Zewail City of Science and Technology

Control Systems
REE 310

Self Balancing Robot

Ahmed Adel	201901464
Ahmed Ibrahim	201902227
Gehan Sherif Helmy	201901989
Mohamed Ali Saleh	201902216
Muhammad Khalid Anwer	201901493
Osama Shehata	201900776

June 21, 2022

Contents

1 Introduction and background

A self-balancing robot can be thought of as a car with two wheels and additionally requiring maintaining stability and balance. Balancing is achieved by continuously measuring the angle of deviation between the robot's normal axis and the vertical axis, and, in case of a slight deviation, the robot starts moving in the direction that gives it momentum to reverse the deviation. Stability is achieved by providing the ability to move the robot while remaining balanced at the same time. Upon achieving stability, the self-balancing robot could be used as a light transportation method with less power consumption and less equipment than a similar-sized car.

Self-balancing robots have numerous applications. An example is **Segway** robot, provided is shown in fig. 1. This robot is commonly used on short trips as to the supermarket , for example, and is powered by electricity rather than harmful gases. **Segway** is capable of holding people up to 70KG, which makes it a perfect transportation solution for short distances for many people. It was developed such that it maintains stability and reduces safety risks. Another application for maintaining the stability of two wheels is **Hoverboard**, which could be thought of as a really nice game. It has widespread in Egypt, especially in public gardens.

This project provides an implementation of a small-scale prototype for a self-balancing robot. The aim is to make the robot able to balance and stabilize it self only with no capabilities to carry loads.

Figure 1: Segway

Figure 2: Hoverboard

2 Literature review

This section represents analysis of two papers on the same topic.

2.1 Design and Control of Two-wheeled Self-Balancing Robot using Arduino [1]

Authors of this paper tried to provide a detailed review of achievements in building self-balancing robots. They had three main components for their system, controlled by **Arduino nano**.

Firstly, they used a stepper motor powered by the motor driver; they justified their choice of using a stepper motor rather than a regular DC motor by stating that the stepper motor is more accurate in movements, making

it more stable.

The second main component they used was the **MPU 6050** sensor for calculating the acceleration and angle between robot's normal axis and the vrtical axis. The two parameters were then used to control the behavior of the system.

Finally, they used **PID controller**. A **Bluetooth module** was used for communication between the system and mobile phone application. The mathematical expression for their PID controller is shown in equation 1.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{1}$$

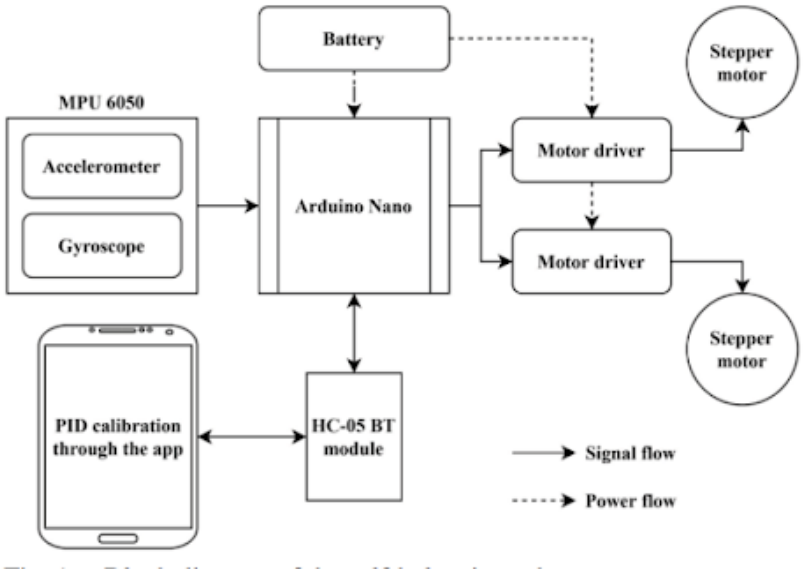


Figure 3: Block diagram of Singh Shekhawat paper

2.2 Development of Self-Balancing Robot [2]

Professor Baskaran and his student **Raja** provided us with some considerations for choosing specific parts of the robot. They started by defining the self-balancing robot as an inverted pendulum and stated that the main goal of their project was to maintain the center of the robot under the wheels, consequently if the robot tilted forwarded, the wheels should move forward to maintain the condition; otherwise, it would fall down.

