Tilt Set-point Correction System for Balancing Robot Using PID Controller

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

***Abstract*—Balancing robot is a robot that relies on two wheels in the process of movement. Basically, to be able to remain standing balanced, the control requires an angle value to be used as tilt set-point. That angle value is a balance point of the robot itself which is the robot's center of gravity. Generally, to find the correct balance point, requires manual measurement or through trial and error, depends on the robot's mechanical design. However, when the robot is at balance state and its balance point changes because of the mechanical moving parts or bringing a payload, the robot will move towards the heaviest side and then fall. In this research, a cascade PID control system is developed for balancing robot to keep it balanced without changing the set-point even if the balance point changes. Two parameter is used as feedback for error variable, angle and distance error. When the robot is about to fall, distance taken from the starting position will be calculated and used to correct angle error so that the robot will still balance without changing the set-point but manipulating the control's error value. Based on the research that has been done, payload that can be brought by the robot is up to 350 grams.**

Keywords—balancing robot, PID control, center of gravity correction, rotary encoder, IMU

I. INTRODUCTION

Advance in our technology today has grown rapidly. Robotic field for example, researchers has been developing various types of robot to do human's work. Robots can be easily do the human's work automatically with their different forms and actuators. Starting from wheeled robots to humanoid, robots have been programmed to do their job.

In order to move the robot's actuator, a control algorithm is needed. Therefore, control theory is used as base to control a robot or system. While in theory of general feedback, it is mentioned that the more feedback is used, the better the control [1]. So that a control system that has a lot of input for correction would be better than a system that only uses one input.

For example on a balancing robot, which is like a two-wheeled mobile robot. The interesting part is a problem on its balancing control that makes it a popular research topic because of its mechanical design that resembles an inverted pendulum, very unstable to be controller [2] – [7]. This is a challenge for researchers, to design a control in order to make the robot standing balanced. A lot research has been discussing about balancing robot control, such as Linear Quadratic Regulator (LQR), Pole-Placement Controller, Fuzzy Logic Controller (FLC), and PID Controller [8]. Most of these controls are using input from tilt angle. While the input of wheel speed is only used to adjust the steering and maneuver.

This research focused to make the balancing robot able to bring an unknown payload weight. The control system needs to correct the balance point. So that when the robot is at the balance state, and its center of gravity changes because of payload taken, the robot will still balance and correct the tilt angle that needs to be maintained.

II. RELATED WORKS

A. Two Wheels Self-Balancing Robot

Two wheels self-balancing robot has two driving wheels while it has four degrees of freedom, i.e., three of planar motions and one of tilt-angular motion. In addition, its dynamic are also nonlinear such that the two wheels self-balancing robot is an under-actuated nonlinear system [9].

This system is different from the cart and pendulum system, because this system is driven by two wheels and its motion is not planar, and in addition, the motors driving the wheels are directly mounted on the pendulum body, while the actuator does not directly drive the pendulum in the cart and pendulum systems [9]-[10]. Therefore, two wheels self-balancing robot is continuously rotating both wheels to the same direction as the direction where the robot's body is falling until the robot reaches a certain tilt which has been set as set-point tilt.

There are many research done on this system, for example The Equibot, The Segbot, uBot-5, nBot, JOE, and U-turns [9] which all of them focused on the self-balancing using many types of control method. But there is a balancing robot made by WowWee [11], a commercial product which has many features, such as maneuver, and including payload balancing. It is able to bring a payload with the same weight as its weight.

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

978-1-4799-8975-1/15/$31.00 ©2015 IEEE

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 [9].

As stated by [9] 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 (Proportional-Integral-Derivative controller) 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 [12]. Three of them can be used together or individually depends on the response wanted for a plant.

Based on [13], 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 [9] 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 [14].

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 is 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 [14], useful features of the DMP on MPU-6050 module for this research is 3 Axis Low Power Quaternions – gyro only quaternions. This feature when enabled 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 accel 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 31.5cm height and 21.5cm width when the gripper is open. In standing position which make it more similar like inverted pendulum model, two wheels which is connected directly to the motor was placed in right and left side of the robot's lower body as actuator, while the gripper is on top of robot's body to hold a load.

