GU J Sci, Part C, 9(4): 597-607 (2021)

Gazi University

Journal of Science

PART C: DESIGN AND TECHNOLOGY

http://dergipark.gov.tr/gujsc

Use of PID control during Education in Reinforcement Learning on Two Wheel Balance Robot

Emrah ATAC[1] Kazim YILDIZ[2] E. Emre. ULKU[2]

[1]*Marmara University, Institute of Pure and Applied Sciences, Mechatronics Engineering, Istanbul/TURKEY*

[2]*Marmara University, Faculty of Technology, Computer Engineering, Istanbul/TURKEY*

Article Info

Research article
Received:21.06.2021
Revision:14.10.2021
Accepted:06.11.2021

Keywords

Reinforcement
Machine Learning
Balancing Robot
PID control

Abstract

This study's primary objective was to try to shorten the training time of the Reinforcement Learning (RL) method, which is one of the Machine Learning methods, by using the proportional-integral-derivative (PID) control method during training. In this study, a balancing robot with two wheels that can be controlled independently on the same axis is used. While the robot is in balance, the RL software block follows how the PID block maintains the balance, and the RL blog learned how to behave against disturbing factors without physical falling/rising. In the training of RL, it is necessary to create approximately 500 policy/reward/path equations between the current state and future state matrices. The number of equations will increase considerably when subjects such as old position and acceleration are added. Approximately 1000 trial/error is required for training purposes in alone RL. This means many falling/rising cycles. With the method we present, the RL block has learned to keep the robot in balance without falling and requiring human intervention in 900 trials. The time spent for a fall/stand-up with RL alone was measured to be about 30 seconds (approximately 9 hours for 1000 attempts). On the other hand, PID-assisted learning took less than 4 hours of training since falling did not occur in many trials. This shows that the training period is shortened by approximately 60%.

1. INTRODUCTION

Two-wheel self-balancing robots (TWSBR) are systems with controllable degrees of freedom lower than total degrees of freedom [1]. The balance of two-wheeled robots has been a demanding issue in the field of system control. In this context, TWSBR emerges as a very important research topic in self-learning smart systems. In comparison with traditional robots, the brain of such robots can constantly evolve according to external environments. This intelligence has been produced similar to the human brain by people who work in some area of science [9]. Some of the most famous of TWSBR are Boston Dynamic Handle [5], Ascento [6] and Segway [7].

The motion equations of wheeled balance robots are largely similar to inverted pendulum equations [23] and they can be used in the same way for TWSBRs. Many researchers have worked on stability analysis and control system design of TWSBR. However, less research has been devoted to the dynamic modeling problem [4]. Self-balancing of TWSBR is controlled by embedded software. This software is fed online by sensors and transducers [9] with the information they receive from the outside environment. Researchers have proposed several control approaches: Liangliang Cui horse et al. worked on the method of Support Vector Regression [10], Chia-Hong Chen et al. suggested Fuzzy Logic [2], and [4] Proportional Integral Derivative (PID), Linear Quadratic Regulator (LQR) and pole placement control methods were investigated. Ebin Philip, Sharath Golluri investigated cascade PID control systems for autonomous balanced driving robots [15]. Ren Hongge et al. represented a learning approaching named Bionic Learning [21]. Some of the above approaches of algorithms for TWSBRs are based on neural networks. One of its

*Corresponding author, e-mail: kazim.yildiz@marmara.edu.tr
DOI: 10.29109/gujsc.955562

biggest advantages is its high fault tolerance. Nevertheless, its deficits are poor learning ability and sensitivity to external noise. So it is difficult for the controller to reach a steady state [9]. PID and LQR require the correct derivation of the mathematical equations of the system established as a traditional approach. Farias and friends developed an algorithm to control the position of a wheeled mobile robot. The main advantage is learning procedure which is done automatically with recursive procedure [24]. Farias and friends proposed an 3D simulation environment for control the position of a wheeled mobile robot. They proposed and algorithm about two phases, which are learning and operational stage. So balancing the robot is performed with this two sage algorithm [25] Cui and friends proposed the adaptive optimal control problem for robot. The paper presents a solution for adaptive control which is learning based. Experiments were carried out to show the efficiency of the new adaptive suboptimal controller in balancing the wheel-legged robot [26].

In addition to traditional PID algorithms, existing Neural Networks (NN) have proven to be a good method in the field of control. Better adaptability to environmental and mass changes can be achieved, either by using basic NN alone or by using repetitive NN. Although various solutions give an appropriate responses in solving the balancing issue, it is frequently very difficult to compare distinct controllers, particularly those diverse NN algorithms, because of several models of the robot used for testing, dissimilar empirical circumstances, and value of parameters. All these differences make it difficult to analyze algorithms for achieving and controlling balance [3].

