Movement Control of Two Wheels Balancing Robot using Cascaded PID Controller

Derry Pratama*, Eko Henfri Binugroho+, Fernando Ardilla*
*Computer Engineering, +Mechatronic Engineering
Electronic Engineering Polytechnic Institute of Surabaya
Surabaya, Indonesia
Email: derryprimero@gmail.com, nando@pens.ac.id, sragen@pens.ac.id

***Abstract* — Balancing robot which is proposed in this paper is a robot that relies on two wheels in the process of movement. Unlike the other mobile robot which is mechanically stable in its standing position, balancing robot need a balancing control which requires an angle value to be used as tilt feedback. The balancing control will control the robot, so it can maintain its standing position. Beside the balancing control itself, the movement of balancing robot needs its own control in order to control the movement while keeping the robot balanced. Both controllers will be combined since will both of them control the same wheel as the actuator. In this paper we proposed a cascaded PID control algorithm to combine the balancing and movement or distance controller. The movement of the robot is controlled using a distance controller that use rotary encoder sensor to measure its traveled distance. The experiment shows that the robot is able to climb up on 30 degree sloping board. By cascading the distance control to the balancing control, the robot is able to move forward, turning, and reach the desired position by calculating the body's tilt angle.**

Keywords— balancing robot, cascaded PID control, balancing control, distance control

I. INTRODUCTION

Robotic technology to help human's work is growing fast nowadays. There are many researches that develop many kinds of robot with different size, shape, and movement. Starting from stationary robot that does not move its position, to mobile robot that moves freely to any place. One example for stationary robot is arm robot which is commonly used as industrial robot that resembles an inverted human arm mounted on a base [1]. Many mobile robot researches also have been done and implemented in the real application. One of them is a military bomb-defusing robot called DRDO Daksh which is fully automated. It can climb staircases, negotiate steep slopes, and navigate narrow corridors [2]. Not only in the industrial and military applications, robots also have been developed in the other area of application, such as; entertainment [3], exploration, and education [4].

There are many kinds of mobile robot based on its actuator, for example wheeled robot, legged robot, swimming robot, and flying robot [5]. Wheeled robot body is moved by the wheels. Based on the amount of wheel, there is a single wheeled robot called REGIS [6], two wheeled self-balancing robot called nBot [7], or more like Curiosity [8].

Many of the balancing robot developed now days are two wheeled robot model, for example; The Equibot, The Segbot, uBot-5, JOE, and U-turns [9]. The balancing robot principle also has been implemented for transportation application. One of them is Segway PT, a commercial personal transporter product founded by Dean Kamen. It uses the same principle as balancing robot but the movement depends on user's body. To move forward/backward, user needs to lean towards the wanted direction and turning is controlled by the steer [10-11]. But many researchers related to balancing robot topics are still focused on the balance control it self.

This paper focuses on the movement of control balancing robot. Robot is designed to move according to remote input from android device. Three controls will be designed, first is to control the robot balance, the second one distance controller for the forward or backward movement control, and the last one is for maneuver or steering. Both balancing and distance control will be cascaded to make the robot stay balanced while moving This cascaded PID control will correct the angle error from the balancing controller by using the speed error from the distance controller.

II. RELATED WORKS

A. Two Wheels Self-Balancing Robot Principles

According to [12], two wheels self-balancing robot has two wheels to stay balanced and has four degrees of freedom, one is tilt-angular motion and the other three is planar motion. It has nonlinear dynamic since it is driven by two wheels which is located under the robot's body. Unlike the cart and pendulum system, the actuator of two wheels self-balancing robot is directly mounted on the pendulum which is the robot's body itself. The robot needs to continuously rotating both wheels and move towards to the side where the robot's body is falling [12-13]. Therefore, because the motor has been controlled to make the pendulum stay balanced, in order to move the robot for maneuver, another controller is needed.

In two wheel self-balancing robot, the pendulum's tilt will affect its movement at balance state. Forward tilting will wove the robot forward, while backward tilting will make it moves backward [14].

B. Linear Control of Wheeled Inverted Pendulum

Wheeled inverted pendulums are intrinsically nonlinear and their dynamics will be described by nonlinear differential

equations. The presence of nonlinearities in systems complicates analysis. Despite this, for the case of controller design, it is often possible to obtain a linearized model of the system. If the system operates around an operating point, and the signals involved are small signals, a linear model that approximates the nonlinear system in the region of operation can be obtained [12].

