Journal of Robotics and Control (JRC)
Volume 6, Issue 3, 2025
ISSN: 2715-5072, DOI: 10.18196/jrc.v6i3.26260

Utilizing FPGAs for Real Time Control and Stabilization of Multibody Mechatronic Systems

Ibrahim Abdel-Hady [1], Mona A. Bayoumi [2], Nader A. Mansour [3], Ayman A. Nada [4]
[1, 2, 3] Benha Faculty of Engineering, Benha University, 13512 Benha, Egypt
[3] Egypt University of Informatics (EUI), New Administrative Capital, Cairo, Egypt
[4] Department of Robotics and Mechatronics, Egypt-Japan University of Science and Technology E-JUST, Alexandria 21934, Egypt
Email: [1] ibrahim.abdulhadi@bhit.bu.edu.eg, [2] mona.elawa@bhit.bu.edu.eg, [3] nader.mansour@eui.edu.eg, [4] ayman.nada@ejust.edu.eg

***Abstract*—This work aims to design and implement an FPGA-based embedded controller for multibody mechatronic systems. The application considered is a wheeled self-balancing robot. The main objective is to stabilize the inverted body of the robot in its vertical position. It is done by precisely driving its wheels forward and reverse direction until stabilization occurs within a small distance. Using the multibody dynamics approach for the control system design of such systems is a challenging task due to the resulting differential nonlinear algebraic equations. In this article, development, verification, and simulation were done for the resulting model. The bumgrate stabilization method was used with the parameter $\alpha = \beta = 10$ shows acceptable violations of $\pm 10^{-6}$ and $\pm 10^{-4}$ for both holonomic and nonholonomic constraints, respectively, during the dynamic equations solution. Next, we designed and simulated optimal feedback controllers and classical PID controllers for both linear and nonlinear multibody models. In addition, our testing of the digital PID controller on an FPGA shows that the steady-state error for successful stabilization is around ± 0.2 degrees, and it takes 2 seconds for the zero-tilt angle setpoint to settle. Finally, we implemented a PID controller on the NI-SBRIO 9631 with a $266MHz$ real-time processor, and a $40MHz$ Xilinx FPGA target. Using such RIO board enables the rapid development of such control systems. The results of this implementation reveals that the controller is able to stabilize the robot in the range of the tilt angles $\theta^3 = \pm 15^\circ$ due to the DC motors torque specifications. Furthermore, the paper proves the effectiveness of using the coordinate partitioning method for the state-space formulation of such under-actuated nonlinear systems.**

***Keywords*—Multibody Dynamics; Optimal Feedback; Pid Control; Real-Time Embedded Control; Under-Actuated Systems**.

I. Introduction

A. Background

Reconfigurable field programmable gate arrays adoption are growing rapidly in many industrial [1], [2], embedded, and control mechatronic applications [3], [4]. There are many applications utilized FPGAs in the fields of robotics [5], [6], mobile robots [7]–[9], autonomous robot navigation [10], [11], serial manipulators [12], [13], design and control of mechatronic systems [14] [15] [16], and energy management systems [17], [18],real-time control systems [2], [3], [14], [19]–[23], and deep learning accelerators built-in hardware [24]–[27]. In the field of control engineering, developing FPGAs is of great importance in completing robotic control systems. Furthermore, it was found that model-based control methods are the most efficient and cost-effective. This model must interpret how the system's multiple parts move relative to each other. So, the reconfigurable input/output (RIO) board such as myRIO kit, SbRIO, or cRIO, which includes an FPGA target, is well-suited for implementation of the control design for the real-time system, so that it can achieve the required trajectory [20], [28], [29]. Also, it is well suited for fast development and prototyping in the control of micro-robotic systems [4], as the NI-myRIO based FPGA was used for that purpose. Also, it is used for hardware in loop simulations as in [30].

FPGAs have long been within the circle of engineers, developers, and communication researchers with high expertise in VHDL or verilog, implementing the digital signal processor algorithms on FPGAs. That requires large efforts due to the complexities of programming FPGAs. Graphical languages such as the LabVIEW FPGA module help non-experienced engineers with the stated languages to easily design, test, and deploy algorithms and controllers on these FPGA targets. Features included in this module replace thousands of script-like written VHDL commands. This helps in the fast development of the self-balancing robot real-time control systems instead of using complex VHDL-based subroutines [3], [29], [31], [32].

Regarding labview based-programming with RIO FPGA devices with embedded controllers,the author in [33] discussed the different methodologies for building controllers on these devices. Furthermore, he discussed the floating-point (FP) and fixed-point (FXP) considerations on the design of digital controllers on FPGAs. Additionally, he explained the design cycle for building a controller on FPGA target. Beside this, the author highlighted the importance of resource saving by suggesting of utilization of BRAMs to reduce resources consumption on the

target.

On the other side, for the implementation of accurate control systems on FPGAs, the multibody dynamics (MBDs) approach is well-suited for building the under-actuated system model in a general formulation. As it is widely used for modelling and simulation of the mechanical systems that have been recognized as a cornerstone in the dynamic analysis [34]–[37], design [36], [38]–[40], control [35], [41]–[43], mechanical system parameters identification [36], [44]–[46] and integration with machine learning based computer simulation [47]–[49]. MBDs result in nonlinear differential-algebraic constraint equations of index-1 that govern the complex physical systems' motion [50]–[52]. The resulting model in the augmented formulation as in (1). Where $\mathbf{M}$ is the system mass matrix, $\mathbf{C}_q$ is the constraints function Jacobian matrix, $\ddot{\mathbf{q}}$ is the generalized acceleration vector, λ lagrange multipliers associated to each constraint equation in the constraint function $\mathbf{C}$. Also, $\mathbf{Q}$ is the total external force vector and $\mathbf{Q}_d$ is the quadratic velocity vector.

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_q^T \\ \mathbf{C}_q & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{Q}_d \end{bmatrix} \tag{1}$$

Numerical difficulties may arise when solving the dynamic equations of motion, especially in the existence of nonholonomic constraints. One of these interesting problems is the constraint instability during the solution of the general augmented EOMs [53]. As it doesn't explicitly use the constraints at both position $\mathcal{C}$ and velocity level $\dot{\mathcal{C}}$, i.e. they are not satisfied during the solution (numerical integration process) to obtain positions and velocities of the generalized system coordinates. This leads to large errors during long simulation time, especially in bad initial conditions of this integration process [37], [54]. Consequently, Methods to control and eliminate these violations should exist. Different methods were adopted, namely could be implemented. The first one is the direct correction algorithms like in [55], such as the Newton–Raphson method (post stabilization). These methods were stated and used (some of them) in the literature [56]–[58]. In this method, at each integration step, the constraints function is satisfied only at the acceleration level and that doesn't confirm that this function is satisfied at velocity and position levels. so that, errors may occur. Thus correction is made at velocity and position levels to get them back to their zero state before the next integration step. This is done iteratively using newton's differences to update the coordinates at position level C. and is done at one step at velocity level $\dot{C}$. On using this method at the existence of constraints at velocity level,the constraint violation in velocities is eliminated with only a single step. This may cause errors, and the constraint equations violates. for that, the approach is not suited for both moderate and long simulations [56], [57].

