Benha Journal of Applied Sciences (BJAS) **Print: ISSN 2356–9751**
Vol. (10) Issue (4) (2025), (1-8) **Online: ISSN 2356–976x**

http://bjas.bu.edu.eg
Engineering Sciences

PID Controller Design and Implementation for Multibody Mechatronic Systems

Ibrahim Abdel-Hady[1], Mona A. Bayoumi [2], Nader A.Mansour[1, 3] and Ayman A. Nada[4]
[1]*Mechanical Engineering Dept., Engineering Faculty, Benha University, Benha, Egypt*
[2]*Electrical Engineering Dept., Engineering Faculty, Benha University, Benha, Egypt*
[3]*Egypt University of Informatics (EUI), New Administrative Capital, Cairo, Egypt*
[4]*Mechatronics and Robotics Engineering Dept., Egypt-Japan University of Science and Technology, Alexandria, Egypt*
Email: ibrahim.abdulhadi@bhit.bu.edu.eg

Abstract

The multibody dynamics (MBDs) approach is mainly used for modelling and simulating mechanical systems, especially if they have complex dynamic analysis. Stabilization controllers are added to provide constraint stability during validation of the multibody equations. For controller design, model linearization should be addressed to obtain the state-space formulation. On the other hand, field-programmable gate arrays (FPGAs) are widely used for the fast development of control systems and embedded applications. Thus, they are suitable for the stabilization of the underactuated mechatronic applications. Thus, MATLAB software is used to build the symbolic model and the computational simulation of an inverted wheeled robot. Then, the PID controller is designed and simulated for this application. Simulation results show the effectiveness of the multibody dynamics approach in the formulation of the state space model-based controllers of mechatronic applications. The LabVIEW FPGA module is used to implement the controller on the sbRIO 9631 single RIO board. The results show that the settling time is 2 seconds, and the steady-state error is ±0.5 degrees at the zero-tilt angle set point.

Keywords: Mulltibody Modelling, MBDs, Self-balancing Robot, FPGAs, real-time PID

1. Introduction

Reconfigurable FPGAs are becoming of great interest in many fields, such as industrial [1] [2], embedded systems, and control mechatronic applications [3] [4] [5]. A wide range of applications benefit Fast FPGAs in the fields of robotics, as in [6], [7], [8], autonomous robot fields [9], serial robots [10], [11], [12], and the design of mechatronic systems [13], [4]. Using graphical programming FPGAs like MATLAB and LabView software simplifies the design process and contributes to the fast development and testing of control systems. RIO devices are widely provided by National Instruments for industrial embedded and DAQ systems. [13]

Modeling is essential for the precise design of control systems. This study addresses the multibody dynamics (MBD) technique. This is extensively utilized in the dynamic analysis and modeling of mechanical systems. It has served as the foundation for studying of interconnected rigid and flexible bodies[1]. The general MBD model is provided by eq. (1).

$$\begin{bmatrix} \widehat{\boldsymbol{M}} & \boldsymbol{C}_{\boldsymbol{q}}^{\boldsymbol{T}} \\ \boldsymbol{C}_{\boldsymbol{q}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{Q}} \\ \widehat{\boldsymbol{Q}}_{\boldsymbol{d}} \end{bmatrix} \quad (1)$$

Where, $\widehat{\boldsymbol{M}}$ is the total system mass matrix and $\boldsymbol{C}_{\boldsymbol{q}}^{\boldsymbol{T}}$ is the constraints jacobian matrix, $\ddot{\boldsymbol{q}}$ is the coordinates acceleration vector, λ is the lagrange multipliers correspond to each constraint equation defined for the mechanical system. While $\widehat{\boldsymbol{Q}}$ is the system external forces and $\widehat{\boldsymbol{Q}}_{\boldsymbol{d}}$ is the vector contains the quadratic velocity terms that is defined by eq.(2) as follows in [14][15],

$$\widehat{\boldsymbol{Q}}_{\boldsymbol{d}} = -\left(\boldsymbol{C}_{\boldsymbol{q}}\dot{\boldsymbol{q}}\right)_{\boldsymbol{q}}\dot{\boldsymbol{q}} - \boldsymbol{2}(\boldsymbol{C}_{\boldsymbol{qt}})\dot{\boldsymbol{q}} - (\boldsymbol{C}_{\boldsymbol{tt}}) \quad (2)$$

2. Test-Rig Components Describtion.

As shown in both Fig. (1) and Fig. (2), the self-balancing robot (SBR) prototype is simply composed of the robot's main body that is controlled in the upright position with a tilt angle of zero by its wheels. The actuators are two DC motors that are aligned with the robot wheels axis. Robot motors need a voltage regulator to map the incoming control action to the DC motor voltage supply range. This is done by using a 12-volt DC motor dual driver with a high switching frequency that reaches up to 20 kHz. The tilt angle is the most crucial in the balancing task. So, a low-cost accelerometer is utilized to do this task. Provided with some noise filter (low-pass filter) to enhance its readings of acceleration from which the tilt is obtained.

