Self-balancing Robot Control Using Fractional-Order PID Controller

K. Kankhunthod, V. Kongratana, A. Numsomran and V. Tipsuwanporn, *Member, IAENG*

***Abstract*—This paper presents a two-wheeled robot control that can balance upright on its own by controlling the angular speed of the motor to keep the robot upright using measured data from the gyroscope sensor. The aim of this study is to demonstrate the design of fractional-order PID controller (FOPID) that has more controllability and the ability to adjust outperform traditional PID controller. The design of optimal FOPID controller based on integer-order PID parameters are explained and validated its performance compared with the traditional PID controller using Matlab simulation. As well as the real system experiment is implemented on Raspberry Pi using IIR filters cascaded second-order section form II. The study revealed the appropriate concept of implementation of FOPID on the real system.**

***Index Terms*—two-wheels self-balancing robot, raspberry pi, control system, fractional-order PID controller, digital IIR filter**

I. Introduction

SELF-balancing robot is standing on two-wheels and keeps itself balance without falling as well-known small personal transporter called "Segway". Self-balancing robot based on an inverted pendulum theory is a popular research topic in several years. An inverted pendulum is a challenge control problem because the system is non-linear and completely unstable [1]. To control the robot upright, the external force form motor is needed to compensate for the angular displacement of the robot. The most important part to stabilize the robot's system is a controller. The most widely used and simple controller is a PID controller (proportional-integral-derivative controller) which appropriates for improving both transient and steady-state responses. However, the PID controller is less effective with high order and high external noise system [2]. This paper introduces fractional order *PID* controller (FOPID) $PI^{\lambda}D^{\mu}$ controller based on fractional calculus. The non-integer operators λ and μ are the order of integral and derivative parts respectively, therefore FOPID have more adjustable parameters than traditional PID so that it has higher performance to control high order and delay time system, especially the performance of the non-linear control system overcome traditional PID controller [3]. FOPID will be deeply described in section II. There are more author several approaches about a self-balancing robot follow literature reviews [4-8] In section II, we will discuss theories related to this paper include of fractional calculus, fractional order $PI^{\lambda}D^{\mu}$ controller, digital IIR filter, and Kalman filter that necessary to eliminate the measurement error from the tilt sensor. Section III discusses in mechanical structure, mathematical model, and state-space of the robot. Section IV demonstrates PID and FOPID controller design and their simulation results. Section V demonstrates to realization implemented both controllers on the real system and result of PID controller on the real system.

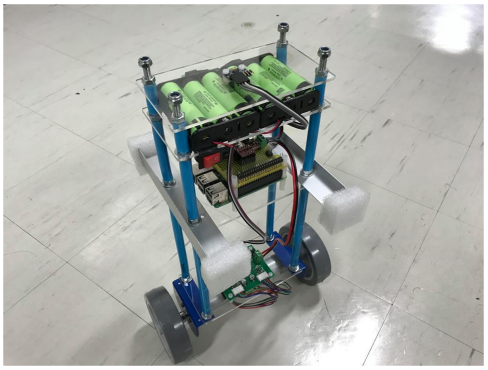

Fig. 1. Self-balancing robot.

K. Kankhunthod is a master of engineering student with Instrumentation Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

Assoc. Prof. V. Kongratana is with the Department of Instrumentation and Control Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

Assoc. Prof. Dr. A. Numsomran is with the department of Instrumentation and Control Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

Assoc. Prof. Dr. V. Tipsuwanporn is with the Department of Instrumentation and Control Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. (e-mail: vittaya.ti@kmitl.ac.th).

II. Theories Related To The Self-Balancing Robot

A. Fractional Calculus

As far as we know that the conventional calculus has integer orders for both derivative and integral operations while the fractional-order calculus has non-integer order operations ${}_aD_t^{\alpha}$ in (1).

$$ {}_aD_t^{\alpha} = \begin{cases} \frac{d^{\alpha}}{dt^{\alpha}} & \Re(\alpha) > 0, \\ 1 & \Re(\alpha) = 0, \\ \int_a^t (dt)^{-\alpha} & \Re(\alpha) < 0, \end{cases} \tag{1} $$

where a, t denote the limits of the operation and α denotes the fractional order. The fractional differ-integral has multi-definitions [9] as follows:

1) Riemann-Liouville definition [10]:

$$ {}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(m-\alpha)} \left(\frac{d}{dt}\right)^m \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-m+1}} d\tau \tag{2} $$

For $m-1 < \alpha < m, m \in N$ where $\Gamma(\cdot)$ is Euler's gamma Function.