The second method, coordinate partitioning method, at which the generalized coordinates are grouped into dependent and independent. This is done at position, velocity, and acceleration levels. The equations of motion are integrated only for independent coordinates. Then solving the non-linear constraints vector using the obtained independent coordinates iteratively using Newton–Raphson algorithm to have the dependent coordinates. On finding the total coordinates, The dependent velocities are obtained. Then, finally, solve the whole set of equations of motion to get the total acceleration Vector. The number of independent coordinates reflects the system DOFs. In such method, the violations are eliminated since all the constraints at the coordinate, velocity, and acceleration levels are solved. Its accuracy relies on the selection of the independent and dependent coordinates. Which is considered one obvious disadvantage of that method [56], [59]–[61].

The third method is the constraint stabilization control methods such as the baumgarte stabilization method stated in [62], [63], penalty method presented in [64]and the PID method [65].

Constraint stabilization methods are widely addressed due to their simplicity in the computational formulation and implementation. Although their major drawback is the method in choosing the stabilization parameters that leads to the simulation failure, it is considered in this article. Due to the simplicity of the implementation and the suitability for controlling both holonomic and nonholonomic constraints stabilization, baumgarte stabilization method will be utilized in this article [56].

In this context we state the different modern and classic control systems for control of self-balancing robots. Either two wheels or unicycle robots such as [66], a nonlinear $\mathbf{H}_\infty$ controller was designed and applied for two-wheeled self-balanced vehicles (velocity tilting angle). In [67], the robot model based on lagrangian dynamics was addressed. Then, they designed a linear feedback controller for the self-balancing robot with cRio. In [68], they designed a new approach for nonlinear $\mathbf{H}_\infty$ controller. While in the thesis [28], it developed the self-balancing robot classic PID controller based on the myRio device. Also, in [69], they proposed an adaptive controller for two wheels self-balancing robot combining SMC and neural network. While in [7], the two-wheeled self- balancing robot (TWSBR) was controlled to avoid obstacles using fuzzy that was built using RISC-V (an open-source instruction set architecture (ISA) and free to use on FPGA). Another research article, [8], used open source FPGA tools, such as IceStudio and the IceZum Alhambra board. Going to another type of self-balancing segway such as unicycle robots which uses a balancing and another driving mechanisms in [70], it designed a robust $\mathbf{H}_2$ controller and applied to a reaction wheel unicycle robot. In [71], authores redesigned what was done in [70] and applied PID, LQR, and SMC controllers. Similarly, in [72] and [73] which designed and implemented a PID, fuzzy and LQR control for the wheeled balancing robot in order to keep it stable when subjected to uncertainties and different heights. Furthermore, the Dynamic modelling and characteristics analysis of lateral-pendulum unicycle robot were done in [74].while, [75] designed

a novel adaptive interval type-2 fuzzy controller (AIT2FC) and it was used for a single-wheel vehicle (SWV), who successfully implemented a single-wheel vehicle (SWV) based on a novel adaptive interval type-2 fuzzy control system. The efficiency of the (AIT2FC) was proved by real-time control of SWVs control problem. In, [76], it used labview with arduino test rig to test both a lead-lad compensator and fuzzy logic controller. For recent optimization case, this article in [77] addressed the optimization ofthe sliding mode control parameters using the firefly algorithm.

B. Problem Statement

The main problem is to design and simulate a fast controller based on the multibody model for such under-actuated mechatronic system. The validation of the multibody system dynamics is done through solving the nonlinear differential equations in (1) This encounters problems due to constraints violations during the integration process. The other side of the problem is that the self-balancing robots need fast control actions for fast stabilization at the preset balancing point.

C. Proposed Solution

In this article, Baumgarte stabilization was considered as a PD controller. At each integration step, it was utilized to get the constraints back to their manifolds. thhere are two factors that are multiplied by the constraints function at position and velocity levels and summed to the quadratic velocity vector, $\mathbf{Q}_d$, associated with these constraints to bring back the constraints function at the acceleration level $\ddot{C}$ back to its manifold. That method keeps violations controlled to their lowest level [56], [63]. Beside this method, the coordinate partitioning was used to formulate independent coordinate acceleration to obtain the system equations of motion [62]. Furthermore, well-designed controllers lead to perfect stabilization. Classic PID and optimal state feedback controllers are well suited for the under-actuated systems. The proposed hardware controller is NI-sbrio 9631 RIO board that include FPGA target is used for the purpose of control due to their higher performance and accuracy over other embedded controllers [9].

D. Contribution

The research contribution lies in the formulation of a general multibody dynamic model with both holonomic and non-holonomic constraint stabilization for the self-balancing robot (SBR). Then, the coordinate partitioning method was used to build the linearized system state space model for the planar 2D multibody model. This is the last part: designing and simulating the state-feedback optimal controller, focusing on pole placement and linear quadratic regulator (LQR) controllers. We successfully designed and implemented a PID controller for robot stabilization in a vertical position. The Sbrio-Fpga was utilized to construct the embedded-based real-time PID controller subroutines for this mechatronic system (SBR).

E. Organization of the Article

This section gives a summary of the methods used in this article. The methods section begins with, the subsection II-A, at which each term in the multibody model dynamics equation of motion in (1) is explained in a detailed manner. Starting with robot description, selection of generalized coordinates, formulation of holonomic and nonholonomic constraints, mass matrix, the quadratic velocity vector, and the external forces for the under-actuated systems. After that, the coordinate partitioning method was used to address model-based controlled design in the next sections. Following this, the multibody model simulation and stabilization for both the augmented and coordinate partitioning methods according to the flow chart in Fig. 3 were implemented. Then, the DC motor model with its estimated parameters was included to fully define the robot model. At the end of this section, the state space model was obtained.

In this subsection, subsection II-B, the controller design was explained. Two-state feedback, pole placement and LQR, control methods were designed for both the linearized and nonlinear models obtained in the subsection II-A. Also, model modification was done for friction and damping that may exist due to real hardware joints. Later in this section, we designed a PID controller and its corresponding discrete transfer function for digital implementation. Both subsections, subsection II-C and II-D explained the real hardware implementation of the proposed PID controller on the Sbrio 9631 FPGA RIO platform. Finally, section III briefly discussed the obtained results, followed by, section IV that discussed the conclusion and the expected future work related to the addressed topic.

II. METHODS

A. Self-Balancing Robot Dynamic Model

1) Robot Description:

In this section, We state the full steps applied to build the wheeled balancing robot multibody model. The self-balancing robot structure, is shown in Fig. 1. It mainly consists of the following components:

- **Robot Wheels**: For planar modeling, we only consider one wheel for the robot's movement and stabilization. Frame 2 defines the robot wheel.
- **The inverted Body**: The wheel is attached to this vertical component through the revolute joint, which rises upwards from the robot base. It acts as an inverted pendulum, which is the common configuration for self-balancing robots. Typically, the tilt sensor, such as the utilized ADXL335 analog accelerometer sensor, monitors the pendulum's angular po-

sition to help the robot maintain its balance. This body is defined by frame 3.