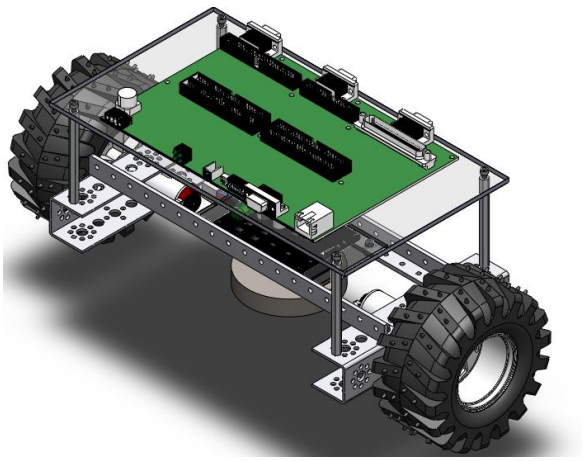


Fig (1) Robot CAD Model and Components.

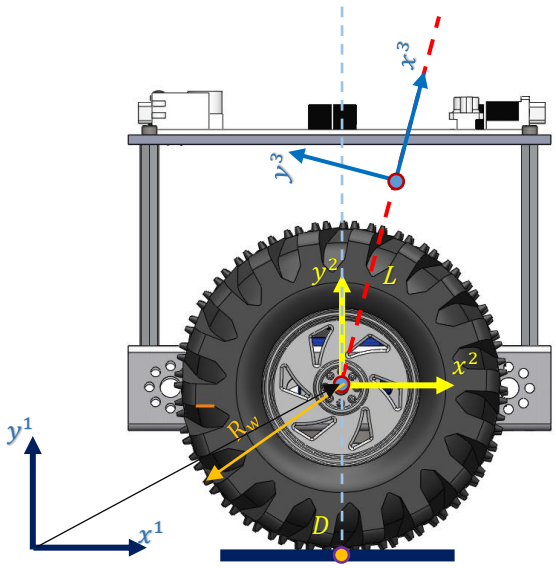


Fig (2) Robot planar configuration.

3. The Two Dimensional MBDs Model

3.1 Constarints Equations

As shown in Fig (2), The robot is simplified for the planar case by two bodies. Each is denoted by a body frame denoted by $x^i y^i$with $i = 2,3$ and the fixed frame is denoted by $i = 1$. The first body includes the robot wheel with its coupling with the motor. It is described by its radius R_w , mass m_w^2 and mass moment of inertia about its z-axis I_w^2 . Similarly For body 2. That is considered one rigid body that includes the masses of the DC motors, motor driver, robot chassis, battery, sensor and the RIO board. Its center of gravity is located above the wheel axis by distance l^3 , its mass m_{pen}^3 and mass moment of inertia about its z-axis I_{pen}^3.

On using the MBDs approach to develop the robot model, one can define the generalized coordinates vector by, $\boldsymbol{q^T} = [R_x^2 \;\; R_y^2 \;\; \theta^2 \;\; R_x^3 \;\; R_y^3 \;\; \theta^3]$. Where, R_x^2 is the position of the wheel centre of gravity in the global x^1 direction, similarly R_y^2 in the global y^1 and $\theta^2$is the angular displacement of the wheel. . Similarly for body 2 generalized coordinates.

For the holonomic constraints, they are described by one revolute joint between the two bodies, body 1 and body 2, and the wheel must be in contact with the ground. For the nonholonomic one, the robot wheel velocity at the contact point D for pure rolling constraints should equal zero. Thus, the total constraints vector can be defined as in eq. (3)

$$\boldsymbol{C^{(Total)}} = \begin{bmatrix} (R_x^2 - R_x^3) + (l^3 cos\theta^3) \\ (R_y^2 - R_y^3) + (l^3 sin\theta^3) \\ R_y^2 - R_w \\ \dot{R}_x^2 - R_w\dot{\theta}^2 \end{bmatrix}_{4\times 1} = \mathbf{0} \quad (3)$$

3.2 Constarints Jacobian

Referring to eq.(1) one should obtain the constraint Jacobian matrix $\boldsymbol{C_q}$ and this is defined by,

$$\boldsymbol{C_q} = \begin{bmatrix} \boldsymbol{C_{q(1-3)}} \\ \boldsymbol{C_{q(4)}} \end{bmatrix}$$

Where, $\boldsymbol{C_{q(1-3)}}$ associated with the holonomic type of constraints $\boldsymbol{C^{(Total)}}_{\boldsymbol{1\to 3}}$ and $\boldsymbol{C_{q(4)}}$ for the nonholonomic rolling constraint $\boldsymbol{C^{(Total)}}_{\boldsymbol{4}}$. Thus

$$\boldsymbol{C_{q(1-3)}} = \frac{\boldsymbol{\partial C^{(Total)}}_{\boldsymbol{1\to 3}}}{\boldsymbol{\partial q}} \quad (4)$$

And,

$$\boldsymbol{C_{q(4)}} = \frac{\boldsymbol{\partial C^{(Total)}}_{\boldsymbol{4}}}{\boldsymbol{\partial \dot{q}}} \quad (5)$$

The resulting Jacobian is a matrix of size 4×6.

3.3 Robot Mass matrix, The Quadritic velocity and forces vectors,

The system mass matrix is defined as in [14][15] by,

$$\boldsymbol{M^i} = \begin{bmatrix} m^i & 0 & 0 \\ 0 & m^i & 0 \\ 0 & 0 & {I^i}_{zz} \end{bmatrix} \quad (6)$$

Where, $i = 2,3$ for the both bodies, and ${I^i}_{zz}$ defines the inertia about the z axis of body i.

