3rd International Conference on
Robotics Automation and Non-Destructive Evaluation,
Chennai, India, 23 April 2022.
https://doi.org/10.13180/RANE.2022.23.04.05

ROBOT DEVELOPMENT USING SINGLE STEPPER DRIVER

VASVANI ASHISH MAHESHBHAI and DEEPAK KUMAR
Department of Mechanical Engineering, National Institute of Technology Jamshedpur
Adityapur, Jamshedpur, Pin -831014, Jharkhand, India
E-mail: 2017pgmeci14@nitjsr.ac.in, deepak.me@nitjsr.ac.in
www.nitjsr.ac.in

One of the important parameters for the development of specific robots is the mechanical stability of robots. Due to increased applications of robots in various fields such as marines, aerospace, biomedical, and so on, it has now become an important area of research. In this paper, the inverted pendulum concept was used to create a two-wheeled robot (TWR). To balance its vertical position, a Proportional Integrate Derivative (PID) controller was used. The TWR is made up of one Inertial Mass Unit (IMU) sensor, two stepper motors, an Arduino Nano microcontroller, and a stepper motor driver. Both motors were actuated by a single stepper driver in the developed design. Both motors were driven in opposite directions by a single stepper motor driver to ensure forward and backward motion of the robot. The IMU sensor was used to measure angular velocity along the pitch axis and acceleration along the other two axes. The measured data are used to calculate the inclination of TWR (θ) with respect to its vertical position. Accelerometer data was used to compensate for gyro drifts using a complementary filter. With finely tuned PID constants, acceleration to wheels was provided as per inclination angle (θ). The performance of the developed TWR is found to be almost identical to that of an Arduino UNO-based Self-Balancing Robot [20], despite the fact that it has fewer components, is lighter, and costs less.

1. Introduction

Nowadays, human life is involved with robots in many ways. They can perform many complicated tasks easily, quickly and with high precision [1]. Zimit et al. [1] have used Lagrangian method to develop a dynamic model of their two-wheeled self-balanced (TWSB) robot. They effectively implemented the Proportional–Integral–Derivative (PID) control loop and exhibited various results with different PID values. Eldhose et al. [2] made self-balanced robot using an arduino UNO and L23D driver module with dc motors. They calibrated offset values of MPU 6050 sensor and tuned the PID by trial & error method to stabilize the robot. Most Self-balanced robots are based on the principle of Inverted pendulum. F. Grasser et al. [5] had developed a mobile robot based on inverted pendulum principle and named it JOE. It was an early version of a self-balancing robot designed using inertia sensors, motor encoders and digital signal processor board (for implementation of the controller). It weighs 12 kg and its height was 65 cm. Junfeng Wu et al. [6] worked on a fuzzy PD controller to prevent the robot from falling and achieved self-balanced control of the robot. They had studied the GBOT1001 [7] and established the mathematical model of this system, using fuzzy PD control theory to control the robot. Jian Fang [8] has used particle swarm algorithm to optimize the parameter matrix of the Linear Quadratic Regulator (LQR) controller, which can help a robot in self-balancing. Brian Bonafilia et al. [9] have also used LQR for feedback and simulated results using Kalman Filter to stabilize robot in upright position. Qingwen Qian et al. [10] had designed parallel double fuzzy controller based on information fusion technology and simulated in MATLAB. Bature et al. [11] have done comparison of different controllers' viz. Fuzzy Logic Controller (FLC), LQR and PID by implementing on their robot. They observed that PID controller robot gives almost similar results as robot with LQR controller robot. Juang and Lum [12] had used PID and Proportional Integration- Proportional Derivative (PI-PD) controller in the same robot individually and compared. They observed that it can achieve more stability with PI-PD controller, and it can also compensate the centre of gravity (C.G.) misalignment. Rasoul and Mehdi [13] implemented three controllers on self-balancing robot viz. PD, PID and Fuzzy-PID. They compared the results using individual controller on Self-Balancing Robot (SBR) and found a slight vibration on the body of the robot due to increased steady state error after implementing PD controller. They reduced the vibration of robot and made it more stable by implementing PID controller. They also observed that robot body might fell down by applying external forces on it. SBR is developed by tuning various parameters with Fuzzy logic to reduce this effect significantly. Bhatti Omer et al. [14] have done comparative analysis between a simple PID-controller and an auto-tuned PID controller. They employed relay method to auto-tune PID parameters. They have found reduced steady state error, rise time and settling time with auto-tuned controller. Further, Robot can update its PID co- efficient in case of battery-drainage over time and change of mass during the operation with auto-tuned controllers. The only drawback of this controller is large overshoot. Unluturk Ali et al. [15] observed the effects of P, PI and PID controllers on self-balancing robot with the help of visual computer interface based on Qt-creator [16]. They observed that P-controller, alone is not sufficient for robot's balance. Robot can balance itself for a short period with PI controller and robot can stand in upright position longer with proper values of PID controller.