- **Ground**: The global fixed coordinate is defined by XY frame.
- **Key assumptions considered for the system planar MBDs Model** :
  1) The robot wheel and the inverted mass are rigid bodies.
  2) The mechanical and electrical system losses are initially set to zero.
  3) The wheels and the surface are in constant contact. They roll without skidding or slipping.
  4) The inverted body inclination angle from equilibrium is small for linearization.

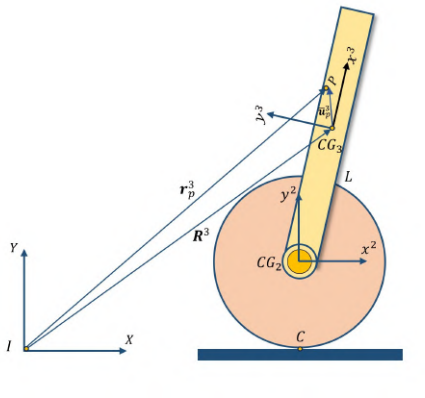

Fig. 1. Self-balancing Robot Generalized Coordinates

2) Multibody Dynamics Model (MBDs):

The generalized coordinates vector for planar MBDs can be selected and written as,

$$\mathbf{q}^T = \begin{bmatrix} \mathbf{q}^{2^T} & \mathbf{q}^{3^T} \end{bmatrix}^T \tag{2}$$

Where, $\mathbf{q}^i = \begin{bmatrix} \mathbf{R}^{i^T} & \theta^{i^T} \end{bmatrix}^T = \begin{bmatrix} R_x^i & R_y^i & \theta^i \end{bmatrix}^T$ is the generalized coordinates of the body i,$i = 1, 2$. $\mathbf{R}^i$ is the body i centre of gravity position defined in the global coordinate system $\mathbf{XY}$ and θ^i is body i orientation about the vector normal to the $\mathbf{XY}$ plane. The coordinate system defined by $(\mathbf{x}^2, \mathbf{y}^2)$ is fixed to the wheel and this frames rotates with the wheel. Similarly for the pendulum coordinate system. It is defined by $(\mathbf{x}^3, \mathbf{y}^3)$ and this frame moves with this body. The derived system equations of motions in (1), is extended for systems that include both holonomic and nonholonomic constraints to be,as in [78], [79],

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_q^{h^T} & \dot{\mathbf{C}}_{\dot{\mathbf{q}}}^{nh^T} \\ \mathbf{C}_q^h & \mathbf{0} & \mathbf{0} \\ \dot{\mathbf{C}}_{\dot{\mathbf{q}}}^{nh} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda^h \\ \lambda^{nh} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{Q}_d^h \\ \mathbf{Q}_d^{nh} \end{bmatrix} \tag{3}$$

Where ,$\mathbf{M}$ is the assembled robot mass matrix such that $\mathbf{M} = diagonal\left(\mathbf{M}^i\right)$ with each $\mathbf{M}^i = diagonal(m^i, m^i, I_z^i)$. The matrix $\mathbf{C_q}$ defines the Jacobian matrix and, λ is the vector of lagrange multipliers. The quadritic vector, $\mathbf{Q}_d$, absorbs all quadratic terms of velocity derived fro the second derivatives by time of the constraints function. $\mathbf{I_z^i}$ is the body i mass moment of inertia about the vector $\mathbf{Z}$ passing through the body centre of gravity and normal to $\mathbf{XY}$ plane. Note that the superscript h and nh are used through the article refer to holonomic and nonholonomic respectively.

a) Robot Holonomic Constraints: as shown in Fig. 1, the joint connecting robot wheel and the pendulum is defined by one revolute Joint. That is defined in the constraint vector $\mathbf{C}^{h(R)}$. Also, the wheel must be constrained to be in contact with the ground at point C defined by a contact constraint. At which robot wheel radius R_w is always the vertical position of wheel centre of gravity ($R_y^2 = R_w$). For the nonholonomic constraints, there exist only the wheel pure rolling constraint as it should roll without sliding in the direction of motion. It can be defined by $\mathbf{C}^{nh}$.

$$\mathbf{C}^{h(R)} = \begin{bmatrix} \left(\mathbf{r}^{2CG_2} - \mathbf{r}^{3CG_2}\right)^T & \mathbf{h_j}^T \mathbf{r}^{2C} \end{bmatrix}^T \tag{4}$$

Where,

$$\mathbf{h_j}^T = \begin{bmatrix} 0 & 1 \end{bmatrix}, \; and \;\; \mathbf{h_i}^T = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

and the global position vector of the point C is defined by,

$$\mathbf{r}^{2C} = \mathbf{R}^2 + \mathbf{A}^2 \bar{\mathbf{u}}_C^2$$

Where, $\bar{\mathbf{u}}_C^2$ defines the local position vector of point C defined in frame 2.

$$\bar{\mathbf{u}}_C^2 = \begin{bmatrix} -R_w sin(\theta^2) \\ -R_w cos(\theta^2) \end{bmatrix}, \; \mathbf{A}^i = \begin{bmatrix} cos(\theta^i) & -sin(\theta^i) \\ sin(\theta^i) & cos(\theta^i) \end{bmatrix}^T$$

Similarly for the position vectors for the revolute joint common point between the two bodies CG_2. They are defined in frame 2 and 3 by,

$$\mathbf{r}^{2CG2} = \mathbf{R}^2 + \mathbf{A}^2 \bar{\mathbf{u}}_{CG2}^2 \tag{5}$$

$$\mathbf{r}^{3CG2} = \mathbf{R}^3 + \mathbf{A}^3 \bar{\mathbf{u}}_{CG2}^3 \tag{6}$$

and the local position of point CG_2 in both stated frames as,

$$\bar{\mathbf{u}}_{CG2}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; \bar{\mathbf{u}}_{CG2}^3 = \begin{bmatrix} -L \\ 0 \end{bmatrix}^T$$

Where, L is the position of the pendulum centre of gravity from the wheels axis.

b) Pure Rolling Non-holonomic Constraint $\mathbf{C}^{nh}$*:*

$$\mathbf{C}^{nh} = \begin{bmatrix} \mathbf{h_i}^T \dot{\mathbf{r}}^{2C} \end{bmatrix} \tag{7}$$

and the global velocity vector is defined by,

$$\dot{\mathbf{r}}^{2C} = \dot{\mathbf{R}}^2 + \dot{\mathbf{A}}^2 \bar{\mathbf{u}}_C^2$$

After simplification, the total assembled constraints vector is,

$$\mathbf{C}^{Total} = \begin{bmatrix} R_x^2 - R_x^3 + Lcos(\theta^3) \\ R_y^2 - R_y^3 + Lsin(\theta^3) \\ R_y^2 - R_w \\ \dot{R}_x^2 - R_w\dot{\theta}^2 \end{bmatrix} = \mathbf{0} \tag{8}$$