In real-time control, Reinforcement learning is widely used and is an unsupervised learning approach [9]. In the reinforcement learning of the robot world, according to the trial/error method, the controller connects with the external environment through sensors and provides the action with the actuators. The policy defines the way the robot behaves at a given time. In other words, it is the graded relationship between the current situation and the environment. The algorithm must work to find the maximum value of the generated policies. “Long Short Term Memory” (LSTM) was used in this study. While the LSTM uses the angle to make decisions about a particular action, it also uses the angle change rate and knowledge learned in past steps. This adds stability to the robot's behavior. The output value is generated according to the results. At the same time, the internal status is also updated. In this way, it is ensured that the experiences gained in the past situations affect future decisions [3].

In this study, how to get help from traditional PID in RL training is discussed. Thus, it was hoped that the transition from the traditional field to RL would be easier for many fields. For this purpose, there are two blocks in the software of this study. First, the traditional PID block keeps the robot in balance. Raw Angle and acceleration information is obtained from the sensor at 100 samples per second. The data is filtered and reduced to 10 samples. thus providing resistance to noise. The robot stands by providing motor movement for each data. PWM is created by processing the motor movement with proportional (Kp), integral (Ki) and derivative (Kd) coefficients of the data. The motor driver is powered by PWM. Thus, the movement of the motor to keep the robot in balance is performed.

The coefficients of Kp = 0.23 Ki = 0.01 Kd = 0.1 are found by trial/error method. First, Ki = 0.0, Kd = 0.0, and an oscillatory movement was performed while the Kp value was 0.23. When Kp = 0.23 and Kd = 0.0, oscillating motion is more balanced and short oscillating stance is obtained at Ki value 0.01. finally, an acceptable balance was achieved when Kd was 0.1. it took about 40 man/hour to obtain these values. It works even if the values are not completely satisfactory. The second block, the RL block, follows angle and wheel control to assign a reward value to the principles. It periodically sends a disturbing signal of varying intensities to the wheel control. So it captures more angle situations and gives reward value to its policies based on how PID controls the motor. This value is between -1 and 1.

2.MATERIALS AND METHODS

The general view of all physical components of the robot is described in this section. The system is inherently unstable. That's why it tends to tip over. The microcontroller needs to constantly monitor the robot's current angle and be fast to keep its balance. A little bit of math and a little patience is required to realize them.

Generally, the system of the robot is as Figure 1. Figure 1 is a simplified view of the balancing robot. CG is the center of gravity.

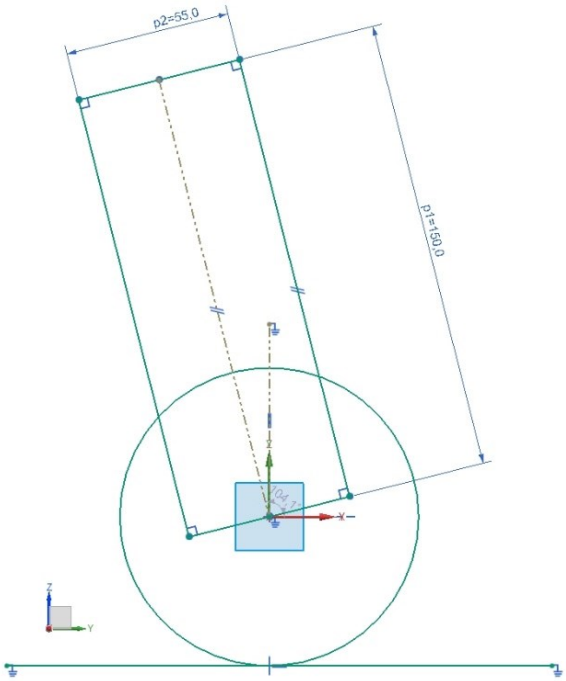

Figure 1. *General dynamics of the system*

The system can be written as equation 1:

$$\ddot{\theta} = \frac{g}{l}\sin\theta + \frac{u}{l}\cos\theta \quad (1)$$

where: l is the span from the robot's center of mass to the axis of wheel rotation center, g represents the value of gravity acceleration and u ($\dot{x}$) represents the linear acceleration through the x-axis. While θ represents the angle of inclination, $\ddot{\theta}$ is the angular acceleration. Figure 2 shows the general dimensions of the robot and details.