The next step is to find the quadratic velocity term $\boldsymbol{\hat{Q}_d}$, as in eq.(2)

$$\boldsymbol{\hat{Q}_d} = \begin{bmatrix} l^3\,\dot{\theta}^3 cos\theta^3 \\ l^3\,\dot{\theta}^3\, sin\theta^3 \\ 0 \\ 0 \end{bmatrix} \quad (7)$$

Finally, the external forces vector for the robot without the actuating torque could be defined using the virtual work principle in [15], and can be defined by,

$$\boldsymbol{\hat{Q}} = \begin{bmatrix} 0 \\ -m_w^2\, g \\ 0 \\ 0 \\ -m_{pen}^3 g \\ 0 \end{bmatrix} \quad (8)$$

3.4 Coordinate Partitioning Method

For the case of finding independent states for later linearization of the planar model, coordinate partitioning method, is fully explained in [15][17] and [18], could be used. In this method, the generalized coordinates are partitioned to two generalized coordinates vectors as, $\boldsymbol{q^T} = [\boldsymbol{q_d} \;\; \boldsymbol{q_i}]$ where $\boldsymbol{q_i} = [R_x^2 \;\; \theta^3]$ are the independent coordinates and the remaining terms, $\boldsymbol{q_d} = [R_y^2 \;\; \theta^2 \;\; R_x^3 \;\; R_y^3]$, are the dependent coordinates. Before that, some model modifications are made. one could incorporate the DC motor simplified model to the MBDs model through

the motor torque and the input DC voltage relation defined in [19]. It could be summarized by the linear relationship assumption between the motor torque and the input voltage,

$$\tau_w = \frac{N_G\, k_t}{R_a} v \tag{9}$$

Where, N_G is the motor gear ratio and k_t is the torque constant and R_a is the motor armature resistance. Also by adding the friction forces using the classical Coulomb friction model on robot wheels as defined in [20], and the damping at the revolute joint by [21]. The resulting equations from the partitioning method for independent accelerations are,

$$\ddot{\boldsymbol{q}}_{\boldsymbol{\iota}} = \bar{\boldsymbol{M}}^{+}\bar{\boldsymbol{Q}}_{\boldsymbol{B}} \tag{10}$$

Where,

$$\ddot{\boldsymbol{q}}_{\boldsymbol{\iota}} = \begin{bmatrix} \ddot{R}_x^2 \\ \ddot{\theta}^3 \end{bmatrix} \tag{11}$$

And the linearized equation terms about the robot stabilization point the equilibrium point of Tilt angle could be made by using the first-order Taylor expansion in [22] or by setting the equilibrium point for the robot at tilt angle of $\theta^3 = 90 + \theta$. According to Frame 3 orientation. Where θ is the angle measured from the upright stable position of the inverted body. Thus $\sin(\theta^3) = sin(\theta + 90) = cos(\theta) = 0$. Similarly, $\cos(\theta + 90) = -sin\theta = -\theta$, and the tilt angular velocity $\dot{\theta}^3 = 0\ and\ , \dot{\theta}^{3^2} = 0$. The resulting substitutions in the model are (the assembled mass matrix and the total force vector are as follows,

$$\bar{\boldsymbol{M}} = \begin{bmatrix} m_w^2 + m_{pen}^3 + \frac{I_w^2}{R_w{}^2} & -l^3\, m_{pen}^3 sin\theta^3 \\ -l^3\, m_{pen}^3 sin\theta^3 & m_{pen}^3 l^{3^2} + I_{pen}^3 \end{bmatrix}_{\mathbf{2\times2}} \tag{12}$$

And,

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} (K_d\dot{\theta}^3/R_w - f_r\boldsymbol{tanh}\left(k_f\dot{R}_x^2\right) \times \\ \left(m_w^2\boldsymbol{g} + m_{pen}^3\boldsymbol{g}\right) - \theta^3 + 2K_tN_GV/R_wR_a) \\ 2K_tN_GV/R_w - K_d\dot{\theta}^3 - l^3m_{pen}^3\boldsymbol{g} - \theta^3 \end{bmatrix}_{\mathbf{2\times1}} \tag{13}$$

Where, $K_d = 0.1$ is the damping coefficient, $\mathrm{f_r} = 0.2$ is the dry friction coefficient and the k_f is the Coulomb friction coefficient. Note that Coulomb-tanh alternative is used instead of the classical Coulomb friction model as defined in [20], because it ensures continuity at zero velocities.

At this point The dynamic model was solved using MATLAB software with the ODE45 function, employing a time step of Δt=0.001 seconds, working together with the Bumgrate stabilizing approach outlined in [15] for the augmented formulation in (1) to control the constraint violations. Fig (3) illustrates the simulation results, depicting the tilt angle and its angular velocity derived from the dynamic model of equation (1) and the model produced by the coordinate partitioning method.

Property	Data
m_w^2	$0.25\ kg$
m_{pen}^3	$2.045\ kg$
l^3	$0.027\ m$
I_w^2	$0.00045\ kg/m^2$
I_{pen}^3	$0.00616\ kg/m^2$
R_w	$0.06\ m$
g	$9.81\ m/s^2$

