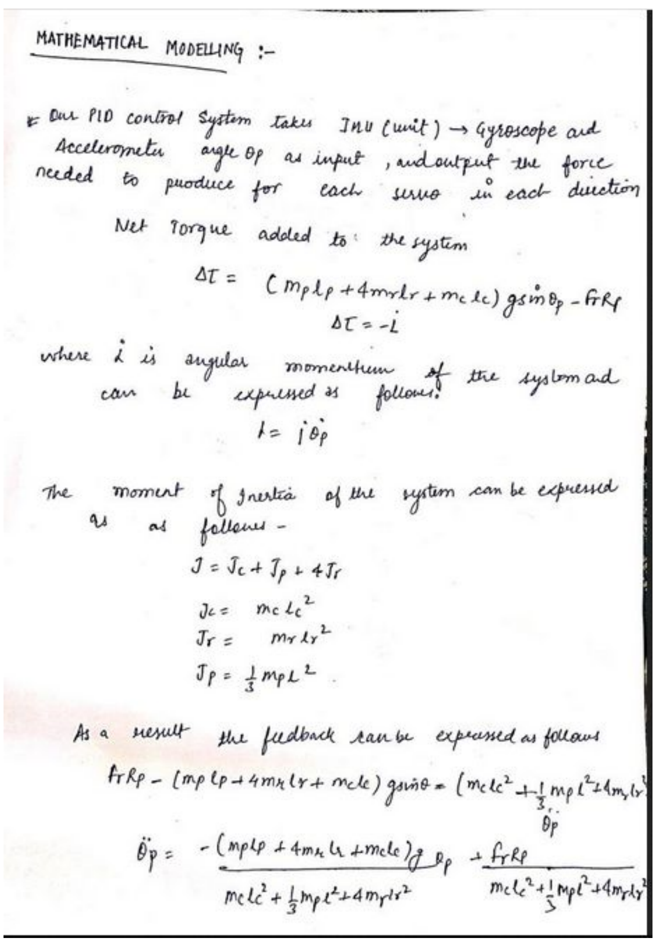


Fig 1 Design of the proposed project

II. CONTROL OBJECTIVE

- To illustrate the techniques involved in balancing a platform.
- To work on exact movements and accurate control of the platform, with the use of various algorithms and filtering process
- To illustrate the techniques involved in balancing a platform.
- To work on precise movements and accurate control of the platform, with the use of various algorithms and filtering process
- To understand the working of IMU. IMU work involves understanding the pin configurations of the IMUIMU and configuring the correct libraries for the IMU.
- To know the correct connections required for all the peripheral hardware to communicate with the microcontroller.

III. MATHEMATICAL MODELLING



MATHEMATICAL MODELLING :-

* Our PID control System takes IMU (unit) → Gyroscope and Accelerometer angle θ_p as input, and output the force needed to produce for each servo in each direction

Net Torque added to the system

$$\Delta \tau = (m_p l_p + 4 m_r l_r + m_c l_c) g \sin\theta_p - F_r R_f$$

$$\Delta \tau = -\dot{L}$$

where $\dot{L}$ is angular momentum of the system and can be expressed as follows:

$$L = J \dot{\theta}_p$$

The moment of Inertia of the system can be expressed as as follows -

$$J = J_c + J_p + 4 J_r$$

$$J_c = m_c l_c^2$$

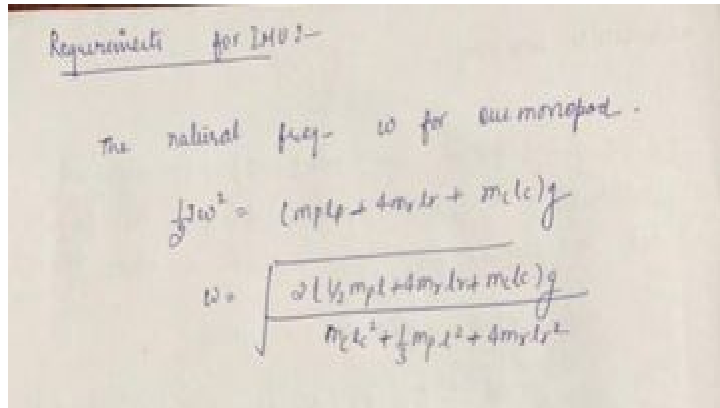
$$J_r = m_r l_r^2$$

$$J_p = \frac{1}{3} m_p L^2$$

As a result the feedback can be expressed as follows

$$F_r R_p - (m_p l_p + 4 m_r l_r + m_c l_c) g \sin\theta = (m_c l_c^2 + \frac{1}{3} m_p l^2 + 4 m_r l_r^2) \ddot{\theta}_p$$