Francisco and Federico [17] had made a low cost (less than €35) educational robot: Arduino-A1, which included an android and Arduino system. The developed robot can be used as an educational tool in schools and colleges. They included a combination of sensors and communication resources: GPS, compass, light sensor, accelerometer, camera, Wi-Fi, Local Area Network (LAN), Bluetooth and Internet connection capabilities. It can be easily programmed and modified by users. Yeonhoon Kim et al. [18] used Kane's method of three Degree of Freedom (DOF) modeling to conduct dynamic modeling to analyze the mechanism of two-wheeled robot. They constructed a mechanical robot with tilt sensors, a gyroscope and encoders. His team has analyzed the effect of inclined surface and turning motion on stability of the robot. They succeeded in implementation of robot in balancing, rectilinear motion and spinning motion on 2-dimensional surface. Azhar et al. [19] have used two DOF PID controller to control self-balancing robot. Motor speed is controlled by inner loop and vertical position of robot is maintained in the specified boundary by outer loop. SBR have faster response, increased disturbance rejection and smaller error in comparison to the traditional PID controlled by Two DOF PID controller

Zeng et al. [20] have developed self-balancing robot using Arduino UNO R3 micro-controller board. They have used DC motors as actuators, PID control loop as controller and Kalman filter to fuse data of sensors. Further, they also implemented Bluetooth module to send signal to robot via mobile application. An and Li [21] have done modelling and analysis of self-balancing robot by using the principle of energy conservation and Lagrange equation. They controlled two sub-systems namely self-balance and yaw rotation via PID and LQR controller. In their design, both subsystems were controlled by PID initially and then after self-balance subsystem was controlled by LQR. They compared both sub-systems by doing simulation in MATLAB and found that, for controlling self-balancing subsystem, LQR gives more suitable results than PID. Zhuang et al. [22] have done dynamic modelling of self-balancing robot through a Lagrange equation of generalized co-ordinates of robot system. They designed the non-linear control algorithm of the robot balance movement, and used MATLAB for simulation.

They found that pitch angle and angle of wheels become stable in short time and robot can be stabilized faster and smoothly with their designed controller. Nakai et al. [23] used a recursive robot regulator. They used a formation, which follow a leader. To check the robustness, they made communication fault in blind areas. When communication is established again all robots that lost communication return to their position in formation. Further, they successfully simulated three possible faults to test robustness of controllers.

In this paper, Two Wheeled Robot (TWR) is designed and developed by using a single stepper driver and PID control loop. It is simple compare to advanced controller such as PI-PD controller, fuzzy PID controller and does not require any matrix manipulation as in LQR. Further, Arduino Nano-micro controller, which is based on open source platform Arduino.cc [24], is used which has the same functions of Arduino UNO with benefits of compact size and less cost. In addition, stepper motors are used for precise control of the robot. In General, separate motor driver is required for separate stepper motor, but in developed design, a single driver for both the motors has been provided. Here, one thing is to be kept in mind that both the motors should be driven in the opposite direction, since they are facing opposite to each other. The developed unique design reduces the cost, space and simplifies the code (for Arduino Microcontroller). Pulse width will be changed to drive stepper motors according to PID output variable, which in turn changes speed of motors that is proportional to an inclination angle (θ).

2. Mechanical System Design

Initial Design of robot is modeled in CAD software (Autodesk Inventor Professional) (Fig. 1). Design was taken from Joop Brooking's model [25] for reference. In the developed design, additional mass is provided to upper part of the robot, so that the center of mass will be at higher position relative to the wheel axis. It will increase moment of inertia of robot, which causes reduction in angular acceleration of robot. Further, it will also, help robot to balance better [4]. The dimensions and inertial properties of a robot are listed in the Table 1.

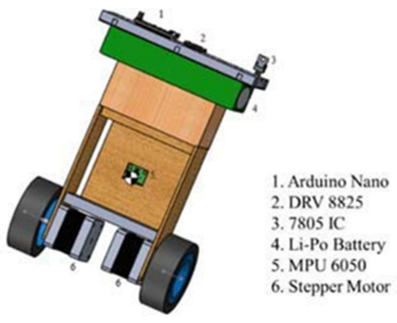

Figure 1. CAD model of Self-Balancing Robot.