2) Grünwald-Letnikov definition:

$$ D_t^{\alpha} f(t) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\left[\frac{t-a}{h}\right]} (-1)^j \binom{\alpha}{j} f(t-jh) \tag{3} $$

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

In equation (3) part inside [] are integer. The Laplace transform of α^{th} order derivative of a signal $x(t)$ with $\alpha \in R_+$ (assuming zero initial conditions) is given by (4).

$$\mathcal{L}\{D^{\alpha}x(t)\} = s^{\alpha}X(s). \tag{4}$$

B. Fractional-order $PI^{\lambda}D^{\mu}$ controller

Fractional PID controller or FOPID is well known as $PI^{\lambda}D^{\mu}$ controller based on the fractional calculus, λ and μ are non-integer order of integral and derivative part respectively, therefor FOPID has total five parameters (K_p, K_i, K_d, λ and μ) cause ability to control the dynamic system precisely. In (5), $u(t)$ is FOPID control input in time domain and $G_c(s)$ denotes the fractional-order PID controller transfer function.

$$u(t) = K_p e(t) + K_i D^{-\lambda} e(t) + K_d D^{\mu} e(t) \tag{5}$$

$$G_c(s) = K_p + \frac{K_i}{s^{\lambda}} + K_d s^{\mu} \tag{6}$$

When taking $\lambda = \mu = 1$ result is the traditional PID controller.

C. Kalman Filter

Kalman filter introduced by Rudolph E. Kalman in 1960 is the best well know filter theory [11] and widely used [12]. Kalman filter is kind of pure time domain filter which differs from most filter like a low-pass filter that is a frequency domain designed [13]. Kalman filter can eliminate noise and recover the real data by comparing error covariance between previous (7) and current states (8) [11]. Kalman filter time update equation:

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{7}$$

Kalman filter measurement update equation:

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ P_k &= (I - K_k H)P_k^- \end{aligned} \tag{8}$$

Where A, B are parameters of state, Q, R denote error covariance of process and measurement respectively, P denotes an estimated error covariance, K is Kalman filter gain, H is predicted measurement, more information about Kalman filter in [12] and about self-balancing robot with Kalman filter are in [14, 15].

D. Digital IIR Filters

Infinite impulse response (IIR) filter contain feedback paths that can keep infinite impulse response [16]. IIR filter can be model in transfer function form, $H(z)$ that consists of poles (b_i) and zeros (a_i), as the following Equation.

$$H(z) = k\frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}} \tag{9}$$

IIR cascaded second-order section form II transfer function defined as follows.

$$H(z) = k \times \prod_{n=0}^{N-1} \frac{b_n[0] + b_n[1]z^{-1} + b_n[2]z^{-2}}{1 + a_n[1]z^{-1} + a_n[2]z^{-2}} \tag{10}$$

Where N is the number of sections, b_n is the set of forward coefficients, and a_n is the set of reverse coefficients. To implement the PID controller on the microcontroller, the transfer function of controller is replaced by the IIR cascaded second-order section form II filter because it has less delay term than IIR filters described in (10) which will be discussed in the implementation section.

III. Modeling for Self-Balancing Robot and State-Space Representation

A. Mechanical Struction

The Structure of the self-balancing robot is shown in Fig.1. The chassis of the robot made by 4 threaded rods, layered with 3 mm acrylic plates for the equipment installed. Raspberry Pi 3 model B is used as the main controller because it's tiny high-performance computer with 1.4 GHz, 64-bit quad-core processor [8], low power consumption with an ability to wirelessly access from another device over the same network that allows monitoring parameters while the robot is operating. The angular position and angular acceleration obtained from MPU-9150 Gyroscope and also accelerometer etc. It can communicate via I2C protocol. EMG30 gear motor with built-in encoder [17] and MD-25 drive board [18] designed for EMG30 was used as main drive system that also communicates via I2C. as shown by Fig.2. Gonçalves et. al. [19] have modeled EMG30 gear motor both mechanical and electrical and simulation resulted in estimated parameters of this motor that have been using in the simulation section of this paper.