It is obvious that system degree of freedom is two, and is calculated as the following equation,

$$N_{DOF} = 3N_B - N_C \tag{9}$$

Where, N_B is the number of bodies = 2, and N_C is the number of holonomic constraints equations = 4 then $N_{DOF} = 2$ which are the position R_x^2 and tilt angle θ^3. Note the fact that, the stated pure rolling constraint leads to a holonomic constrain that in embedded (hidden) in the nonholonmic constraint equation ($R_x^2 - R_w\theta^2$) which is an integrable equation (just for planar case). But on dealing with spatial system as in [54], it is fully nonholonomic.

c) Holonomic Jacobian Matrix $\mathbf{C}_q^h$:

$$\mathbf{C}_q^h = \frac{\partial \mathbf{C}^h}{\partial q} \tag{10}$$

$$\mathbf{C}_q^h = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & Lsin(\theta^3) \\ 0 & 1 & 0 & 0 & 1 & Lcos(\theta^3) \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

d) Nonholonomic Jacobian Matrix $\mathbf{C}_{\dot{q}}^{nh}$:

$$\mathbf{C}_{\dot{q}}^{nh} = \frac{\partial \mathbf{C}^{nh}}{\partial \dot{q}} \tag{12}$$

$$\mathbf{C}_q^{nh} = \begin{bmatrix} 1 & 0 & R_w & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

Refering to (3), The system mass matrix was found as,

$$\mathbf{M} = \begin{bmatrix} m^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & m^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{zz}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & m^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & m^3 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz}^3 \end{bmatrix} \tag{14}$$

Recall, from (3) the holonomic quadratic $\mathbf{Q}_d^h$ is the vector that absorbs all quadratic terms of velocity associated with holonomic constraints was found as follows [37],

$$\mathbf{C}^h(\mathbf{q},t) = \mathbf{0} \tag{15}$$

$$\dot{\mathbf{C}}^h(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{C}_{\mathbf{q}}^h\dot{\mathbf{q}} + \mathbf{C}_t^h = \mathbf{0} \tag{16}$$

$$\ddot{\mathbf{C}}^h(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{C}_{\mathbf{q}}^h\ddot{\mathbf{q}} + \left(\mathbf{C}_{\mathbf{q}}^{\mathbf{h}}\dot{\mathbf{q}}\right)_q + 2\mathbf{C}_{\mathbf{qt}}^h\dot{\mathbf{q}} + \mathbf{C}_{tt}^h = \mathbf{0} \tag{17}$$

$$\mathbf{Q}_d^h = -\left(\mathbf{C}_{\mathbf{q}}^h\dot{\mathbf{q}}\right)_{\mathbf{q}}\dot{\mathbf{q}} - 2\mathbf{C}_{\mathbf{q}t}^h\dot{\mathbf{q}} - \mathbf{C}_{tt}^h. \tag{18}$$

$$\mathbf{Q}_d^h = \begin{bmatrix} L\dot{\theta^3}cos(\theta^3) \\ L\dot{\theta^3}sin(\theta^3) \\ 0 \end{bmatrix} \tag{19}$$

In the same manner, $\mathbf{Q}_d^{nh}$ is the vector that absorbs all quadratic terms of velocity associated with nonholonomic constraints,

$$\dot{\mathbf{C}}^{nh}(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{H}(\mathbf{q},\dot{\mathbf{q}},\mathbf{t}) + \mathbf{g}(\mathbf{q},\mathbf{t}) = \mathbf{0} \tag{20}$$

$$\ddot{\mathbf{C}}^{nh}(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{H}\ddot{\mathbf{q}} + (\mathbf{H}\dot{\mathbf{q}})_q\dot{\mathbf{q}} + (\mathbf{g}_q + \mathbf{H}_t)\dot{\mathbf{q}} + \mathbf{g}_t = \mathbf{0}$$

$$\mathbf{Q}_d^{nh} = -\left(\mathbf{H}\dot{\mathbf{q}}\right)_{\mathbf{q}}\dot{\mathbf{q}} - \left(\mathbf{g}_{\mathbf{q}} + \mathbf{H}_t\right)\dot{\mathbf{q}} - \mathbf{g}_t. \tag{21}$$

$$\mathbf{Q}_d^{nh} = 0 \tag{22}$$

e) Total External Forces $\mathbf{Q}_{ex}$: Based on virtual Work Principal as in [3], [52], the external forces are,

$$\delta\mathbf{W}_{ex}^i = \mathbf{Q}_{ex}^{i^T}\delta\mathbf{q}^i \tag{23}$$

$$\delta\mathbf{W}_{ex}^2 = -m^2g\delta R_y^2 \tag{24}$$

$$\delta\mathbf{W}_{ex}^3 = -m^3g\delta R_y^3 \tag{25}$$

$$\mathbf{Q} = \begin{bmatrix} 0 \\ -m^2g \\ 0 \\ 0 \\ -m^3g \\ 0 \end{bmatrix} \tag{26}$$

3) Coordinate Partitioning and Linearization of the Planar MultiBody Model:

The coordinate partitioning method explained in [35], [51], [52], [61], is used to get the robot independent coordinates to facilitate the state-space representation model, the generalized coordinates could be partitioned into independent, $\mathbf{q}_i$, and dependent, $\mathbf{q}_d$, coordinates as,

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_d & \mathbf{q}_i \end{bmatrix}^T \tag{27}$$

Both coordinates are defined respectively,

$$\mathbf{q}_d = \begin{bmatrix} R_y^2 & \theta^2 & R_x^3 & R_y^3 \end{bmatrix}^T \tag{28}$$

$$\mathbf{q}_i = \begin{bmatrix} R_x^2 & \theta^3 \end{bmatrix}^T \tag{29}$$

Now, the Jacobian matrix should be also partitioned into dependent and independent Jacobian matrices $\mathbf{C}_{q_d}$, $\mathbf{C}_{q_i}$,

$$\mathbf{C}_{q_d} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & R_w & 0 & 0 \end{bmatrix} \tag{30}$$

$$\mathbf{C}_{q_i} = \begin{bmatrix} 1 & -Lsin(\theta^3) \\ 0 & Lcos(\theta^3) \\ 0 & 0 \\ 1 & R_w \end{bmatrix} \tag{31}$$

Now for finding the transformation matrices Calculate the following Matrix,that are used to eliminate Lagrange multipliers by

$$\mathbf{C}_{di} = -\mathbf{C}_{q_d}^+\mathbf{C}_{q_i} \tag{32}$$

where, $\mathbf{C}_{q_d}^{+}$ is Moore-Penrose pseudo inverse matrix of the dependent Jacobian Matrix. Thus, the transformation matrix $\mathbf{C}_{di}$ must be modified to accommodate with the generalized coordinates by addition of a unity matrix of size 2×2 to be $\mathbf{B_i}$

$$\mathbf{B}_i = \begin{bmatrix} \mathbf{C}_{di} \\ \mathbf{I}_{2\times 2} \end{bmatrix} \tag{33}$$

$$\mathbf{B}_i = \begin{bmatrix} 0 & 0 \\ -1/R_w & 0 \\ 1 & -Lsin(\theta^3) \\ 0 & Lcos(\theta^3) \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{34}$$