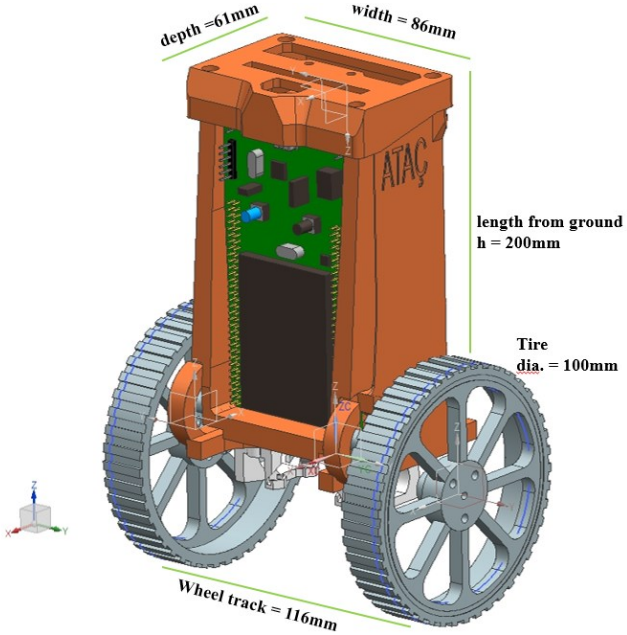

Figure 2. *General dimensions of the robot*

400rpm 6v motors are used as actuators. The geared actuators drive the wheel which is 100mm in diameter. To measure the angle, the MPU-6500, which includes a MEMS-based accelerometer and gyroscope, was used on the top of the robot. In this module one "Digital Motion Processor" (DMP) is located. The accelerometer which evaluates acceleration in three "x, y, z" axes is also located. But it has a drawback: sensitivity to noise. The gyroscope, which is stable against noise. It measures the angular ratio around all x, y, z axes. As the battery, an external charging battery with 5000mah USB output for 5v microcontroller supply and a 7v lithium-ion battery pack is used for the motor. STM32F4 discovery board was used as a microcontroller. MX1508 module was used as a motor driver due to its small size. This module allows PWM driving. In Figure 3 the general block diagram of the system is shown.

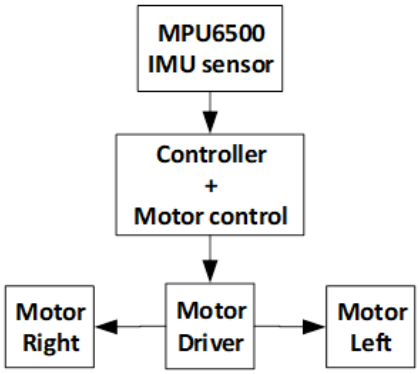

Figure 3. *General block diagram of the system*

When the system is started, the angle information from the MPU6500 sensor starts to be read online by the STM32F4. This angle is the θ value that can be seen in Figure 1. The control software (PID or RL) embedded in the microcontroller produces various values according to the θ angle. Using these values, the motor control software manages the motors in the PWM method via the driver circuit.

PWM duty cycle is like the muscle power of the robot. When it's 100 (which means 100%), it means the motor is spinning at full speed 400rpm. When the duty cycle is 0 at 50 to 200 rpm, there is no energy to the motor. The important parameters of the system are voltage current and moment of inertia. In mathematical modeling, these must be obtained correctly. However, in these studies, the duty cycle was determined by trial and error so that the PID could keep it in balance. If the θ angle is not close to the set value of 0 at the first startup of the robot, the system will not work. When the robot is brought into balance, the PID works to keep it in balance. RL training for the first about 4 hours is performed as in Figure 7. Subsequently, RL training is considered complete. RL continues the balance job.

In the software run on the STM32, the end of the training is determined simply: The training is terminated when all the positions in the state vector have values according to the movements in the action vector. After that, only the RL block manages the balancing process. It is clear that if the resolution of the position vector (number of elements) is increased, the training time will increase to a certain extent. This also means increased processing load for the processor and provides a more stable balance. An optimum point must be determined here.

Figure 4 shows the robot and its internal structure. It is important that each element and cable are firmly fixed. The robot body is printed by 3d printer.