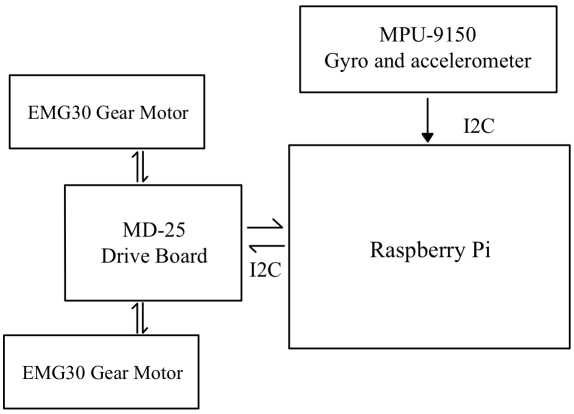

Fig. 2. Structure of Self-Balancing Robot.

B. Mathematical model of self-balancing robot

The mathematical model of robot is separated into 3 parts as follows.

1) DC motor's model: The circuit of a DC motor show in Fig.3

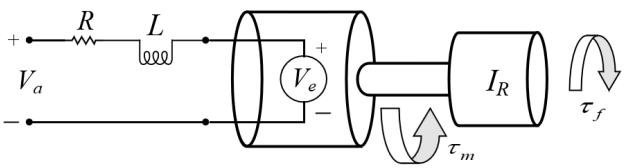

Fig. 3. Circuit of a DC motor.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Equation.11 represent Kirchhoff's voltage law of DC motor where V_a is voltage applied to DC motor, R and L are equivalent resistance and inductance respectively, i is motor current, V_e is back emf voltage.

$$V_a = Ri + L\frac{\mathrm{d}i}{\mathrm{d}t} + V_e \tag{11}$$

Back emf voltage e V is a linear function of shaft angular velocity , ω as show in (12).

$$V_e = k_e\omega \tag{12}$$

Torque produced from a DC motor and total torque are represented in (13) and (14) respectively.

$$\tau_m = i_{total}k_m \tag{13}$$

$$\tau_{total} = \tau_m - \tau_f \tag{14}$$

Where τ_m is no load torque produced from motor, τ_f is friction torque, τ_{total} is total current of a motor. Then combine (13) and (14) we get the relation between total torque and total current show in (15).

$$\tau_{total} = i_{total}k_m - \tau_f \tag{15}$$

$$\tau_{total} = \frac{(V_a - V_e)}{R}k_m - \tau_f \tag{16}$$

Substitute (12) into (16) the governing equation of DC motor obtained (17). I_R is moment of inertia of a wheel.

$$\dot{\omega} = \frac{k_m V_a}{I_r R} - \frac{k_m k_e \omega}{I_r R} - \frac{\tau_f}{I_r} \tag{17}$$

Equation (18) and (19) are state and output equations of state-space model

$$\begin{bmatrix}\dot{\theta}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & \frac{-k_m k_e}{I_W R}\end{bmatrix}\begin{bmatrix}\theta\\ \omega\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{k_m}{I_W R} & -\frac{1}{I_W}\end{bmatrix}\begin{bmatrix}V_a\\ \tau_f\end{bmatrix} \tag{18}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \omega\end{bmatrix} + \begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}V_a\\ \tau_a\end{bmatrix} \tag{19}$$

2) Robot's Model: As mentioned previously, self-balancing robot's model is also called two-wheels inverted pendulum that has similar behavior to the inverted pendulum on cart's model [20]. The cart's model was replaced with two wheels described as follows.

Consider each left and right wheel applied Newton's second law and the relation between torque, force, and radius the equations of left and right wheel obtained represent in (20) and (21).

$$M_W\ddot{x} = \frac{k_m}{Rr}V_a - \frac{k_m k_e}{Rr^2}\dot{x} - \frac{I_W}{r^2}\ddot{x} - H_L \tag{20}$$

$$M_W\ddot{x} = \frac{k_m}{Rr}V_a - \frac{k_m k_e}{Rr^2}\dot{x} - \frac{I_W}{r^2}\ddot{x} - H_R \tag{21}$$

Combining both (20) and (21) we have:

$$2\left(M_W + \frac{I_W}{r^2}\right)\ddot{x} = \frac{2k_m}{Rr}V_a - \frac{2k_m k_e}{Rr^2}\dot{x} - (H_L + H_R) \tag{22}$$