Use the transformation $\mathbf{B_i}$ to get the independent mass matrix $\bar{\mathbf{M}}$ according to the partitioning coordinates. By rearranging the system mass matrix in (14) then solve the following,

$$\bar{\mathbf{M}} = \mathbf{B}_i^T \mathbf{M} \mathbf{B}_i \tag{35}$$

$$\bar{\mathbf{M}} = \begin{bmatrix} m^2 + m^3 + I_{zz}^3/R_W^2 & -Lm^3 sin(\theta^3) \\ -Lm^3 sin(\theta^3) & L^2 m^3 + I_{zz}^3 \end{bmatrix} \tag{36}$$

Similarly, the transformed force matrix is $\bar{\mathbf{Q}}_B$

$$\bar{\mathbf{Q}}_B = \mathbf{B}_i^T \mathbf{Q} - \mathbf{B}_i^T \mathbf{M} \gamma \tag{37}$$

Where,

$$\gamma = \begin{bmatrix} \mathbf{c}_d \\ 0 \\ 0 \end{bmatrix} \tag{38}$$

$$\mathbf{C}_d = \mathbf{C}_{q_d}^{+} \mathbf{Q}_d \tag{39}$$

Note that the Lagrange multipliers vector λ was eliminated on using the transformation matrix $\mathbf{B}_i$. Now,

$$\ddot{\mathbf{q}}_i = \bar{\mathbf{M}}^{+} \bar{\mathbf{Q}}_B \tag{40}$$

Where,

$$\ddot{\mathbf{q}}_i = \begin{bmatrix} \ddot{\mathbf{R}}_x^2 \\ \ddot{\theta}^3 \end{bmatrix}$$

4) Constraints Stabilization Using Baumgarte method:

As discussed in the introduction, the second-order equations of motion along with the augmented, $\ddot{C}(q, \dot{q}, t)$, constraints are unstable. Due to this, small errors/perturbations arising from numerical errors that introduced by the integration process cannot be corrected naturally, and they only tend to increase with time. Baumgarte introduced a feedback controller that is used to put the system violations under control if they occur on the position or velocity constraint equations [62], [63]. That method brings $\ddot{C}$ back to their manifolds by,

$$\ddot{C} + 2\alpha \dot{C} + \beta^2 C^h = 0 \tag{41}$$

Where, the constants, α and β, are positive. They are the weights used to damp out the violations in the constraints defined at both position and velocity levels. And play the great role of control terms. As stated that in the background in section I-A and in [56], [62] that the utilized method keeps the violations (errors) under control. These constants may be in the range of $(1\ to\ 10)$. It needs trial and errors to decide the best selection of the $\alpha$and β parameters.The first good choice of these values is α= β= 1 for multibody systems consisting of rigid bodies that converge the constraints without oscillation. The use of the Baumgarte stabilization method is carried out simply in the computational subroutines by employing,

$$Q_d - C = 0 \tag{42}$$

this leads to

$$Q_d - (2\alpha \dot{C} + \beta^2 C) = 0 \tag{43}$$

and by direct substitution of (43) in the system equations of motion in (3) instead of using Q_d only during the numerical integration process. Till Now, by Putting the system of nonlinear equations in the form of states. Following that, by solving the model for the specified states. The state equation is defined as,

$$\dot{\mathbf{X}}(t) = \mathbf{f}(\mathbf{x(t)}, \mathbf{u(t)}) = \ \mathbf{0} \tag{44}$$

where, the system states are the wheel displacement, R_x^2, and its time derivative $\dot{R}_x^2$. the pendulum tilt angle,θ^3 and its derivative $\dot{\theta}^3$. they are listed by,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} R_x^2 \\ \theta^3 \\ \dot{R}_x^2 \\ \dot{\theta}^3 \end{bmatrix} \tag{45}$$

Validation of the SBR nonlinear model for both the full generalized coordinates augmented formulation technique and the coordinate partitioning technique was done through the simulation. By using MATLAB/functions and mathscripts along with the common integrator ode45 with predetermined tolerance to simulate both models with nonlinear dynamics. Table I include the model parameters used in the simulation. While the chart in Fig. 2 explains the solution of the multbody model along with using Bumgrate violations stabilization method.

TABLE I. ROBOT SPECIFICATIONS

Parameter	Data	Description
m^2	$0.25\ kg$	wheel mass
m^3	$2.045\ kg$	pendillum mass
L	$0.027\ m$	pendillum CG. position
I_{zz}^2	$0.00045\ kg.m^2$	wheel mass moment of inertia
I_{zz}^3	$0.0061602\ kg.m^2$	pendilum mass moment of inertia
R_w	$0.06\ m$	wheel radius
g	$9.81\ m/s^2$	gravitational constant

At the start, proper robot initial configuration $\mathbf{q}_0$ and $\dot{\mathbf{q}}_0$ are selected to avoid violations at the start and to ensure accurate and fast convergence during integration process. Also, at this step, the initial and final simulation times with the

solution step $\delta t = 0.001sec$ are defined. After that, system mass matrix, jacobian matrices and the quadratic vectors for both holonomic and nonholonomic constrains are defined in the form of functions in MATLAB to be called at each step of time. Following that the solution of (3) at each step of time is done for the coordinates accelerations $\ddot{\mathbf{q}}$ and lagrange multiplies vectors $\lambda^h$and λ^{nh}. After that integration is done using ode45 function for the coordinates at velocity $\dot{\mathbf{q}}$ and position $\mathbf{q}$ levels.

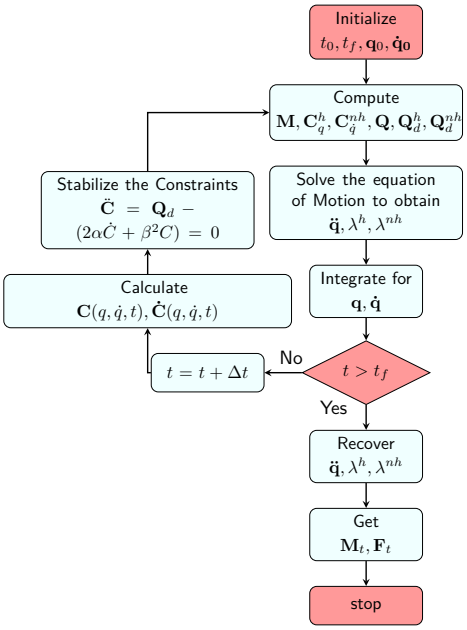

Fig. 2. Multibody Model Simulation With Baumgrate Stabilization Method Flow Chart

At each integration step, Bumgrate stabilization defined in (43) is calculated to control the constraints violations. At the end of simulation time, the coordinates accelerations are obtained by direct substitution of the obtained coordinates at both postion and velocity levels during the solution. Additionally, the reaction forces and moment can be calculated from lagrange multipliers vectors [52], [54].