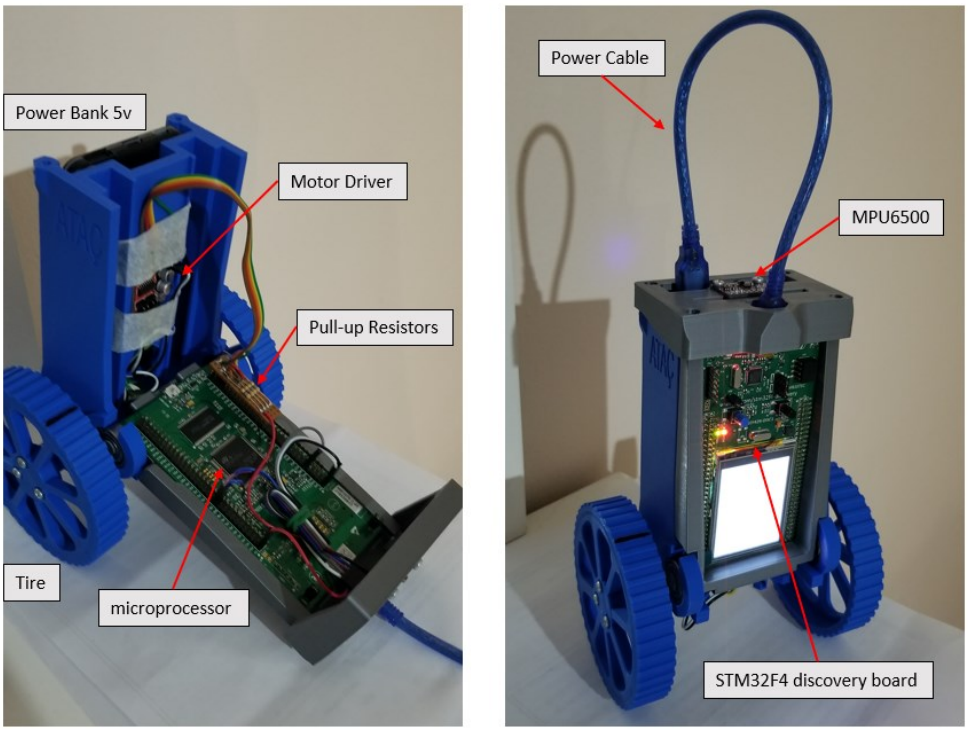

Figure 4. *Designed robot*

2.1. PID controller

Since TWSBR has highly unbalanced dynamics, it is absolutely necessary to use a control system to stabilize it around the set point (vertical position). In this context, the PID controller with basic PID block is shown in Figure 5 and equation (2) were applied to ensure the perpendicularity of the robot. The PID controller's main purpose is to keep the robot's tilt angle close to the setpoint $\theta = 0$. In our case, the IMU is 0° as the robot is mounted horizontally at the top. The setpoint rate is given to the microcontroller (STM32F4) as a reference signal and errorr $e(t)$ which is calculated from the angle is also given through minus feedback. The error is then used to calculate the adjustment signal or input signal $u(t)$ as given below in equation (2).

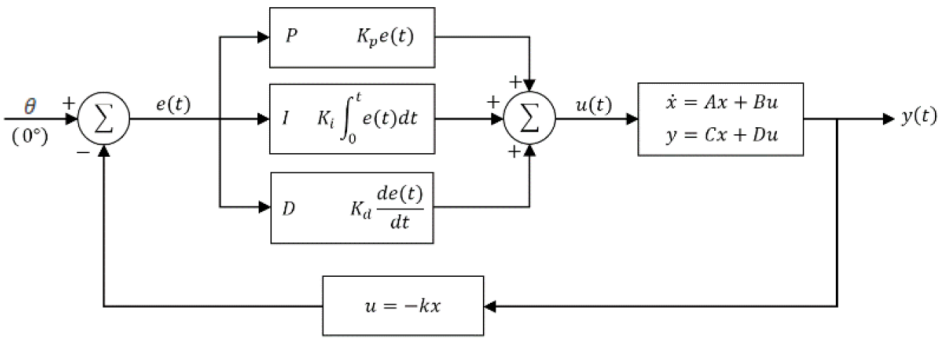

Figure 5.*Basic PID block*

The block can be written as equation 2

$$u(t) = \mathrm{Kp} * \mathrm{e(t)} + Ki * \int_0^t e(t)dt + Kd\frac{de(t)}{dt} \qquad (2)$$

Figure 5 and equation (2) show the flow and mathematical equation of the PID block. Here, $e(t)$ is an error, u is feedback, P I and D are mathematical correction blocks, $u(t)$ is control signal. In our study, Kp Ki and Kd values are 0.23, 0.01 and 0.1, respectively [14].

2.2.Reinforcement Q Learning

The Reinforcement learning includes a reward/punishment assessment. In this method, the action which gets the highest reward value in a given situation is found. One of the hallmarks of RL is that the agent performs the action and then receives a reward. In this reward system, it interacts with the environment to define the most appropriate policy (i.e. the highest rewarded action) through trial and error [11].

Figure 6 shows the interaction of agent / environment (in this case the environment is the robot's vertical angle theta) in reinforcement learning. "Agent" and "Environment" are two main elements of reinforcement learning. “Agent” is decribed as the learner responsible for decision making and the "environment" interacts with it. The learning process involves the setting that presents the agent with the current situation or situation in which the agent will choose appropriate action. After that environment will then generate a reward value based on the new situation and action taken.

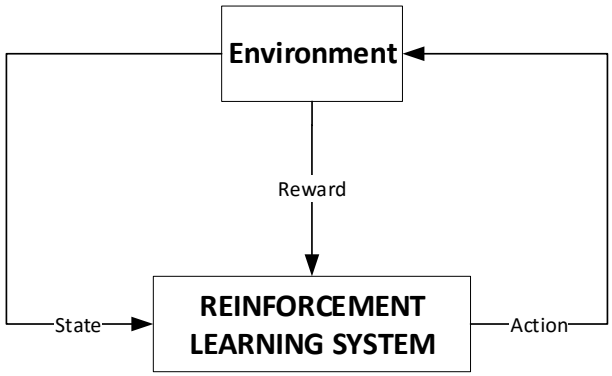

***Figure 6**. Basic RL block diagram*

Q learning is a model-independent reinforcement learning method, presented by And Watkings. The learning process is the embodied concept that intelligence is a process that is deeply dependent on the robot's interaction with the environment. An independent agent, usually formulated as a “Markov Decision Process” (MDP). A gyro sensor, actuator, and learning cycle are used in reinforcement learning. The agent gets input from the gyro sensor from the medium representing the current state (S). The most appropriate action (A) is selected based on the current siuation, knowledge and goals. After receiving feedback from the environment by receiving reward values (R), the agent learns to receive positive rewards in the future [17] [20].

