2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) | 978-1-7281-9111-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCEEE49695.2021.9429579

Design and Implementation of Two-Wheeled Self-Balancing Robot Using PID Controller

A.Taifour Ali*,Ahmed M. O. Mohamed†, Asad S. A. Salim‡, El-Amin O. M. El-Amin§, Osman M. K. Ahmed¶

School of Electrical and Nuclear Engineering
Sudan University of Science and Technology, Sudan
Email: *awadallatayfor@sustech.edu, †ahmedsolomonn@gmail.com, ‡asaad7salem@gmail.com,
§aminomerm@gmail.com,¶osman.mohd.off@gmail.com

Abstract—**In this work, a Proportional, Integral, and Differential (PID) controller was implemented to control the stability of the Two-Wheeled Self-Balancing Robot (TWSBR). The PID parameters were manually tuned and the best performing set of parameters was chosen in the final implementation. Here the robot is implemented using an Arduino, an Inertial Measurement Unit (IMU), and two geared DC motors, all components were mounted on a plywood and steel chassis. The two DC motors applied the desired amount of torque in the right direction to enable the structure to maintain a vertical equilibrium state. Also, the mathematical model of the robot was obtained by using mechanical principles. A real-time response was plotted by connecting the Arduino to MATLAB using serial communication. The result of this work showed that the TWSBR could achieve good stability with only angle feedback and a manually tuned PID controller implemented using an Arduino. Finally, recommendations were given for reaching a better performance for future projects.**

I. Introduction

The idea of a self-balancing robot is very popular among control engineering students, as it demonstrates basic concepts of control like feedback, mathematical modeling, transfer function, controller design, etc. This idea is related to the inverted pendulum: by flipping a normal pendulum upside down and mount it onto a cart we get the inverted pendulum or IP. It's a system that has to be actively adjusted to remain stable by using a feedback loop and a controller to compensate for the error in real-time. That's why most of the control engineering textbooks like [8] and [7] use it to explain modeling and controller design. The TWSBR is considered one of the robots that are changing the way we work and live, bringing humans closer to a more automated future where robots like drones, robotic arms, and cleaning robots do most of the routine jobs with better precision, speed, and safety than human beings. Like mentioned before to make the system stable, a feedback loop is needed with a controller to compensate for the error in the tilting angle. Here, a Proportional, Integral, and Differential (PID) controller is used. The robot can sense tilt with the help of two sensors, an accelerometer, and a gyroscope embedded in an Inertial Measurement Unit (IMU). Then drives the wheels using two geared DC motors. All calculations and processing are done by an Arduino. Another feature of the TWSBR is that it can control each wheel independently of the other. In this case, it can turn left or right or even circle its center. These robots can carry and balance different objects on top of them without losing equilibrium. And some other designs may provide wireless communication for remote control. These robots are commercially available and called a Segway [12]. Another example might be the hoverboard for entertainment purposes. Although simple, the PID controller is the most used in the industrial field. Its simplicity plus cheap implementation methods make it unmatched in many applications. Also, many of the industrial devices can be linearized without much error, so using a commercially available PID controller is a common practice [10]. Here, after linearizing our model, we can use a PID controller and obtain a good performance. The angle of the robot is calculated using the signal from both the accelerometer and gyroscope by combining them with a complementary filter, to ensure an accurate angle measurement over the short and long periods.

II. Related Works

Zimit et al. [13] experimented with PID tuning for a Two-Wheeled Self-Balance (TWSB) Robot in both balancing and trajectory following. The authors derived the mathematical model of the robot by using the Lagrangian method, they represented the system in a state-space matrix. By utilizing an Arduino, stepper motors, an accelerometer, and a gyroscope, the authors designed and implemented the robot. The result showed that the designed PID controller managed to stabilize the robot and realize the trajectory following.

Mai et al. [5] combined the PID theory with fuzzy logic to control a TWSB robot. The authors designed and implemented four control loops using an STM32 as a microcontroller. Firstly, there is a PID loop to control the orientation. Secondly, a PD loop is used for position control. The third loop is a PI one that provides the needed tilt. And lastly, there is the fourth and the main loop, a fuzzy controller that keeps the equilibrium of the robot. Adding to the STM32, an Inertial Measurement Unit (IMU), and two DC motors are used for implementing the robot. The authors decided to use a complementary filter to accurately calculate the tilt angle for its easy and light implementation compared to other filters like the Kalman filter.

