Electrical Science & Engineering

https://ojs.bilpublishing.com/index.php/ese

ARTICLE

Research on Self-balancing Two Wheels Mobile Robot Control System Analysis

Hla Myo Tun[1*] Myat Su Nwe[2] Zaw Min Naing[3] Maung Maung Latt[4] Devasis Pradhan[5] Prasanna Kumar Sahu[6]

1. Faculty of Electrical and Computer Engineering, Yangon Technological University, Yangon, Myanmar
2. Department of Mechanical Engineering, King Lauk Phya Institute of Technology Myaungmya, Myaungmya, Myanmar
3. Department of Research and Innovation, Ministry of Science and Technology, Yangon, Myanmar
4. Department of Electronic Engineering, Yangon Technological University, Yangon, Myanmar
5. Department of Electronics & Communication Engineering, Acharya Institute of Technology, Bengaluru, India
6. Department of Electrical Engineering, National Institute of Technology, Rourkela, Odisha, India

ARTICLE INFO

Article history
Received: 24 January 2022
Revised: 17 March 2022
Accepted: 28 March 2022
Published Online: 13 April 2022

Keywords:
Mobile robot
Self-balancing robot
Control system design
PID controller
Dynamic control system analysis

ABSTRACT

The paper presents the research on self-balancing two-wheels mobile robot control system analysis with experimental studies. The research problem in this work is to stabilize the mobile robot with self-control and to carry the sensitive things without failing in a long span period. The main objective of this study is to focus on the mathematical modelling of mobile robot from laboratory scale to real world applications. The numerical expression with mathematical modelling is very important to control the mobile robot system with linearization. The fundamental concepts of dynamic system stability were utilized for maintaining the stability of the constructed mobile robot system. The controller design is also important for checking the stability and the appropriate controller design is proportional, integral, and derivative – PID controller and Linear Quadratic Regulator (LQR). The steady state error could be reduced by using such kind of PID controller. The simulation of numerical expression on mathematical modeling was conducted in MATLAB environments. The confirmation results from the simulation techniques were applied to construct the hardware design of mobile robot system for practical study. The results from simulation approaches and experimental approaches are matched in various kinds of analyses. The constructed mobile robot system was designed and analyzed in the control system design laboratory of Yangon Technological University (YTU).

*Corresponding Author:
Hla Myo Tun,
Faculty of Electrical and Computer Engineering, Yangon Technological University, Yangon, Myanmar;
Email: hlamyotun@ytu.edu.mm

DOI: https: //doi.org/10.30564/ese.v4i1.4398

1. Introduction

Mobile robots are empowered for military and industries and became popular in helping domestic works like a four-wheel vacuum cleaner and hospitals like a motorised wheelchair, among those various robots such as a four-wheel robot, dog robot, spider robot, etc. [1,2]. The two-wheel self-balancing robot invented by Segway is one of the innovative robot solutions from personal robotics applications to private transportation. The idea of developing Segway personal transport, which can balance on its two wheels, has drawn interest from many researchers worldwide. Such kind of robot can solve several challenges in industry and society [3-5]. Because being a small and light cart allows humans to travel in small areas or factories where cars cannot enter with less pollution and less power consumption. Moreover, a Segway robot is interesting because it combines sensory capability and intelligent control methods to keep the robot in equilibrium [6-8].

The two-wheeled self-balancing robot is similar to the configuration of Segway based on inverted pendulum theory. The increasing affordability of commercial off-the-shelf (COTS) sensing components and microprocessor boards has become one of the key reasons to widely research and report. The two-wheeled self-balancing robot is a kind of robot that can balance itself on two wheels [9-11]. This robot chassis was designed in the SolidWorks software and fabricated with a laser cutter using acrylic materials. Then it was mounted on two 12 V DC motors 500 rpm with the encoder. In this project, an inertial measurement unit (IMU) called MPU6050, consisting of 3 axis accelerometer and a three-axis gyroscope, was chosen to measure the robot orientation (roll, pitch, and yaw). It consists of some weak points in the gyroscope and accelerometer when they are used alone [12].

Consequently, the data received from IMU were converted to the tilt angle and passed through one of the filters. A complimentary filter is used for data fusion to get the perfect angle. To balance the device, a controller plays an essential role in the self-balancing robot. PID controller is used in this project to control the speed of the motor via the PWM duty cycle. The robot's power can be given by using a battery placed on top of the chassis [13]. In this way, the aim of balancing the robot was fully achieved.

Because of higher speed due to less friction from fewer wheels, more straightforward structural design, lesser parts that need replacing over time, such as wheels, this two-wheeled self-balancing robot can be used in various applications with different perspectives such as an intelligent gardener in agricultural fields, an autonomous trolley in hospitals, shopping malls, offices, airports, healthcare applications or an intelligent robot to guide blind or disable people [14]. The general objectives to perform this research are to design and fabricate a relevant balancing two-wheeled robot chassis, to derive the mathematical model [15] of the system, to illustrate the schemes and techniques involved in balancing unstable robotic environments on two wheels using proper motors, sensors and a suitable microprocessor, and to design a complete stable discrete digital control system in the appropriate software.

In this work, we emphasized the derivation of a mathematical model based on the inverted pendulum theory, design of two-wheeled robot in SolidWorks software, implementation of the filter for IMU sensor and PID control, integration of the robot's controller with hardware, and testing, analysing and troubleshooting for the whole system. The authors have solved the stabilizing problem for the mobile robot with self-control and to carry the sensitive things without failing in a long span period. The detailed analyses and numerical expression on mathematical modellings were discussed in the following section. The research motivation on the mobile robot system implementation is how to link the simulation approaches to experimental approaches in real time applications.

The paper is organised as follows. Section II presents the dynamic mathematical model of the targeted mobile robot. Section III mentions the control system design. Section IV points out the hardware implementation. Section V gives the results and discussions on the theoretical and experimental studies. The last section is the conclusion of the research work.

2. Mathematical Modelling

The mathematical modelling for the self-balancing mobile robot system could be completed based on the respective section on the dynamical system appraoches. The first portion is concerning the linear model of a DC motor for that mobile robot and the second portion is the dynamic model of mobile robot.

2.1 Linear Model of a Direct Current (DC) Motor

Two 500 rpm DC motors are used to power the robot. In this section, the state-space model of the DC motor, which is necessary for deriving the balancing robot's dynamic model, is expressed. Consequently, the relationship between the input voltage to the motors and the control torque required to stabilise the robot is obtained.

Figure 1 expresses an effective linear model for a direct current motor. When a voltage is applied to the dc motor, a current (i) is generated. As a result, a motor torque, which is proportional to the current, is produced. The equation can be expressed as

$$\tau_m = k_m \times i \tag{1}$$

An electromotive force (emf) is induced when the coil of a motor is spinning through a magnetic field due to an applied voltage source. In other words, the motor's shaft is turning, and emf is generated when the motor is in an operation mode. Thus, back emf voltage increases proportionally to the shaft velocity w, which can be written as

$$V_{en} = k_e \times \omega \tag{2}$$

By using Kirchoff's Voltage Law, the linear differential equation of the DC motor's electrical circuit can be written as

$$-V_s + i \times R + L \times \frac{di}{dt} + V_{en} = 0 \tag{3}$$

$$\frac{di}{dt} = \frac{V_s}{L} - \frac{iR}{L} - \frac{V_{en}}{L} \tag{4}$$

$$\frac{di}{dt} = -\frac{R}{L} \times i - \frac{k_e}{L} \times \omega + \frac{V_s}{L} \tag{5}$$

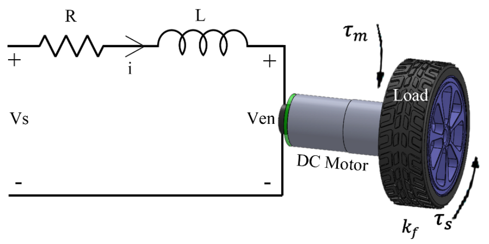

Figure 1. Diagram of a DC motor

According to Newton's Law of Motion, the summation of all torques produced on the shaft is linearly related to the acceleration of the shaft by the inertial load of the armature. The motion equation can be written

$$\sum M = I_R \times \alpha \tag{6}$$

$$\sum M = I_R \times \dot{\omega} \tag{7}$$

$$\tau_m - k_f \omega - \tau_s = I_R \dot{\omega} \tag{8}$$

$$k_m i - k_f \omega - \tau_s = I_R \frac{d\omega}{dt} \tag{9}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} i - \frac{k_f}{I_R} \omega - \frac{1}{I_R} \tau_s \tag{10}$$