To find the most proper training tactics, investigators present the concept of quality function. Recently, the Q-function is a commonly used quality function. In Q-learning, the reward is based on the pair of status-action, and the update rule can be written as equation 3;

$$V(S) = max_a Q(s,a)$$

$$Q_{t+1}(s,a) = (1-c)Q_t(s,a) + c(r + \gamma V(s')) \quad (3)$$

where c is learning coefficient, γ discount factor, r active reward ve s' future state [9]. State Vector = [-5.0 ; -4.9 ; … -0.2 ; -0.1 ; 0 ; 0.1 ; 0.2 ; … 4.9 ; 5.0]. These right angles are the values of θ.

Action Vector = [-100 ; 95 ; … ; -15 ; -10 ; -5 ; 0 ; 5 ; 10 ; 15 ; … 95 ; 100]. This indicates the motor's PWM duty time. 100 means go in one direction at maximum speed. - values rotate the motor in the opposite direction. Simultaneous fixations of the two vectors' value to 0 indicate the static equilibrium position.

Training mode: Robot opens and balances on its own wheels with PID. Meanwhile, the RL block is in training mode. PID keeps the robot upright during this training period. The RL block which is in the training mode, reads the data from the sensor and the duty cycle produced by the PID. Since the robot stands upright with PID, the reward process is matched with high values and appropriate state-action. In this process, if the robot was not kept alive with PID, the training process with each reward cycle would have taken hours.

In the RL training phase, it sends a disruptive PWM duty cycle value to the robot, which is in balance with the PID. It keeps the maximum distortion value of the angle, here the relevant action vector element is valued by assigning a value to the current PWM duty cycle by taking the old state and state (current IMU value) while coming to the maximum angle and returning to the equilibrium state from the maximum angle. The same old state and states will appear in the correction of disruptors. In this case, whichever action vector element is evaluated most times will have the highest reward. Thus, while keeping the RL alone in balance, the PWM duty cycle is driven with the most valuable action vector element in the same old state and state, and the motor is moved. There is a flowchart of the training phase can be seen in Figure 7.

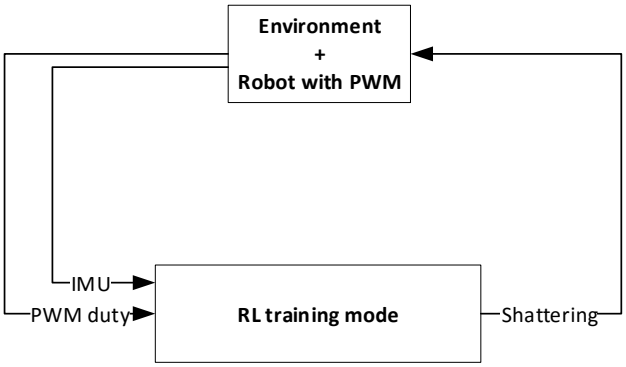

***Figure** 7. Training mode for RL block*

In RL standard training, the path between the state vector and the action vector is determined and rewarded. The reward is given higher, the new situation after the action is closer to the set point. Award value set [-1; 1] has a resolution of 0.1 in the range. The RL block sends destabilizing signals to the motor of the robot that is in balance with the PID. Then it reads the angle change and monitors the balance by PWM duty changes. It also takes the previous value as input and saves them as the high reward path in its policy / agent. For example, even if the robot is in full equilibrium with the PID, the RL block sends a disturbing signal to the motor drivers. Measures and saves the new state (this maximum disturbed location) with the IMU. This value is 3 degrees. PID drives motors with PWM in contrast. The RL block software reads and saves the duty time of the PWM. It saves a high reward path between these two values. Thus, rewarded paths are assigned to all elements in the state vector.

Figure 8 shows the flowchart of the RL block after training. As shown here, when the training period is over, the "RL active" mode is turned on and the PID is turned off. The trained RL now stabilizes the robot: a PWM duty cycle is generated by selecting the robot, the environmental state, the state vector, and the action vector element evaluated according to the previous state. Thus, the motor is driven and balance is achieved.