As stated by [12] the nonlinear state space equations of the wheeled inverted pendulum described as:

$$\dot{x} = f(x,u) \quad (1)$$

$$u = Kx \quad (2)$$

$$y = Cx \quad (3)$$

Because C in Eq. 3 is already a constant, the linearized state space representation of the system, where A and B are constant matrices, is given by:

$$\dot{x} = Ax + Bu \quad (4)$$

$$y = Cx \quad (5)$$

C. PID Controller

PID is a controller to determine the precision of instrumentation system with feedback characteristics. PID control component consist of three types: Proportional, Integrative, and Derivative. If interpreted in terms of time: P constants depends on the value of the current error, I constant is the accumulation of previous error value, while the D constant is the future prediction error, based on the changes rate [13]. Three of them can be used together or individually depends on the response wanted for a plant.

Based on [14], if n is the discrete time of t then the discrete equation of PID is:

$$u(n) = K_p E(n) + K_i \sum_{k=0}^{n} E(k) + K_d(E(n) - E(n-1)) \quad (6)$$

Where

u : Controller output.
E : $Error = Setpoint - Process\ Value$.
Kp : Proportional constant gain.
Ki : Integral constant gain.
Kd : Derivative constant gain.

Since this research based on wheeled balancing robot, using the linearized state system [12] on Eq. 4, where x_d is the desired value and the x is the actual value, the error value (E) can be defined as

$$E = x_d - x \quad (7)$$

D. Motion Driver

Motion Driver is an embedded software stack of the sensor driver layer that easily configures and leverages many of the features of the InvenSense motion tracking solutions. The motion devices supported are MPU6050 / MPU6500 / MPU9150 / MPU9250. Many of the features of the hardware and the on board Digital Motion Processor (DMP) are encapsulated into modular APIs which can be used and referenced [17].

The DMP has many features which can be dynamically turned off and on at run-time. Individual feature can also be disabled while leaving others running. All DMP data are outputted to the FIFO except for pedometer. The DMP can also be programmed to generate an interrupt via gesture or if data ready. Based on [17], the DMP on MPU-6050 modules used for this research are 3 Axis Low Power Quaternions–gyro only quaternions. This feature will integrate the gyro data at 200Hz while outputting the sensor fusion data to the FIFO at the user requested rate. The 200Hz integration will allow for more accurate sensor fusion data. In MD6, if this feature is enabled, the driver will push the 3-axis quaternion into the MPL library and the MPL will handle the accelerometer and compass integrations.

III. System Design and Methodology

Robot used in this system is a two wheeled mobile robot which was designed to operate in standing position with 17.4cm height and 16.7cm width as depicted in Fig. 1. It is similar with inverted pendulum model, two wheels which is mounted directly to the motor and placed on the right and left side of the body as actuator of the pendulum which is the robot's body itself.

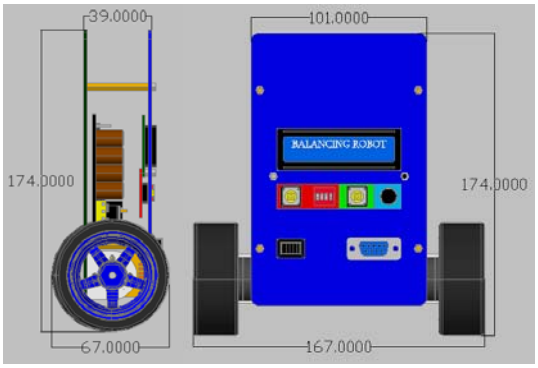

Fig. 1 Robot's mechanical design, left-side and front-side

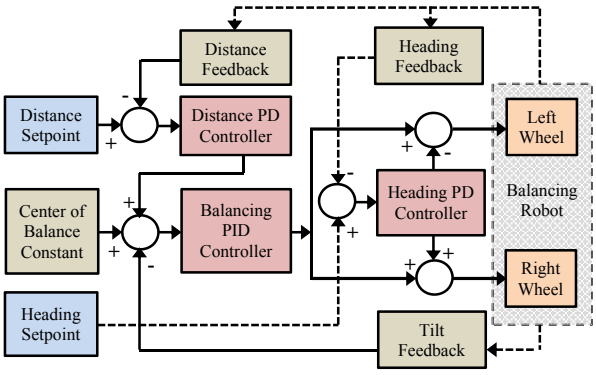

Fig. 2 Overall PID Diagram

Robot is designed to have the center of gravity located in the center when the robot is standing 90 degrees from the