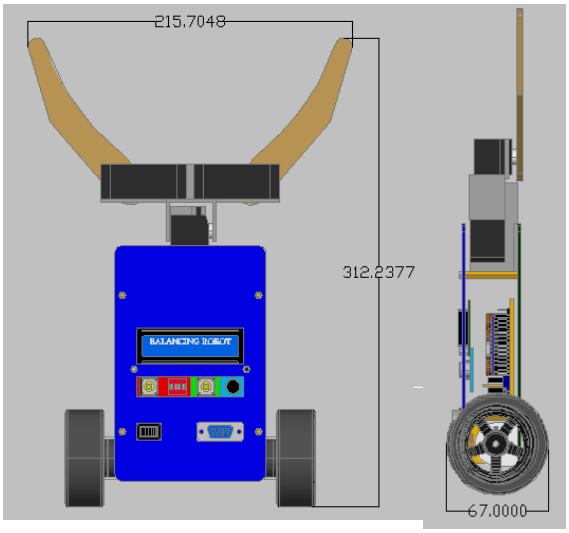

Fig. 1. Robot's mechanical design, front-side and left-side

This mechanical designed to have a center of gravity located in center part of overall robot's body to make it easier to reach the balance state. Rotary encoder is connected directly to the motor while the tilt sensor placed at the center backside of the robot's body.

A. Payload Balancing PID Control Method

In order to make the robot stand balanced while carrying a load, error calculation in PID control divided in two. So there will be tilt error and distance error calculations.

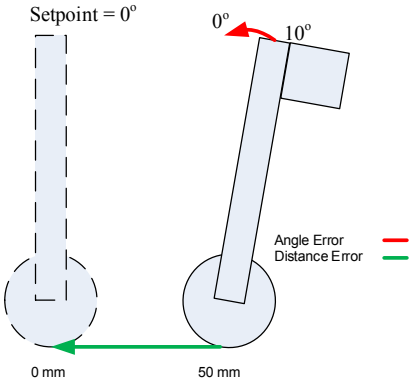

Fig. 2. Tilt and distance error

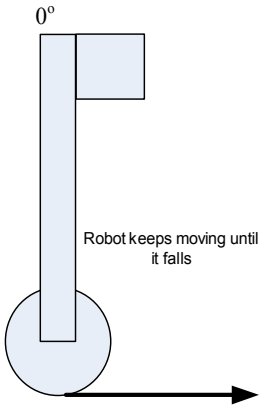

Fig. 3. Using tilt error only

Just like balancing robot in general, the value of tilt error obtained from the difference between the value of current robot angle and the set-point angle, while the value of distance error is calculated from the difference between current robot mileage and set-point mileage.

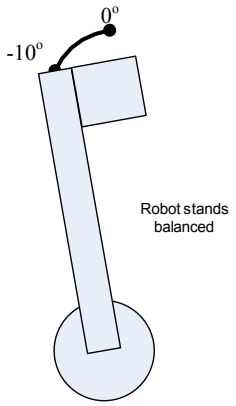

Fig. 4. Using tilt and distance error

Mileage itself is calculated from rotary encoder pulse which is then divided by two. Mileage is a distance of robot displacement from starting position. Summation of both error value will generate another error value that has a dynamic zero reference.

PID control function that will be used in this robot has a mileage parameter and a tilt parameter. Mileage parameter will go through the PD control calculation which then output of PD control of the mileage parameter will be used as additional feedback to calculate the error in robot's tilt control.

Center of gravity of this robot's mechanical design is in center from the front-side and back-side, so the balance point for this robot without carrying a load is near 0° from the floor perpendicular line. This tilt angle will be the constant for tilt control set-point, while the robot's mileage parameter calculated from rotary encoder will correct this constant continuously by PD control.

In this case, for mileage parameter set-point for PD control is 0 because when the robot needs to stand still at the starting position and getting a disturbance, it will try to go back to its first position by leaning toward its starting position. That will keep the robot balanced when there is a continuous disturbance like when carrying a load which changes its center of gravity.