Assume that the motor's inductance (L) and motor's friction (k_f) is approximately zero. Equation (3) and Equation (10) become Equation (12) and Equation (13), respectively.

$$-V_s + iR + V_e = 0 \tag{11}$$

$$i = -\frac{k_e}{R} \omega + \frac{1}{R} V_s \tag{12}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} i - \frac{1}{I_R} \tau_s \tag{13}$$

By substituting Equation (12) into Equation (13), an approximation for the DC motor, which is only a function of the current motor speed (ω), applied voltage (V_s) and applied torque (τ_s) is obtained.

$$\frac{d\omega}{dt} = \frac{k_m}{I_R}\left(-\frac{k_e}{R}\omega + \frac{1}{R}V_s\right) - \frac{1}{I_R}\tau_s \tag{14}$$

$$\frac{d\omega}{dt} = -\frac{k_m k_e}{I_R R}\omega + \frac{k_m}{I_R R}V_s - \frac{1}{I_R}\tau_s \tag{15}$$

A state-space model can represent the dynamic model of the motor. This state-space consists of two state variables: angle (θ) and velocity (ω). The input variables for the motor is the applied voltage (V_s) and applied torque (τ_s).

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{k_m k_e}{I_R R} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{k_m}{I_R R} & -\frac{1}{I_R} \end{bmatrix} \begin{bmatrix} V_s \\ \tau_s \end{bmatrix} \tag{16}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} V_s \\ \tau_s \end{bmatrix} \tag{17}$$

The outcome of Equation (17) is the estimated mobile robot position in experimental studies.

2.2 Dynamic Modelling of Mobile Robot System

This two-wheeled self-balancing robot is a kind of complex dynamic system because of the nature of an inverted pendulum on a cart. Two equations of motion, which can completely describe the heart of the balancing robot, are driven by analysing the robot's chassis and wheels separately.

The mathematical model will accommodate such forces as disturbances, and the motor's torque can influence the robot's behaviour. First of all, the equation of motion associated with the left and right wheels is obtained. Since the equation for the left and right wheels are absolutely analogous, only the equation for the right wheel is given. Figure 2 expresses the free body diagram for both wheels.

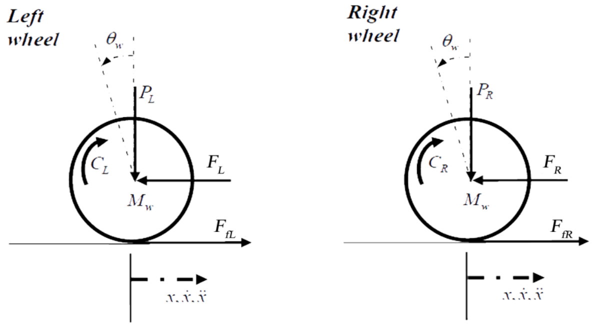

Figure 2. Free body diagram of the wheels

According to Newton's law of motion, the summation of forces on the horizontal x-direction is

$$\sum F_x = Ma \tag{18}$$

$$F_{fR} - F_R = M_w \ddot{x} \tag{19}$$

Adding the moments about the centroid of the wheel gives Equation (20).

$$\sum M_O = I\alpha \tag{20}$$

$$C_R - F_{fR} r = I_w \ddot{\theta}_w \tag{21}$$

The motor torque can be expressed from DC motor dynamics, and the equation can be described as,

$$\tau_m = I_R \frac{d\omega}{dt} + \tau_s \tag{22}$$

By rearranging the equation and substituting Equation (15) into Equation (22), the output torque to the wheels is attained.

$$\tau_m = I_R \left[-\frac{k_m k_e}{I_R R} \dot{\theta}_w + \frac{k_m}{I_R R} V_s - \frac{1}{I_R} \tau_s \right] + \tau_s \tag{23}$$

$$C = -\frac{k_m k_e}{R} \dot{\theta}_w + \frac{k_m}{R} V_s \tag{24}$$

Therefore, Equation (4) becomes

$$-\frac{k_m k_e}{R} \dot{\theta}_w + \frac{k_m}{R} V_s - F_{fR} r = I_w \ddot{\theta}_w \tag{25}$$

Thus,

$$F_{fR} = -\frac{k_m k_e}{Rr} \dot{\theta}_w + \frac{k_m}{Rr} V_s - \frac{I_w}{r} \ddot{\theta}_w \tag{26}$$

The equation of left and right wheels can be obtained by substituting Equation (26) into Equation (19).

For the right wheel,

$$M_w \ddot{x} = -\frac{k_m k_e}{Rr} \dot{\theta}_w + \frac{k_m}{Rr} V_s - \frac{I_w}{r} \ddot{\theta}_w - F_R \tag{27}$$

For the left wheel,

$$M_w \ddot{x} = -\frac{k_m k_e}{Rr} \dot{\theta}_w + \frac{k_m}{Rr} V_s - \frac{I_w}{r} \ddot{\theta}_w - F_L \tag{28}$$

The angular rotation can be converted into linear motion by straightforward transformation because the linear motion is acting on the central point of the wheel.

$$\ddot{\theta}_w r = \ddot{x} \rightarrow \ddot{\theta}_w = \frac{\ddot{x}}{r} \tag{29}$$

$$\dot{\theta}_w r = \dot{x} \rightarrow \dot{\theta}_w = \frac{\dot{x}}{r} \tag{30}$$

By the linear transformation, Equation (27) and Equation (28) becomes:

For the right wheel,

$$M_w \ddot{x} = -\frac{k_m k_e}{Rr^2} \dot{x} + \frac{k_m}{Rr} V_s - \frac{I_w}{r^2} \ddot{x} - F_R \tag{31}$$

For the left wheel,

$$M_w \ddot{x} = -\frac{k_m k_e}{Rr^2} \dot{x} + \frac{k_m}{Rr} V_s - \frac{I_w}{r^2} \ddot{x} - F_L \tag{32}$$

By adding Equation (31) and Equation (32), the first equation of motion for the balancing robot is obtained.

$$2\left[M_w + \frac{I_w}{r^2} \right] \ddot{x} = -\frac{2k_m k_e}{Rr^2} \dot{x} + \frac{2k_m}{Rr} V_s - \left(F_L + F_R\right) \tag{33}$$

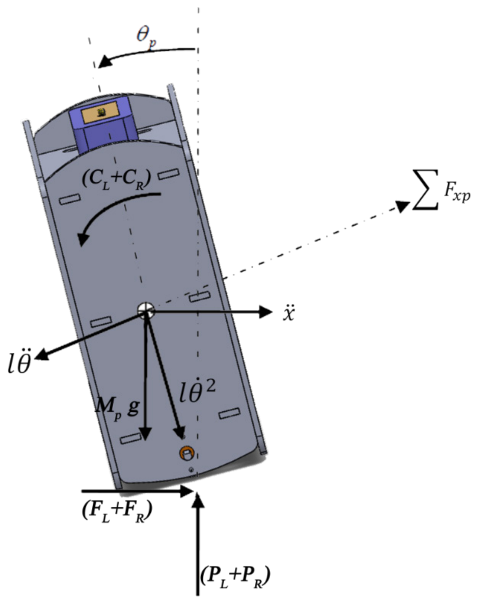

Figure 3. Free Body Diagram of the Chassis

The free body diagram of the chassis is illustrated in Figure 3. In this figure, the Newton's law of motion for self-balancing mobile robot dynamic system based on the mathematical expressions. The mathematical modelling of that free body diagram could be evaluated in the following section.

$$\sum F_P = M_P \ddot{x} \tag{34}$$

$$\left(F_L + F_R\right) - M_p l \ddot{\theta}_p \cos\theta_p + M_p l \dot{\theta}_p^2 \sin\theta_p = M_p \ddot{x} \tag{35}$$

The self-motion could be observed by changing the parameter of "p" for left and right position.