floor to make it easier reaching the balance state. The tilt sensor is placed inside and parallel with the robot's body. For the distance, rotary encoder is mounted directly on the DC motor. From Fig 2, the center of balancing constant of the robot is defined manually as the set-point for balance angle PID control. The error feedback for this controller use a combination of angle error from tilt sensor and the output of distance control converted from rotary encoder pulses into degree to correct the angle set-point.

$$\gamma(n) = \beta(n) + \alpha(n) \quad (8)$$

$$\beta(n) = \left(C_g - \theta(n)\right) \quad (9)$$

Where:

γ : Overall balance error.
β : Angle error of center of balance.
C_g : Center of balance constant.
θ : Actual robot's body angle.
α : Angle offset, an output of distance PD control.

When the robot is tilting forward, the θ becomes negative and when the robot is moving forward and the distance control calculated the output, α will be positive against θ. So when the target distance is increased in the distance set-point, the robot's body will increase the tilt angle forward as long as the distance target not achieved.

A. Steering PD Control

In order to control the robot's maneuver, steering control is needed. Therefore, in this case a PD control is used to control the orientation of the robot. Orientation data is taken from rotary encoder which will count the pulses generated by the wheel. According to [18], robot orientation can be calculated as:

$$x_h = (x_l - x_r) \quad (10)$$

Where x_h is the heading of the robot, and x_l is the left pulses which is subtracted by the right rotary encoder pulses (x_r). This heading value (x_h) will be used as feedback for heading PD control.

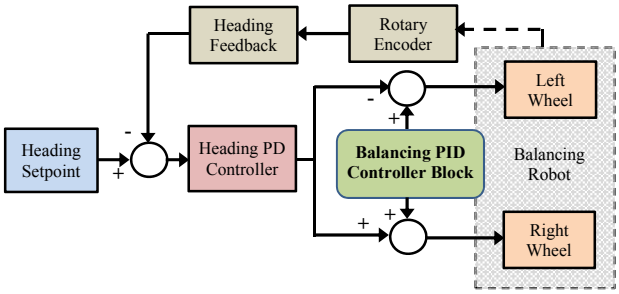

Fig. 3 Heading Control Diagram

As depicted in Fig. 3, heading set-point is a variable to control the desired robot's orientation. The equation can be described as:

$$E_h = x_{hs} - x_h \quad (11)$$

$$v_h(n) = K_{ph}E_h(n) + K_{dh}(E_h(n) - E_h(n-1)) \quad (12)$$

Where:

E_h : Heading error.
x_{hs} : Desired heading.
x_h : Actual heading.
K_{ph} : Proportional constant for heading PD control.
K_{dh} : Derivative constant for heading PD control.

Based on the experiment of trial and error, the best value for proportional constant is 2.0 while the derivative constant is 0.75.

B. Distance PD Control

In this research distance is used as a parameter for movement, forward or backward. Output of this control affect the robot's tilt error by being an input for the angle PID control. According to [18], distance can be calculated as:

$$x_j = \frac{(x_l + x_r)}{2} \quad (13)$$

Where x_j is the distance pulse averaged from two rotary encoder mounted on both right (x_r) and left wheel (x_l).

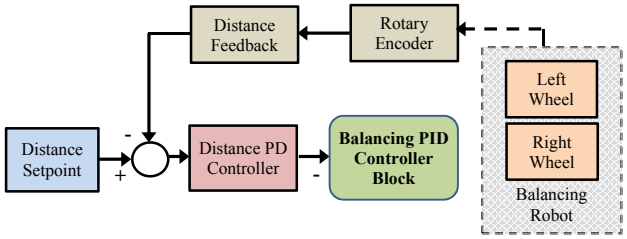

Fig. 4 Distance PD Control Diagram

From Fig 4, it can be seen that the output is used as error negative reference for balancing PID controller block, and not directly control the motor velocity. Distance error and control can be calculated as:

$$E_x = x_{js} - x_j \quad (14)$$

$$\alpha(n) = K_{pj}E_x(n) + K_{dj}(E_x(n) - E_x(n-1)) \quad (15)$$

Where:

E_x : Distance error.
x_{js} : Desired distance / position.
x_j : Actual distance.
K_{pj} : Proportional constant for distance PD control.
K_{dj} : Derivative constant for distance PD control.
α : Angle offset, distance PD control output.