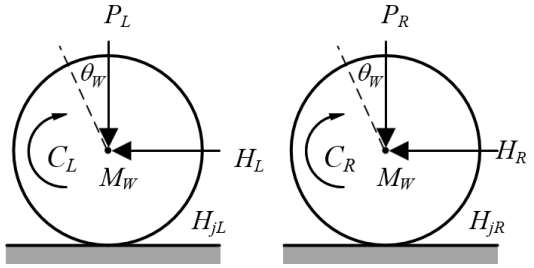

Fig. 4. Free body diagram of robot's wheels.

3) Chassis's model: According to part of inverted pendulum model in the free body diagram of chassis shown in Fig.5

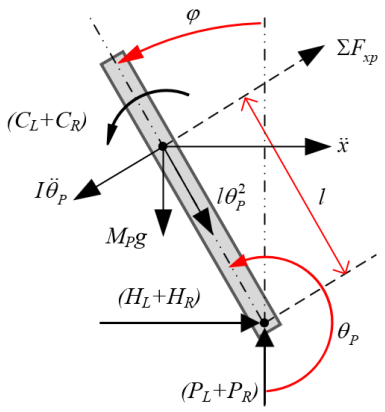

Fig. 5. Free body diagram of robot's chassis.

The Equations of robot's chassis calculated and integrated with the DC motor's model (for more information see also [20], [21]) shown as follows.

$$(I_P + l^2M_P)\ddot{\theta}_P - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a + M_P g l\sin\theta_P = -M_P l\ddot{x}\cos\theta_P \tag{23}$$

$$\frac{2k_m}{Rr}V_a = (2M_W + \frac{2I_W}{r^2} + M_P)\ddot{x} + \frac{2k_m k_e}{Rr}\dot{x} + M_P l\ddot{\theta}_P\ cos\theta_P - M_P l\theta_P^2\cos\theta_P \tag{24}$$

Linearize system with vertical upward equilibrium position condition, $\theta = \pi$, if φ denotes the deviation of the pendulum's position from equilibrium (assume as a small deviation), that is, $\theta_P = \pi + \varphi$ then we get the following conditions.

$$\cos\theta_P = \cos(\pi + \varphi) \approx -1 \tag{25}$$

$$\sin\theta_P = \sin(\pi + \varphi) \approx -\varphi \tag{26}$$

$$(\dot{\theta}_P)^2 = (\dot{\varphi}_P)^2 \approx 0 \tag{27}$$

Then substitute approximations in (25), (26) and (27) into (23) and (24) result to two main governing equations as follows.

$$\ddot{\varphi} = \frac{M_P l}{(I_P + l^2M_P)}\ddot{x} + \frac{2k_m k_e}{Rr(I_P + l^2M_P)}\dot{x} - \frac{2k_m}{R(I_P + l^2M_P)}V_a + \frac{M_P g l}{(I_P + l^2M_P)}\varphi \tag{28}$$

$$\ddot{x} = \frac{M_P l}{\left(\frac{2I_W}{r^2} + M_P + 2M_W\right)}\ddot{\varphi} - \frac{2k_m k_e}{Rr^2\left(\frac{2I_W}{r^2} + M_P + 2M_W\right)}\dot{x} + \frac{2k_m}{R\left(\frac{2I_W}{r^2} + M_P + 2M_W\right)}V_a \tag{29}$$

Combine (28) and (29) into state-space model shown in (30) and (31).

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\varphi}\\ \ddot{\varphi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2k_m k_e(M_P l r - I_P - M_P g l^2)}{Rr^2\alpha} & \frac{M_P g l^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{2k_m k_e(r\beta - M_P l)}{Rr^2\alpha} & \frac{M_P g l \beta}{\alpha} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \varphi\\ \dot{\varphi}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2k_m(-M_P l r + I_P + M_P l^2)}{Rr\alpha}\\ 0\\ \frac{2k_m(-r\beta + M_P l)}{Rr\alpha}\end{bmatrix} \quad (30)$$

$$y = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \varphi\\ \dot{\varphi}\end{bmatrix} \quad (31)$$

Where $\beta = 2M_W + \frac{2I_W}{r^2} + M_P$ and $\alpha = I_P\beta + M_P l^2\left(M_W \frac{I_W}{r^2}\right)$