In basic control for balancing which only uses tilt error as parameter, when the tilt set-point that has been set is not the same with the real tilt balance point, robot will keep moving towards to its real center of gravity until it falls. This is because when the robot receive a load, its center of gravity changed and the control calculation became wrong because the tilt it maintains is no longer the right tilt to make it balanced. Just before the robot falls because of carrying load, there is a mileage parameter that changes. Researchers try to use that parameter as correction for tilt based control.

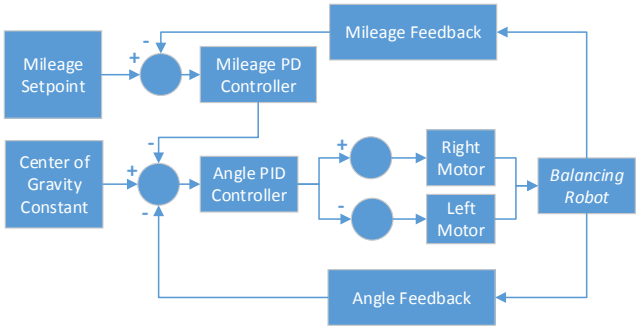

Fig. 5. Cascaded PID Diagram

Mileage will be used as correction for tilt based PID control, so that when the robot carries a load mileage will keep increasing and this value will be an input for PD control with 0 set-point and the output will be an input for error in tilt PID control. It will make the robot leaning toward the opposite direction of carried load.

According to Fig. 5, and Eq. 7 here is the formula for error calculation for tilt PID control:

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

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

Where

γ : angle error after summed with offset.
β : angle error from robot's center of gravity.
C_g : robot's center of gravity constant.
θ : current robot's angle read by the sensor.
α : mileage PD control output in degree unit.

When the robot leans forward, the value of θ became negative, when the robot moves forward, α became positive, opposing the value of θ. So that when the robot receives the load disturbance and its center of gravity changes to a new one, the farther robot moves to the new center of gravity direction, the more it leans to the starting location direction which is opposing its moving direction. This causing the system to found a new balance point with certain tilt and certain distance from starting location where the value of $\beta(n)$ and $\alpha(n)$ has the same but opposing value. With this method, robot can seek a new balance point but also requires the robot to move from starting location to calculate distance in order to maintain new tilt.

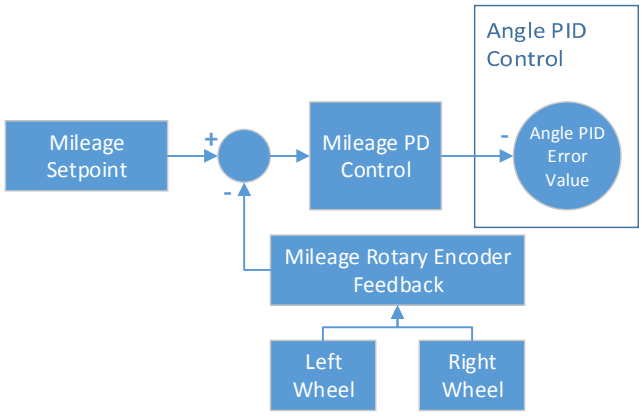

Fig. 6. Mileage PD Controller Diagram

Since the input of mileage PD control is from the pulse output of rotary encoder of both right and left wheel, the mileage (x_j) taken by the robot can be calculated as:

$$x_j = \frac{(x_l + x_r)}{2} \quad (10)$$

Where x_l is the left wheel's mileage, and x_r is the right wheel's mileage. x_j is the average of both rotary pulse value. Because of $\alpha(n)$ used in Eq. 8 is the output of PD controller using input from mileage error $\alpha(n)$ can be described as:

$$E_x = x_{js} - x_j \quad (11)$$

$$\alpha(n) = K_{pj}E_x(n) + K_{dj}(E_x(n) - E_x(n-1)) \quad (12)$$

Where

E_x : mileage error as input for mileage PD controller.
x_{js} : mileage set-point.
x_j : current robot's mileage read by rotary encoder.
K_{pj} : proportional constant for mileage PD control.
K_{dj} : derivative constant for mileage PD control.
α : angle offset, mileage PD control output.