Based on trial and error experiment, best value found for proportional constant is 0.0005 while for the derivative constant is 0.75.

C. Balancing PID Control

This control is used to maintain the robot balance by controlling the robot standing position by using its tilt angle feedback. Because of the angle needs to be corrected while moving, this control input is the sum of both angle error (β)

and angle offset (α) which is the output of distance PD controller. After the α value calculated by distance PD control, it will be summed with angle error, the diagram can be seen at Fig 5.

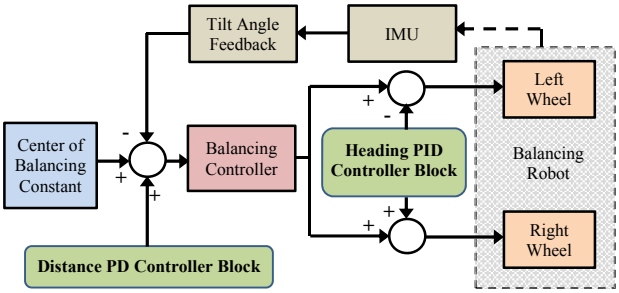

Fig. 5 Balancing PID Control Diagram

Center of gravity is a constant in degree unit that is the balance point of the robot mechanical body. Output of this control is applied to the motor speed. Based on the discrete PID equation, this control can be described as:

$$v_s(n) = u(n) \quad (16)$$

$$v_s(n) = (K_{ps}\gamma(n) + K_{is}\sum_{k=0}^{n}\gamma(k) + K_{ds}(\gamma(n) - \gamma(n-1))) \quad (17)$$

Where:

v_s : Motor speed.
K_{ps} : Desired distance / position.
K_{is} : Actual distance.
K_{ds} : Proportional constant for distance PD control.
γ : Derivative constant for distance PD control.

Based on experiment, the best value for the proportional constant is 175.0, derivative constant is 8.5, while for the integral constant is 5.0.

IV. Result

The PID gain constant of all control in this system remains unchanged in this experiment. There will be a variable for speed input which is the value that has been set to change the distance set-point. The distance is increased by the speed offset value. These experiments are done at flat surface and sloping surface.

A. Standby Balancing Response

The purpose of this experiment is to see how the response of the robot stopped with zero target distance, means that the robot's desired position is in its place. The graph will show the robot's control response in steady state, without any disturbance. From the Fig. 6, it can be seen that the robot should not move from its starting position, so the speed input is totally zero.

In Fig. 7, the standby speed value is close to zero. It because the speed input is actually zero, so the controller will maintain the zero speed. In t = 52, there is a maximum oscillation up to 2 centimeter error, it is caused by the balancing controller that act to control the wheels to balance the robot.