They also used two motors for the two wheels and indicated that the proper value for torque to maintain the stability would be halved between the two wheels. Furthermore, they stated that the wheels must have at least 30-45 mm rubber to be able to overcome most terrains and weather conditions.

The paper starts by assuming that the robot moves only in one direction, they did that as it would simplify their calculations a lot. Professor Baskara used some main components for his project, like **Arduino Uno**, **motor controller**, **motor**, **MPU 6050**, and **PID controller**. Their block diagram is provided in fig. 4.

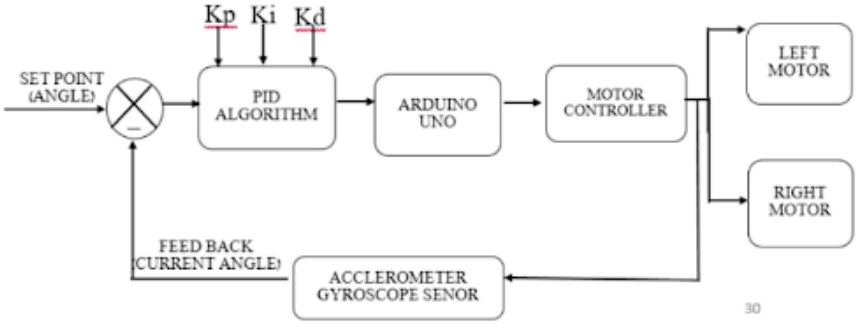


Figure 4: Block diagram of Professor Baskara's paper

3 Functional Block diagram

The reference is the angle that we want to reach in order to make the robot balanced (pi/2). The input to the arduino board is the error signal e(t) which is the difference between the reference that we want to reach and the actual angle that is measured by the sensor MPU6050. After the error is given to the Arduino Board, we use a PID Controller to improve the signal and try to make it as close as possible to the reference signal. The block diagram for the PID controller is as follow: The proportional constant is to make the output signal from

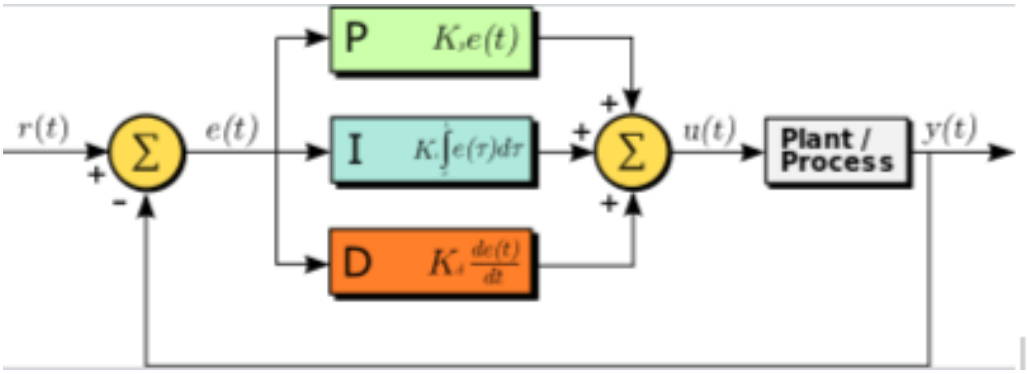


Figure 5: PID Controller Block Diagram

the Arduino as close as possible to the reference signal, the integral constant is to minimise steady state error and the differential constant is to decrease overshoot/undershoot. After the PID controller performs its action. We pass the signal to a Motor control algorithm. The output of such an algorithm is the PWM which will control the speed of the motor. The PWM is passed to a motor driver l298n. The output voltage of the L289N is then passed to two motors. And the two motors are connected to the wheels. Using the sensor MPU6050 we measure the current angle of the wheels and take it as feedback to construct the error signal e(t). The whole block diagram is as follow:

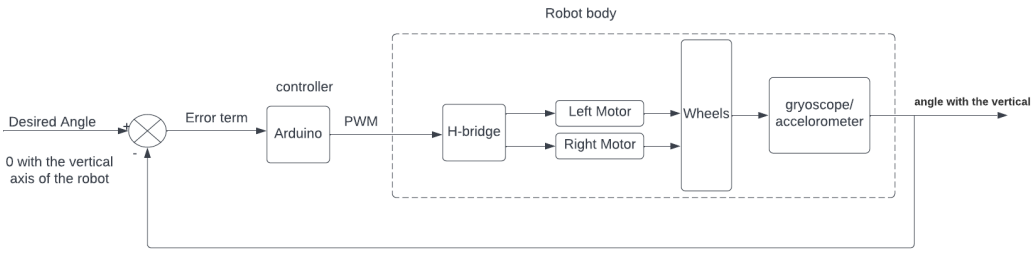


Figure 6: Self Balancing Robot Block Diagram Using PID Controller

4 Schematic diagram

Components used in the schematic:

- **Arduino Uno**: Arduino Uno is simply a microcontroller, which has enough digital and analog pins for our project; as it has 14 I/O digital pins, and 6 I/IO analog pins. It's hardcoded using C-language, which makes it a perfect option for our responsive response. It's powered by only 6-12 V. Arduino Uno has an Analog-digital converter (ADC), but unfortunately, it does not have a Digital-analog converter(DAC), but We can easily use Pulse width modulation(PWM).
- **DC motor:** we use two Dc motors, motor for each wheel.
- **L298N motor driver:** We choose to use L298N as it is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors.
- **Accelerometer and gyroscope(MPU 6050):** Feedback system is the main point in a controlled system, by which the system can catch errors, and rapidly change its behaviors to maintain its stable state. For our problem, we must detect any inclination caused by external forces, our mathematical model for such a case is the vertical angle deviation, and acceleration of the robot it's forced to fall down in.
- **Batteries:** Many of us have engaged in car projects, and our problem was in batteries being done every couple of hours during the development stage, therapy, we determined to use rechargeable ones, we thought it would be much cheaper. We would also buy a batteries holder to hold four batteries each would give 3V; for a total of 12V. Finally, connect these batteries to Arduino via a DC connector.