Table 1. Properties of robot from CAD model.

Mass	1048.54 grams
Volume	907169.05 mm^3
Centre of Mass	
X	17.05 mm

Y	72.62 mm
Z	117.72 mm
Mass moment of Inertia	2230348.88 g*mm^2

The Block diagram is shown in the Fig. 2 for the circuit connection of two wheeled robot (TWR). Arduino Nano control board (Fig. 3 a).) was used instead of Arduino UNO to reduce the cost of the TWR. It can be easily programmable by Arduino IDE open source software [24]. Its code is based on C language and even simpler than C language. DRV8825 stepper driver (Fig. 3 b).)) was also, used to drive the stepper motors for micro stepping up to 1/32. It works well compare to A4988 driver which can provide micro stepping up to 1/16 only. Designed TWR is driven on 1/16th micro stepping with the driver. Further, Li-Po battery provides 12 V to run the stepper motors while, other electronic components like Arduino Nano, MPU 6050 and DRV8825 require 5V. To achieve this, 7805 IC (Fig. 3 c).) was used to convert 12V to 5V as it is compact and not much costly

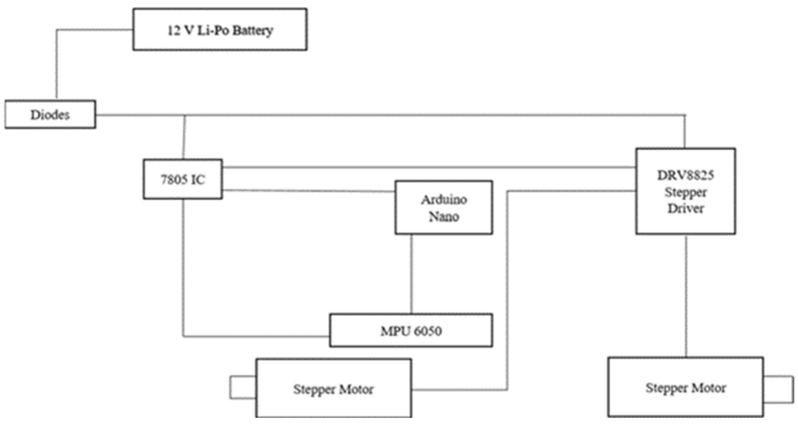

Figure 2. Block Diagram of Circuit Connection.

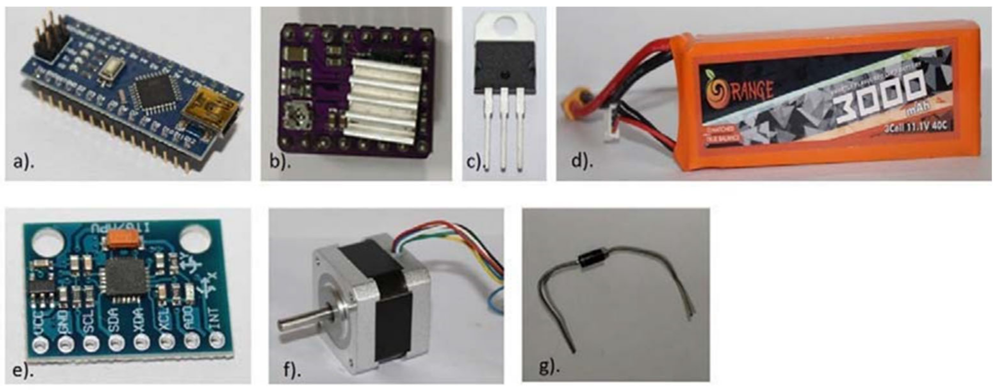

Figure 3. a). Arduino NANO Microcontroller, b). DRV8825Stepper Motor Driver, c). Stepper Motor, d). 7805 5V Voltage Regulator, e). MPU 6050 3-Axis Accelerometer and 3-Axis Gyroscope, f). 1N4001 Diode, g). 12V Li-Po Battery