TABLE I
PAREMETERS OF THE SYSTEM

Parameters	Description	Value
g	Gravitational acceleration	9.81 m/s^2
r	Wheel's radius	0.05 m.
M_W	Mass of a wheel	0.13 kg.
M_P	Mass of pendulum	1.24 kg.
l	Length to chassis's center of mass	0.213 m.
I_W	Wheel's moment of inertia	0.0002899 $kg.m^2$
I_P	Chassis's moment of inertia	0.05626 $Kg.m^2$
k_m	Motor's torque constant	0.2774 $N.m/A$
k_e	Back EMF constant	0.509 $V/(rad/s)$
R	Equivalent resistance of motor	7.101 Ω

IV. CONTROLLER DESIGN AND SIMULATION RESULT

This section discusses the design technique of PID and FOPID controller and then simulate and compare control results using Simulink at the end of this section.

A. PID controller design

After state-space and parameters of self-balancing robot obtained in the previous section. The PID controller's gains (K_p, K_i, K_d) obtained by minimizing the difference between reference position ($\varphi = 0$, balance in vertical upward) and actual output using trial and error method based on mathematical model and balance ability of real robot's system.

Step 1: Create a closed-loop control system in Simulink represented in Fig.6, Auto-tune parameters using PID controller toolbox with 30-degree initial condition, adjusting proper response time and transient behavior. Controller output value required between -12 to 12 V. Too fast response time and strong transient behavior lead to the insufficient torque produced by two motors cause the robot can't keep itself upward.

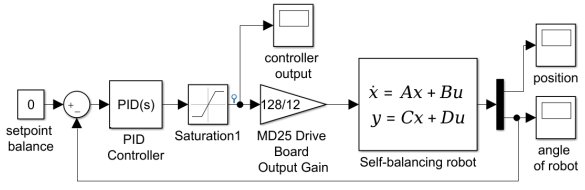

Fig. 6. Block diagram of closed-loop control.

Step 2: Applying PID controller gains to Raspberry Pi board using block diagram language available in CODESYS V3.5 (discussing at section V), start step 1 over again until achieving the best result. Optimal PID gains represented in Table II.

B. FOPID controller design

The design of FOPID controller performed by fractionalorder PID controller optimization tool form FOMCON toolbox for MATLAB called by command `fpid_optim` using the following steps.

Step 1: Convert state-space of the system to transfer function using `ss2tf()` for optimizing by FOPID controller optimization toolbox.

Step 2: Fix K_p, K_i, K_d with optimized integer-order PID parameters obtained from section A. $\lambda, \mu = 1$ and set search limit for $\lambda = [0.01, 10], \mu = [0.01, 2]$, controller output $= [-12, 12]$.

Step 3: Using integrate square of error (ISE) performance matrix for faster response. Set setpoint to 0.078 rad (small deviation angle because `fpid_optim` toolbox focuses on transfer function optimization with zero initial condition there is not possible to set the setpoint to 0). Optimal FOPID controller's parameters are also represented in Table. II

TABLE II
PARAMETERS OF THE CONTROLLER

Parameters	PID controller	FOPID controller
K_p	194.534	194.534
K_i	1432.586	1432.586
K_d	5.551	5.551
λ	1	1.357
μ	1	0.803

The simulation results of both controllers with 2 sec simulation time compared in Fig.7, Fig.7 and Table III. show that the result of closed-loop control with proper FOPID controller (red line) overshoot improved from 29.22% to 18.45%, rise time decreases from 74.74 ms to 46.91 ms compared with integer-order PID controller. The fractional-order $PI^{\lambda}D^{\mu}$ controller can achieve the system stability, great transient response and robustness compared with conventional PID controller.