In this case, x_{js} is zero because the starting position is the position that needs to be maintained overtime. In this case, best K_{pj} and K_{dj} value found by trial and error experiment was $K_{pj} = 0.00050$ and $K_{dj} = 0.7500$.

After $\alpha(n)$ value is obtained, then $\gamma(n)$ can be calculated to process the next angle PID control. The detailed angle PID controller diagram can be seen on Fig. 7.

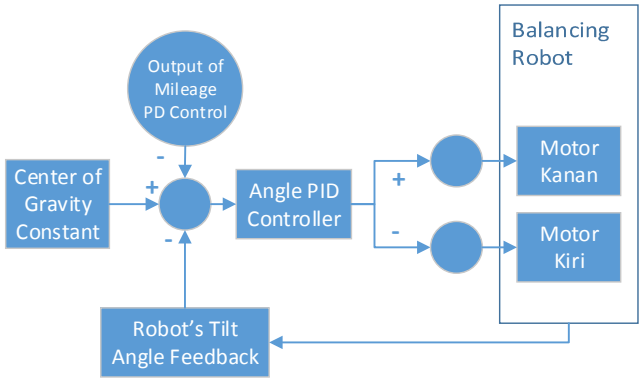

Fig. 7. Angle PID Controller Diagram

Based on discrete PID equation on Eq. 6, and error input using γ in Eq. 8, the equation for motor speed control can be described as:

$$v_s(n) = u(n) \quad (13)$$

$$v_s(n) = (K_{ps}\gamma(n) + K_{is}\sum_{k=0}^{n}\gamma(k) + K_{ds}(\gamma(n) - \gamma(n-1))) \quad (14)$$

Where

v_s : motor speed, output of angle PID controller.
K_{ps} : proportional constant for angle PID controller.
K_{is} : integral constant for PID controller.
K_{ds} : derivative constant for PID controller.
γ : angle error summed with offset correction.

This second control using error variable based on tilt angle read by the sensor summed with the output of previous mileage PD control. Based on trial and error experiment, it was found that the best constant for $K_{ps} = 175.0$, $K_{is} = 5.0$, and $K_{ds} = 8.5$ which is able to control the robot balance and position.

IV. Result

In this experiment, PID control is used as load balancing system will be tested with some loads which are different in weights from 0g to 350g. The robot itself is weighted 600g. This means robot must carry a load up to 58.3% of its own weight. This experiment goal is to find out responsiveness and stability of the system in balancing the robot while carrying a load and how much weight can it take.

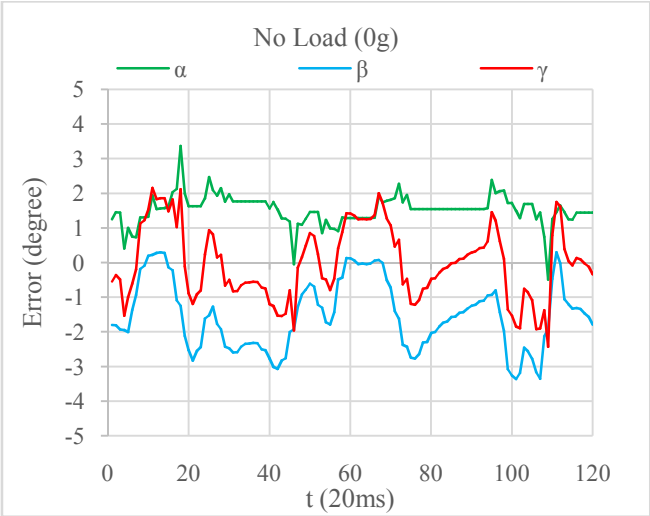

Fig. 8. Error in degree while not carrying any load

In this stage, system will keep the robot balance at set-point angle, but since the additional mechanic part is added as load basket, instead of zero, the set-point is about -0.5 degree, that's why the total error (γ) stays at 0 and -1 degree. In this data, the displacement offset angle (α) of robot from starting location can be seen, it is not far from zero since the changes of center of gravity is very small.

Distance offset angle error value (α) near zero means the robot is not moving from the starting location. Angle error value (β) near zero means the robot's tilt is perpendicular to the floor. Total error value (γ) near zero means the robot is balanced.