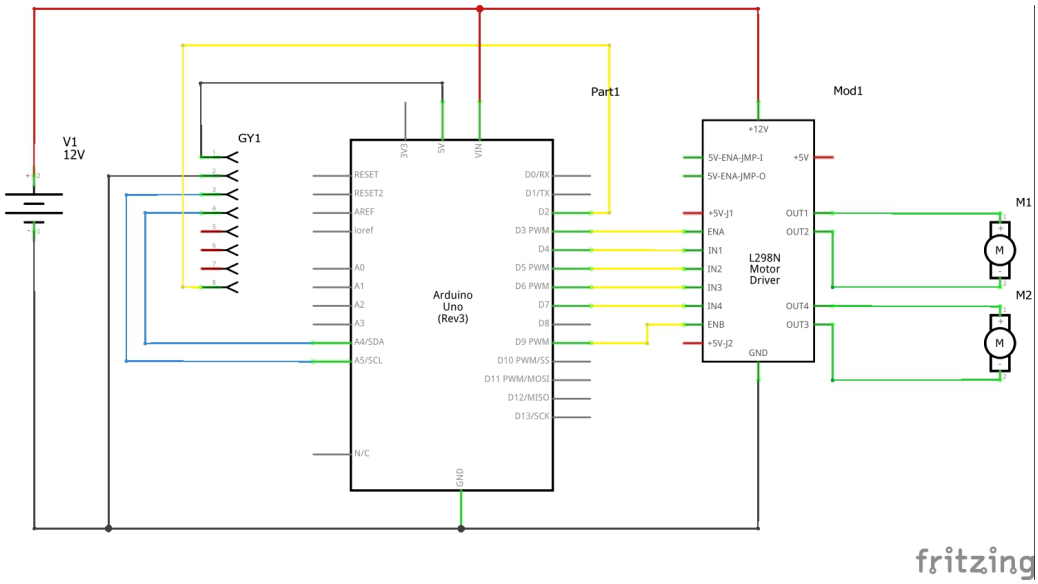


Figure 7

5 List of components

Component name	Quantity
Arduino Uno	x1
DC motor	x2
Dual H-bridge motor driver using L298N	x1
MPU 6050	x1
Battery	x1
DC connector	x1
Acrylic plastic sheets	x2

6 CAD Design For 3D Printed Parts

We will have to print seven parts. We used here SOLID-WORKS to design our robot. we provide different angle view for our robot in the following figures.

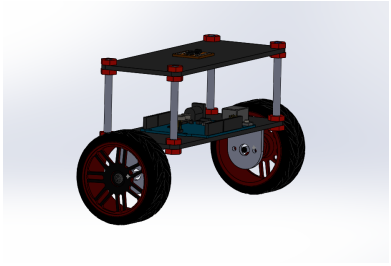


Figure 8

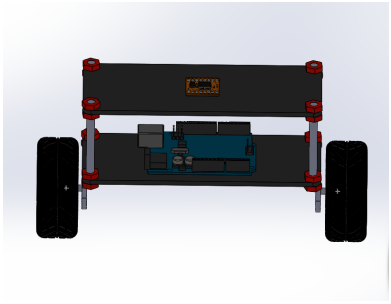


Figure 9

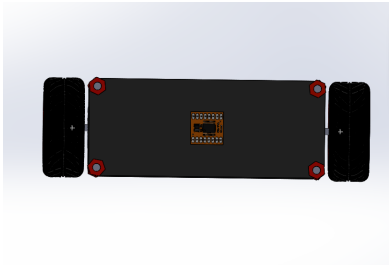
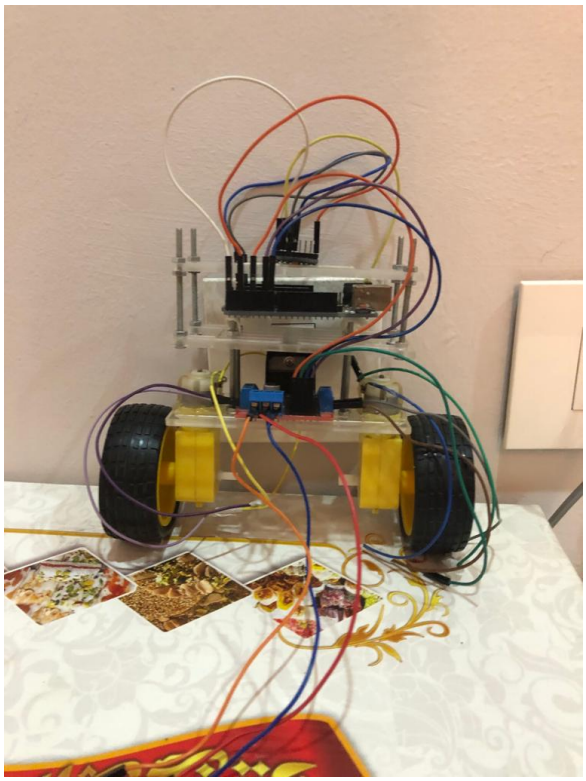
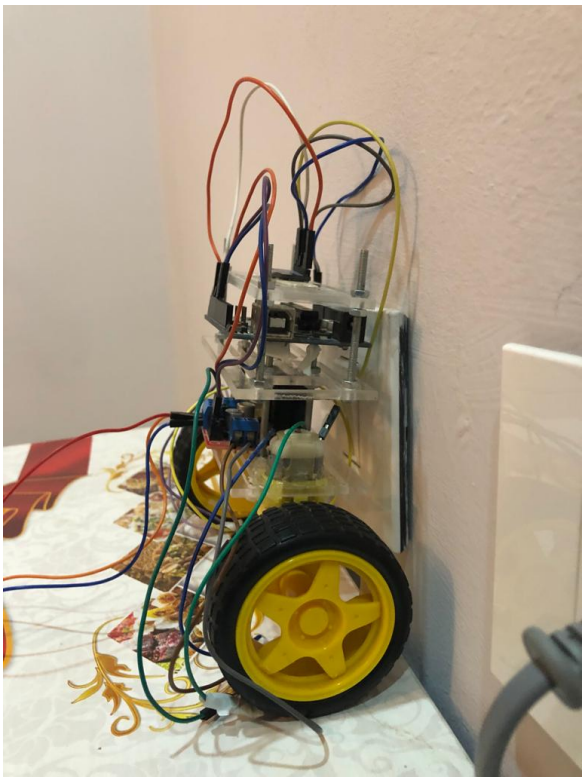