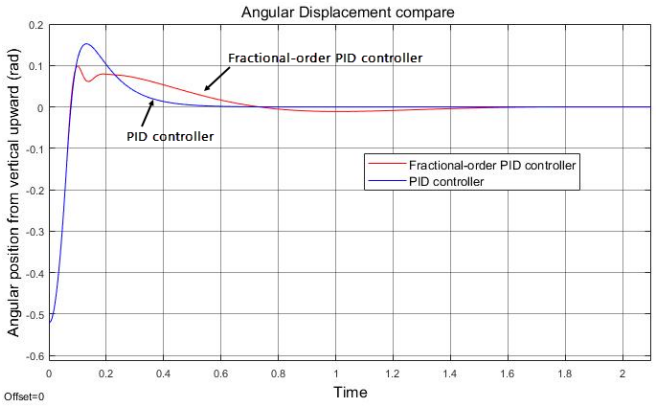

Fig. 7. Angle displacement results compared PID and FOPID controller.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

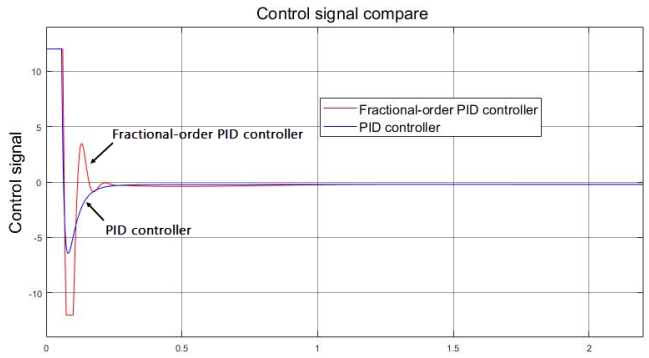

Fig. 8. Control signal of PID and FOPID controller.

TABLE III
RESULT SPECIFICATION OF PID AND FOPID CONTROLLER

Parameters	PID controller	FOPID controller
Rise Time	74.74 ms	46.91 ms
Delay Time	49 ms	49 ms
Peak Time	0.13 sec	0.10 sec
Settling Time %	335 ms	544 ms
Percent Overshot	29.22 %	18.45 %

V. FOPID CONTROLLER IMPLEMENTATION ON REAL SYSTEM

This section briefly demonstrates implementation of the controller into microcontroller board (Raspberry Pi 3) in term of IIR cascaded second-order section form II transfer function using CODESYS V3.5 Fig.9 represent the concept of a closed-loop control system of the self-balancing robot.

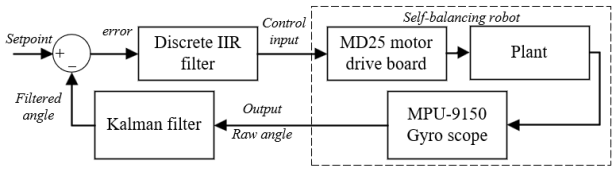

Fig. 9. Block diagram of closed-loop control.

Using PID implementation MATLAB toolbox pidim to convert continuous PID controller and Fractional-order PID controller to discrete-time transfer function with zero-order hold discretization method and convert to IIR cascaded second-order section form II matrix by `d2sos()` as follows.

PID controller:

```
bn[i] = 1.0000, -1.9777, 0.9779
an[i] = -1.3679, 0.3679
System DC gain = 5551.4
```

FOPID controller:

```
bn[i]= 1,0.4050, -1.3503  an[i]= -0.8630, 0.1859
       1,0.5810, -1.5259          -1.5740, 0.6192
       1,0.8942, -1.8914          -1.8681, 0.8724
       1,0.9652, -1.9649          -1.9615, 0.9619
       1,0.9943, -1.9943          -1.9890, 0.9890
       1,0.9954, -1.9954          -1.9968, 0.9969
       1,0.9988, -1.9988          -1.9991, 0.9991
       1,0.9997, -1.9997          -1.9997, 0.9997
       1,0.9999, -1.9999          -1.9999, 0.9999
       1,1.0000, -2.0000          -2.0000, 1.0000
       1,1.0000, -2.0000          -2.0000, 1.0000
       1,0.9945, -1.9944          -2.0000, 1.0000

System DC gain =1619.5590
```

Create IIR cascaded second-order section form II function block using CODESYS V3.5 for controlling real system of the self-balancing robot the angular displacement of the robot with PID and FOPID controller shown in Fig.10 and Fig. 11