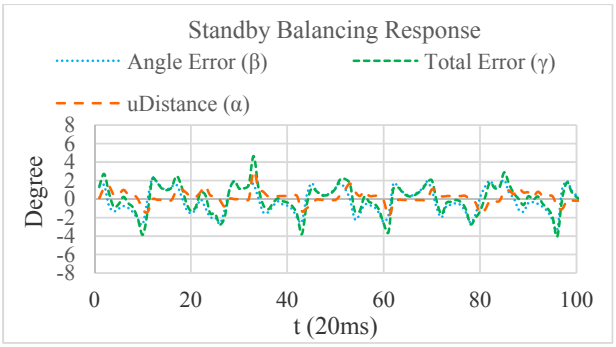

Fig. 6 Standby Balancing Response

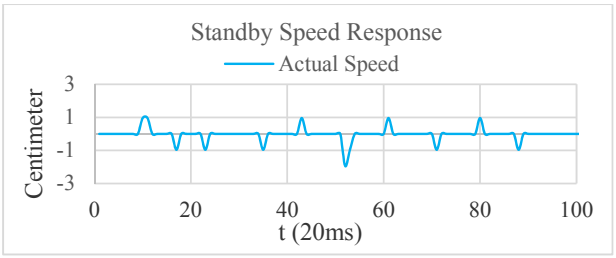

Fig. 7 Standby Balancing Speed Offset Response

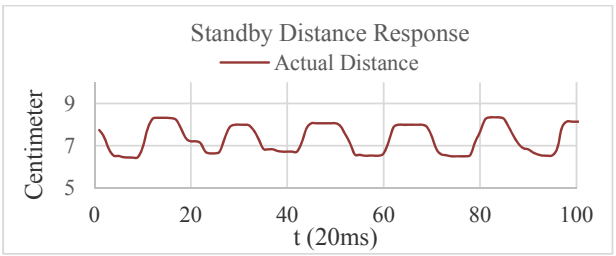

Fig. 8 Standby Distance Response

From Fig. 8, it is shown that even if the robot is controlled to stay at the starting position, the actual distance from the starting point is not actually zero. The shift is happening because of the center of gravity that has been set on angle PID controller is not perfectly correct, so the system corrects it by calculating the distance from starting position so the body's tilt can be corrected. From Fig 6, it can be seen that the value of α which is the output of distance PD control is about 0.23 degree. It is affecting the current tilt set-point, that's why the angle error (β) is controlled to be at 0.47 degree. So, when the robot distance set-point is zero, a small oscillation is happen. It is caused by the the action of balancing control system which uses the wheels as output too. This experiment shows that the error on distance while balancing steadily is about ±1 cm from its steady position.

B. Forward Moving Response

This experiment shows the robot's response when the speed offset is changed. In Fig. 9 it can be seen that at t =

1290, the speed offset is changed so the desired distance is changed. The effect can be clearly seen by looking at the rising of uDistance (α), output of distance control which response right after the speed offset is changed. It is affecting the angle error to make the robot's move to the desired distance.

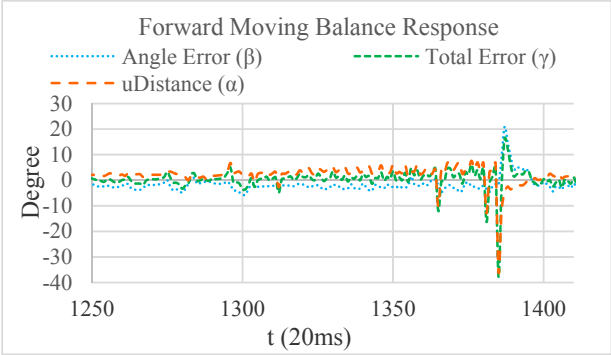

Fig. 9 Forward Moving Balance Response

The constant changes on the peak of α shows that the robot is constantly moving, right after the speed input is changed.

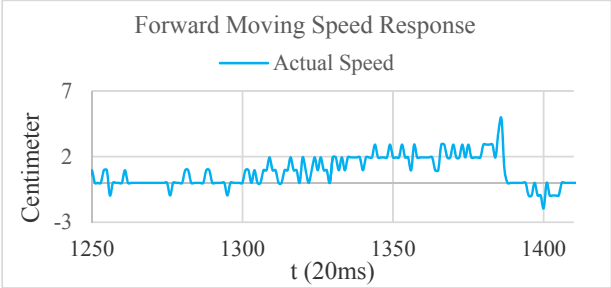

Fig. 10 Forward Moving Speed Response

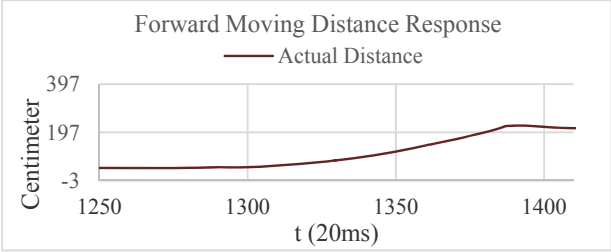

Fig. 11 Forward Moving Distance Response

The movement of the robot can be seen clearly at Fig. 12. It shows the robot's distance from the starting point changing and suddenly stopped at t = 1390.

C. *Steering Response*

This experiment measure the response of the robot's stability while fully turning right without changing the speed. The orientation data at Fig. 15 shows the actual value of orientation measured from the rotary encoder and calculated using Eq. 13.