The results from the solution of the nonlinear dynamic equations are shown in Fig. 3, Fig. 5, Fig. 4 and Fig. 6 respectively. On using the initial conditions defined in (46), the pendulum falls from its preset initial position and keeps swinging about the stable equilibrium position($\theta^3 = \pi/2$) as shown in Fig. 3. As a consequence, the wheel also keeps oscillating and its position varies nearly in the range of [-0.114 :-0.06], as shown in Fig. 5, under the effect of IP momentum as we neglect the friction effect on the robot wheels. Due to the pure rolling constrains set before.

Furthermore, Fig. 4 and Fig. 6 show the angular and linear velocities of the inverted body and robot wheel respectively. The angular velocity starts from zero and varies in the range of ± 17.5 rads/s while the wheel linear velocity varies in the range of ± 0.42 m/s.

$$\mathbf{x_0} = \begin{bmatrix} -0.0868 & (\pi/2) + (\pi/18) & 0 & 0 \end{bmatrix} \quad (46)$$

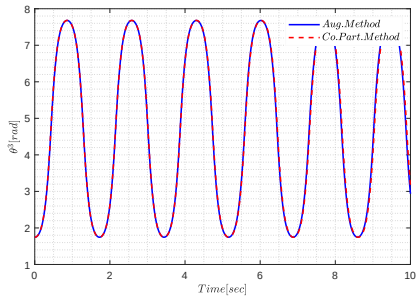

Fig. 3. The Inverted Pendulum Tilt Angle θ^3

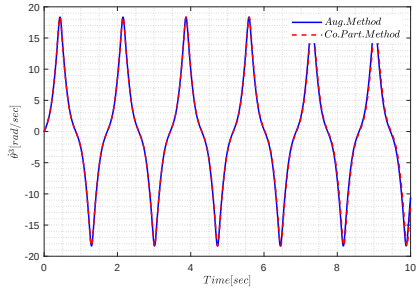

Fig. 4. inverted pendulum angular velocity $\dot{\theta}^3$

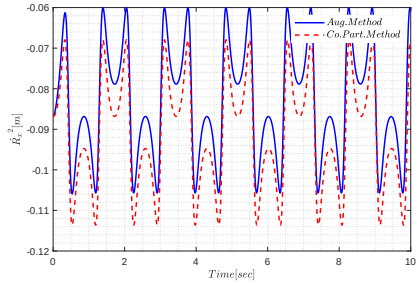

Fig. 5. Robot wheel position R_x^2

Also, Fig. 7 shows the holonomic constraints functions error C^h during the simulation time t. As indicated, this error is initially relatively large of a value -9.5×10^{-3} and the effectiveness of using baumgrate stabilization. it damp this error to be within the range of $\pm 2 \times 10^{-6}$. While Fig. 8 refer to the violation in the nonholonomic constraints. it keeps the error within the range of $\pm 10^{-4}$ for C^{nh}. Which both are acceptable ranges for the simulation of the dynamic response [52] of the uncontrolled Self balancing robot planar model.

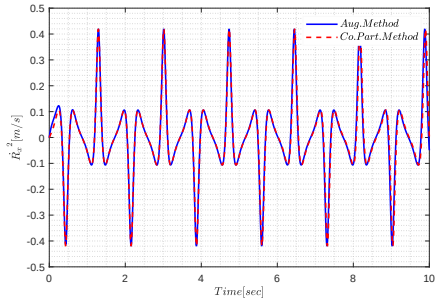

Fig. 6. Robot wheel linear velocity $\dot{R}_x^2$

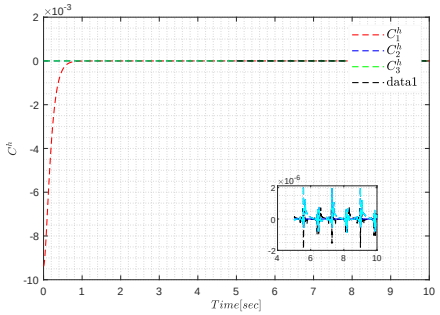

Fig. 7. Holonomic constraints violation C^h

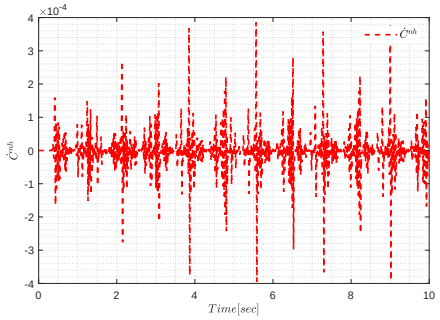

Fig. 8. Nonholonomic constraints violation $\dot{C}^{nh}$

5) DC Motor Model:

In order to include DC motor model in the multibody dynamics model to complete the full system block diagram. we must derive the related equations.
For the motor electrical circuit defined in [81], [82]

$$v = e + i_a R_a + l_{induc}\frac{di}{dt} \tag{47}$$

Where,i_a is the armature current, e the back emf and R_a is the armature resistance. As noted from Table II l_{induc} is very small when compared to the armature resistance R_a thus $l_{induc} = 0$. Note that the mechanical system dynamics can be considered slow when compared to the motor electrical system. For this reason, the current transients term $\frac{di}{dt}$ can be neglected as the model addressed in [83]. There by,

$$v = e + i_a R_a \tag{48}$$

TABLE II. DC MOTOR ESTIMATED PARAMETERS

Parameter	Data	Description
N_G	34.02	motor gear ratio
k_b	$1e^{-2}\ Vs/rad$	back EMF constant
J_m	$7.9431e^{-7}\ kg.m^2$	motor inertia
l_{induc}	$8.5744e^{-6}\ H$	rotor inductance
R	18.334 Ω	motor resistance
b	$1.1355e^{-5}\ Nm/s$	viscous friction constant
K_t	$0.014703\ Nm/A$	torque constant

Where, the back emf and armature current could be defined by,

$$e = k_b\dot{\theta}\ ,\ and\quad i_a = \frac{\tau_m}{k_t} \tag{49}$$

and k_t,k_b are torque and back emf constants. By substitution from (49) in (48) to get,

$$v = k_b\dot{\theta}\ + \frac{\tau_m}{k_t}R_a \tag{50}$$

as shown in Table II

$$let\ k_t = k_b \tag{51}$$

on the other side, the motor mechanical system,

$$\tau_m = \tau_{load} + b\dot{\theta}_{\ m} + J\ddot{\theta}_m \tag{52}$$

where, b and J, are the motor damping coefficient and inertia respectively. By substitution of (50) and 51 in (52) to get the DC motor equation, (53), that relates both the mechanical and electrical systems.