Thus

$$\left(F_L + F_R\right) = M_p l \ddot{\theta}_p \cos\theta_p - M_p l \dot{\theta}_p^2 \sin\theta_p + M_p \ddot{x} \tag{36}$$

Summing the forces perpendicular to the chassis can also give the second equation of motion for the system.

$$\sum F_{xp} = M_p \ddot{x} \cos\theta_p \tag{37}$$

$$\begin{aligned}&\left(F_L+F_R\right)\cos\theta_p+\left(P_L+P_R\right)\sin\theta_p\\&-M_p g\cos\theta_p-M_p l\ddot{\theta}_p=M_p\ddot{x}\cos\theta_p\end{aligned}\quad(38)$$

To get rid of the F and P terms in the above equation, by summing the moments around the centre of mass of chassis, Equation (39) is obtained.

$$\sum M_O=I\alpha\quad(39)$$

$$\begin{aligned}&-\left(F_L+F_R\right)l\cos\theta_p-\left(P_L+P_R\right)l\sin\theta_p\\&-\left(C_L+C_R\right)=I_p\ddot{\theta}_p\end{aligned}\quad(40)$$

The torque applied on the chassis from the motor as defined in Equation (40) and after linear transformation,

$$\left(C_L+C_R\right)=-\frac{2k_m k_e}{Rr}\dot{x}+\frac{2k_m}{R}V_s\quad(41)$$

Substituting this into Equation (40) gives,

$$\begin{aligned}&-\left(F_L+F_R\right)l\cos\theta_p-\left(P_L+P_R\right)l\sin\theta_p\\&-\left(-\frac{2k_m k_e}{Rr}\dot{x}+\frac{2k_m}{R}V\right)=I_p\ddot{\theta}_p\end{aligned}\quad(42)$$

Thus,

$$\begin{aligned}&-\left(F_L+F_R\right)l\cos\theta_p-\left(P_L+P_R\right)l\sin\theta_p\\&=I_p\ddot{\theta}_p-\frac{2k_m k_e}{Rr}\dot{x}+\frac{2k_m}{R}V_s\end{aligned}\quad(43)$$

By multiplying Equation (38) by –l,

$$\begin{aligned}&-\left[\left(F_L+F_R\right)l\cos\theta_p+\left(P_L+P_R\right)l\sin\theta_p\right]\\&+M_p l\mathrm{g}\sin\theta_p+M_p l^2\ddot{\theta}_p=-M_p l\ddot{x}\cos\theta_p\end{aligned}\quad(44)$$

Substitute Equation (43) in Equation (44),

$$\begin{aligned}&I_p\ddot{\theta}_p-\frac{2k_m k_e}{Rr}\dot{x}+\frac{2k_m}{R}V_s+M_p l\mathrm{g}\sin\theta_p\\&+M_p l^2\ddot{\theta}_p=-M_p l\ddot{x}\cos\theta_p\end{aligned}\quad(45)$$

To eliminate $\left(F_L+F_R\right)$ from the motor dynamics, Equation (36) is substituted into Equation (16),

$$\begin{aligned}&2\left[M_w+\frac{I_w}{r^2}\right]\ddot{x}=-\frac{2k_m k_e}{Rr^2}\dot{x}+\frac{2k_m}{Rr}V_s\\&-\left(M_p l\ddot{\theta}_p\cos\theta_p+M_p l\dot{\theta}_p^2\sin\theta_p+M_p\ddot{x}_R\right)\end{aligned}\quad(46)$$

$$\begin{aligned}&2\left[M_w+\frac{I_w}{r^2}\right]\ddot{x}=-\frac{2k_m k_e}{Rr^2}\dot{x}+\frac{2k_m}{Rr}V_s\\&-M_p l\ddot{\theta}_p\cos\theta_p+M_p l\dot{\theta}_p^2\sin\theta_p-M_p\ddot{x}_R\end{aligned}\quad(47)$$

Rearranging Equation (45) and Equation (46) gives the nonlinear equations of motion of the system.

$$\begin{aligned}&\left[I_p+M_p l^2\right]\ddot{\theta}_p-\frac{2k_m k_e}{Rr}\dot{x}+\frac{2k_m}{R}V_s\\&+M_p l\mathrm{g}\sin\theta_p=-M_p l\ddot{x}\cos\theta_p\end{aligned}\quad(48)$$

$$\begin{aligned}&\left[2M_w+\frac{2l_w}{r^2}+M_p\right]\ddot{x}+\frac{2k_m k_e}{Rr^2}\dot{x}\\&+M_p l\ddot{\theta}_p\cos\theta_p-M_p l\dot{\theta}_p^2\sin\theta_p=\frac{2k_m}{Rr}V_s\end{aligned}\quad(49)$$

The above two equations can be linearised by assuming $\theta_p=\pi-\phi$, where ϕ represents a small angle from the vertically upward direction. This simplification was used to enable a linear model to be obtained so linear state space controllers could be implemented.

Therefore,

$$\cos\theta_p=\cos\left(\pi-\phi\right)\approx-1\quad(50)$$

$$\sin\theta_p=\sin\left(\pi-\phi\right)\approx-\phi\quad(51)$$

$$\dot{\theta}_p=\dot{\phi}^2\approx 0\text{ and }\ddot{\theta}_p=\ddot{\phi}\quad(52)$$

The linearised equation of motion is

$$\begin{aligned}&\left[I_p+M_p l^2\right]\ddot{\phi}-\frac{2k_m k_e}{Rr}\dot{x}+\frac{2k_m}{R}V_s\\&-M_p l\mathrm{g}\phi=M_p l\ddot{x}\end{aligned}\quad(53)$$

$$\begin{aligned}&\left[2M_w+\frac{2I_w}{Rr}+M_p\right]\ddot{x}+\frac{2k_m k_e}{Rr^2}\dot{x}\\&-M_p l\ddot{\phi}=\frac{2k_m}{Rr}V_s\end{aligned}\quad(54)$$

When Equation (53) and Equation (54) are rearranged, the state space equation of the system is obtained.

$$\begin{aligned}\ddot{\phi}=&\frac{M_p l}{\left(I_p+M_p l^2\right)}\ddot{x}+\frac{2k_m k_e}{Rr\left(I_p+M_p l^2\right)}\dot{x}\\&-\frac{2k_m}{R\left(I_p+M_p l^2\right)}V_s+\frac{M_p g l}{\left(I_p+M_p l^2\right)}\phi\end{aligned}\quad(55)$$

$$\begin{aligned}\ddot{x}=&\frac{2k_m}{Rr\left(2M_w+\frac{2I_w}{r^2}+M_p\right)}V_s\\&-\frac{2k_m k_e}{Rr^2\left(2M_w+\frac{2I_w}{r^2}+M_p\right)}\dot{x}\\&+\frac{M_p l}{\left(2M_w+\frac{2I_w}{r^2}+M_p\right)}\ddot{\phi}\end{aligned}\quad(56)$$

By substituting Equation (55) into Equation (49), substituting Equation (43) into Equation (31), and after a series of algebraic manipulation, the state space equation for the system is obtained.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \dfrac{2k_m k_e\left(M_p lr-I_p-M_p l^2\right)}{Rr^2\alpha} & \dfrac{M_p^2 gl^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \dfrac{2k_m k_e\left(r\beta-M_p l\right)}{Rr^2\alpha} & \dfrac{M_p gl\beta}{\alpha} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}$$

$$+\begin{bmatrix}0\\ \dfrac{2k_m\left(I_p+M_p l^2-M_p lr\right)}{Rr\alpha}\\ 0\\ \dfrac{2k_m\left(M_p l-r\beta\right)}{Rr\alpha}\end{bmatrix}V_s \quad (57)$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+0\times V_s \quad (58)$$

Where,

$$\alpha=\left[I_p\beta+2M_pl^2\left(M_w+\frac{I_w}{r^2}\right)\right] \quad (59)$$

$$\beta=\left[2M_w+\frac{2I_w}{r^2}+M_p\right] \quad (60)$$

The detail descriptions for all mathematical expression are offered in the background theory of mobile robot system in several books.

In the abovementioned model, it is assumed that the wheels of the vehicle will always stay in contact with the ground and that there is no slip at the wheels. Therefore, cornering forces are also considered negligible.