Imtiaz et al. [1] conducted a comparison between different controllers where they modeled, analyzed, and designed the

Two Wheeled Self Balancing Robot. Mathematical models were developed using the Lagrange equations. Then, Linear Quadratic Regulator (LQR), Proportional Integral Derivative (PID), and pole placement control methods were executed in simulation. A thorough assessment of each controller was performed. The comparison showed that the execution of the PID Controller is simpler as compared to LQR and Pole Placement as it is not built on the state-space model. Although they are difficult to implement, LQR and Pole Placement outperformed the PID implementation, yet the state-space model makes them challenging to apply. As a result, the PID controller was implemented in the actual design thanks to its simplicity. Which as expected was able to balance the robot.

Jacknoon et al. [2] adopted an optimization methodology to tune the LQR and PID controller parameters to control the inverted pendulum using the integral square error (ISE) as a cost function. The target is to move the cart to a specific point and at the same time keep and maintain the upright position of the pendulum. The first step is to obtain the state-space model for the system which is necessary for the LQR implementation. Then Ant Colony Optimization (ACO) algorithm is utilized to tune both the PID and the LQR controller. A simulation has been carried out with the help of MATLAB Simulink alongside with MATLAB script. The results indicate that (ACO) algorithm is remarkably effective to tune the parameters to achieve an optimum response. Actual results show that the use of the PID controller together with the LQR controller has a far superior response than using them individually.

In this study, Raheem et al. [9] designed two controllers to stabilize the TWSBR. A PID controller and a fuzzy logic one. MATLAB software has been used to simulate the system, and it showed an impressive performance for the fuzzy logic controller over the PID controller. Although both designs managed to reach the steady-state eventually, the fuzzy logic implementation easily surpassed the standard PID controller; it took the fuzzy controller about 2 seconds to settle at the vertical position while the PID stumbles for 4 seconds before it stabilizes. But implementing a fuzzy logic controller requires a more powerful microcontroller than the Arduino. This limitation is what made us chose the PID controller.

Hau-Shiue Juang and Kai-Yew Lum [3] used a PI-PD controller to keep a TWSBR upright. Both stability and position locking was a result of their work. The authors implemented the PI-PD controller using an Arduino. An encoder was added to each of the motors, providing speed and position signals. This feedback along with the angle signal made the system more stable compared to a basic PID controller. However, in our implementation, it was only the angle signal that provided feedback, aiming to reduce the cost of the extra encoders while achieving decent results.

III. Hardware Design

A. Mechanical part

The TWSBR's mechanical part consists of the body and two wheels as shown in figure 1. The material used to make the body is wood. It was cut into three shelves using a CNC machine. The shelves were connected using iron poles and nuts. Zippers were used to fasten the electrical parts to the shelves. The wheels were made of Teflon, it is a white solid material similar to plastic in shape and density which is easy to machine. The wheels were cut by a lathe with an 80mm diameter. The dimensions of the physical model are shown in figure 1.

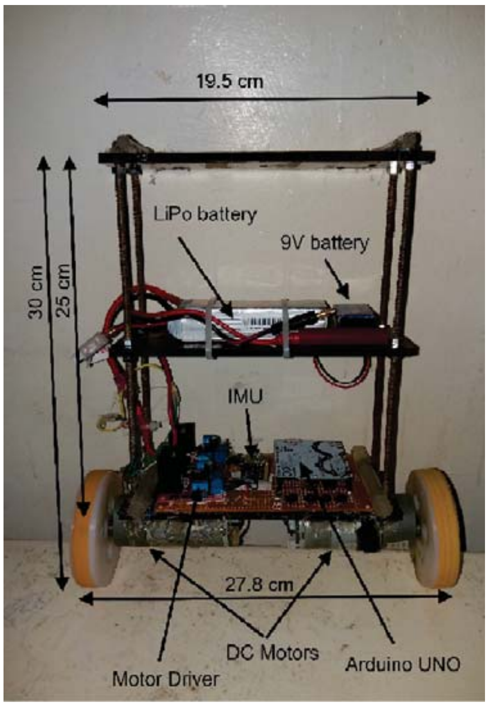

Fig. 1: Robot Body Design and Dimensions.

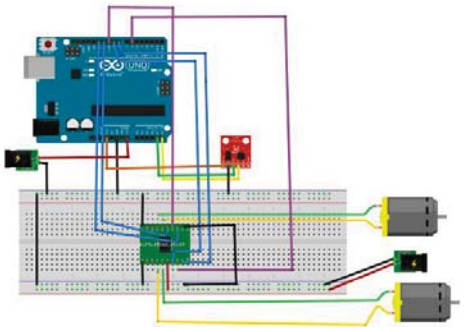
Fig. 2: Robot Electrical Schematic.