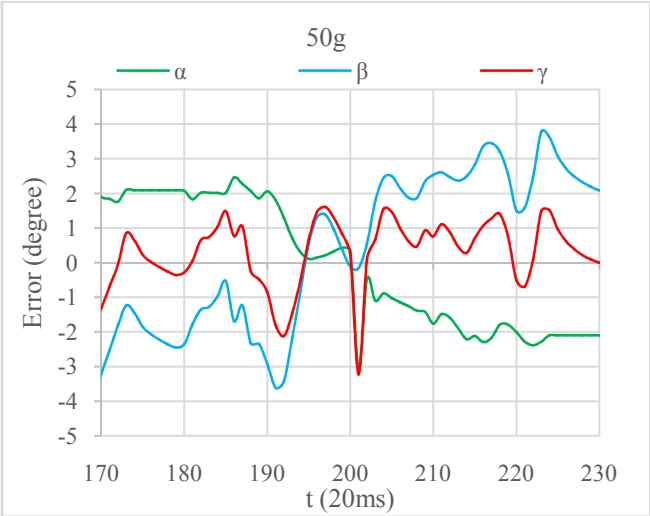

Fig. 9. Error in degree while carrying 50g load

Experiment continues with 50g load added to the robot while was in steady state. Now we can see the changes of distance error (α) which now in degree because it's an output from PD controller. As the distance error (α) increase, the angle error β is following it with the opposite value causing the total error near zero.