3. Experimental Results and Discussions

The experimental results based on separated hardware implementation and overall system implementation were presented in this section. And also the outcomes of research are mentioned based on the discussions on several analyses.

3.1 Overall Operation of the System

The self-balancing two-wheeled robot is based on an inverted pendulum theory. Without any control system, it cannot maintain its upright position. To hold the robot in an excellent place, the controller should know at which angle the robot is tilting. This process can be done by the IMU sensor, which consists of a three-axis accelerometer and a three-axis gyroscope. The accelerometer is responsible for measuring the tilt angle, and the gyroscope shows the robot's angle position. However, each sensor data is not entirely accurate due to the presence of noise and drift in each sensor. To overcome this problem, a complementary filter is required to implement for fusing the sensor data. In this robot hardware, a control algorithm is written in an intelligent electronic device known as an Arduino UNO. UNO is used to collect the robot's orientation in a single axis from the IMU sensor and provide an appropriate control signal to control both left and suitable motor by tuning a Proportional, Integral and Derivative (PID) gain value. The output from this PID controller is a pulse width modulation (PWM) signal. Consequently, it can control the speed of the motor through a dual full-bridge motor driver (L298N). Thus to balance the two-wheeled self-balancing robot, tuning the PID gain values and filtering the sensor data play an essential role in this unstable system.

3.2 Closed-loop PID Controller Testing Based on Mathematical Model

PID controller is simulated in MATLAB/SIMULINK using the accurate parameter in the robot's mathematical model. The mathematical model equation was driven in Equation (56), and the robot's precise parameters have been calculated in section III.

Figure 4 demonstrates the Closed Loop Impulse Response of the System with PID. This simulation is constructed according to the state-space model. An impulse response is used to test the PID closed-loop control system. The first proportional gain (K_p=15) controls the position of the robot and the second PID controller controls the robot's angle. The input source is an impulse signal, and it is similar to the robot, which is suddenly pushed in the objective case. This impulse force will cause the robot to change its stable position and angle for a short period. The result can be seen in Figure 5.

Figure 5(a) means that the robot's position has changed from 0 meters to around 0.8 meters first due to the applied impulse force. This will create the robot to lean nearly 18 degrees. Consequently, the robot's angle must be controlled with proper PID gain values to maintain its stability. Unless the robot's angle is held, it will fall. With the help of PID controllers in both position and angle, the robot will return to its initial position after 2 seconds, and at the same time, the robot's tilting angle will return to zero degrees. Among these two Figures 5(a) and 5(b), the right-hand side graph can give a better result when the robot's displacement and tilt angle measurements are compared.

In the second experiment, a unit step signal is used as an input to the system, and then the robot's position and angle are checked after passing through the PID controller in both cases. The main unsimilar point between Figure 4 and Figure 6 is the input signal. Figure 4 uses an impulse signal, while Figure 6 uses a unit step signal. The simulation result can be demonstrated in Figure 7. This time let's assume that the user wants to move the robot to one meter and stay in that position, like giving a unit step function to the robot's position in the simulation. In order to observe this, the robot should lean for at most around 15 degrees. With the help of these PID gain values, the robot could manage to reach that set point position within 3 seconds. After getting to the desired position, the robot's tilting angle will be returned to zero. In other words, the robot will return to its balanced position. Among these, two graphs the right-hand side graph can give a better result.

Last but not least, the PID controller is used to control only the robot's angle instead of the robot's position. MATLAB simulation diagram is shown in Figure 8.

This phenomenon is quite similar to setting the robot to lean forward to a certain degree constantly. The simulation result can be seen in Figure 9. In this simulation, the robot is set to lean forward for one degree constantly. Thus in Figure 9, at first, the robot is oscillating and trying to maintain in one degree; as a result, the robot will move continuously. Therefore, between these two figures: Figure 9(a) and Figure 9(b), Figure 9(b) can be assumed to be the best one by comparing the robot's angle graph.

One of the interesting facts in this simulation test is that all the PID gain values that are used in those three different simulations are the same value. This means that appropriate PID gain values for both unit step and unit impulse function in controlling both position and angle have already been tuned after realising the characteristic of the PID controller and trying a try and error method for tuning the gain values. Moreover, PID gain values used in every simulation (a) can balance the robot in software simulation and hardware testing.

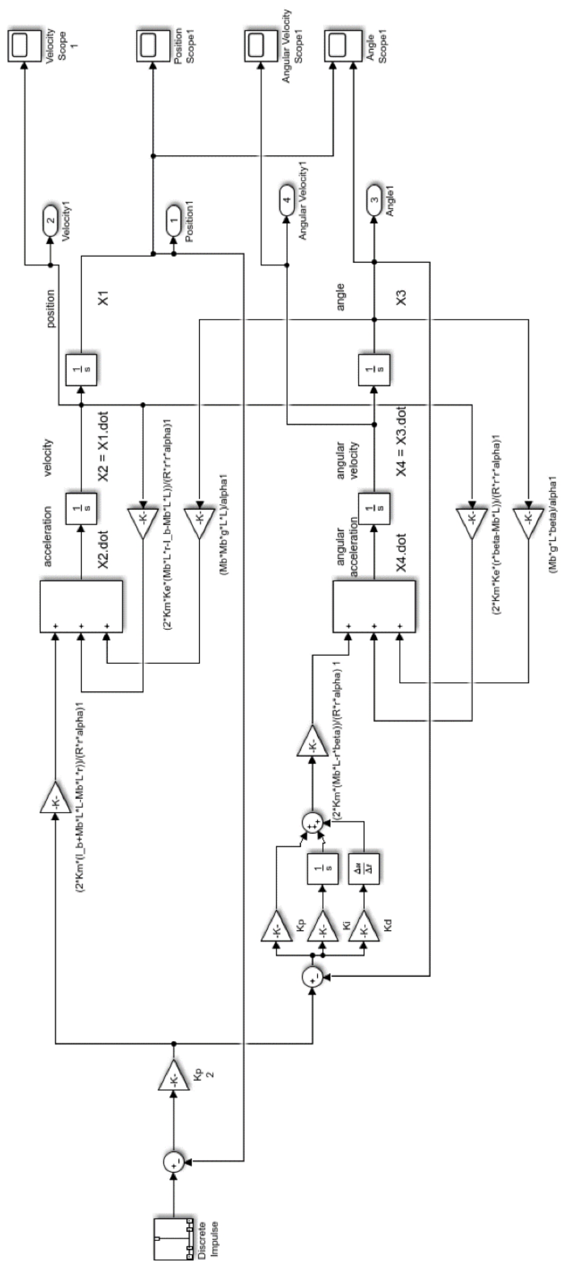

Figure 4. Closed Loop Impulse Response of the System with PID

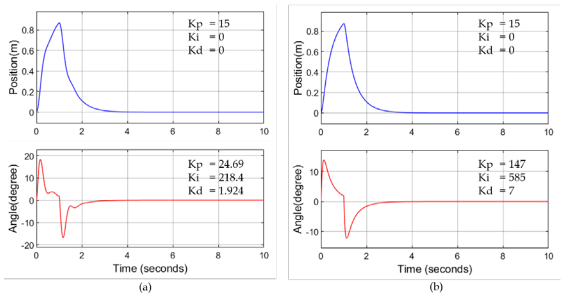

Figure 5. Test and Result of Closed Loop Impulse Response with PID Controller in Both Position and Angle