$$\tau_{load} + b\dot{\theta}_{\ m} + J\ddot{\theta}_m = v\frac{k_t}{R_a} - \frac{{k_t}^2}{R_a}\ \dot{\theta}_{\ m} \tag{53}$$

the relation between motor and wheel torque is,

$$\tau_{load} = \frac{\tau_w}{NG} \tag{54}$$

where, NG is the motor gearbox ratio. By simplification of (53)

$$\tau_w = v\frac{NG\ k_t}{R_a} - NG\left(\frac{{k_t}^2}{R_a} + b\right)\dot{\theta}_{\ m} - NG\ J\ddot{\theta}_m \tag{55}$$

while the relation between the wheel linear velocity and the motor angular velocity at the revolute joint or the wheel axis defined by,

$$\dot{R}_x^2 = R_w\frac{\dot{\theta}_{\ m}}{N_G} \tag{56}$$

Assume that J is very small, this ignores the acceleration term, $\ddot{\theta}_m$, and it is eliminated. Which effectively assume that the motor reaches its final value of angular velocity once input voltage is applied. This results in a reduced order model [84]. Torque is related to both the applied voltage and the inertia of the system.The resulting torque equation is utilized for later in linearization with the overall multibody model. Finally the key

equation that relates motor output control torque of robot wheel and the input DC voltage ,

$$\tau_w = \frac{NG\ k_t}{R_a} v - \left(\frac{{k_t}^2}{R_a} + b\right) \frac{{N_G}^2}{R_w} \dot{R}_x^2 \tag{57}$$

6) DC-Motor Parameters Estimation:

We use the Matlab parameter estimation toolbox to simulate the motor parameters in real-time using a DC motor with hardware-in-the-loop (HIL). We followed the strategy found in [68], [85]. The motor driver receives an excitation voltage from the main controller board as in Fig. 9.

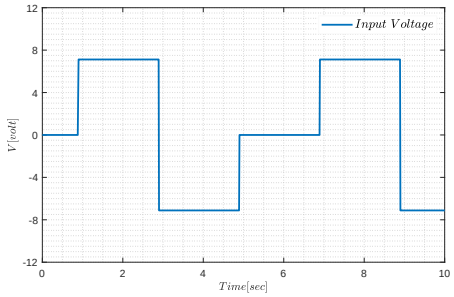

Fig. 9. Input Voltage For Motor Parameters Estimation (V)

This voltage input is a step signal its amplitude varies between $\{0, \pm 7\}$ to see the dc motor speed response at each change on the step signal. Next, we measure the motor speed corresponding to this excitation which is indicated in Fig. 10 this is indicated by the pink colour (Real $\dot{\theta}^2$). Finally, by using the parameter estimation toolbox in MATLAB software. In this toolbox, the method used for optimization is the nonlinear least squares algorithm for model fitting, with a parameter tolerance of 0.001 and a maximum of 100 iterations. The parameters resulted from the estimation process using are listed in Table II.

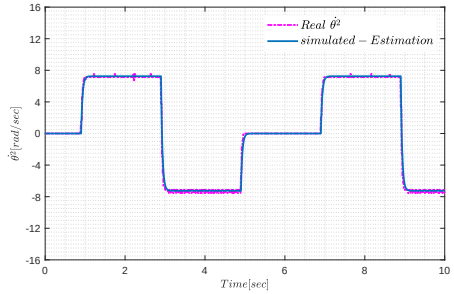

Fig. 10. DC Motor Angular Velocity Response $\dot{\theta}^2$ (Real Measurement-Simulated)

To validate the estimated parameters, we construct the DC motor model in Matlab Simulink. Both Fig. 11 and Fig. 12 show the DC motor model provided with its new estimated parameters found in Table II. Fig. 11 is the main block indicated by the red color in Fig. 12. It starts with the control signal input node (V) that signal enters a summation point that subtracts the back EMF signal generated due to the rotor angular velocity. This error signal is multiplied by the electrical circuit transfer function (l_{induc}, R) that produces the motor armature current I. We use a saturation function to limit the current to the rated value of the used DC motor. We find the motor torque by multiplying this value with the motor torque constant (T.constant).

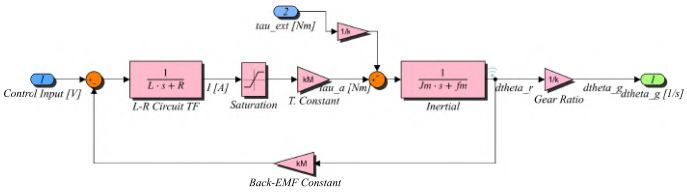

Fig. 11. DC Motor Model/Simulink

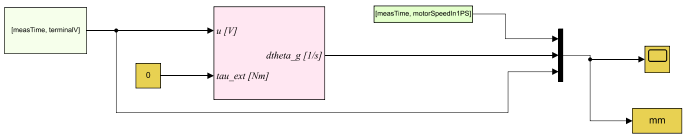

Fig. 12. Validation Model/Simulink

For the mechanical side of the DC motor, the motor torque is the input, and the output is rotor angular velocity. In addition to this, Fig. 12 constitutes the full simulation model at which the excitation voltage along with the motor estimated parameters were given to the DC motor model, and the outputs are the real and simulation results of the rotor angular velocities in rad/sec. These results were plotted and indicated in Fig. 9. This method ensures the success of the estimation process. The output angular velocity from the simulation process is indicated by the blue color, and it is almost identical to the real measured data in the pink color.

7) State Space Representation:

Since we need to design a controller for self-balancing robot stabilization, the state space model is to be found. Before this step, the modification of the force matrix in (37) was done by using the half of the input torque ($\tau_w/2$) from (57) as we deal with two DC Motors for the right and left wheels that will keep the robot balanced. Following this the nonlinear MBM was linearized around the equilibrium operating point ($\mathbf{x_0}, \mathbf{u_0}$). Where, $\mathbf{x_0}$ was defined in (46) and $\mathbf{u_0} = 0$. Recall, the state space model and Taylor series are utilized, by neglecting terms of second and higher orders as discussed in [86], [87], taking into our consideration that $f_i(\mathbf{x_0}, \mathbf{u_0}) = \mathbf{0}$ and the variations about that operating point are, $\widetilde{x}_j = x_j - x_{j_0}$, $\widetilde{u}_k = u_k - u_{k_0}$ and $\dot{x}_i = \dot{\widetilde{x}}_i$. Then the resulted linearized i^th state space equation, at $x_0 = [0; pi/2; 0; 0]$ are,

$$\dot{\widetilde{\mathbf{x}}} = \mathbf{A}\widetilde{\mathbf{x}} + \mathbf{B}\widetilde{\mathbf{u}} \tag{58}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_{32} & A_{34} & 0 \\ 0 & A_{42} & A_{44} & 0 \end{bmatrix} \tag{59}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ B_{31} \\ B_{34} \end{bmatrix} \tag{60}$$

Where, the term A_{32}, A_{34}, A_{42} , A_{44}, B_{31} and B_{34} are stated in section IV. By substitution of the all model parameters in Table I and the equilibrium operating point results in the state space model matrices,

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1.9337 & -1.6202 & 0 \\ 0 & 84.7510 & -11.6927 & 0 \end{bmatrix} \tag{61}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0.1963 \\ 1.4168 \end{bmatrix} \tag{62}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \ \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{63}$$

Note that, robot parameters in Table I were estimated from the robot CAD model, the measured and calculated data of the measurable parameters. In this stage, we could use the results in (61),(62) and (63) and check the stability, controllability and observability for the controller design about the stabilization point.