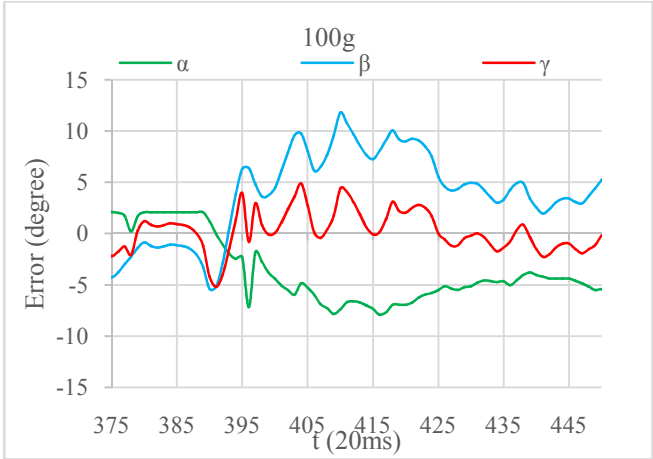

Fig. 10. Error in degree while carrying 100g load

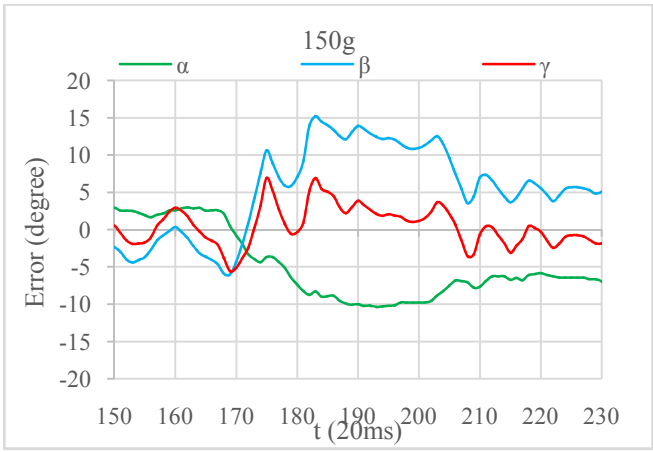

Fig. 11. Error in degree while carrying 150g load

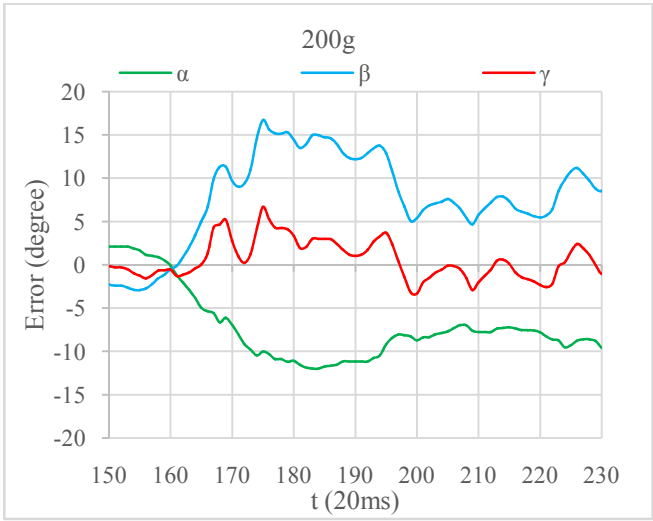

Fig. 12. Error in degree while carrying 200g load

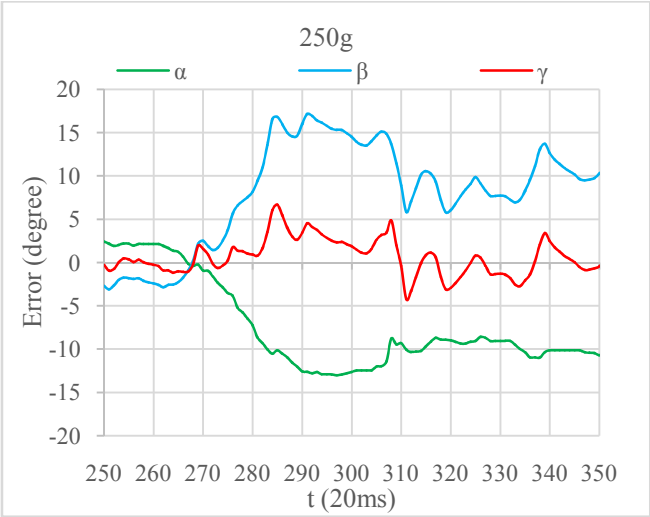

Fig. 13. Error in degree while carrying 250g load

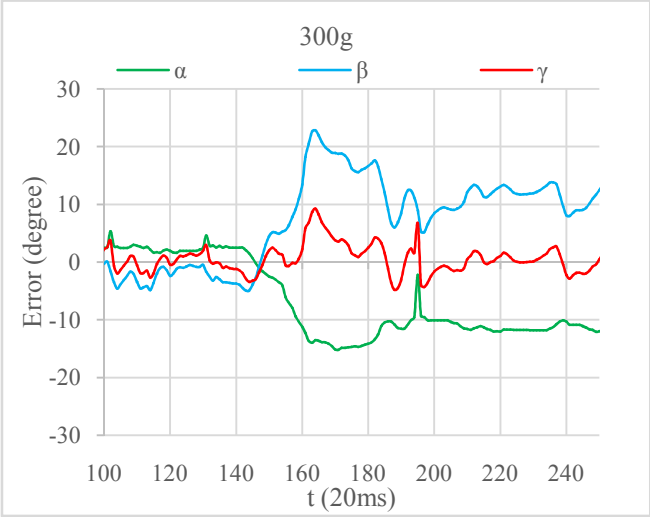

Fig. 14. Error in degree while carrying 300g load

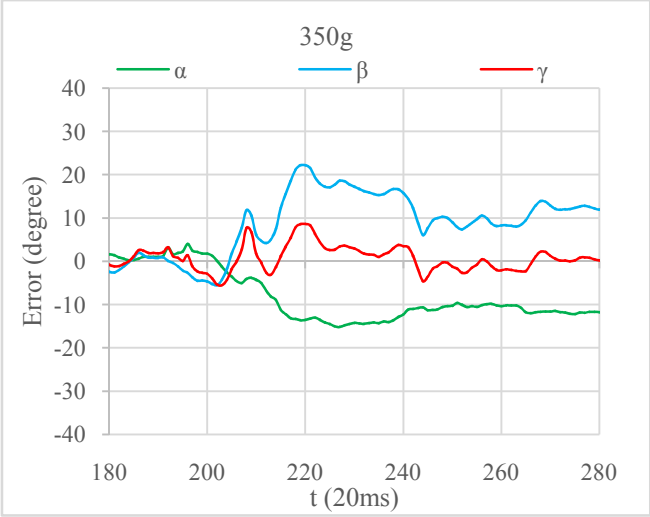

Fig. 15. Error in degree while carrying 350g load

At 100g load, angle error (β) reaches about 8 degrees from original center of gravity because the distance it takes is getting farther to get the correct balance point. System still can balance with 100g load with similar steady response as it 50g. No big difference at all for different payload's weight except the tilt of the robot. Angle error (β) is normal and it happens because the robot leans toward to the starting location to match the load weight and seek the correct balance point. Given load weighted at 50% of the robot's weight, the system can still balance at normal total error rate.

The last experiment shows that the robot can still stand balanced while carrying load more that 50% of its own weight with tolerable error rate. It can be seen that when the weight carried by the robot increase, the distance it takes to balance is also increasing until the total error is equal zero.

V. Conclusions

From the experiment, researchers conclude that:

1. System is able to bring a payload up to 350 grams.
2. Controllable tilt angle is in range of 20° to -20° from the balance point.
3. The more weight of payload that brought by the robot, the farther the distance taken by the system to correct the tilt set-point.
4. Average time taken by the system to get back to steady state after the system receive the payload is 1.13 second.
5. PID control that uses tilt and position error can maintain the robot balance even when its center of gravity changed. Offset value of position error will cause the system change the angle set-point that need to maintain so that when there is a load in backside of robot, it will compensate it by leaning to the front-side.

This control system is able to maintain the balance and correct the tilt angle set-point while bringing a payload with unknown weight. This could solve the problem for the linearized error value at Eq. 4, where the desired value for balancing robot is a center of gravity constant, can be modified into Eq. 8 to find the new center of gravity constant.