A lithium-ion polymer battery (Fig. 3 d).) was used due to its feature to recharge. It is based on lithium-ion technology in which polymer electrolyte is used instead of a liquid electrolyte. It also, provides higher specific energy than other lithium battery types. TWR also utilizes MPU 6050 sensor (Fig. 3 e).) for the angular measurement. Stepper motors (Fig. 3 f).) are used as actuators due to its unique feature to move in discrete steps. Further, it also, shows no performance loss when the battery voltage drops. The Step angle of stepper motor used is 3.750. Also, with micro stepping of 1/16th, a step angle of 0.2343750 was achieved. 1N4001 diodes (Fig. 3 g).) were used to ensure safety of components against the reverse connection of the battery. Furthermore, capacitors were used for protection of DRV and IC against voltage spikes. All the components were purchased from Robokits India [26] except Li-Po battery. Li-Po battery was purchased from robu.in [27]. All components related to jumper wires on the breadboard and body of the robot is shown in the Fig. 4 a). The designed robot is divided in to three parts: the bottom part, the middle part, and the top part. The bottom part has two motors, middle part has IMU and top part includes wooden block, battery and breadboard containing all the (other) components (Fig. 4 b).). In the developed robot (TWR), bi-polar stepper motor is used, which consists of two coils, and one permanent magnet. Leads of these coils are termed as A1, A2, B1 and B2 and Shaft of the stepper motor, which is connected, to the permanent magnet will rotate in the clockwise (C.W) or counter-clockwise (C.C.W) direction. The angular direction depends on the sequence of voltage supply (through stepper driver) to these leads. Here, '+' means '+12V' and '–' means ground. Table 2 shows the sequence of supply voltage applied on the lead in order to drive the shaft C.W. or C.C.W.

Table 2. Sequence of voltage applied on leads in general case [31].

		A1	**A2**	**B1**	**B2**	
	1	-	+	+	-	
C.W.↓	**2**	-	+	-	+	**C.C.W.↑**
	3	+	-	-	+	
	4	+	-	+	-	

Since, both stepper motors are facing opposite to each other, shaft of both motors should be driven in the opposite direction to ensure motion of a robot in straight direction (i.e. forward or backward). Generally, two stepper motors require two different stepper drivers. However, in the developed robot, both stepper motors are being driven with a single stepper driver only. Connection of these motors is shown in the Fig. 5.

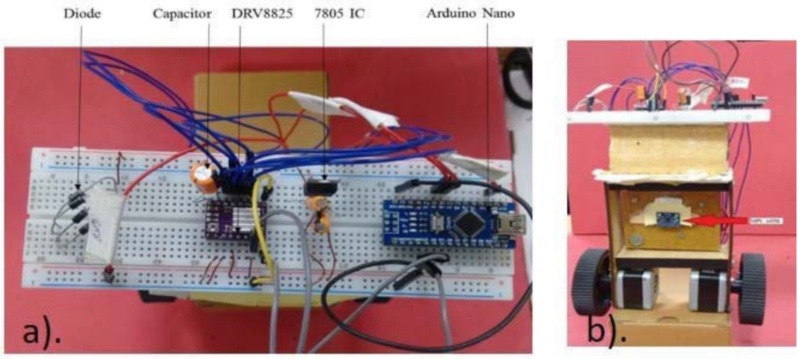

Figure 4. a). Circuit on Breadboard (Top View), b). Position of MPU 6050 (Front View)

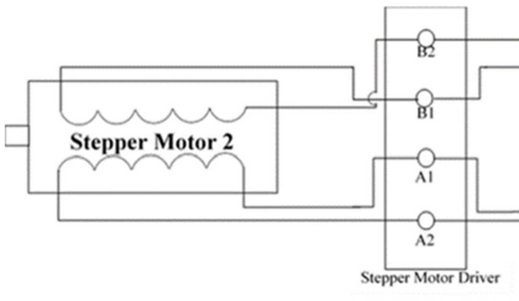

Figure 5. Connections of stepper motors with stepper driver.

The leads of B1 and B2 were exchanged for motor 2 to ensure the opposite direction of rotation for both shafts. Thus, B2 for motor 1 will work as B1 for motor 2 and B1 for motor1 will work as B2 for motor 2. Therefore, a sequence of supplied voltage will be as per Table 3 in our case. By comparing the sequence of Table 3 with the sequence given in Table 2, it is clear that when shaft of motor 1 will rotate in counter-clockwise direction, the shaft of motor 2 will rotate in the clockwise direction and vice-versa. The various parameters of developed robot are listed in Table 4.

Table 3. Sequence of voltage applied on leads in our case.

	Motor 1				**Motor 2**			
	A1	**A2**	**B1**	**B2**	**A1**	**A2**	**B1**	**B2**
1	-	+	+	-	-	+	-	+
2	-	+	-	+	-	+	+	-
3	+	-	-	+	+	-	+	-
4	+	-	+	-	+	-	-	+

Table 4. Robot Geometrical Parameters.