$$\ddot{\theta}_p = \frac{-(m_p l_p + 4 m_r l_r + m_c l_c) g}{m_c l_c^2 + \frac{1}{3} m_p l^2 + 4 m_r l_r^2} \theta_p + \frac{F_r R_p}{m_c l_c^2 + \frac{1}{3} m_p l^2 + 4 m_r l_r^2}$$

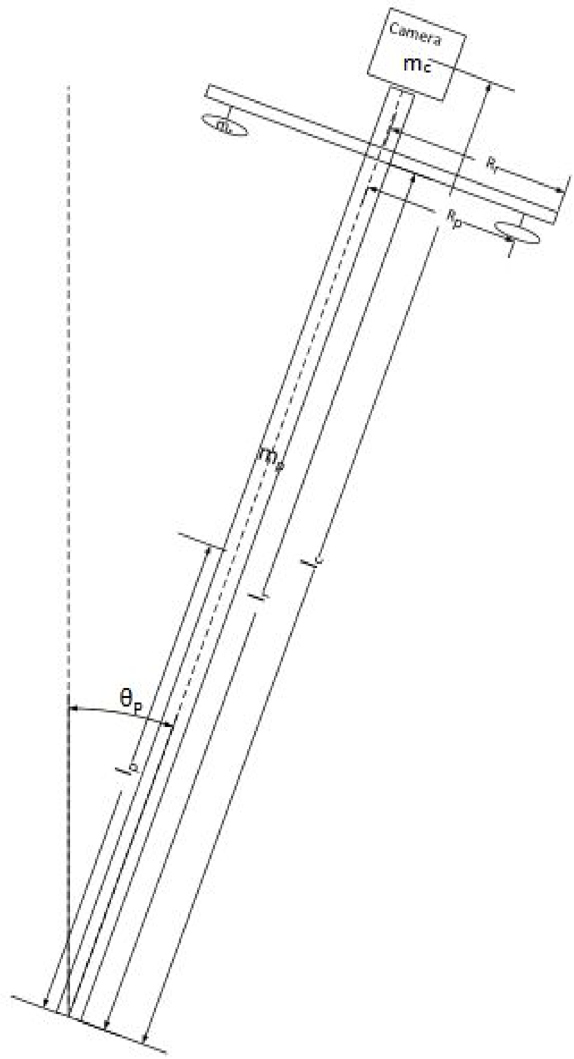


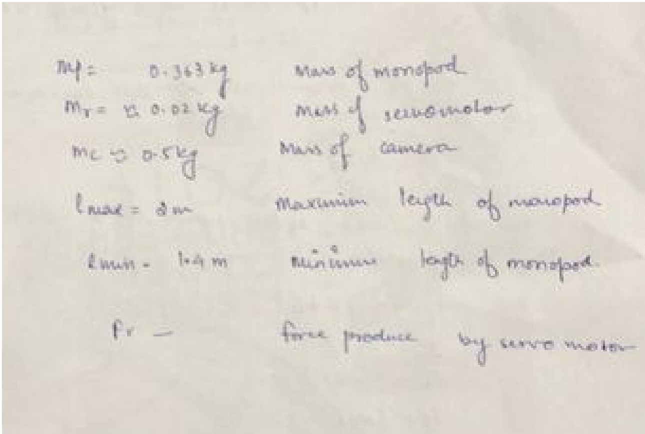
Requirements for IMU :-

The natural freq- ω for our monopod -

$$\frac{1}{2} J \omega^2 = (m_p l_p + 4 m_r l_r + m_c l_c) g$$

$$\omega = \sqrt{\frac{2(\frac{1}{2} m_p l + 4 m_r l_r + m_c l_c) g}{m_c l_c^2 + \frac{1}{3} m_p l^2 + 4 m_r l_r^2}}$$





IV OBSERVATION

A. TIME DOMAIN ANALYSIS

we derived the open-loop transfer functions of the system as the following.