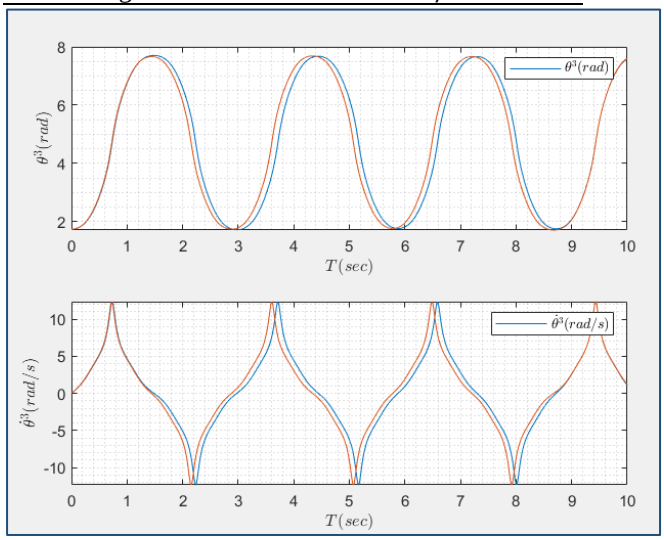


Fig (3) Robot Tilt angle Simulation of both Augmented (blue color) and coordinate partitioning (orange color) method MBDs

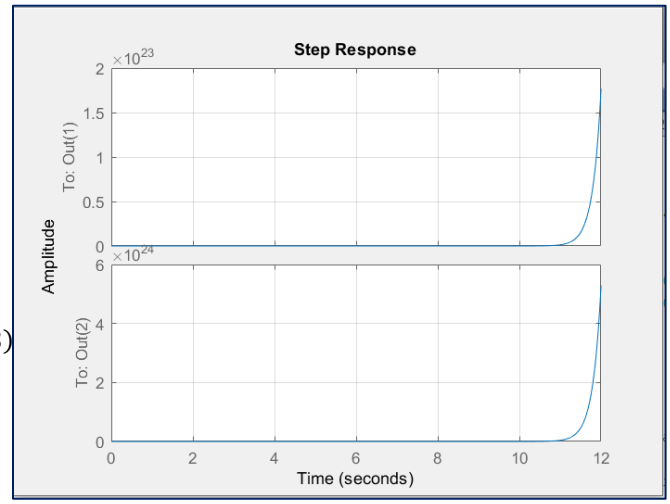


Fig (4) Step Response of the linearized model

4. State Space Model (SSM) and Control

After linearization around the robot equilibrium point of its tilt angle, the system states can be defined by the robot degrees of freedom and their time derivatives as, $x_1 = R_x^2$, $x_2 = \theta^3$, $x_3 = \dot{R}_x^2$,and $x_4 = \dot{\theta}^3$. Then the system state space model is,

$$\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{A}\tilde{\boldsymbol{x}} + \boldsymbol{B}\tilde{\boldsymbol{u}} \tag{14}$$

$$\tilde{\boldsymbol{y}} = \boldsymbol{C}\tilde{\boldsymbol{x}} + \boldsymbol{D}\tilde{\boldsymbol{u}} \tag{15}$$

Where,

$$\boldsymbol{A}=\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1.9337 & -1.6202 & 0\\ 0 & 84.7510 & -11.6927 & 0\end{bmatrix}, \boldsymbol{B}=\begin{bmatrix}0\\ 0\\ 0.1963\\ 1.4168\end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}, \boldsymbol{and}\ \ \boldsymbol{D}=\begin{bmatrix}0\\ 0\end{bmatrix}$$

Before the controller design, the controllability and the observability matrices should be checked for the SSM. It was found that the system is controllable and observable as their matrices are of full rank 4 as the system states as defined in [22][23]. Furthermore, stability analysis shows that the system is unstable as seen from the step response and the system open-loop poles. Fig. (4) shows that the robot is unstable as the system poles are $[0, -111.4968, 8.4725, -8.3466]$. And at least one pole is at the right-hand side of the S-Plane of the pole-zero map. In this paper, our main target is to stabilize the robot only. Thus, our main concern is the tilt angle related to the input voltage to the robot DC motor. The transfer function is defined from the SSM by,

$$\frac{\theta^3}{V}=\frac{137.5\,s\ -\ 9543}{s^4+111.4\,s^3-\ 84.8\,s^2-7885\,s} \qquad (16)$$

For controller design, A classical PID controller is defined in[22][23] and the control action as,

$$u=k_p+k_i\frac{1}{s}+k_d s \qquad (17)$$

PID was designed for robot stabilization with the help of MATLAB pidtool along with the simulink model of eq.(16) for tuning the best parameters for the system response. The best parameters were found as, $k_p=16.8, k_i=94.05$ and $k_d=0.7501$. With rise time of 0.02 sec, maximum overshoot of 19% and settling time of 0.19 sec. with zero steady state error. Now recall, the controller discrete time transfer function could be defined as [1] by,

$$u(z)=k_p+k_i\frac{t_s}{2}\frac{z+1}{z-1}+\frac{k_d}{t_s}\frac{z-1}{z} \qquad (18)$$

where, t_s is the sampling time (loop time). The model response of the closed loop system with the designed PID controller is shown in Fig.(5). The designed PID controller is transformed into eq.(18) for implementation on LabView FPGA sbRio 9631 board.

The motivation behind the utilization of the classic PID controllers is that they follow a straightforward method for implementation on the hardware-based FPGAs. They also require less computational power, making them very efficient, and this is well-suited for real-time control at high sampling rates. Furthermore, this requires less resource consumption of the target FPGA on the limited memory sbRIO 9631. While advanced methods like MPC usually need a lot of computing power and can be complicated to implement on the RIO FPGA, they also have to handle floating-point operations, which are changed into fixed-point operations. This limits the accuracy of computations. In addition to this, it requires higher FPGA resources.