Height	176 mm
Width(wheel end to end)	166 mm
Thickness	43 mm
Wheel Diameter	72 mm
Wheel width	18 mm

3. Robot Control System

The MPU6050 is a combination of a 3-axis gyroscope, and a 3-axis accelerometer. Angular velocity with respect to three axes is measured by a gyroscope. Acceleration (in terms of g) along three axes is measured by accelerometer. Here, MPU6050 (Fig. 4 b).) is placed such as X-axis, Y-axis and Z-axis act as yaw axis, pitch axis and roll axis, respectively. Library: MPU6050.h inscribed by Jeff Rowberg [28] was used to get the readings from IMU sensor. Inclination of a robot is measured by two axis of accelerometer and single axis of gyroscope.

3.1. *Using Accelerometer*

Acceleration values along x-axis (accX) and z-axis (accZ) are required to measure the angle of inclination of a robot from the vertical position using accelerometer. Angle between the line to the point specified by the coordinates (accZ, accX) and the positive x-axis can be obtained by using tan2 (accZ, accX) with math.h library. Henceforth, desired angle of inclination (α) is computed. Here, the positive sign (+ve) indicates counter- clockwise angles (when robot leans forward), and negative sign (-ve) indicates the clockwise angles (when robot leans backward).

3.2. *Using Gyroscope*

Angular velocity along y-axis (pitch axis) is required to measure the inclination of robot using a gyroscope. The Value of a gyroscope with respect to Y axis (gyroY) was obtained from MPU sensor. gyroY value is converted to degrees per second and then converted value is multiplied by 0.005 to obtain angle travelled in loop time (i.e. 5 milliseconds). Finally, calculated angles are added to formerInclination (γ) to obtain presentInclination (θ).

3.3. *Combination of the Results with a Complementary Filter*

TWR had two measurements, accInclination (α) (Inclination obtained from accelerometer data) (Fig. 6 a).) and gyroInclination (β) (Inclination obtained from gyroscope data) as shown in the Fig. 6 b). Sudden horizontal movement affects accelerometer and gyroscope data gradually drifts away from the actual values. Short duration signals affect accelerometer data and long duration signals affect gyroscope data. Thus, these readings are complimentary to each other. An accurate and stable measurement of angle can be obtained by combining these inclinations with a proper complimentary filter. Basically, the complementary filter is a combination of a high pass filter and a low pass filter. The low pass filter acts on the accelerometer and a high pass filter acts on the gyroscope to sift the noise and drift out of the measured data.

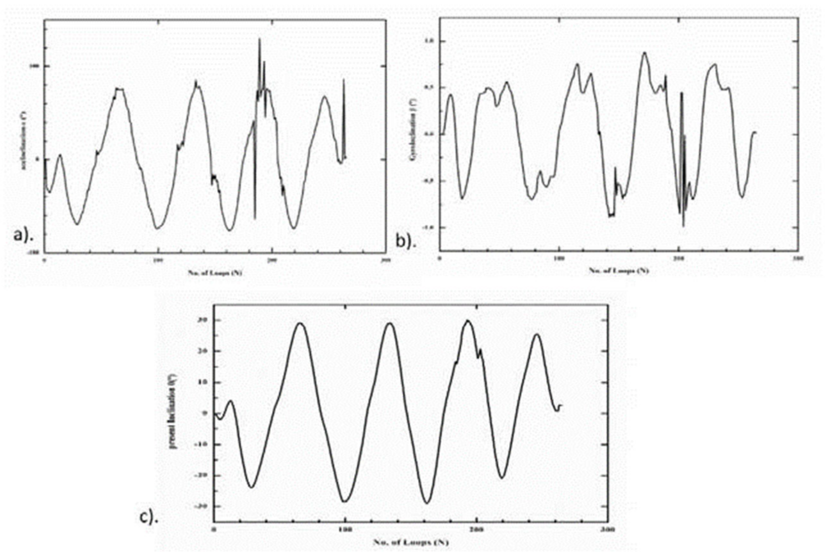

Figure 6. a). accInclination (α) *vs*. No. of loops (N) b). gyroInclination (β) *vs*. No. of Loops (N)
c). presentInclination (θ) *vs*. No. of Loops (N)

$\theta = \omega*(\beta) + (1-\omega)*(\alpha)$ (1)

$\omega = \frac{t}{t+dt} = \frac{0.75}{0.75+0.005} = 0.9934$ (2)

Filter co-efficient (ω & 1-ω) are computed from Equation. (2) as 0.9934 and 0.0066 for 0.75s, filter time constant. The only signal, which is longer than 0.75s, can pass through the low pass filter and only signal which is shorter than 0.75s can pass through the high pass filter. Further, the response of a filter can be changed by changing the value of time constant. More horizontal component of acceleration can pass though complementary filter if time constant is decreased. The graph of presentInclination (θ) (using Equation. (1)) is shown in the Fig.6 c).