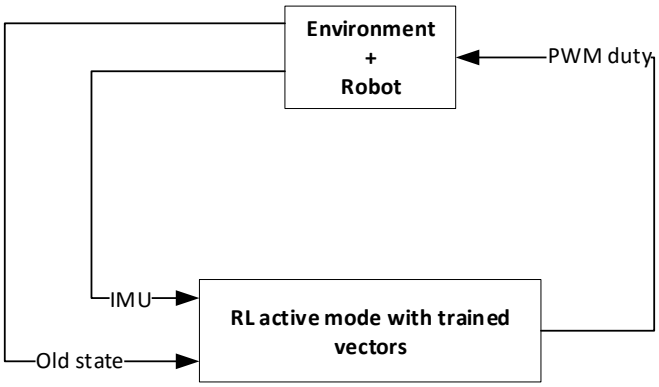

Figure 8. *Active mode for RL block*

An example situation is that at time n the Robot is in equilibrium at 0 degrees. With the measurement from the IMU, State_Vector[n] = 0. Let State_Vector[n-1] be 0 at the previous instant (Old state). These values enter the RL block. As learned from the block trainings, Action_Vector[n] = 0 comes out as the highest reward (very close to -1) and PWM duty cycle time value is assigned to the motor driver so that the motor does not move. Small imbalance at time n+1 When State_Vector[n+1] = 0.9; It enters the RL block with State_Vector[n] = 0. The most rewarded Action_Vector[n] = -80 is the output. This gives energy to the engine in a way that the duty cycle will be 20 in the direction that will correct the balance. At the time of n+2, State_Vector[n+2] = -0.2 and as old state State_Vector[n+1] = 0.9 when it feeds the RL block, Action_Vector[n] = 5 and the robot balance approaches 0. It enters the stable area around 0. When the state vector resolution is 0.1, the approximate oscillation is +-0.1 degrees (fig 9). The higher the resolution of this vector, the smaller the oscillations. At very small resolutions, such as 0.001, although the sensor can measure it, the motor cannot respond to it. Gear gaps prevent the wheel from turning.

The transition from a balanced state to an unstable state and returning to a balanced state with action is defined as a full training cycle. The greater the number of training given, which is directly related to the action vector, the more stable the robot seems to be. One of the important criteria is action vector sensitivity. The motor as an actuator is fine-tuned to correspond to reaction time. High resolution will increase accuracy, but the microcontroller clock speed may not reach the appropriate hertz. Furthermore, due to motor inertia, there may not be enough time for action response.

3.RESULTS

For each In this article, reinforcement learning is used to provide control of TWSBR. First, the training process was carried out with the classical PWM method. Then RL performed the robot's balance stance.

The recommended control method has been classical Reinforcement Q Learning. Here, the state and action vectors are determined in the software at the appropriate resolution (0.1step). In this state, the RL block could be taken to training by hovering, or the mathematical model could be extracted and the training process could be carried out on the computer and transferred to the robot. Here, the PID parameters were determined to differ from the previous applications, and the RL part was trained with the robot standing in balance, by reducing the amount of falling.

The Q-Learning method was chosen using the angle of slope, angular velocity as state elements and clockwise-counterclockwise rotations with different PWM duty cycles as action elements. During the training, only PID training was tried to save time. It is clear that the standard RL training will pass 500 fall

/ rise cycles. No more than 10 drops were observed in education supported by PID. Thus, the training was completed in a short time. Vectors with a resolution of 0.1 provided the appropriate solution. For better balance, the range of vectors can be expanded and resolution can be lowered. However, this will put a load on the processor. At a resolution of 0.001, it is unnecessary as the motor cannot give the necessary responses. At higher resolutions, the sensor cannot provide appropriate data.

One of the difficulties of PID is that it requires a model to determine its parameters. Or it must be proceeded with trial and error, which is a tiring job. Moreover, poorly determined parameters persistent error late sitting causes situations such as oscillation, because the RL block undergoes an incorrect training period and similarly correct oscillatory results. In order to overcome this, both PID parameters can be updated and the RL learning process can be operated by working together with the Autotune PID parameters and RL. Table 1 shows the values policy/agents which are taken.

Table 1. *Policy/agent summary table and reward values*

#Policy	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
State_old	*-0.1*	*0*	*-3.3*	*0*	*4.6*	*2.7*	*1.2*	*4.1*	*-0.8*
action	*0*	*-5*	*-80*	*-70*	*90*	*-50*	*-40*	*65*	*-50*
state	*0*	*0.9*	*-0.9*	*1.1*	*0.1*	*-0.7*	*-0.3*	*-0.1*	*0.8*
REWARD	*0.95*	*0.80*	*0.75*	*-0.55*	*0.90*	*0.65*	*0.80*	*0.50*	*0.30*

The demonstration of the robot returning to equilibrium with a small disturbing impact when the robot is in balance is shown in Figure 9. The samples are obtained by recording the angle values in each cycle of the software. The total time is about 1.3 seconds. Angle values of 15 moments (approximately equal) taken during this period are also shown. The red line is the angle values obtained during PID block operation. An attempt was made to deliver a similar blow to the robot under RL control. The blue line represents the RL block. Observe how RL fluctuates on the zero axis. This is related to the resolution of State and Action vectors. It is clear that education needs to be improved. It can be read from Figure 9, Rl block which is not soft and agile enough like PID. Despite all this, the RL block manages to keep the robot standing on its own.