5. Hardware Real-time Controller Impelementation

The self-balancing robot stabilization needs an accurately measured tilt angle for stabilization. To achieve this, we used a low-cost ADXL334 accelerometer. Enhancement of its reading was challenging with the SbRio9631, as the sensor exhibits high noise on small shakes due to the forward and reverse motion of the robot wheels during balancing. A low pass filter was designed using bilinear transformation and implemented on LabView FPGA. Fig (6) shows the actual and filtered signal of the tilt angle by the sensor.

Fig (7) shows the full system components; at the top of the chart is the main programming device PC along with LabView RT and FPGA modules. The software used was NI-LabView 2019 with its supporting modules for control design and simulation. *Section 2 provides a description of the remaining components.* The dc motors were controlled by the motor driver with the PWM Signal Generation code in the FPGA side. The analog control parameters

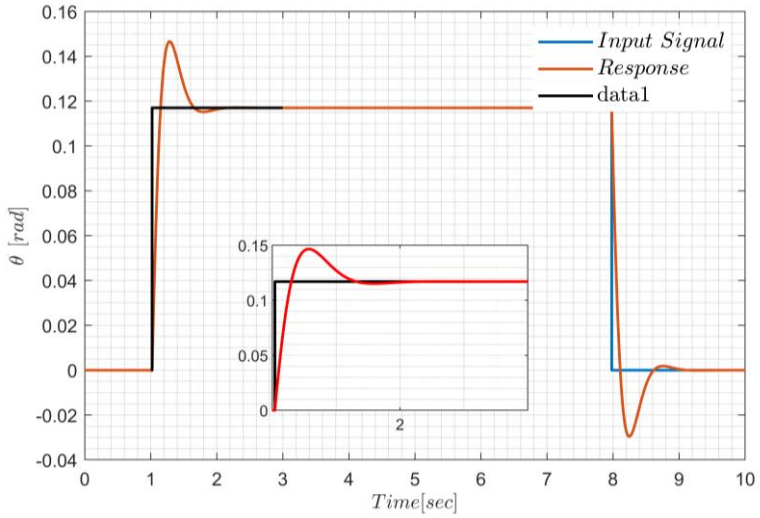


Fig (5) Step Response of the closed loop-controlled system using Simulink $\theta = 0.115\ rad$

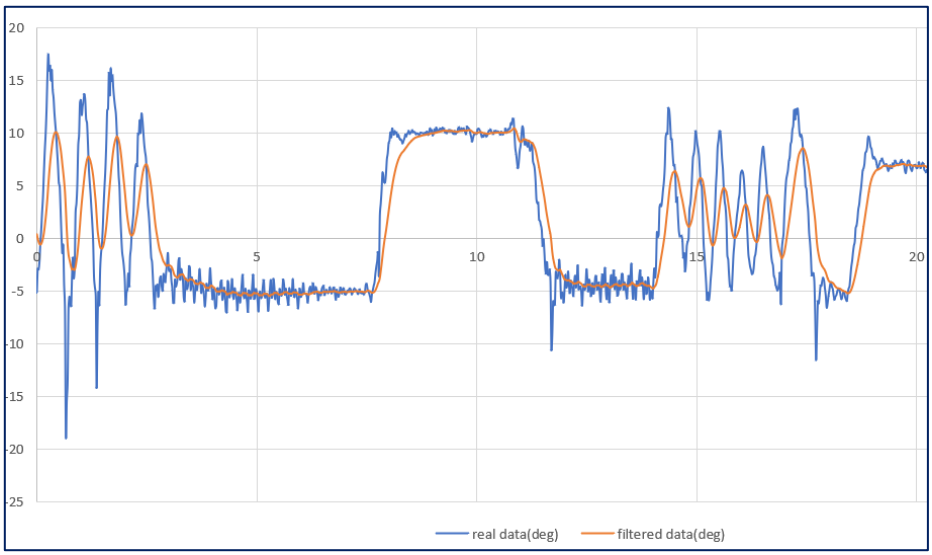


Fig (6) shows the actual and filtered signal of the tilt angle by the sensor.