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad [\frac{rad}{N}]$$

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad [\frac{m}{N}]$$

where

$$q = (M+m)(I+ml^2) - (ml)^2$$

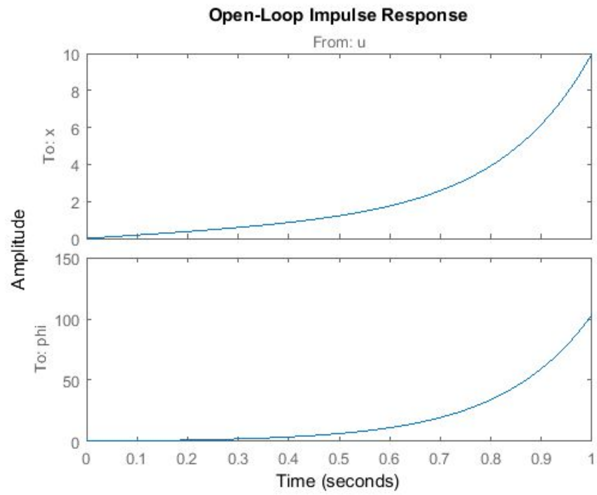
Recall that the above two transfer functions are valid only for small values of the angle ϕ, which is the gyroscopic and accelerometer values of the servo motor from the vertically upward position.

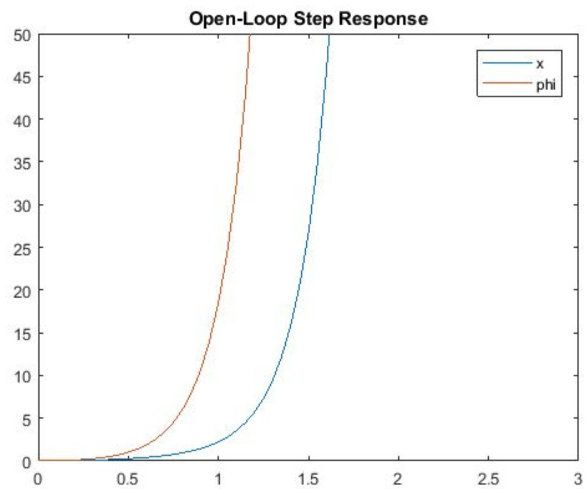
Considering the response of the pendulum to a 1-Nsec impulse applied to the cart, the design requirements for the system are:

- Settling time for θ of less than 5 seconds

Additionally, the specifications for the response of the system to a 0.2-meter step command in cart position are:

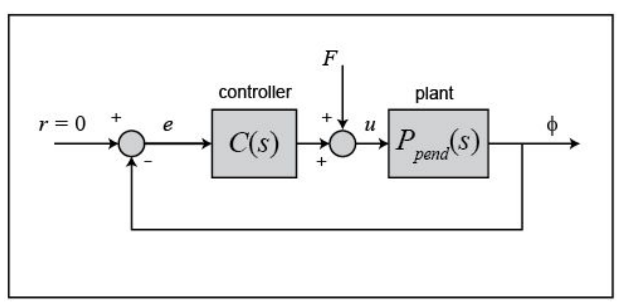
- Settling time for x and θ of smaller than 5 seconds
- Rise time for x of smaller than 0.5 seconds
- Servo motor angle θ never more than 20 degrees (0.35 radians) from the vertical