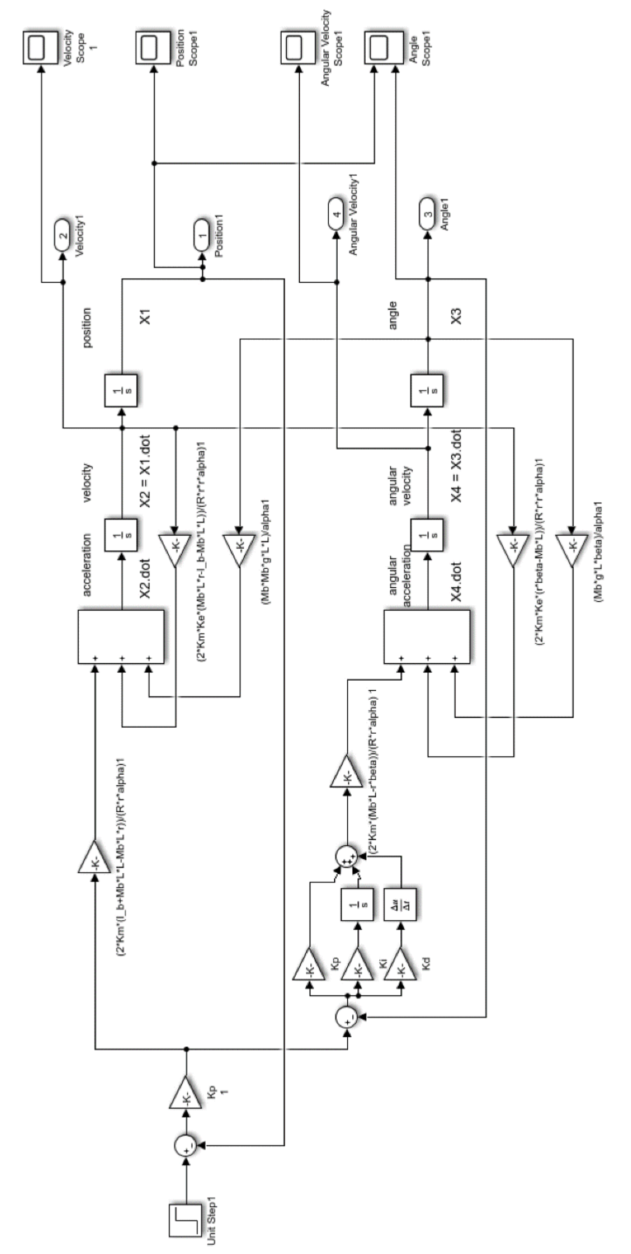

Figure 6. Closed Loop Unit Step Response of the System with PID Controller in Both Position and Angle

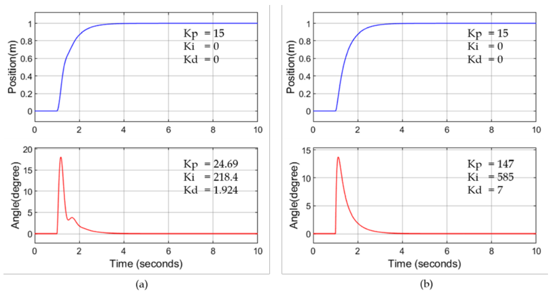

Figure 7. Test and Result of Closed Loop Unit Step Response with PID controller in Both Position and Angle

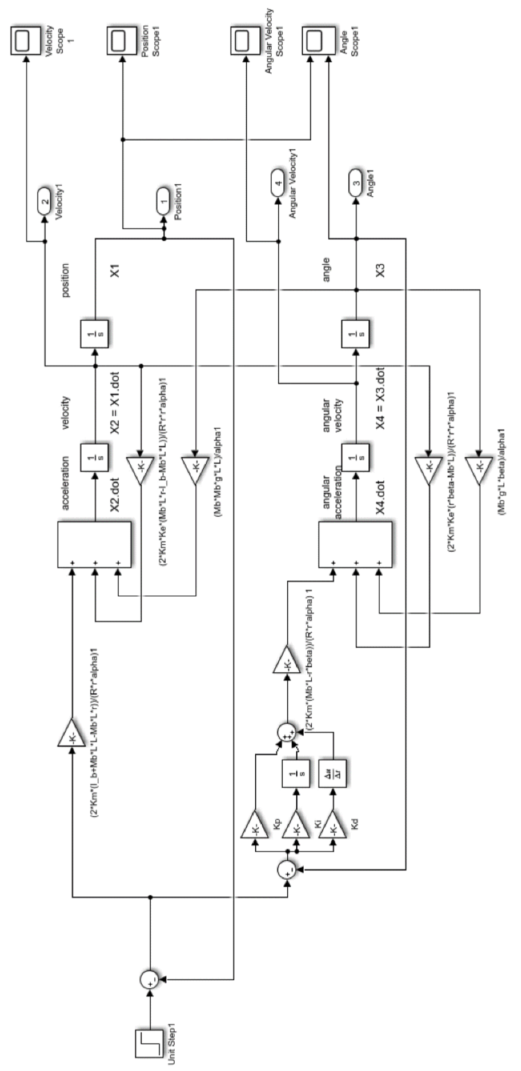

Figure 8. Closed Loop Unit Step Response of the System with PID Controller in Angle

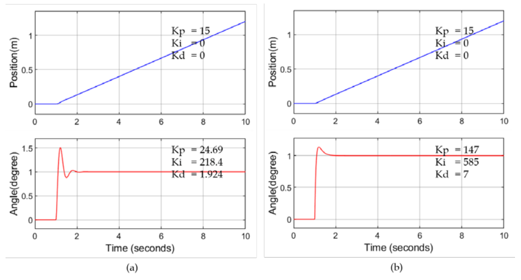

Figure 9. Test and Result of Closed Loop Unit Step Response with PID controller in Both Position and Angle

3.3 Closed Loop LQR Controller Testing Based on Mathematical Model

After substituting the robot's parameter into the state-space model derived in section III, Equation (60) is obtained.

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -113 & 10.81 & 0\\ 0 & 0 & 0 & 1\\ 0 & -630.4 & 178.2 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+\begin{bmatrix}0\\ 4.296\\ 0\\ 23.96\end{bmatrix}V_s \qquad (61)$$

For a linear control system to be implemented, the system has to be controllable. This requires that the rank of the $n\times n$ controllability matrix $C=\left[BAB...A^{n-1}B\right]$ is n or modulus of C is not equal to zero.

The algebraic Ricatti equation is solved using MATLAB, and the control gain K is evaluated for different values of Q and R weighting matrices. The response of the system is simulated as well.

The Q matrix assumes the form of

$$Q=\begin{bmatrix}a & 0 & 0 & 0\\ 0 & b & 0 & 0\\ 0 & 0 & c & 0\\ 0 & 0 & 0 & d\end{bmatrix} \qquad (62)$$

where the values of a,b,c and d are the weightings for the states $x,\dot{x},\phi,\dot{\phi}$ while the weighing matrix R is a scalar value as there is only one control input to the system. The values in Q and R matrices are adjusted to penalise the system state space and input. The main aim of this control system is to make the states of the system converge to zero at the shortest time possible. Thus the control engineer should adjust the Q and R matrices to obtain the desired response. The simulation result for this self-balancing robot is illustrated in Figure 10.

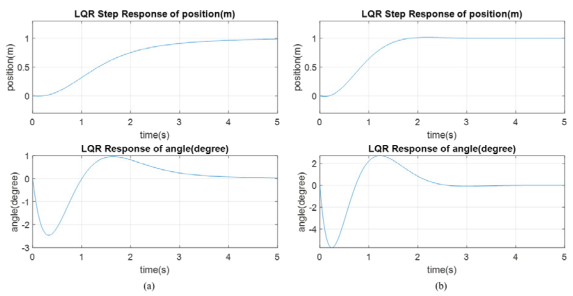

Figure 10. Test and Result of Closed Loop LQR controller

Figure 10(a) uses a=10, b=10, c=60, d=100 for Q matrix and R=0.001. Using these values, the robot will reach the desired position 1m after 4 seconds by oscillating the robot between –2 degrees and 1 degree. Figure 10(b) uses a=100, b=10, c=600, d=100 for Q matrix and R=0.001. At these values, the robot will reach the desired position 1 m after 2 seconds by oscillating the robot between –5 degrees and 2.3 degrees. Among these two LQR simulations, Figure 10(b) can provide better performance than that of (a). Moreover, by comparing this LQR simulation result (b) with the PID control simulation result in Figure 7(b), the LQR controller can perform better because the robot will reach the desired distance of 1 meter within 2 seconds the tilting angle of a maximum 5 degrees. But with the PID control, the robot needs to tilt about 15 degrees to reach the desired position within 2 seconds.

3.4 Flowchart of the Robot Control System

The flowchart used to carry this project can be seen in Figure 11. Once the necessary components are set, the operation of the flowchart will be explained in this section.