B. Electrical part

The electrical part of the robot consists of DC geared motor, LiPo battery, Arduino microcontroller, IMU, and Motor Driver. figure 2 shows the electrical schematic of the robot.

IV. Mathematical Modeling

Consider the free-body diagrams shown in figure 3. Furthermore, assume that the coordinates of the centroid (center

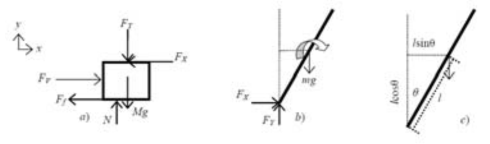

Fig. 3: Free body diagram.

of gravity) of the pendulum, are given by [6]:

$$x_G = x + l.sin\theta \tag{1}$$

$$y_G = lcos\theta \tag{2}$$

Where l is the distance from the rotation pivot of the pendulum to its center of gravity and x represents the cart's position coordinates in the x axis.

For the horizontal motion of the cart, Newton's second law of motion:

$$\sum F = M\frac{d^2x}{dt^2} \tag{3}$$

Can be written as:

$$M\frac{d^2x}{dt^2} = F_v - F_x - F_f \tag{4}$$

Assume that the friction force can be written as:

$$F_f = \gamma\frac{dx}{dt} \tag{5}$$

The horizontal motion of the pendulum can be written as:

$$F_x = m\frac{d^2x_G}{dt^2} \tag{6}$$

The second derivative of x_G can be determined using equation 1 is:

$$\frac{d^2x_G}{dt^2} = \frac{d^2x}{dt^2} - lsin\theta(\frac{d\theta}{dt})^2 + lcos\theta\frac{d^2x}{dt^2} \tag{7}$$

combining all the above into equation 4, the final form for the horizontal motion of the card can be given as:

$$(M+m)\frac{d^2x}{dt^2}+\gamma\frac{dx}{dt2} = F_v+mlsin\theta(\frac{d\theta}{dt})^2-mlcos\theta\frac{d^2\theta}{dt^2} \tag{8}$$

For the vertical motion of the pendulum:

$$F_y - mg = m\frac{d^2y_G}{dt^2} \tag{9}$$

Similar to the horizontal case the final vertical motion can be written as:

$$F_y = mg + m(-lcos\theta(\frac{d\theta}{dt})^2 - lsin\theta\frac{d^2\theta}{dt^2}) \tag{10}$$

For the pendulum, summing the moment around its center of gravity gives:

$$F_ylsin\theta - F_xlcos\theta = I\frac{d^2\theta}{dt^2}) \tag{11}$$

Substituting for F_x and F_y and simplifying gives:

$$(I + ml^2)\frac{d^2\theta}{dt^2} = mglsin\theta - mlcos\theta(\frac{d^2x}{dt^2}) \tag{12}$$

Therefore equations (8) and (12) represent the equations of motion for the inverted pendulum on a moving cart. The model of the system given by equations (8) and (12) is nonlinear and it must be linearized for the design of the PID controller. Linearization is performed at $x = 0$ m and $\theta = 0$ radians. Furthermore, it will be assumed that sinceθ is small (This is justifiable when controlling an object as it should not deviate greatly from the assumed steady-state value), also by neglecting the friction constant we get:

$$\sin\theta \approx \theta \tag{13}$$

$$\cos\theta \approx 1 \tag{14}$$

$$(\frac{d\theta}{dt})^2 \approx 0 \tag{15}$$

$$\gamma \approx 0 \tag{16}$$

Under these assumptions, equations (8) and (12) can be rewritten as:

$$(M + m)\frac{d^2x}{dt^2} = F_v - ml\frac{d^2\theta}{dt^2} \tag{17}$$

$$(I + ml^2)\frac{d^2\theta}{dt^2} = mgl - ml\frac{d^2x}{dt^2} \tag{18}$$

Using the Laplace transform:

$$F_v(s) = (M + m)s^2X(s) + mls^2\theta(s) \tag{19}$$

$$0 = mls^2X(s) + (I + ml^2)s^2\theta(s) - mgl\theta(s) \tag{20}$$

The Transform Function of the robot:

$$\frac{\theta(s)}{F_v(s)} = \frac{1}{[ml\frac{(M+m)(I+ml^2)}{ml}]s^2 + (M + m)g} \tag{21}$$