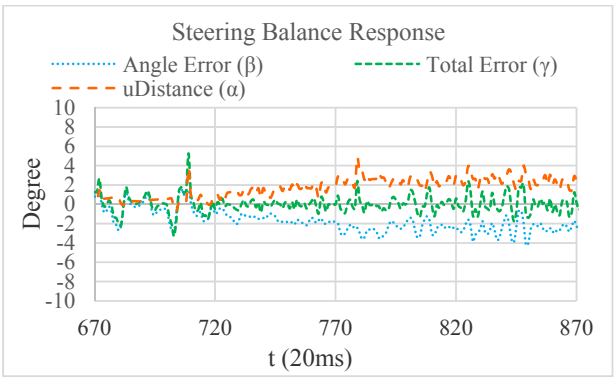

Fig. 12 Full Turn Response

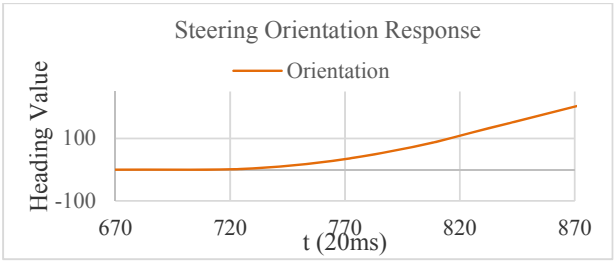

Fig. 13 Steering Orientation Response

From the Fig. 12, at t = 711, the full turn is started and the response of the robot is shown. The robot starts to oscillate bigger up to 3.75 degree. While at the uDistance (α) is shown that the robot changes the center of gravity automatically because its body shifted and caused the output of distance control to change.

In the Fig. 15, the robot started to change its orientation in t = 730. The experiment shows that the orientation change in orientation is almost linear.

D. *Climbing Response*

In this experiment, a sloping board with 30 degrees inclination from the ground is prepared. The robot will be driven to the top of the sloping board and then stops. The goal is to find how the robot responses and how the robot corrects its body inclination while moving on slope surfaces. There are five data taken for this experiment, speed input, distance control output, angle control output and the overall error.

In the graph shown by Fig. 16, it can be seen that the robot starts to change its balance point at t = 335. The angle error shows that it's going far from zero means that the balance point is already changed while the robot trying to climb up the slope.

The actual speed of the robot is shown at Fig. 15 Climb up Speed Response, where the robot speed starts to increase slowly at t = 270, and then finally slowing down at t = 335. The robot starts to move again until t = 400. In that moment, the speed input is suddenly stopped at zero, it causes the balance control oscillated but still can stand in a slope with 30 degree inclination from the ground.

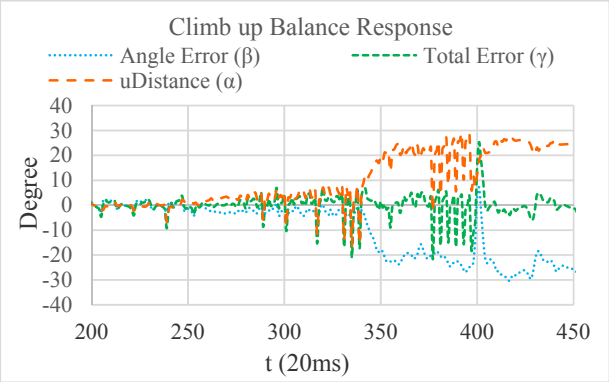

Fig. 14 Climb up Balance Response

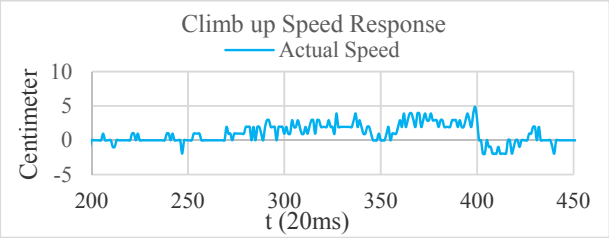

Fig. 15 Climb up Speed Response

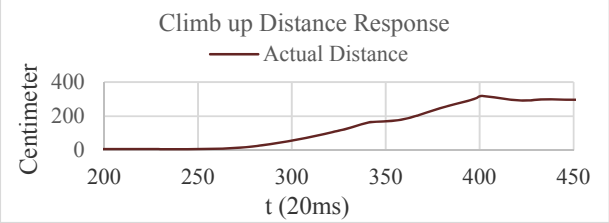

Fig. 16 Climb up Distance Response

At t = 240, the speed input is start to increase, which will cause the robots to increase its distance by the speed offset. At t = 335, the robot starts to climb, as we can see the big difference in β and α in Fig 16, and a small movement of the robot at t = 335 in Fig. 19. The robot keeps climbing then it stops at the top at t = 400.