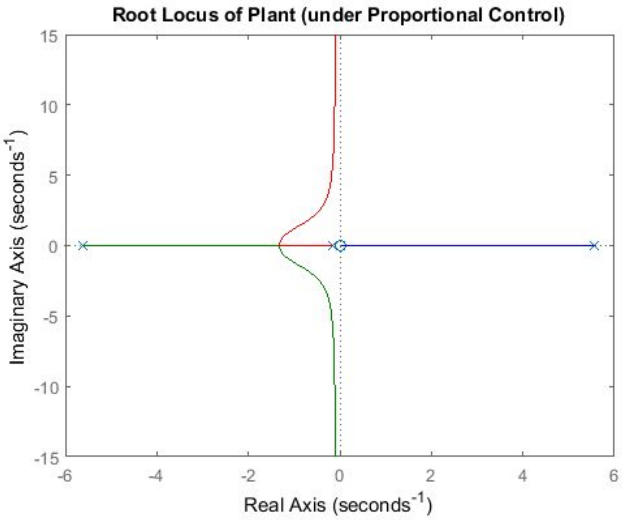


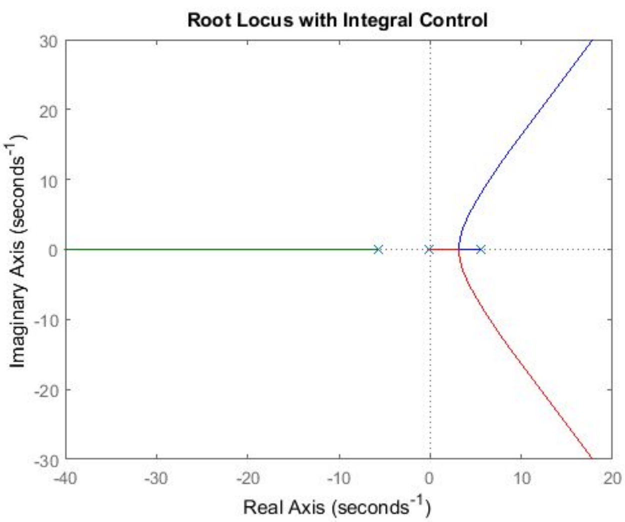
Open-loop step Response

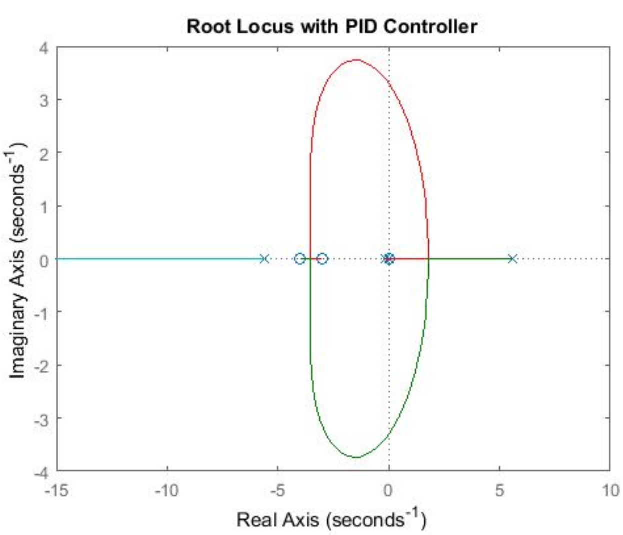
B. ROOT LOCUS ANALYSIS



$$T(s) = \frac{\Phi(s)}{F(s)} = \frac{P_{pend}(s)}{1 + C(s)P_{pend}(s)}$$





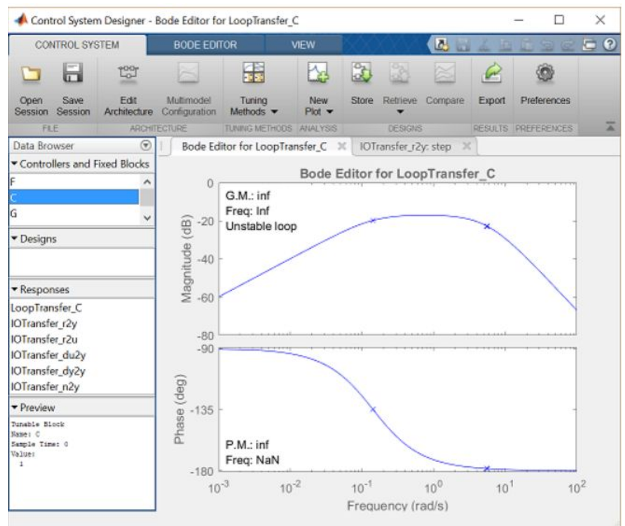


C. FREQUENCY DOMAIN ANALYSIS

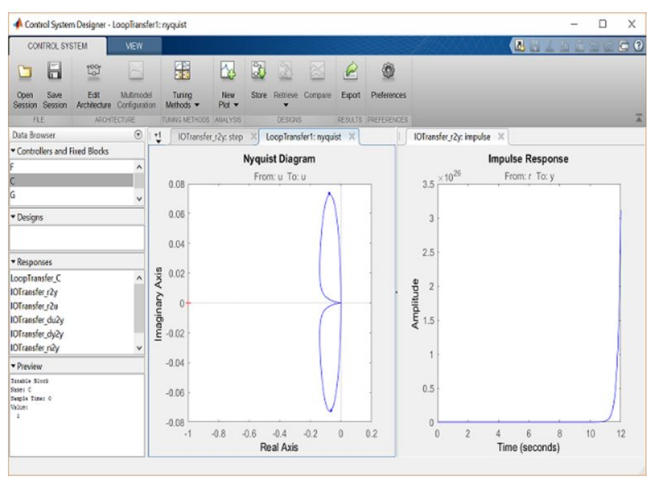
The controller we are planning will specifically attempt to maintain the platform vertically upward when the system is subjected to a 1-Nsec impulse. Under these conditions, the design guidelines are:

- Settling time of fewer than 5 seconds

Closed-loop response without compensation:-