Having both electrical and mechanical parts, the DC motor has the following transfer function [11]:

$$\frac{T_m(s)}{E_a(s)} = \frac{\frac{K_t}{La}s}{s^2 + \frac{(R_a)s}{L_a} + \frac{(K_tKb)}{J_mL_a}} \tag{22}$$

V. System Controller Design

A. PID Controller

A PID controller is a simple three terms equation. Two terms are functions of the error between the measured angle of the robot and the desired angle, which is 0 degrees., the third term is just a constant. Each term is multiplied in a parameter. These three parameters are called proportional, integral, and derivative parameters. They are denoted, Kp, Ki, and Kd respectfully. Below is the equation for a PID controller output:

$$u(t) = K_p\, e(t) + K_i\int_0^t e(t)\,\mathrm{d}t + K_d\frac{e(t)}{\mathrm{d}t} \tag{23}$$

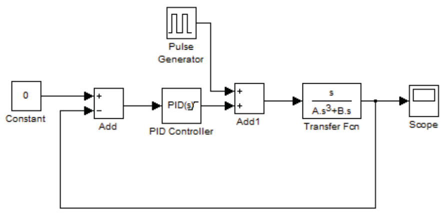

Fig. 4: SIMULINK block diagram of the system.

Experiment No.	K_p	K_i	K_d
1	1	0	0
2	5	0	0
3	10	0	0
4	15	0	0
5	20	0	0
6	30	0	0
7	30	0	0.2
8	30	0	0.5
9	30	0	0.7
10	30	0	1
11	40	0	1
12	40	0.4	1
13	40	0.8	1
14	40	1	1

TABLE I: Manual tuning of PID controller

B. SIMULINK Simulation

To simulate the impulse response of the transfer function, SIMULINK is used. The tuning of the PID controller was done manually using SIMULINK PID Controller block, as shown in figure 4 :

There are lots of methods to tune the PID parameters, some even use genetic algorithms for that [4]. Here, we relied on manual tuning, which is considered as the quick and dirty way to do that. This method requires some experience to be able to fine-tune the parameters and usually takes a lot of time to reach the desired performance. For the TWSBR, we began by increasing Kp until the robot started to tilt back and forth and oscillated a lot. After that, we increased Kd to minimize the overshoot, and ensuring the system was sufficiently dampened. Here, we were very careful not to increase Kd too much, as this will cause a massive overshoot and the whole robot will tip over. After that, Ki was tuned for the elimination of any offsets from the set-point.

The best parameters that gives the best response are: $K_p = 40, K_i = 1, K_d = 1$

VI. EXPERIMENTAL RESULTS

In this section, we examine the tuning procedure conducted to reach the final result. The results here describe the whole testing process of the project. We began our testing by only

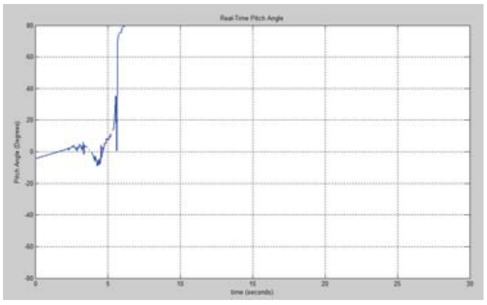

Fig. 5: $K_p = 1, K_i = 0, K_d = 0$

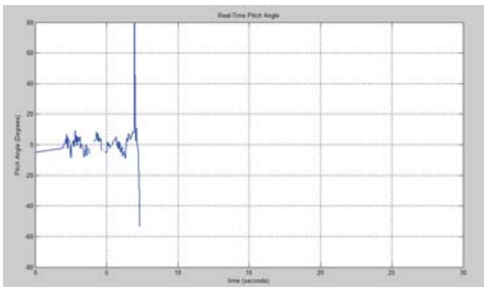

Fig. 6: $K_p = 30, K_i = 0, K_d = 0.5$

tuning K_p. While increasing K_p we observed the oscillation of the robot, and once reached an amount that caused it to overshoot in the other direction, we then started to tune K_d. During this phase, we kept an eye for the settling of the robot. Increasing K_d until a good response was observed, this good response was the attempting of the robot to balance its self and succeeding in the fastest possible time. After that K_i was tuned to remove any offset, this means if the robot balanced itself to a degree that's close to the desired input, 0 degrees, the PID controller would eliminate this offset. As shown in figure 5, without any derivative or integral error compensation and with only unity proportional feedback, the robot will definitely fall on the spot.