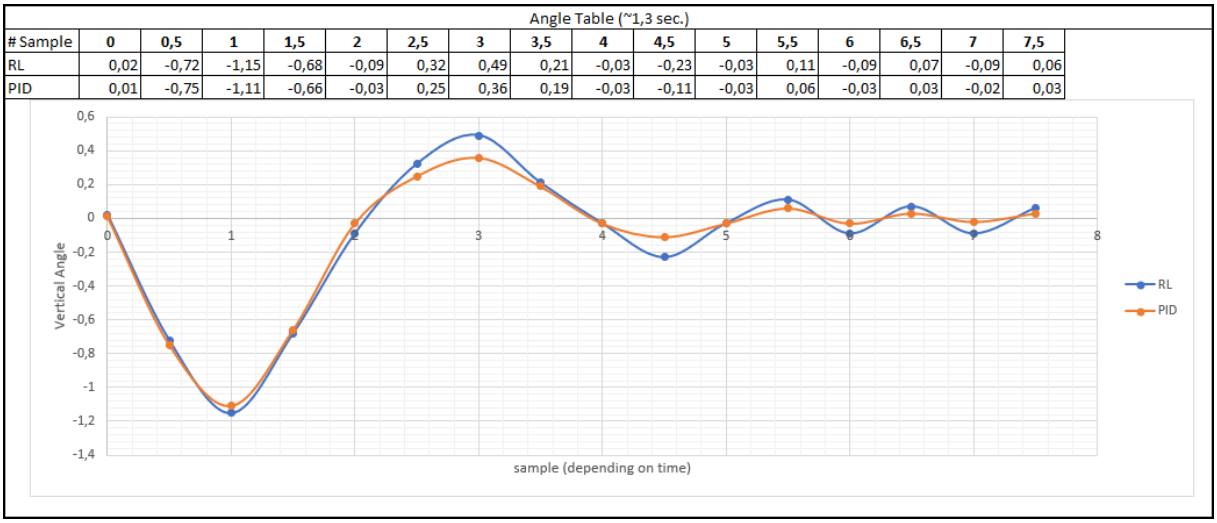

Figure 9. *Sample/degree graph of robot*

On the other hand, for a smoother operation for RL: The resolution of the state vector can be automatically increased and the resolution of the action vector can be adjusted automatically. This can be examined in future studies. According to the disturbance signal, its sign and its size (the robot in balance), the way and the amount of distortion of the angle can also be monitored and only if an RL training method can be developed from here is a good topic that can be examined in future studies. Finally, since a real application

is performed, the robot created must have a solid rigid body and well-fixed internal elements. Electronic cards that are not fixed tend to fail.

4.CONCLUSIONS

In the present In this article, the training process of RL in optimum control of TWSBR with approximate PID parameter values with math model was discussed. Both the PID control method and RL control method designs were considered. The proposed control program, to realize optimum control of the robot, is an online feedback Q-Learning method which is used commonly. The Q-Learning training process is shortened by supporting the PID algorithm. While PID control partially requires the mathematical equations of the robot, Q-Learning control does not require a mathematical model by determining the state and action vectors at the appropriate resolution. While the Q-learning algorithm that is not need mathematical equation can start with a random policy of control and progress through a trial / error process, the PID algorithm has a better steady-state response. During the training process, an average PID control observer was helpful. In this way, when applying Q-learning control algorithms, the numerical difficulties caused by the large-volume matrices and instability caused by the system were overcome. As a result, the time spent for a fall/stand-up with RL alone was measured to be about 30 seconds (approximately 9 hours for 1000 attempts). On the other hand, PID-assisted learning took less than 4 hours of training, since falling did not occur in many trials. The study of PID use in RL training is thus accomplished by sending an unbalanced PWM duty cycle value. This shows that the training period is shortened by approximately 60%.

REFERENCES

[1] Ali Ghaffari, Azadeh Shariati, Amir H. Shamekhi. “A modified dynamical formulation for two-wheeled self-balancing robots” (Article - DOI 10.1007/s11071-015-2321-9)

[2] Chıa-Hong Chen, Jong-Hann Jean, Dao-Xıang Xu. “Applıcatıon Of Fuzzy Control For Self-Balancıng Two-Wheel Vehıcle” (Article - 978-1-4577-0308-9/11/$26.00 ©2011 IEEE)

[3] Raudys A, Subonien A. “A Review of Self-balancing Robot Reinforcement Learning Algorithms” (Article - ICIST 2020, CCIS 1283, pp. 159–170, 2020.)