4. PID Controller

PID is abbreviation for Proportional, Integral, and Derivative. Combination of these three controllers will produce a control signal (Fig. 7).

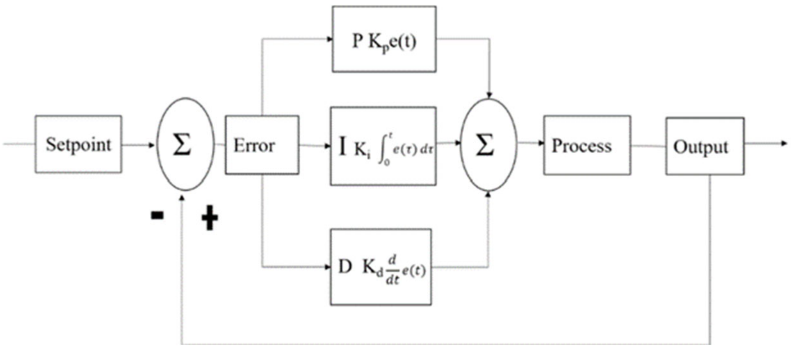

Figure 7. PID Controller [28]

The P-controller gives an output, which is proportionate to the deviation (θ). For the developed TWR, presentInclination (θ) is represented by deviation. The P- controller has limitation in the form of steady state error. Further, I-controller is required to overcome limitations of P-controller. It integrates the deviation over a period until it reaches to zero. Overshoot is the limitation of I-controller. The D-controller overcomes above limitations by predicting future behavior of deviation (θ). Combining all these results, output can be generated, which is used to drive the stepper motors

Output = $K_p*e(t) + K_i*\int e(t)dt + K_d*\frac{d}{dt}e(t)$ Where, e(t) = setpoint – input (3)

Output = $K_p*(\text{deviation}) + K_i*(\text{total deviation})dt + K_d*\frac{(\text{currentDeviation} - \text{previousDeviation})}{dt}$ (4)

Output = $Kp*(\text{deviation})+Ki*(\text{total deviation})*\text{loopTime} + Kd*\frac{(\text{currentDeviation} - \text{previousDeviation})}{\text{loopTime}}$ (5)

4.1. *Tuning Methods of PID Controller*

Various methods [29] are developed to tune PID controller till now such as trial and error method, Tyreus-Luyben method, Damped oscillation method, Zeigler-Nichols method, etc. For developed TWR, trial and error method is used to tune PID constants. In this method, values are tuned when system is working. Here, first K_i and K_d values are set to zero and then, K_p value is increased until robot started oscillating. Afterwards, K_i value is adjusted to reduce oscillations and K_d value was adjusted to obtain fast response [30]. Table 5 is also taken as reference while tuning PID values. Finally, K_p=30, K_i=1 and K_d=1 value is obtained.

Approximate PID output variable values are listed in Table 6 at different inclinations of robot. These values were obtained from serial monitor of arduino while keeping robot at different angle of inclinations. Here, PID output variable was restricted between -800 µs to 800 µs (values corresponding to -30° and 30° inclination) because it is not possible to balance robot if it inclines greater than 30° [25].

Table 5. PID parameter effect comparison [31].

Closed loop response	Rise time	Overshoot	Settling time	Steady state error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Increase	Decrease	Decrease	No change

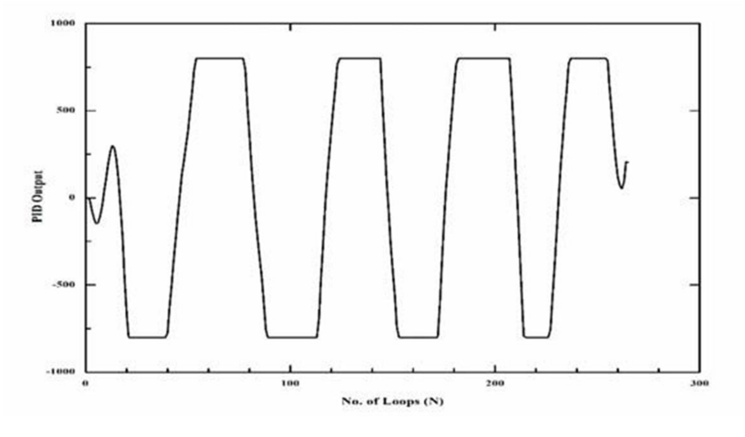

Figure 8. PID Output *vs* No. of Loops

From the Fig. 18, it is clear that PID output variable is limited to 800 µs for inclinations greater than or equal to 30° and -800 µs for inclinations lesser than or equal to -30°. Equation in the form of pulse width is written to drive stepper motors from PID output value. It means that the pulse width will be changed according to PID output variable which in turn changes speed of motors (which is proportional to inclination angle). When inclination of robot is less, it requires less speed (i.e. more pulse width) and when inclination of robot is more, it requires more speed (i.e. less pulse width). Therefore, different pulse width is provided in code for different range. 800 pulse width (in µs) is supported for 0° to 10° inclination and 300 pulse width (in µs) is supported for 10° to 30° inclination.