Increasing K_p to 30 and K_d to 0.5 while keeping $K_i = 0$, we examine in figure 6 a slight oscillation but the robot still tip over after a short time.

Finally, by setting the controller parameters to these values $K_p = 40, K_i = 1, K_d = 1$ we observe in figure 7 that the robot manages to stay upright and doesn't fall. There is still some oscillation in the response due to the poor mechanical design of the robot.

VII. CONCLUSION

An approach to balance a TWSBR using a manually tuned PID controller was introduced in this paper. The robot's mathematical model was derived using the laws of physics and Laplace transforms. It was demonstrated that an Arduino can be used for controlling the robot smoothly. It was found that the best set of values for the PID controller were, $K_p = 40, K_i = 1, K_d = 1$. Although the results were satisfactory, many improvements could've been made. Like building a better mechanical frame, using a Kalman filter, and

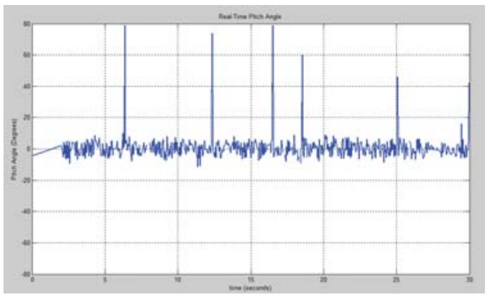

Fig. 7: $K_p = 40, K_i = 1, K_d = 1$

replacing the DC motors with stepper motors. For future work, it's recommended to use a combination of PID and Fuzzy logic controller and to wirelessly tune the controller and also control the robot.

References

[1] M. A. Imtiaz, M. Naveed, N. Bibi, S. Aziz, and S. Z. H. Naqvi. Control system design, analysis implementation of two wheeled self balancing robot (twsbr). In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 431–437, 2018. doi: 10.1109/IEMCON.2018.8614858.

[2] Aman Jacknoon and MA Abido. Ant colony based lqr and pid tuned parameters for controlling inverted pendulum. In *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pages 1–8. IEEE, 2017.

[3] H. Juang and K. Lurrr. Design and control of a two-wheel self-balancing robot using the arduino microcontroller board. In *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pages 634–639, 2013.

[4] Mohamed Magdy, Abdallah El Marhomy, and Mahmoud A. Attia. Modeling of inverted pendulum system with gravitational search algorithm optimized controller. *Ain Shams Engineering Journal*, 10(1):129 – 149, 2019. ISSN 2090-4479. doi: https://doi.org/10.1016/j.asej.2018.11.001. URL http://www.sciencedirect.com/science/article/pii/S209044791830073X.

[5] The Anh Mai, Thai Son Dang, DN Anisimov, and Ekaterina Fedorova. Fuzzy-pid controller for two wheels balancing robot based on stm32 microcontroller. In *2019 International Conference on Engineering Technologies and Computer Science (EnT)*, pages 20–24. IEEE, 2019.

[6] Vicky Mudeng, Barokatun Hassanah, Yun Tonce Kusuma Priyanto, and Okcy Saputra. Design and simulation of two-wheeled balancing mobile robot with pid controller. *International Journal of Sustainable Transportation*, 3 (1):12–19, 2020.

[7] N.S. Nise. *Control Systems Engineering*. 8th edition, 2019. ISBN 9788126537280. URL https://www.wiley.com/en-us/Control+Systems+Engineering%2C+8th+Edition-p-9781119474227.

[8] K. Ogata. *Modern Control Engineering*. Instrumentation and controls series. Prentice Hall, 2010. ISBN 9780136156734. URL https://books.google.com/books?id=Wu5GpNAelzkC.

[9] Firas A Raheem, Bashar F Midhat, and Hussein S Mohammed. Pid and fuzzy logic controller design for balancing robot stabilization. 2017.

[10] Researchgate. URL https://www.researchgate.net/post/What_is_the_percentage_of_the_PID_algorithm_applications_in_industry.

[11] RIT. URL http://edge.rit.edu/edge/P14453/public/Research/DC%20Motor%20Transfer%20Function%20Example.pdf.

[12] wikipedia. URL https://en.wikipedia.org/wiki/Segway.

[13] Aminu Yahaya Zimit, Hwa Jen Yap, Mukhtar Fatihu Hamza, Indrazno Siradjuddin, Billy Hendrik, and Tutut Herawan. Modelling and experimental analysis two-wheeled self balance robot using pid controller. In *International Conference on Computational Science and Its Applications*, pages 683–698. Springer, 2018.