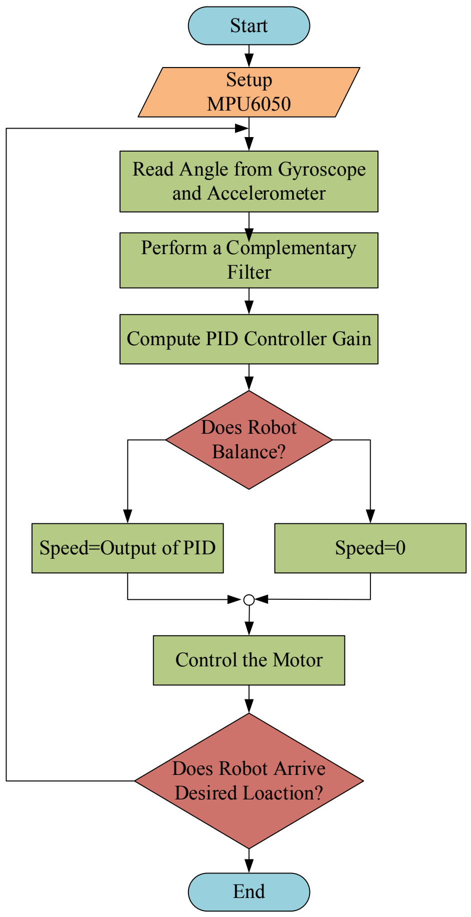
Start
Setup MPU6050
Read Angle from Gyroscope and Accelerometer
Perform a Complementary Filter
Compute PID Controller Gain
Does Robot Balance?
Speed=Output of PID
Speed=0
Control the Motor
Does Robot Arrive Desired Loaction?
End

Figure 11. Flowchart of the Robot Control System

First of all, the MPU6050 sensor is initialised in the setup function. After that, the accelerometer and gyroscope raw data are acquired from the sensor and then passed through the complementary filter to remove drift and noise from MPU6050. Then, it is sent to the PID controller. By tuning the PID gain values, the motor speed can be controlled with a PWM signal. If the sensor senses that the robot is in a balanced condition, the PID controller will command the motor not to move for ten milliseconds which becomes the system's loop time. When the sensor senses that the robot is tilting forward for five degrees, the PID controller will command the motor to accelerate the motor forward until the robot becomes stable. In this way, the robot can balance itself successfully. Finally, when battery power is not enough, the robot will stop running [16]. The specific coding is not mentioned in this paper.

3.5 Hardware Testing

To implement the flowchart program in the hardware properly. Getting angle from the MPU6050 sensor is the first priority for hardware testing.

The Arduino Uno communicates with the MPU6050 through the I2C protocol. Thus serial clock pin (SCL) is connected to the Arduino (A5) pin as well as the serial data pin (SDA) is connected to the Arduino (A4) pin. The sensor's interrupt pin (INT) is then connected to the Arduino pin2. Arduino supplies 3.3V to the sensor by connecting the Vcc pin from the sensor and the 3V3 pin from the Arduino. The two ground pins are made sure to communicate with each other.

After interfacing the sensor and Arduino Uno, the program should be uploaded to test the sensor. Then the raw data are saved in the excel file and plotted in MATLAB. After that, check the nature of the accelerometer and gyroscope sensor and perform complementary in MATLAB code.

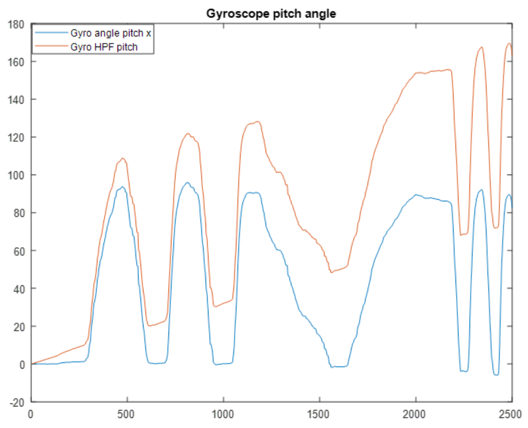

Figure 12. Gyroscope Pitch Angle

The sensor is rotated only in one axis (pitch angle) from 0 to 90 degrees and then rotates again from 0 to 90 degrees for another four times. In Figure 12, the blue line represents the gyroscope pitch angle, and the red line represents the gyroscope high pass filter pitch angle. It was evident that the gyroscope has a bias. Thus without passing a high pass filter, the sensor data are drift gradually. However, in the filtered data, a small drift still presents in between 2000 and 2500 samples.

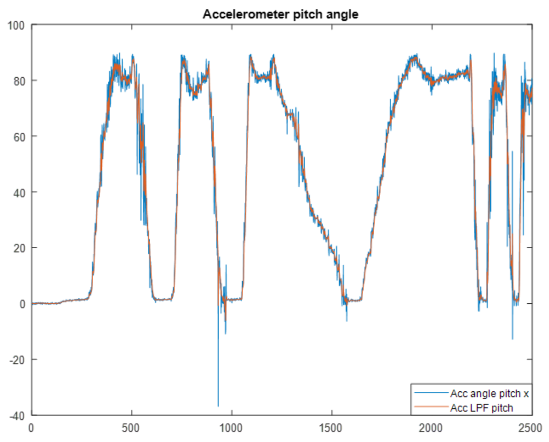

Figure 13. Accelerometer Pitch Angle

Figure 13 represents the nature of the accelerometer sensor. The blue line represents the accelerometer pitch angle, and the red line represents the accelerometer low pass filter pitch angle. Even there is no bias in the accelerometer, and it is pretty sensitive to noise. As a result, a low pass filter is needed to filter out this noise. Moreover, even the sensor is rotated from 0 to 90 degrees for six times, the accelerometer sensor data and low pass filtered data cannot represent absolute 90 degrees when only the accelerometer sensor is used to calculate the angle.

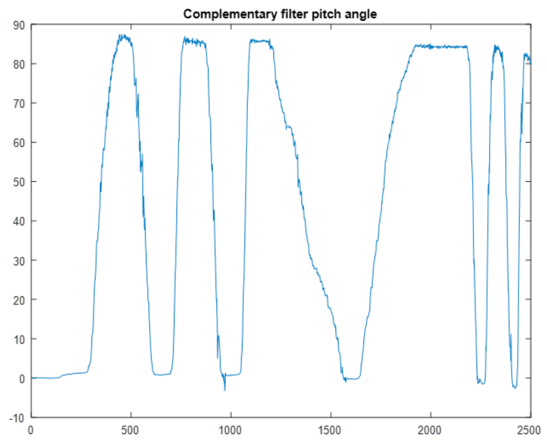

Figure 14. Complementary Filter Pitch Angle

In order to solve the problem occurring in both accelerometer and gyroscope sensors, a complementary filter is used to get a better result. For example, Figure 14 combines the two sensors' data, and this filter is designed to trust the gyroscope sensor data more than that of the accelerometer.

3.6 Implementing PID Controller to the Robot Control System

After testing each component before implementing the whole system, the PID control algorithm is written in the Arduino. This PID controller plays an essential role in the self-balancing robot system. The reason is that without a controller or a poor control design, the robot cannot balance itself and is stable in one position. Thus, in this case, tuning the PID controller plays an important role. Therefore, the author did a lot of experiences to get the perfect controller gain values. Among them, four different PID gain values are selected.

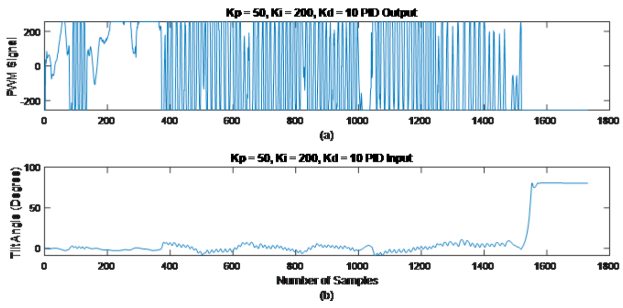

Figure 15. The First PID Tuning in Robot

Figure 15 shows all data conditions of the running robot with PID gain values of $K_p = 50$, $K_i = 200$ and $K_d = 10$. Figure 14(a) represents the PID output (PWM signal) to control the motor speed. Figure 14(b) illustrates the robot's actual tilting angle. In this Figure 14, the robot cannot keep in a balance position because it falls down after operating for a short period. This phenomenon can be seen when the robot's tilting angle reaches 90 degrees.