Figure 10

7 Hardware



(a)



(b)

8 Mathematical Model

We preform mathematical calculations to get the transfer function for the self-balancing robot system. We use the mechanical concept of the inverted pendulum because it is the same as two wheels self-balancing robot concept. The system is unstable and the absence of balancing force will let the pendulum fall. If the pendulum is not inverted, it is stable. Figure 1 represents the model for an inverted pendulum where θ is inclined angle, F is force given to the model, M denotes model mass, y and x are axis of vertical and horizontal, respectively. In our project, We will use DC motors to provide the appropriate velocity, so the robot remains balance. Without the appropriate velocity, the robot will be fallen.

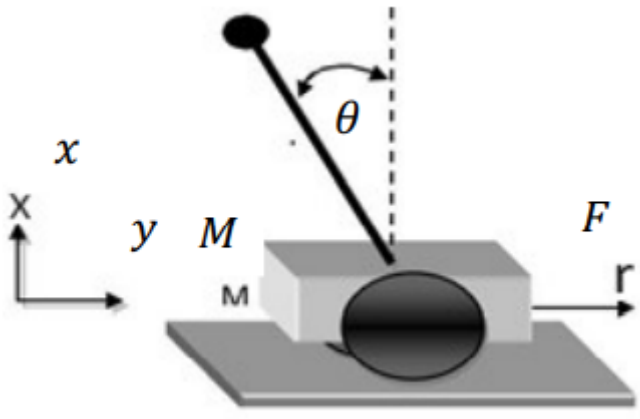


Figure 12: Inverted pendulum

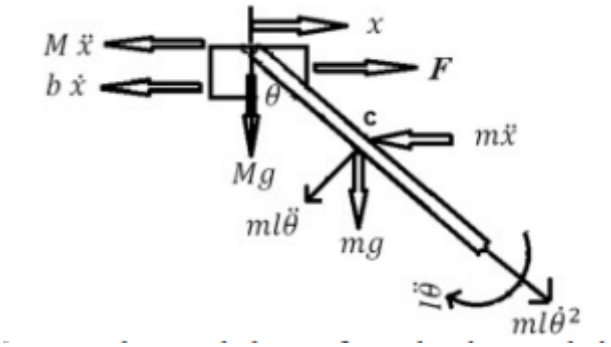


Figure 13: Free body diagram for inverted pendulum

The following table shows the parameters for the system:

Variable	Description	Value
M	Mass of the cart	0.03455 kg
m	Mass of pendulum rod	0.21276 kg
b	Viscous friction coefficient	0.1 N/m/sec
l	One half pendulum rod length	0.04575 m
I	Inertia moment robot	$5.93751x10^{-4}\ kg.m^2$
F	Applied force to the cart	$kg.m/s^2$
g	Gravity	m/s^2
θ	Pendulum angle	rad

Using Newton's second law:

$$\sum f_x = 0$$
$$F - b\dot{x} - M\dot{x} - m\ddot{x} - ml\ddot{\theta}cos\theta + ml\dot{\theta}^2 sin\theta = 0 \tag{2}$$

$$\sum f_y = 0$$
$$Mg + mg - ml\ddot{\theta}sin\theta - ml\dot{\theta}^2 cos\theta = 0 \tag{3}$$

$$\sum M_A = 0$$
$$mglsin\theta + I\ddot{\theta} + ml^2\ddot{\theta} + ml\ddot{c}os\theta = 0 \tag{4}$$

From Equations (1), (2) and (3):

$$(I + ml^2)\theta + mglsin\theta = -ml\ddot{x}cos\theta \tag{5}$$

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos\theta - ml\dot{\theta}^2 sin\theta = F \tag{6}$$

Equations (4) and (5) are two linear equations of the transfer function, where q = π. Assuming $\theta = \pi + \phi$,

$$cos\theta = -1 \tag{7}$$

$$sin\theta = -\phi \tag{8}$$

$$\frac{d^2}{dt^2} = 0 \tag{9}$$

After processing the equations (6), (7), and (8) approach to non-linear, then we obtained two variations of motion equations. The U value represents the input,

$$(I + ml)^2\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{10}$$

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\phi} = U \tag{11}$$

By applying Laplace transform on equations (9) and (10)

$$(I+ml^2)\ddot{\phi}(s)s^2 - mgl\phi(s) = ml\ddot{X}(s)s^2 \tag{12}$$

$$(M+m)\ddot{X}(s)s^2 + b\dot{X}(s)s + ml\phi(s)s^2 = U(s) \tag{13}$$

From equations (11) and (12), we get the transfer function for Two wheels self-balancing robot:

$$\frac{\phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \tag{14}$$

where q = $[(M+m)(I+ml^2) - (ml)^2]$
Substitute Using values form table (1) in equation (13):

$$\frac{\phi(s)}{U(s)} = \frac{59.98s}{s^3 + 0.64s^2 - 145.6s - 58.88} \tag{15}$$

9 System Analysis

Using MATLAB to convert the transfer function to Z-domain:

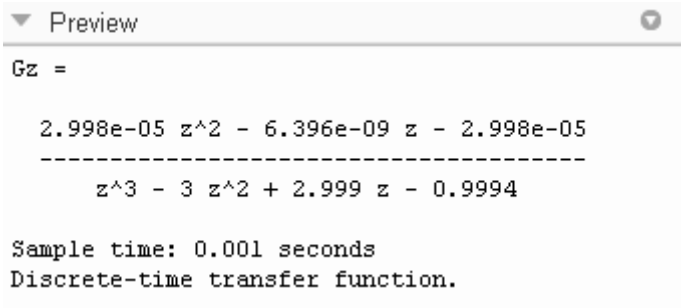


Figure 14

For this transfer function, we designed the following controller using PID Tuner and We have changed the variables of PID controller until we reach maximum overshoot 3.11%:

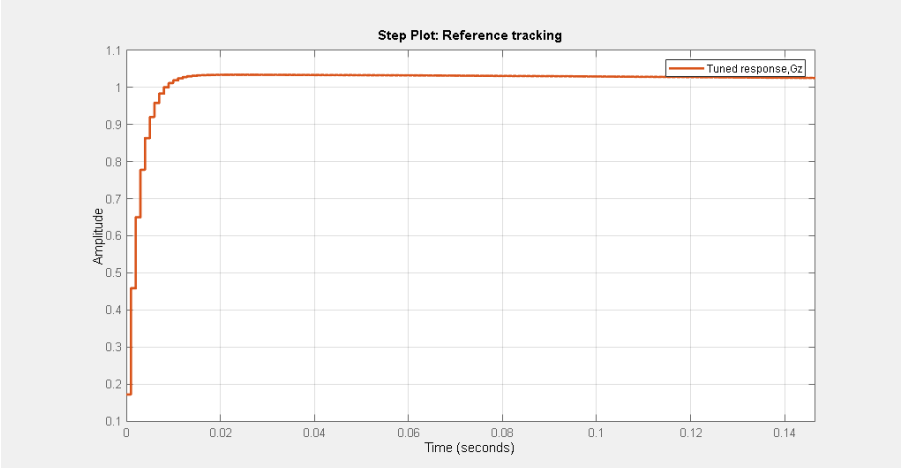


Figure 15

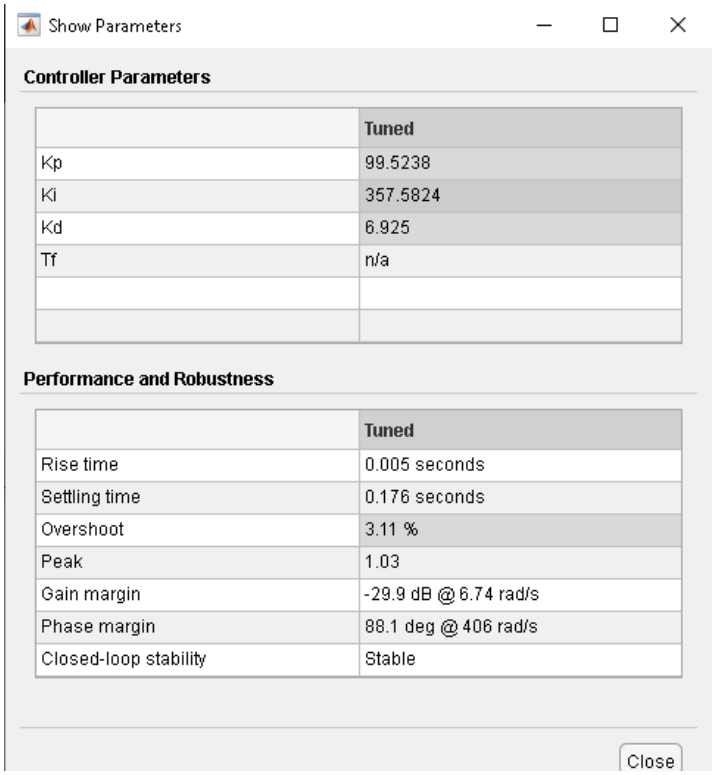

Show Parameters
Controller Parameters
Tuned
Kp
99.5238
Ki
357.5824
Kd
6.925
Tf
n/a
Performance and Robustness
Tuned
Rise time
0.005 seconds
Settling time
0.176 seconds
Overshoot
3.11 %
Peak
1.03
Gain margin
-29.9 dB @ 6.74 rad/s
Phase margin
88.1 deg @ 406 rad/s
Closed-loop stability
Stable
Close

Figure 16

10 References

[1] Mudeng, V., Hassanah, B., Priyanto, Y. and Saputra, O., 2020. Design and Simulation of Two-Wheeled Balancing Mobile Robot with PID Controller. [online] Unijourn.com. Available at: <https://unijourn.com/upload/doc/articleDoc-1587704304985-main.pdf>.

[2] Rohilla, Yogesh & Shekhawat, Anmol. (2020). Design and Control of Two-wheeled Self-Balancing Robot using Arduino. 10.1109/ICOSEC49089.2020.9215421.

[3] Shanmugavel, Baskaran & Raja, Karthik. (2018). Development of Self-Balancing Robot.