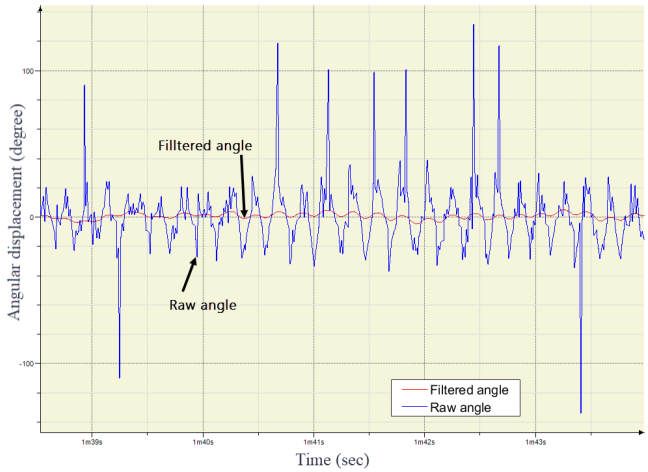

Fig. 10. Angular displacement of robot with PID controller.

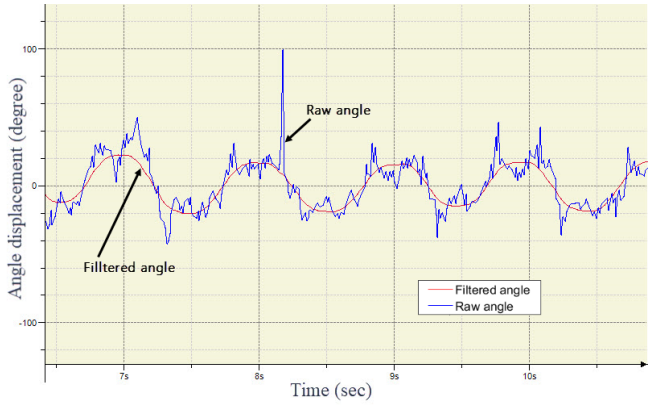

Fig. 11. Angular displacement of robot with FOPID controller.

Fig.10 represents raw angle measured from MPU-9150 gyroscope (blue line) and noiseless angle filtered by Kalman's filters (red line). The result has shown that the angular displacement controlled by the PID controller of the robot is oscillated with amplitude less than ± 5 degree and keep the robot upward.

From Fig.11 the angular displacement of the robot controlled by FOPID controller also oscillates with ± 20 degree amplitude that varies more than the result of PID controller.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Unfortunately, there have two main problems with this implementation. First, the complexity of calculation and many loops per task circle cause the process slow response. The robot can't balance itself. Second, because of i2c communication, we used in this paper is limited speed.

VI. Conclusion

Fractional-order $PI^{\lambda}D^{\mu}$ controller (FOPID) for the self-balancing robot has been studied on this paper. The proper FOPID parameters obtained by FOMCON toolbox for MATLAB based on proper integer-order PID parameters to achieving system performance and stability using MATLAB and SIMULINK. The realize implementation on Raspberry Pi concept for both controllers has been introduced using IIR cascaded second-order section form II transfer function in filter form on CODESYS V3.5, In theoretical term. simulated results show that the FOPID controller can stabilize the system and improve transient response with less percent overshoot and rise time than PID controller. Whereas the implementation of PID controller on real robot system can keep the robot stable better than FOPID controller because of this implementation on raspberry or microcontroller using numerical method of filter lead to over computing and slow overall process. The FOPID implementation on Raspberry Pi has been studying in the future.

The future study will focus on the implementation of fractional-order PID controller on a microcontroller board (Raspberry Pi), the improvement of program cycle speed for FOPID controller calculation and communication speed of robot's components.

References

[1] R.S. Voliansky and A.V. Sadovoi, "Second order sliding mode control of the inverted pendulum," *Proceedings of 2017 International Conference on Modern Electrical and Energy Systems (MEES)*, Nov. 2017. pp. 224-227.

[2] O. Saleem and F. Abbas, "Nonlinear self-tuning of fractional-order PID speed controller for PMDC motor," *2017 13th International Conference on Emerging Technologies (ICET)*, Islamabad, 2017, pp. 1-6.

[3] S. Jiang, M. Li and C. Wang, "Design and simulation of fractional order PID controller for an inverted pendulum system," *2017 IEEE International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale (3M-NANO)*, Shanghai, 2017, pp. 349-352.

[4] M.I. Ali and M.M. Hossen, "A two-wheeled self-balancing robot with dynamics model," *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Dhaka, 2017, pp. 271- 275.

[5] S. Wenxia and C. Wei, "Simulation and debugging of LQR control for two-wheeled self-balanced robot," *2017 Chinese Automation Congress (CAC)*, Jinan, 2017, pp. 2391-2395.