Table 6. PID Output values corresponding to Inclination of robot

Inclination (°)	PID Output (Approximate) (μs)
-30	-800
-20	-600
-10	-300
0	15
10	300
20	600
30	800

Our developed design (TWR) has elevated center of mass and sensor is placed in better position compared to already reported Self Balancing Robot design [12]. Also, Arduino Nano is used in place of Arduino Mega and MPU 6050 is used in place of Lilly Pad ADXL330 and Parallax L3G4200D MEMS gyroscope. Various researcher [2, 20] had used Arduino UNO in the development of their research on self-balanced robot. Arduino Nano provides the same functions with benefits of compact size and less cost. This helps in reducing the cost and space of robot.

5. Conclusion

In this paper, two wheeled robot (TWR) has been designed and developed by system design approach. It was illustrated that the TWR is capable of balance it on its two wheels. This was done using Arduino, IMU sensor and stepper motors with stepper driver. Here, novelty of the design is to drive both stepper motors simultaneously with single stepper driver. Further, Complimentary filter was also, used to obtain smoother measurements from the sensor. When all the components found working properly then, they were connected to assemble robot. Afterwards, Single degree of freedom PID control loop was implemented to balance its vertical position. PID co- efficients were tuned practically and initially, oscillating behavior of robot was obtained by using K_p value. These oscillations were reduced by introduction of K_i value. Finally, faster response was obtained with introduction of K_d value. Tuned values of K_p, K_i and K_d (30, 1 and 1) are calculated by using trial & error method. Low cost Two Wheeled Robot (TWR) has been fabricated. Weight of the developed TWR is 1.04 kg in comparison to available research literatures [5, 11]. Also, Height of developed TWR is 17.6 cm which is almost 1/4th developed by Grasser et al. [5]. Further, Wheel to wheel distance is 16.6 cm which is half of 32.5 cm in case of robot developed by Bature et al. [11]. Thus, developed TWR is light in weight, smaller in size, runs on simple code with no performance loss and less costly (Table 7).

Further, Kalman filter can be used to achieve smoother data from sensor, PID with fuzzy logic or LQR controller can be used for better control and wireless communication with robot can be implemented via Bluetooth or Wi-Fi module.

Acknowledgments

This work was supported by Centre for Creative Learning, Indian Institute of Technology, Gandhinagar, India. The authors wish to thank the staff of the IIT Gandhinagar for providing the facilities for fabrication. This work was not possible without guidance of Manish Jain & Ravi Sinha, Centre for Creative Learning, Indian Institute of Technology, Gandhinagar