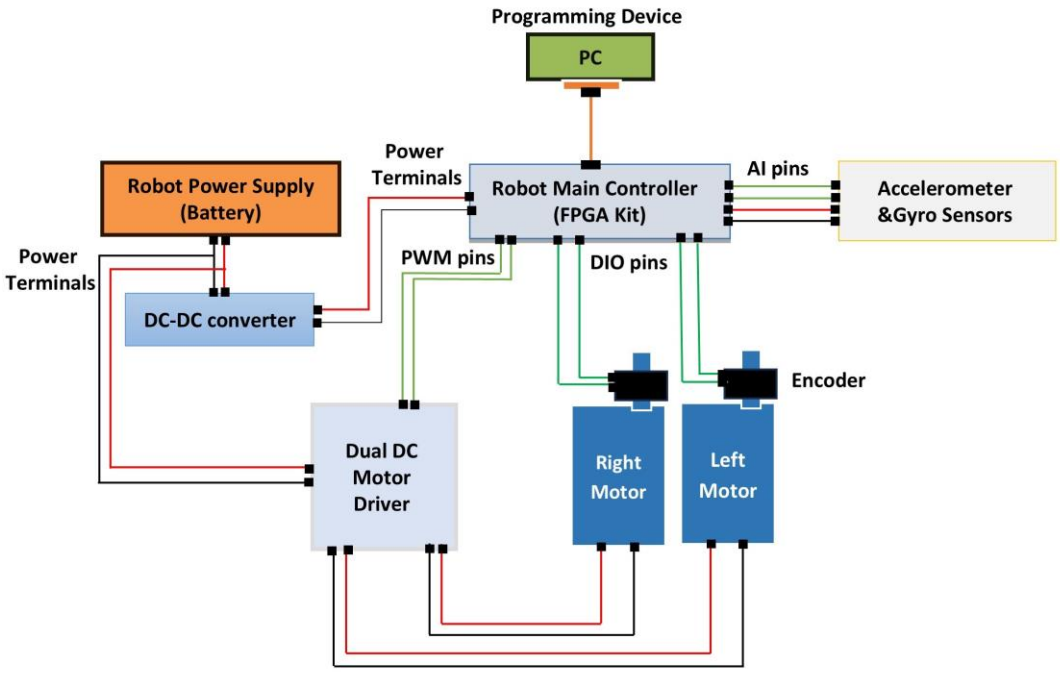


Fig (7) SBR System Hardware Block Diagram

were used in the discrete time PID controller along with the controller loop. Then, the implementation of this controller was done using LABVIEW RT and FPGA modules[16].

For real time implementation, Fig.(8), the LabView RT VI Takes the values on PID controller parameters and the setpoint of the tilt angle and then calculates the control action based on the sensor reading coming from the FPGA part. The output of the controller is limited between ±255 then is mapped to duty cycle that is sent to the FPGA with its value and the direction of actuation of the DC motors to be mapped to a signal by the motor driver to be within 0 to 12 volts.

To drive the robot to the balance condition between 0 to 3.3v the max output of SbRio voltage at the DIO port. That value is regulated

In Fig. (8), each stage is numbered from 1 to 8. Stage 1 the RT processor open the FPGA reference so as to be able to reach FPGA hardware resources. Stage 2, waits for 50 ms before executing the Balancing loop. Stage 3, sets the DC motors duty cycle to zero to avoid any unexpected leakage voltages at the start of the loop. Stage4, the timed loop with loop rate of 0.001 seconds for calculating the control action and sampling the tilt sensor angle reading. That loop include the I/O RT read\write functions invokes the FPGA loops of PWM,

encoder, ADXL Tilt angle reading and motor direction control functions. Stage 5, is the tilt angle low pass filter implementation that takes the angle reading from the FPGA target and damp out the noise. Stage 6, refers to PID controller and motor actuation subVIs. Stage 7, for real time monitoring the tilt and the control actions for balancing the robot. Finally stage 8 ensures closing the reference FPGA on removing power or closing the RT VI.

6. Results

The Results of the PID controller implementation of the FPGA RIO board are shown in Fig (9) and Fig (10). Referring to Fig (9) , in this case, the tilt setpoint is set to zero and after some tuning trials with the proportional, integral and the derivative gains k_p=20, k_i = 0.00181, and k_d = 0.4. It showed good and enhanced results. As it exhibits nearly zero steady-state error. Furthermore, the added disturbance at the t = 7.5 seconds shows that the robot stabilizes in about 4 seconds as in Fig (9). A. To reduce this time one could replace the existing DC motors with another that have higher torque and angular velocity. Also a lead compensator could be designed to respond faster with little oveshoot. While Fig(9).B shows the corresponding DC motor driver input Duty cycle. Another successful case, by changing the set point to -1.5 degrees, the results are shown in Fig (10). The robot exhibits a higher overshoot than in Case 1 and that due to the initial tilt angle as it tries to stabilize the robot about the set point. However, it settles faster to around zero, and at $t = 10\ s$, it settles at nearly -1.7 degrees. And the corresponding duty cycle is shown in Fig (10).

B. Additionally we managed to build the dynamic model and found the state space model based on which the controller design was made and simulated. In addition to this, we managed to enhance the sensor measurement by designing a low pass filter and building it on the real-time FPGA based controller.

7. Conclusion

In conclusion, the multibody modelling and simulation is a powerful approach for model-based controller design and simulation. We used the coordinate partitioning method in the multibody dynamics (MBD) approach for state-space formulation. On the other side, utilizing FPGA technology contributes to rapid prototyping and implementation of multibody mechatronic systems controllers. In this paper, we successfully built the planar multibody model of the wheeled balancing robot. The simulation of the stated dynamic equations validates this model. Furthermore, the digital PID controller was successfully designed, simulated, and implemented on NI-SbRIO with the LabVIEW FPGA module. In the future work, we will consider designing and implementing a state feedback controller as a pole placement and optimal feedback controller LQR on the FPGA RIO board.

8. Acknowledgment

This paper is based upon work supported by Science, Technology & Innovation Funding Authority (STDF) under grant Post Graduate Support (PGSG) Call 2, Project ID (48366) .