[6] B. Shilpa, V. Indu and S.R. Rajasree, "Design of an underactuated self balancing robot using linear quadratic regulator and integral sliding mode controller," *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*, Kollam, 2017, pp. 1- 6.

[7] R. S. Martins and F. Nunes, "Control system for a self-balancing robot," *2017 4th Experiment @International Conference (exp.at'17)*, 2017, pp. 297-302.

[8] Emil Eriksson (2016) *Self-Balancing Robot Control System in CODESYS for Raspberry Pi (Bachelor's Thesis)*, Umeå University, Umeå, Sweden. Retrieved from http://www.divaportal.org/smash/record.jsf?pid=diva2:1090623

[9] A. Tepljakov, E. Petlenkov and J. Belikov, "FOMCON: a MATLAB toolbox for fractional-order system identification and control" *International Journal of Microelectronics and Computer Science*, vol. 2, no. 2, pp. 5162, 2011.

[10] C.A. Monje, Y. Chen, B. Vinagre, D. Xue, and V. Feliu, "Fractionalorder Systems and Controls: Fundamentals and Applications" *ser. Advances in Industrial Control. Springer Verlag, 2010.*

[11] H. Cheng, J. Xiao, Y. Long and T. Zhang, "Wind pendulum modeling based-on improved PID algorithm," *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, Hefei, 2016, pp. 2288-2293.

[12] Greg Welch and Gary Bishop. 1995. "An Introduction to the Kalman Filter" *Technical Report. University of North Carolina at Chapel Hill*, Chapel Hill, NC, USA.

[13] K. Liu, M. Bai and Y. Ni, "Two-wheel self-balanced car based on Kalman filtering and PID algorithm," *2011 IEEE 18th International Conference on Industrial Engineering and Engineering Management*, Changchun, 2011, pp. 281-285.

[14] L. Ouyang, B. Du, A. Peng and Y. Ou, "Design of a Wheeled Inverted Pendulum as a platform for learning based control," *2012 IEEE International Conference on Information and Automation*, Shenyang, 2012, pp. 418-421.

[15] O.K. Sayidmarie, M.O. Tokhi and S.A. Agouri, "Real-time validation of a novel two-wheeled robot with a dynamically moving payload," *The 23rd IEEE International Symposium on Robot and Human Interactive Communication, Edinburgh*, 2014, pp. 102-105.

[16] Steve Winder, Chapter 17 – IIR filter design, Editor(s): Steve Winder, In *EDN Series for Design Engineers, Analog and Digital Filter Design (2nd Edition)*, Newnes, 2002, pp. 395-408, ISBN 9780750675475

[17] R. Electronics, "EMG30, mounting bracket and wheel specification: Robot Electronics," 2016. *[Online]. Available: http://www.robotelectronics. co.uk/htm/emg30.htm.* [Accessed 16 May 2018].

[18] R. Electronics, "MD25 Technical Documentation: Robot Electronics," 2016. *Available: https://www.robot-electronics.co.uk/htm/md25tech.htm.*, [Accessed 16 May 2018].

[19] Gonçalves, J., Lima, J., Costa, P. and Moreira, A. "Modeling and Simulation of the EMG30 Geared Motor with Encoder Resorting to SimTwo," *The Official Robot@Factory Simulator. Advances in Sustainable and Competitive Manufacturing Systems*,pp.307-314.

[20] MathWorks. 2012. "Control Tutorials for MATLAB and SIMULINK, Inverted Pendulum." Available at: *http://ctms.engin.umich.edu/CTMS/index.php.exampleInvertedPendulumsection-SystemModeling.* [Accessed 13 May 2018]

[21] O. Jamil, M. Jamil, Y. Ayaz and K. Ahmad, "Modeling, control of a two-wheeled self-balancing robot," *2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE)*, Islamabad, 2014, pp. 191-199.

Modification date : 28 May 2019
editing detail:
1. author's name below the name of the paper.
old version: "K.Kankhunthodl"
edited version: "K. Kankhunthod"
2. Change the wrong word.
edit the wrong symbol of the Fractional-order PID controller from "P^(lambda) ID^(mu) Controller" to the colected word "PI^(lambda)D^(mu) Controller" To avoid confusion of readers.