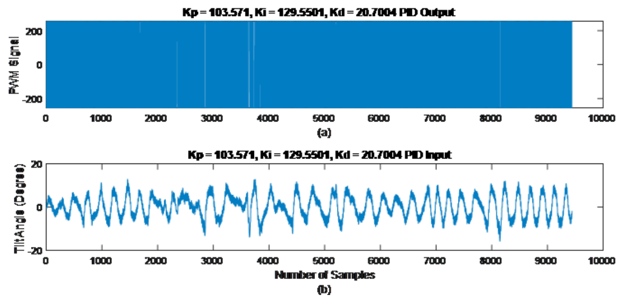

Figure 16. The Second PID Tuning in Robot

Since PID gain values in Figure 14 cannot maintain the robot in an upright position, other PID gain values are tuned. Figure 16 shows all data conditions of the running robot with PID gain values of $K_p = 103.571$, $K_i = 129.5501$, and $K_d = 20.7004$. Figure 15(a) represents the PID output (PWM signal) to control the motor speed. Figure 16(b) illustrates the robot's actual tilting angle. These gain values make the robot oscillate a lot; as a result, the PID controller needs to work at a maximum and a minimum limit of +255 and –255 PWM signals during the running condition. However, it cannot maintain the robot in a stable condition.

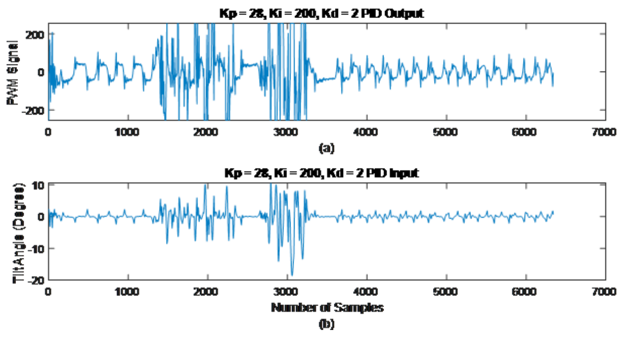

Figure 17. The Third PID Tuning in Robot

Thirdly, PID gain value is changed into $K_p = 28$, $K_i = 200$, and $K_d = 2$. Then the robot's condition is checked. Unlike Figure 16, the PID controller does not need to work at a maximum or minimum limit apart from a disturbance. These gain values can keep the robot in a stable condition since, in Figure 17(b), the robot does not tilt a lot without a disturbance.

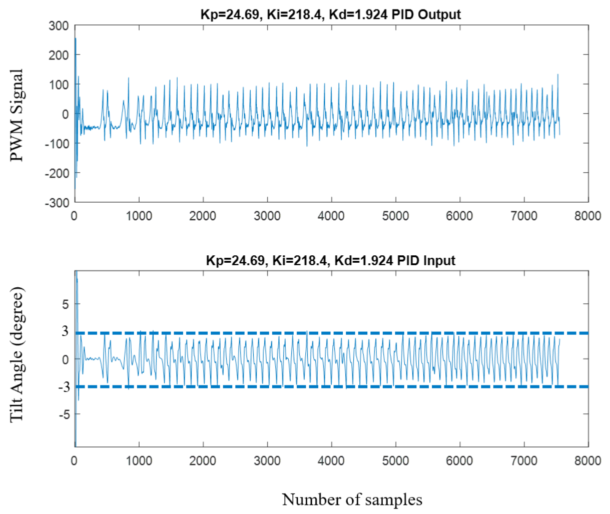

Figure 18. The Balancing Condition of the Robot in Upright Position

This final PID gain value of $K_p = 24.69$, $K_i = 218.4$; and $K_d = 1.924$ can improve the robot's stability. The robot oscillates between –3 degrees and 3 degrees during the experiment. This means that the PID controller controls the motor driver not to let the robot tilt beyond 3 degrees. Thus it can be concluded that this gain value can provide a more stable condition for the self-balancing robot. The result can be seen in Figure 18.

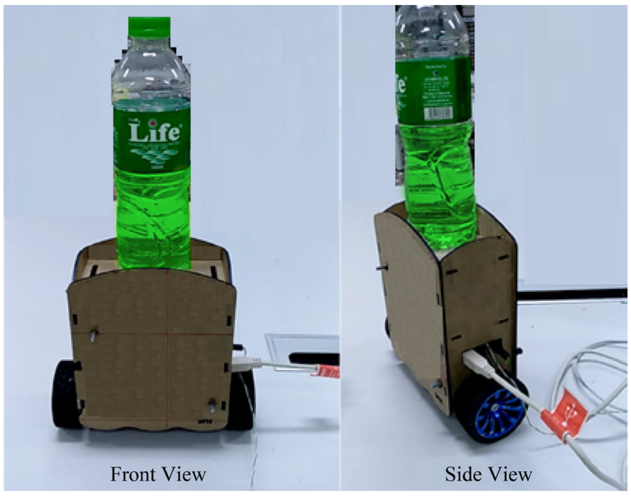

Figure 19. The Balancing Condition of the Robot in the Presence of Extra Weight

Moreover, to know whether this PID gain value of $K_p = 24.69$, $K_i = 218.4$; and $K_d = 1.924$ can maintain the robot's stable condition or not when extra weight is applied to it, a half-filled 0.6 litre of water bottle is placed at the top of the robot's chassis and run the robot. Therefore, this PID controller can also maintain the robot in a stable condition. The front and side view of the robot's running condition can be seen in Figure 19. In order to get this PID to gain value, many experiments are tested during the research program. The simulation tools in this mathematical modelling are based on the MATLAB and Arduino programming. The accuracy could be checked by using the ratio of difference between real and defined position to the defined position in this study.

3.7 Challenging Issues

During this research, a minor challenging issue appeared, such as resizing the depth of the robot chassis because the laser cutter cuts the acrylic sheets into 2 mm instead of cutting into 3mm sheets. However, one major issue was found with the DC motors employed: the first motor, GM25-370-24140 DC motor, does not have enough RPM to rotate when the robot turns to tilt into a large angle. For this reason, it is so complicated to tune the PID control. Fortunately, this problem is solved by replacing it with a 500RPM DC motor.

3.8 Discussions

This research work successfully achieves the goals of balancing unstable two-wheeled robots. Balancing an unstable system is a challenging problem in the research field, and the various controllers can be used in stabilising the system. In this research work, PID and LQR controller is used to control the robot's position and angle. But in hardware testing, only PID controller is used. First, the 3D design of the self-balancing robot was drawn in the Solid Works software to find the best location for each electronic component. By doing so, the author does not have any problem with hardware assembling in practice. Next, the complementary filter has been successfully implemented to filter a gyroscope drift and accelerometer noise, allowing for an accurate estimation of the robot's tilting angle. Arduino must communicate the IMU sensor via I2C communication and control the motor's speed via PWM signal. After passing the filtered sensor data to the microcontroller, the microcontroller compares the actual tilting angle with the setpoint value to check the error. This error is compensated by using a PID controller in hardware. Both PID and LQR control algorithms have successfully compared the software simulation result in MATLAB using the robot's state-space model. LQR can provide better performance than the PID controller. Due to the time limit, only the PID controller is successfully implemented to address the problem of balancing an unstable system in hardware testing. But this PID tuned values of $K_p = 24.69$, $K_i = 218.4$, and $K_d = 1.924$ used in hardware can maintain the robot upright on flat surfaces. Moreover, it can control the robot in a balanced condition with extra weight, as shown in Figure 18. Last but not least, this gain value is also satisfied in software simulation. Therefore, many experiments and simulations were done to achieve a better result. This work is novel for emphasizing on the numerical approaches for experimental approaches with discrete components and utilizing the fundamental concepts of the dynamic control system design. The performance comparison on the mobile robot system was depending on the controller design and robustness of the system. That is the innovative work in this study.

3.9 Limitations

The limitations of the developed mobile robot system are latency of the control signal input to the process by controller output and the translational vector between the simulation approaches to experimental approaches. The problem could be solved to acquire the better performance for experimental studies with the speed of controller design and optimal gain tuning is the contribution in this work.