[4] Muhammad Atif Imtiaz, Mahum Naveed, Nimra Bibi, Sumair Aziz, Syed Zohaib Hassan Naqvi. “Control System Design, Analysis & Implementation of Two Wheeled Self Balancing Robot” (Article - 978-1-5386-7266-2/18/$31.00 ©2018 IEEE)

[5] Boston Dynamic (website) - https://www.bostondynamics.com/handle

[6] Ascento (website) - https://www.ascento.ethz.ch/

[7] Segway - “Segway Inc.: Reference manual, Segway personal transporter (PT).” Segway Inc., Bedford, NH (2006)

[8] R.E. Parr, “Hierarchical Control and Learning for Markov Decision Processes.” University of California: Berkeley, 1998.

[9] Juan Yan, Huibin Yang. “Hierarchical Reinforcement Learning Based Self-balancing Algorithm for Two-wheeled Robots.” (Article - DOI: 10.2174/1874129001610010069)

[10] Liangliang Cui, Yongsheng Ou, Junbo Xin, Dawei Dai, Xiang Gao. “Control of a Two-Wheeled Self-Balancing Robot with Support Vector Regression Method.” (Article - 978-1-4799-4808-6 /14/$31.00 ©2014 IEEE)

[11] Shih-Yu Chang, Ching-Lung Chang. “Using Reinforcement Learning to Achieve Two Wheeled Self Balancing Control” (Article - 978-1-5090-3438-3/16 $31.00 © 2016 IEEE)

[12] Guoping You, Wanghui Zeng. “Design of Two-Wheel Balance Car Based on STM32” (Article - 2474-3828/18/$31.00 ©2018 IEEE)

[13] Tao Zhao, Qian Yu, Songyi Dian, Rui Guo, Shengchuan Li. “Non-singleton General Type-2 Fuzzy Control for a Two-Wheeled Self-Balancing Robot” (Article - https://doi.org/10.1007/s40815-019-00664-4)

[14] Chinmay Samak, Temo Samak. “Design of a Two-Wheel Self-Balancing Robot with the Implementation of a Novel State Feedback for PID Controller using On-Board State Estimation Algorithm” (Article·- ISSN(P): 2250-1592; ISSN(E): 2278-9421)

[15] Ebin Philip, Sharath Golluri. “Implementation of an Autonomous Self-Balancing Robot Using Cascaded PID Strategy” (Article - 978-1-7281-6139-6/20/$31.00 ©2020 IEEE)

[16] Linyuan Guo, Syed Ali Asad Rizvi, Zongli Lin. “Optimal control of a two-wheeled self-balancing robot by reinforcement learning.” (Article - DOI: 10.1002/rnc.5058)

[17] The Anh Mai, D. N. Anisimov, Thai Son Dang, Van Nam Dinh. “Development of a microcontroller-based adaptive fuzzy controller for a two-wheeled self-balancing robot.”(Article - 00542-018-3825-2)

[18] MD Muhaimin Rahman , S. M. Hasanur Rashid, M. M. Hossain. “Implementation of Q learning and deep Q network for controlling a self balancing robot model.” (Article - https://doi.org/10.1186/s40638-018-0091-9)

[19] Penghui Xia, Yanjie Li. “The Control of Two-Wheeled Self-Balancing Vehicle based on Reinforcement Learning in a Continuous domain.” (Article - 978-1-5386-2901-7/17/$31.00 ©2017 IEEE)

[20] Sun Liang, Feimei Gan. “Balance Control of two-wheeled Robot Based on Reinforcement Learning.” (Article - 978-l-61284-088-8/ll/$26.00 ©2011 IEEE)

[21] Ren Hongge, Wang Zhilong, Li Fujin ,Huo Meijie. “The Balance Control of Two-wheeled Robot Based on Bionic Learning Algorithm” (Article - 978-1-4799-3708-0/14/$31.00c 2014 IEEE)

[22] Pete Warden, Daniel Situnayake. “TinyML - Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers” (Book - Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.)

[23] Yunus Çelik, Mahit Güneş. “Designing an Object Tracker Self-Balancing Robot” (Article - Academic Platform Journal of Engineering and Science 6-1, 124-133, 2018)

[24] Gonzalo Farias, Gonzalo Garcia, Guelis Montenegro, Ernesto Fabregas, Sebastian Dormido-Canto and Sebastian Dormido, "Reinforcement Learning for Position Control Problem of a Mobile Robot," (Article - IEEE Access, 8, 152941-152951, 2020, doi: 10.1109/ACCESS.2020.301802).

[25] Gonzalo Farias, Gonzalo Garcia, Guelis Montenegro, Ernesto Fabregas, Sebastian Dormido-Canto and Sebastian Dormido, "Position control of a mobile robot using reinforcement learning." (Article- *IFAC-PapersOnLine* 53.2 (2020): 17393-17398.)

[26] Leilei Cui, Shuai Wang, Jingfan Zhang, Dongsheng Zhang, Jie Lai, Yu Zheng, Zhengyou Zhang, Zhong-Ping Jiang, "Learning-Based Balance Control of Wheel-Legged Robots," (Article- IEEE Robotics and Automation Letters, 6, 4,7667-7674, Oct. 2021, doi: 10.1109/LRA.2021.3100269.)