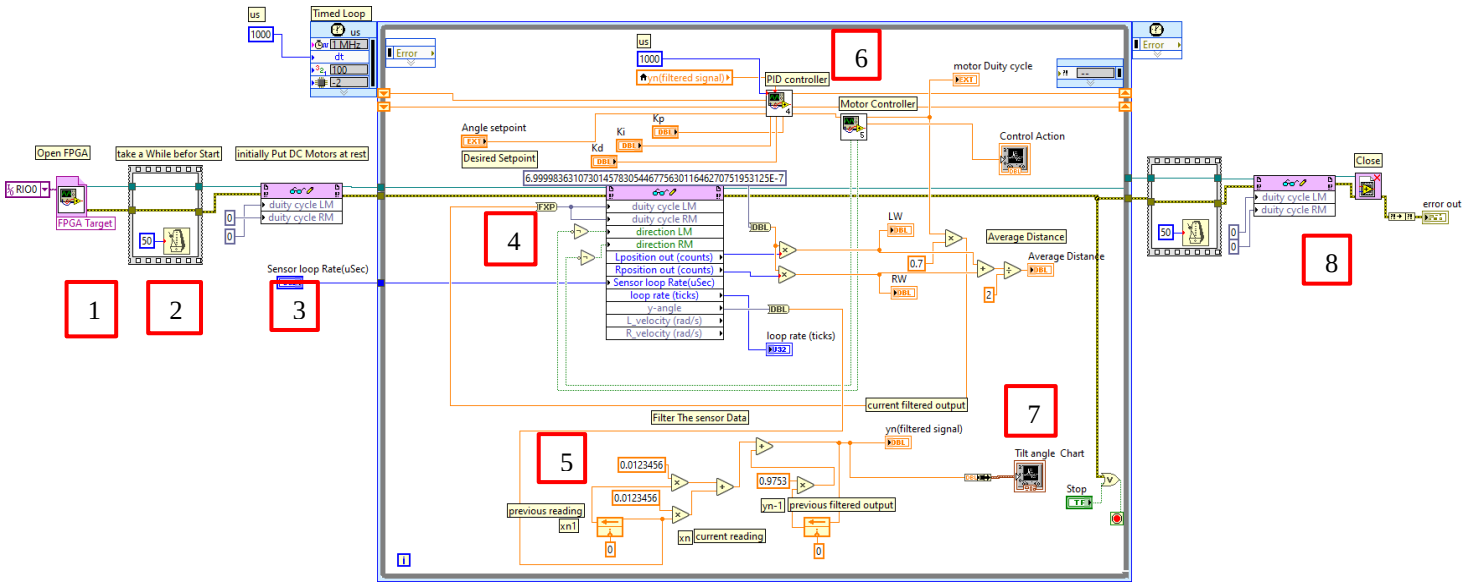
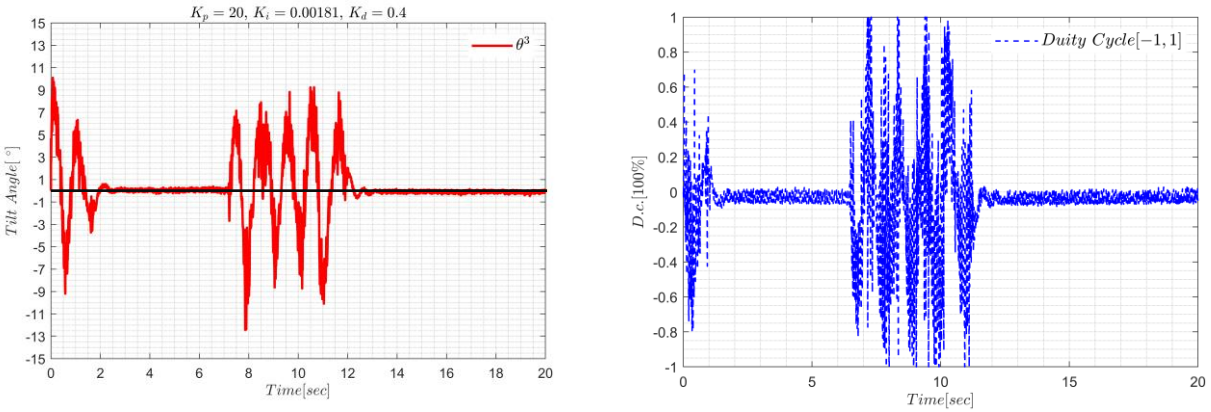
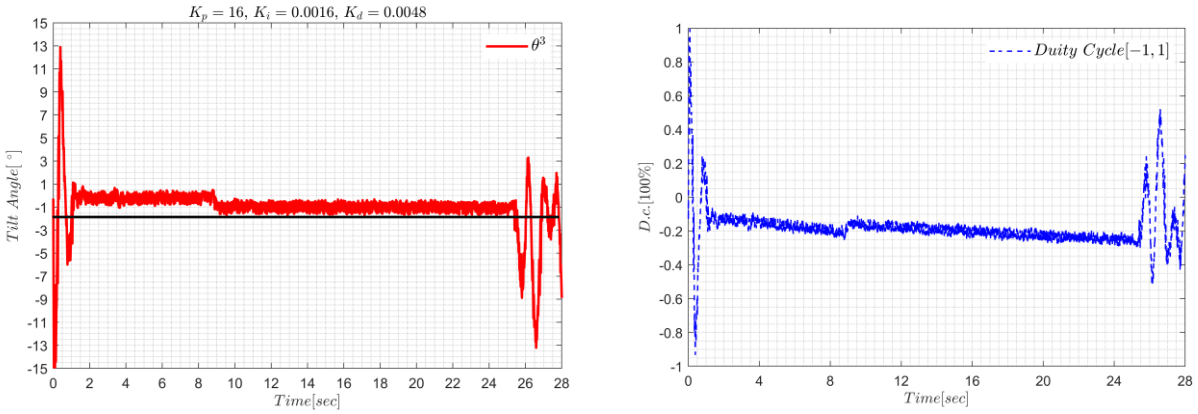


Fig (8) SBR RT PID Controller VI



(A) (B)

Fig (9) A) SBR Tilt angle, B) Duty Cycle at setpoint $\theta^3 = 0^\circ$.