V. Conclusions

From the experiment, researchers conclude that:

1. System is able to keep stable at starting position with maximum balance error at 3.57 degree.
2. When the speed is changed, the maximum balance error is at 7.59 degree from the set-point.
3. The more inclined the surface to climb, the more tilt is needed from the first set-point to make the robot moves forward.
4. The robot is able to do the full turn moves with the balance error maximum at 2.5 degree.
5. PID control that uses tilt and position error can maintain the robot balance while doing maneuver. It also able to move the robot's position by correcting the body's tilt.

This control system is able to do maneuver and changing robot's position while maintaining the body balance. The distance control, able to move the robot's position even there is an obstacle like sloping surface, and has been successfully tested on 30° inclined surface. The steering control is able to change the orientation without disturbing the balance control with an acceptable heading error.

Acknowledgment

This research is conducted by authors with EEPIS Robotics Research Center (ER2C) and workshop laboratory of Computer Engineering in Electronic Engineering Polytechnic Institute of Surabaya.

References

[1] Schilling, Robert. "Fundamentals of robotics." (2013).

[2] "DRDO Daksh." *Wikipedia*. Wikimedia Foundation. Web. 29 Apr. 2015. <https://en.wikipedia.org/wiki/DRDO_Daksh>.

[3] G. Hornby, S.Takamura,T. Yamamoto, and M. Fujita, "Autonomous Evolution of Dynamics Gaits with Two Quadruped Robots", IEEE transactions on Robotics, Vol.21, No.3, pp.402—410, 2005.

[4] Eko Henfri B., Achmad Subhan KH., Dadet Pramadihanto, Rahardhita W. S., "ADROIT V1: Design of Low-cost Modular Educational Robot Platform", International Electronics Symposium, December, 2014.

[5] "Types of Robots." *All on Robots*. Web. 27 Apr. 2015. <http://www.allonrobots.com/types-of-robots.html>.

[6] "REGIS." Versatile Robot. Web. 2 May 2015. <http://versatilerobot.blog.free.fr/index.php?post/2011/08/18/Profil-et-Projet>.

[7] Anderson, David. "NBot, a Two Wheel Balancing Robot." NBot, a Two Wheel Balancing Robot. Web. 15 Apr. 2015. <http://www.geology.smu.edu/~dpa-www/robo/nbot/>.

[8] "Curiosity (rover)." *Wikipedia*. Wikimedia Foundation. Web. 29 Apr. 2015. <https://en.wikipedia.org/wiki/Curiosity_(rover)>.

[9] Li, Zhijun, Chenguang Yang, and Liping Fan. *Advanced control of wheeled inverted pendulum systems*. Springer Science & Business Media, 2012.

[10] "Segway PT." *Wikipedia*. Wikimedia Foundation. Web. 30 Apr. 2015. <https://en.wikipedia.org/wiki/Segway_PT>.

[11] "Segway - The Leader in Personal, Green Transportation." Segway - The Leader in Personal, Green Transportation. Web. 11 Mar. 2015. <http://www.segway.com/>.

[12] Li, Zhijun, Chenguang Yang, and Liping Fan. Advanced control of wheeled inverted pendulum systems. Springer Science & Business Media, 2012.

[13] Pathak, Kaustubh, Jaume Franch, and Sunil K. Agrawal. "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization." Robotics, IEEE Transactions on 21.3 (2005): 505-513.

[14] Hatakeyama, Naoya, and Akira Shimada. "Movement control using zero dynamics of two-wheeled inverted pendulum robot." Advanced Motion Control, 2008. AMC'08. 10th IEEE International Workshop on. IEEE, 2008.

[15] Araki, Mituhiko. *"PID control."* Control systems, robotics and automation 2 (2002): 1-23.

[16] Atmel, AVR221: Discrete PID controller, 2006.

[17] InvenSense, Motion Driver 6.0 – Features Guide, 2014.

[18] Sandi Marta, Bayu. "Path Tracking pada Mobile Robot dengan Umpan Balik Odometry". Program Studi Teknik Komputer, Politeknik Elektronika Negeri Surabaya, Surabaya. 2011.