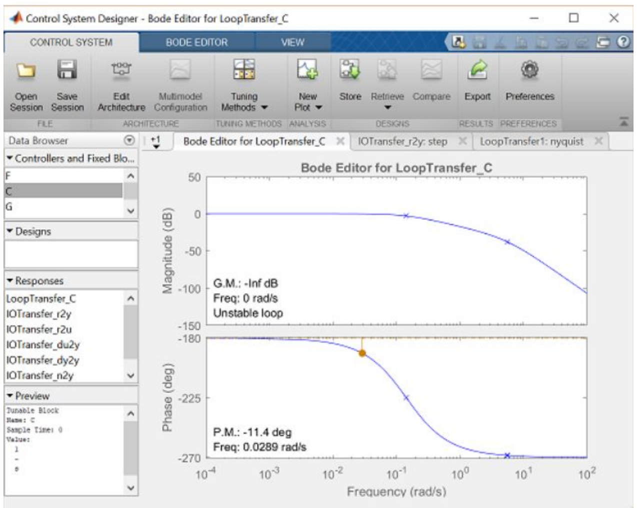
Bode Plot



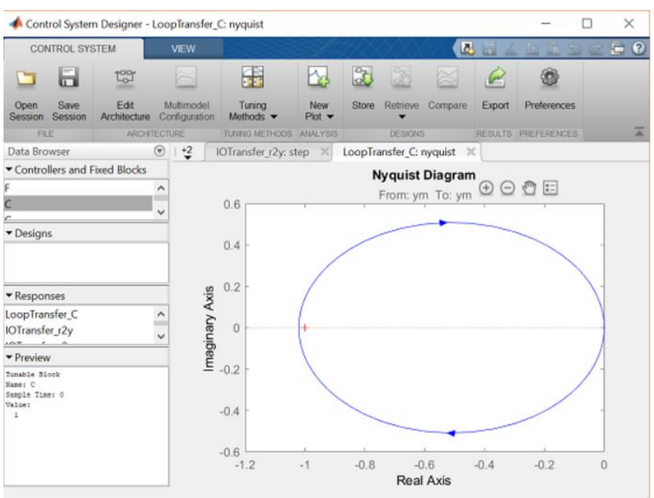
Nyquist Plot

Closed-loop response with compensation

Since the closed-loop system is fissionable without compensation, we need to use our controller to stabilize the system and meet the given requirements. Our first step will be to attach an integrator to eliminate the zero at the origin.

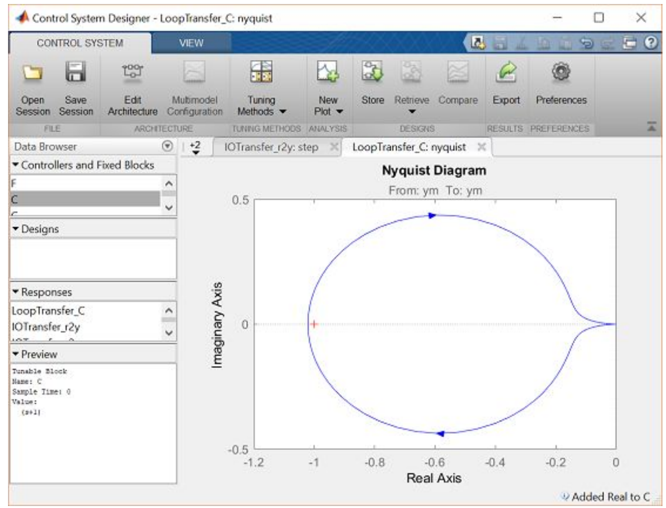


if you study the transfer function closely, you will notice that there is a pole-zero removal at the origin. Even with the extension of this integrator, the closed-loop system is still unstable.