References

1. Zimit, A.Y., Yap, H.J., Hamza, M.F., Siradjuddin, I., Hendrik, B., Herawan, T.: Modelling and Experimental Analysis Two-Wheeled Self Balance Robot Using PID Controller. In: International Conference on Computational Science and Its Applications, Melbourne, Australia, 2-5 July, 2018, pp. 683-698.
2. Eldhose, P.K., Dijoy, J., Anuja, J., Elizabeth P.: Controlling of Two Wheeled Self Balancing Robot using PID. IntJ of Adv Research in Elec., Electronics and Instrumentation Engineering. 7(3): 1513-1518 (2018).
3. Loram, I.D., Lakie, M.: Human balancing of an inverted pendulum: position control by small, ballistic-like, throw and catch movements. The Journal of physiology. 540 (3): 1111-1124 (2002)
4. Tsai, C.C., Ju, S.Y., Hsieh, S.M.: Trajectory tracking of a self-balancing two-wheeled robot using backstepping sliding-mode control and fuzzy basis function networks. In: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference, 2010, pp. 3943-3948.
5. Grasser, F., D'arrigo A., Colombi S., Rufer A.C.: JOE: a mobile, inverted pendulum. IEEE Transactions on industrial electronics. 49(1): 107-14 (2002).
6. Wu J., Zhang W., Wang S.: A two-wheeled self-balancing robot with the fuzzy PD control method. Mathematical Problems in Engineering. 2012. DOI.org/10.1155/2012/469491
7. Googol Technology Self-Balancing Robot GBOT1001 User Manual V1.0. Googol Technology LTD, 2007.
8. Fang, J.: The LQR controller design of two-wheeled self-balancing robot based on the particle swarm optimization algorithm. Mathematical Problems in Engineering. 2014. DOI.org/10.1155/2014/729095
9. Bonafilia, B., Gustafsson, N., Nyman, P., Nilsson, S.: Self-balancing two-wheeled robot. (Web. <http://sebastiannilsson. com/wpcontent/uploads/2013/05/Selfbalancing-two-wheeled-robot-report. pdf. 2015 Jan.)
10. Qian, Q., Wu, J., Wang, Z.: A novel configuration of two-wheeled self-balancing robot. Tehničkivjesnik. 2017; 24(2): 459-464 (2017).
11. Bature, A. A., Buyamin, S., Ahmad, M. N., Muhammad, M.: A comparison of controllers for balancing two wheeled inverted pendulum robot. International Journal of Mechanical & Mechatronics Engineering. 14(3): 62-68, (2014).
12. Juang, H. S., Lum, K. Y.: Design and control of a two-wheel self-balancing robot using the arduino microcontroller board. In: Control and Automation (ICCA), 2013 10th IEEE International Conference, 2013; pp. 634-639.
13. Sadeghian, R., Masoule, M., T.: An experimental study on the PID and Fuzzy-PID controllers on a designed two- wheeled self-balancing autonomous robot. In: Control, Instrumentation, and Automation (ICCIA), 2016 4th International Conference, 2016; pp. 313-318.
14. Bhatti, O. S., Mehmood-ul-Hasan, K., Imtiaz, M. A.: Attitude control and stabilization

of a two-wheeled self- balancing robot. Journal of Control Engineering and Applied Informatics. 17(3): 98-104, (2015).

15. Unluturk, A., Aydogdu, O., Guner, U.: Design and PID control of two wheeled autonomous balance robot. In: Electronics, Computer and Computation (ICECCO), 2013 International Conference, 7 Nov 2013; pp. 260-264.
16. https://www.qt.io/what-is-qt/
17. López-Rodríguez, F., M., Cuesta, F.: Andruino-a1: Low-cost educational mobile robot based on android and arduino. Journal of Intelligent & Robotic Systems, 81(1): 63-76, (2016).
18. Kim, Y., Kim, S. H., Kwak, Y. K.: Dynamic analysis of a nonholonomic two-wheeled inverted pendulum robot. Journal of Intelligent and Robotic Systems, 44(1): 25-46, (2005).
19. Azar, A.T., Ammar, H.H., Barakat, M.H., Saleh, M.A., Abdelwahed, M.A.: Self-balancing Robot Modeling and Control Using Two Degree of Freedom PID Controller. In: International Conference on Advanced Intelligent Systems and Informatics, 2018; pp. 64-76.
20. Zeng, B., Zhang, J., Chen, L., Wang, Y.: Self-balancing car based on ARDUINO UNO R3. In: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018; pp. 1939-1943.
21. An, W., Li, Y.: Simulation and Control of a Two-wheeled Self-balancing Robot. In: Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference, 2013; pp. 456-461.
22. Zhuang, Y., Hu, Z., Yao, Y.: Two-wheeled self-balancing robot dynamic model and controller design. In: Intelligent Control and Automation (WCICA), 2014 11th World Congress, 2014; pp. 1935-1939.
23. Nakai, M.E., Inoue, R.S., Terra, M.H., Grassi, V.: Robust Discrete-Time Markovian Control for Wheeled Mobile Robot Formation: A Fault Tolerant Approach. Journal of Intelligent & Robotic Systems, 91(2), 233-47, (2018).
24. www.arduino.cc
25. Joop Brokking, http://www.brokking.net/yabr_main.html
26. MPU6050.h library, https://github.com/jrowberg/i2cdevlib
27. Stepper Motor Connection Diagrams, Hansen Motors, https://www.hansen-motor.com/connection-diagrams.php
28. Shahrokhi M, Zomorrodi A.: Comparison of PID controller tuning methods. Department of Chemical & Petroleum Engineering Sharif University of Technology, pp. 1-12 (2013)
29. "PID Control, Dr. Stienecker's Site.", https://drstienecker.com/tech-332/7-pid-control/
30. "PID Controller - Structure &amp; Tuning Methods.", https://www.elprocus.com/the-working-of-a-pid-controller/
31. "PID Tuning.", https://www.x-toaster.com/resources/pid-tuning-for-toaster-reflow-oven/