Hopefully, this kind of control system can be applied to research with same inverted pendulum principle, i.e. humanoid robot soccer which has an actuator / hand that can affect / change the robot's center of gravity when it's moving, or two wheeled self-balancing robo-waiter which can bring some food order on its two wheel and deliver it to the customer.

Acknowledgment

This research will not work without the helps from some peoples at Computer Engineering Study Program in Electronic Engineering Polytechnic Institute of Surabaya where this research has been held. I would like to give thanks to my advisor, Fernando Ardilla and Eko Henfri

Binugroho whose been helping me with so much helpful advices on this research. Also thank you very much for my friends who helped me write the paper on this research.

REFERENCES

[1] Antunes, Ricardo, and Vicente Gonzalez. "A Production Model for Construction: A Theoretical Framework." Buildings 5.1 (2015): 209-228.

[2] An, Wei, and Yangmin Li. "Simulation and Control of a Two-wheeled Self-balancing Robot." Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on. IEEE, 2013.

[3] Wen, Yan-Hou, Yu-Sheng Lin, and Yih-Guang Leu. "Design and implementation of the balance of two-wheeled robots." Advanced Robotics and Intelligent Systems (ARIS), 2013 International Conference on. IEEE, 2013.

[4] Zhao, JianWei. "The control and design of Dual-wheel upright self-balance Robot." Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on. IEEE, 2008.

[5] Xiaohui, Yang, et al. "Control experiment of the inverted pendulum using adaptive neural-fuzzy controller." Electrical and Control Engineering (ICECE), 2010 International Conference on. IEEE, 2010.

[6] Choi, Dongil, and Jun-Ho Oh. "Human-friendly motion control of a wheeled inverted pendulum by reduced-order disturbance observer." Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE, 2008.

[7] Nour, M. I. H., J. Ooi, and K. Y. Chan. "Fuzzy logic control vs. conventional PID control of an inverted pendulum robot." Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference on. IEEE, 2007.

[8] Chee, Ong Yin, and M. S. B. Abidin. "Design and development of two wheeled autonomous balancing robot." Research and Development, 2006. SCOReD 2006. 4th Student Conference on. IEEE, 2006.

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

[10] 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.

[11] WowWee. (2015). *MIP Balancing Robot*. Accessed Juny 20, 2014, from http://www.wowwee.com/mip.

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

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

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