(A) (B)

Fig (10) A) SBR Tilt angle, B) Duty Cycle at setpoint $\theta^3 = -1.5^\circ$

References

[1] AR Hamed, EM Shaban, Abdelhaleem, and AM Abdel ghany. 2022. Industrial implementation of state dependent parameter PID+ control for nonlinear time delayed bitumen tank system. Iranian Journal of Science and Technology, Transactions of Electrical Engineering 46, 3 (2022), 743–751.

[2] EM Shaban, AR Hamed, AM Bassiuny, and AM Abdel ghany. 2024. On implementation of nonlinear PID+ controller embedded on FPGA module for industrial system. *International Journal of Dynamics and Control* 12, 7 (2024), 2331–2340.

[3] Ayman A Nada and Mona A Bayoumi. 2024. Development of embedded fuzzy control using reconfigurable FPGA technology. *Automatika* 65, 2 (2024), 609–626.

[4] Ehab Seif Ghith and Farid Adel Aziz Tolba. 2022. Design and optimization of PID controller using various algorithms for micro-robotics system. *Journal of Robotics and Control (JRC)* 3, 3 (2022), 244–256.

[5] Zaher M Kassas. 2011. Methodologies for implementing FPGA-based control systems. *IFAC Proceedings Volumes* 44, 1 (2011), 9911–9916.

[6] Ryuichi Tsutada, Trong-Thuc Hoang, and Cong-Kha Pham. 2022. An Obstacle Avoidance Two-Wheeled Self-Balancing Robot. *International Journal of Mechanical Engineering and Robotics Research* 11, 1 (2022), 1–7.

[7] Juan Ordóñez Cerezo, Encarnación Castillo Morales, and José María Cañas Plaza.. 2019. Control system in open-source FPGA for a self-balancing robot. *Electronics* 8, 2 (2019), 198.

[8] L Vachhani, Arun D Mahindrakar, and K Sridharan. 2010. Mobile robot navigation through a hardware-efficient implementation for control-law-based construction of generalized Voronoi diagram. *IEEE/ASME Transactions on Mechatronics* 16, 6 (2010), 1083–1095.

[9] Agnès Ghorbel, Nader Ben Amor, and Mohamed Jallouli. 2020. Design of a flexible reconfigurable mobile robot localization system using FPGA technology. *SN Applied Sciences* 2, 7 (2020), 1183.

[10] M Deepan Raj, I Gogul, M Thangaraja, and V Sathiesh Kumar. 2017. Static gesture recognition based precise positioning of 5-DOF robotic arm using FPGA. In *2017 Trends in Industrial Measurement and Automation (TIMA)*. IEEE, 1–6.

[11] Zhaorui Zhang, Yao Xin, Benben Liu, Will XY Li, Kit-Hang Lee, Chun-Fai Ng, Danail Stoyanov, Ray CC Cheung, and Ka-Wai Kwok. 2016. FPGA-Based high-Performance collision Detection: an enabling Technique for image-guided robotic surgery. *Frontiers in Robotics and AI* 3 (2016), 51.

[12] C Chandra Mouli, P Jyothi, K Nagabhushan Raju, and C Nagaraja. 2013. Design and implementation of robot arm control using labview and arm controller. *IOSR Journal of Electrical and Electronics Engineering* 6, 5 (2013), 80–84.

[13] Ayman A. Nada, Victor Parque, and Mona A. Bayoumi. 2023. Accelerating the Performance of Fuzzy-FPGA Based Control in LabVIEW for Trajectory Tracking Problems. *IFAC-PapersOnLine* 56, 2 (2023), 3386–3391. doi:10.1016/j.ifacol.2023.10.1486 22nd IFAC World Congress.

[14] Ibrahim Abdel-Hady, Mona A Bayoumi, Nader A Mansour, and Ayman A Nada. 2024. Stabilization of Driving Velocity Constraints for Self-balanced Robot. In *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2354–2359.

[15] Ahmed A Shabana. 2020. *Dynamics of multibody systems.* Cambridge university press.

[16] User Manual National Inst. Co. available: https://www.ni.com/en-lb/support/model.sbrio-9631.html.

[17] Ayman A Nada and Abdullateef H Bashiri. 2017. Selective generalized coordinates partitioning method for multibody systems with non-holonomic constraints. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 58202. American Society of Mechanical Engineers, V006T10A004.

[18] Ayman A Nada and Abdullatif H Bishiri. 2023. Multibody system design based on reference dynamic characteristics: gyroscopic system paradigm. *Mechanics Based Design of Structures and Machines* 51, 6 (2023), 3372–3394.

[19] Minh-Tai Vo, Hoai-Nghia Duong, Vinh-Hao Nguyen, et al. 2023. Combining passivity-based control and linear quadratic regulator to control a rotary inverted pendulum. *Journal of Robotics and Control (JRC)* 4, 4 (2023), 479–490.

[20] Camilo Andres Manrique Escobar, Carmine Maria Pappalardo, and Domenico Guida. 2020. A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole. *Applied Sciences* 10, 24 (2020), 9013.

[21] Jian Huang, Mengshi Zhang, and Toshio Fukuda. 2023. *Robust and Intelligent Control of a Typical Underactuated Robot: Mobile Wheeled Inverted Pendulum.* Springer Nature.

[22] IJ Nagrath and Madan Gopal. 2025. *Control Systems Engineering*. New Age International Pvt Ltd.

[23] R.C. Dorf and R.H. Bishop. 2022. *Modern Control Systems (Fourteenth Edition, Global Edition).* Pearson.