4. Statistics Table for Results Comparison

The experimental results of self-balancing mobile robot system in this study and recent works are given in the following Table 1. The performance comparisons are found in this table with various kinds of conditions.

Table 1. Performance Comparison Table

Works	Controller Design	Accuracy	Performance Level	Robustness
[17]	PD	93%	High	Satisfy
[18]	LQR	94%	High	Satisfy
[19]	PWM	90%	Medium	Satisfy
This Work	PID and LQR	98%	Very High	Satisfy

The control algorithm is based on the real time control system for specific analysis. The application of the mobile robot system is for healthcare service in real time conditions.

5. Conclusions

This research successfully achieved its aims to balance a self-balancing robot based on the inverted pendulum model. Two control strategies have been implemented to address the problem of balance control for the system. The gain matrices obtained from PID simulation are implemented in this self-balancing robot for real-time controller experimentation. During the testing, the robot can maintain its vertical position by slightly adjusting its wheels. Besides PID control, the LQR controller is also successfully tested in MATLAB simulation. The PID controller can easily be tuned by observing the optimal condition. The LQR controller can also better results when the Q matrix and R matrix are adequately set. Many simulations are run to get the best result for this self-balancing robot. In the hardware testing, the PID controller is designed to control a single axis robot tilting angle, y-axis. Rotating in both x and z axes is neglected in implementing the PID control algorithm because this research work focuses only on stabilizing the robot. Also, controlling the position is not considered in hardware testing. However, in MATLAB simulation, both PID and LQR are considered in controlling both robot's position and angle. Thus, considering the yaw axis and x y movement is recommended for trajectory tracking in the robot's hardware. So that later formation control can be performed. Stabilizing the robot on a sloped area and a rough surface area is recommended as a further extension. PID controller cannot maintain stable conditions for slopes and rough surfaces. Thus apart from PID controller, Linear Quadratic Regulator (LQR), pole-placement, fuzzy control, adaptive control, sliding mode control (SMC), etc., are recommended to implement in this robot to decide which controller can perform the best in the future development of the self-balancing robot. Moreover, a manual remote control system like commanding the robot to move forward, backwards and rotate clockwise or counter-clockwise direction via Bluetooth wireless connection can also be considered in this self-balancing robot. The numerical expressions were matched to the experimental studies in this study based on the accuracy percentage of 98%. The performance level is also very high by comparing with the other recent works.

Conflict of Interest

There is no conflict of interest.

Acknowledgment

The authors acknowledge researchers and colleagues from the Faculty of Electrical and Computer Engineering of Yangon Technological University. This work is fully supported by Government Research Funds for 2021-2022 Academic Year. This work is completely measured and analyzed at the Electrical and Electronics Properties Evaluation Laboratory Room (EB-402) of the Myanmar Japan Technological Development Centre 1 (MJTDC 1) of Yangon Technological University (YTU). This work is collaborative research work with Acharya Institute of Technology in India and Chulalongkorn University in Thailand.

References

[1] Peng, K., Ruan, X., Zuo, G., 2012. Dynamic model and balancing control for two-wheeled self-balancing mobile robot on the slopes. Proceedings of the 10th World Congress on Intelligent Control and Automation. pp. 3681-3685. (Accessed 9 September 2021). DOI: https://doi.org/10.1109/WCICA.2012.6359086

[2] Bin, H., Zhen, L.W., Feng, L.H., 2010. The Kinematics Model of a Two-Wheeled Self-Balancing Autonomous Mobile Robot and Its Simulation. 2010 Second International Conference on Computer Engineering and Applications. pp. 64-68. (Accessed 18 September 2021). DOI: https://doi.org/10.1109/ICCEA.2010.169

[3] Gong, D., Li, X., 2013. Dynamics modeling and controller design for a self-balancing unicycle robot. Proceedings of the 32nd Chinese Control Conference. pp. 3205-3209. (Accessed 27 September 2021).

[4] Nikita, T., Prajwal, K.T., 2021. PID Controller Based Two Wheeled Self Balancing Robot. 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI). pp. 1-4. (Accessed 8 October 2021). DOI: https://doi.org/10.1109/ICOEI51242.2021.9453091

[5] Philip, E., Golluri, S., 2020. Implementation of an Autonomous Self-Balancing Robot Using Cascaded

PID Strategy. 2020 6th International Conference on Control, Automation and Robotics (ICCAR). pp. 74-79. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/ICCAR49639.2020.9108049

[6] Jianwei and Xiaogang, 2008. The LQR control and design of dual-wheel upright self-balance Robot," 2008 7th World Congress on Intelligent Control and Automation. pp. 4864-4869. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/WCICA.2008.4593712

[7] Yu, N., Li, Y., Ruan, X., et al., 2013. Research on attitude estimation of small self-balancing two-wheeled robot. Proceedings of the 32nd Chinese Control Conference. pp. 5872-5876. (Accessed 9 September 2021).

[8] Putov, A.V., Ilatovskaya, E.V., Kopichev, M.M., 2021. Self-balancing Robot Autonomous Control System. 2021 10th Mediterranean Conference on Embedded Computing (MECO). pp. 1-4. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/MECO52532.2021.9459720

[9] Sarathy, S., Mariyam Hibah, M.M., Anusooya, S., et al., 2018. Implementation of Efficient Self Balancing Robot. 2018 International Conference on Recent Trends in Electrical, Control and Communication (RTECC). pp. 65-70. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/RTECC.2018.8625624

[10] Sun, W.X., Chen, W., 2017. Simulation and debugging of LQR control for two-wheeled self-balanced robot. 2017 Chinese Automation Congress (CAC). pp. 2391-2395. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/CAC.2017.8243176

[11] Ruan, X.G., Liu, J., Di, H.J., et al., 2008. Design and LQ Control of a two-wheeled self-balancing robot. 2008 27th Chinese Control Conference. pp. 275-279. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/CHICC.2008.4605775

[12] Lv, Q., Wang, K.K., Wang, G.Sh., 2009. Research of LQR controller based on Two-wheeled self-balancing robot. 2009 Chinese Control and Decision Conference. pp. 2343-2348. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/CCDC.2009.5192771

[13] Kongratana, V., Gulphanich, S., Tipsuwanporn, V., et al., 2012. Servo state feedback control of the self balancing robot using MATLAB. 2012 12th International Conference on Control, Automation and Systems. pp. 414-417. (Accessed 9 September 2021).

[14] Gong, Y., Wu, X., Ma, H., 2015. Research on Control Strategy of Two-Wheeled Self-Balancing Robot. 2015 International Conference on Computer Science and Mechanical Automation (CSMA). pp. 281-284. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/CSMA.2015.63

[15] Tun, H.M., Nwe, M.S., Naing, Z.M., 2008. Design analysis of phase lead compensation for typical laser guided missile control system using MATLAB bode plots. 2008 10th International Conference on Control, Automation, Robotics and Vision. pp. 2332-2336. (Accessed 9 September 2021).
DOI: https://doi.org/10.1109/ICARCV.2008.4795897

[16] Tun, H.M., Nwe, M.S., Naing, Z.M., et al., 2022. "Sliding Mode Control-Based Two Wheels Mobile Robot System", The 14th Regional Conference on Electrical and Electronics Engineering (RC-EEE 2021), Chulalongkorn University, Thailand. (Accessed 9 January 2022).

[17] Velagic, J., Kovac, I., Panjevic, A., et al., 2021. Design and Control of Two-Wheeled and Self-Balancing Mobile Robot. pp. 77-82.
DOI: https://doi.org/10.1109/ELMAR52657.2021.9550938

[18] Pajaziti, A., Gara, L., 2019. Navigation of Self-Balancing Mobile Robot Through Sensors. IFAC Papers Online. 52-25, 429-434.
DOI: https://doi.org/10.1016/j.ifacol.2019.12.576

[19] Chhotray, A., Pradhan, M.K., Pandey, K.K., et al., 2016. Kinematic Analysis of a Two-Wheeled Self-Balancing Mobile Robot. Proceedings of the International Conference on Signal, Networks, Computing, and Systems, Lecture Notes in Electrical Engineering. Springer India. 396.
DOI: https://doi.org/10.1007/978-81-322-3589-7_9