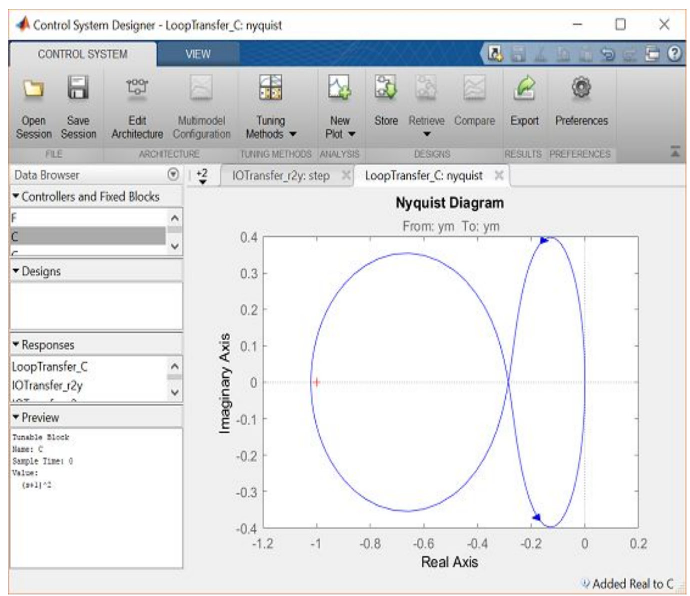


Remark that the open-loop Nyquist plot now encircles the -1 point in the clockwise direction. This implies that the closed-loop system now has two poles in the right-half plane ($Z = P + N = 1 + 1 = 2$). Hence, the closed-loop system is still unstable. We require to add phase in order to get a counterclockwise encirclement. We will do this by attaching a zero to our controller. For beginners, we will place this zero at -1 and view the resulting plots.

This extra zero will automatically change the Bode and Nyquist plots that are already open. The resulting Nyquist plot should resemble as shown below.



As you can see, this change did not provide enough phase. The encirclement around -1 is still clockwise. We will try adding a second zero at -1 in the same manner as was described above. The resulting Nyquist diagram is given below.



IV DESIGN AND TUNNING OF CONTROLLER

The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output, the final form of the PID algorithm is:

Where:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

DCM can be thought of a strong and robust algorithm for precise control of servo motors. It uses Euler angles, Direction Cosine Matrix(DCM) and Quaternion approach. It has inbuilt Filters and Proportional – Integral Control Units which in turn is very effective in giving calibrated outputs

The DCM algorithm uses Proportional-Integral Control units which are almost similar to PID except for the absence of Integral term. The tuning values, proportional constant (*Kp)* and derivative constant (*Kd)* changes in the code depending on how far the position of the platform is from setpoint (balanced point. Recognizing that numerical errors, gyro drift, and gyro offset will gradually accumulate errors in the DCM elements, we use reference vectors to detect the errors, and a proportional plus integral (PI) negative feedback controller between the detected errors and the gyro inputs, to dissipate the errors faster than they can build up. GPS is used to detect yaw error, accelerometers are used to detect pitch and roll.

V CONCLUSION

Through this project we conclude that self-balancing monopod works on the principle of inverted pendulum whose time-domain and frequency domain analysis is done and gets simulated in MATLAB software Apart from that hardware implementation is done with the use of DCM algorithm (PID Algorithm).

Future Work: -

- To Reduce the error in the system
- To calibrate the output more accurately
- To reduce the instability caused due to disbalance of the centre of mass of hardware model.

Acknowledgement

All the group members would like to acknowledge the IOT Lab and IIIT-NR for providing the necessary resources and facilities to implement the project successfully.

References

24

[1]http://www.engr.usask.ca/classes/EE/480/Inverted%20Pendulum.pdf

[2]http://students.iitk.ac.in/eclub/assets/documentations/summer13/Self%20Balancing%20Platform.pdf