B. Controller Design

1) Pole-Placement Controller Design:

In state feedback control law, the theory of asymptotic stability says that the vector for the closed-loop system dynamics (**A**-**BK**), shown in the block diagram in Fig. 13, must have eigenvalues with strictly negative real parts. Control engineers are often concerned with the closed-loop characteristics of the transient response, such as the rise time, t_r, peak time, t_p, maximum percentage overshoot, $\%OS$, and the time at which settling occurs, t_s, of the step response.

For that, in the pole placement technique, we are to shape the system response by finding the desired pole locations according to a desired system step response of the second-order dominant poles. The resulting feedback gain matrix, K, is designed to do this task [73], [87]. Now, the open loop system in (58) has the following eigenvalues ($\mathbf{SI} - \mathbf{A} = \mathbf{0}$) are,

$$\mathbf{Poles} = \begin{bmatrix} 0 & 8.8805 & -10.388 & -4.974 \end{bmatrix} \tag{64}$$

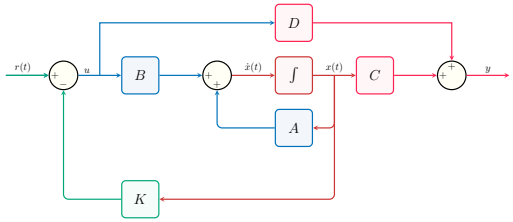

Fig. 13. Closed Loop State Feedback System Block Diagram

Equation (64) indicate that there are at least one pole at the right-hand side of the pole-zero map in the S-plane. Before designing a state feedback controller, we must first check the system's controllability. We must check the stability of the system in (58) that it should be controllable. By checking the rank of the controllability and observability matrices. This can be easily done by using Matlab commands $\mathbf{Ctrb}(\mathbf{A}, \mathbf{B})$ and $\mathbf{Obsv}(\mathbf{A}, \mathbf{C})$ respectively . It was found that the rank of both the controllability the observability matrices were four. Thus the system is controllable and observable,

In this context we used Ackermann's formula for the calculation of the state feedback gain vector $\mathbf{K}$ in terms of the desired closed-loop characteristic polynomial [87], [88]. The second step is shaping the system dynamics response. First of all, we design an approximation of the second-order system dominant poles to pull the system poles $\{0, 8.8805\}$ to the stability region, and by using the desired following Table III, this yields the desired dominant second-order transfer function poles as the first and second poles in (66).

TABLE III. DESIRED DOMINANT POLES FOR THE 2^{nd} ORDER SYSTEM

Symbol	Data	Description
$\%OverShoot$	25%	maximum overshoot
ζ	0.69	damping ratio
t_s	2 sec	settling time
ω_n	2.90 rad/s	undamped natural frequancy

The other remaining poles were chosen to be in the $-ve$ real axis of the s-plane far away with multiples from the origin to bring the system to stability region (non-dominant poles). Therefore, by using the data in Table III, we can determine the desired dominant poles location. The undamped natural frequency can be found by,

$$\omega_n = 4/(\zeta t_s) \tag{65}$$

and the desired closed loop poles are,

$$Desired_{Poles} = \begin{bmatrix} -2 + 2.10i & -2 - 2.10i & -500 & -10 \end{bmatrix} \tag{66}$$

The system's resulting feedback gain vector for the desired stated response criteria was calculated to be,

$$K_{place} = \begin{bmatrix} -756.356 & 1363.475 & -445.143 & 151.235 \end{bmatrix} \tag{67}$$

On testing the controller design, the simulation of the controller on both linearized and nonlinear controller was done. We initialized the model simulation with the initial conditions $\mathbf{x_0}$,

$$\mathbf{x_0} = \begin{bmatrix} -0.0868241 & (\pi/2) - (\pi/18) & 0 & 0 \end{bmatrix} \quad (68)$$

While the reference state vector is the tracking trajectory, $\mathbf{x_r}$,

$$\mathbf{x_r} = \begin{bmatrix} 0.1 & \pi/2 & 0 & 0 \end{bmatrix} \quad (69)$$

Note that the initial values for the linearized model are almost the same as the nonlinear one, except for the initial angle, which is measured from the linearization point $\theta^3 = pi/2$. So, its initialization should have a small drift from that point, for example, $-\pi/18\ rad$.

The results of the controller simulation with both linearized and nonlinear models, with the red and blue colours respectively, are shown in Fig. 14 for displacement and Fig. 15 for tilt angle. It is clear that system requirements in Table III have been met.

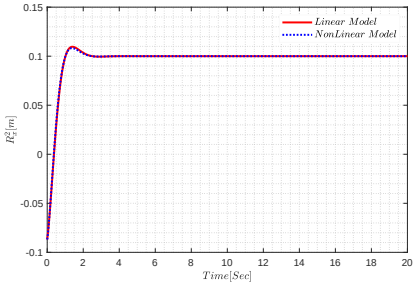

Fig. 14. Robot Wheel Position R_x^2 Using the Pole Placement Controller

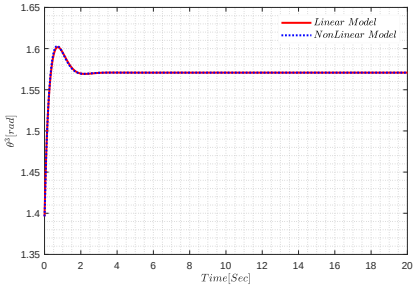

Fig. 15. The Inverted Body Tilt Angle θ^3 Using the Pole Placement Controller

The set point,in Fig. 14, for the wheel displacement, R_x^2 was set to $0.1\ m$ and initially it was set to $-0.0086\ m$. It took about one second to reach the 95% of the desired response. The maximum overshoot is about 10% with the settling time of two seconds. The second Fig. 15, the desired tilt angle was set to $\theta^3 = pi/2\ rad$ while the initial value was set to $\theta^3 = 1.3963\ rad$ as initial drift from the balancing set point. the response to this setpoint shows that the inverted body settles at the balancing point $pi/2$ at nearly two seconds with maximum overshoot of 3%. Fig. 16 and Fig. 17 indicates the linear and angular velocity of the first two states and similarly meets the specified characteristics.

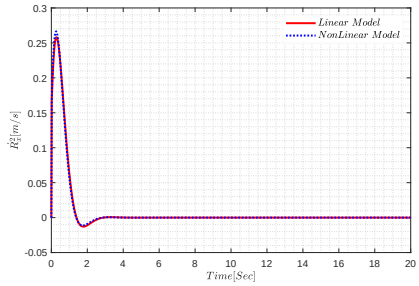

Fig. 16. The Robot Wheel Velocity $\dot{R}_x^2$ Using the Pole Placement Controller

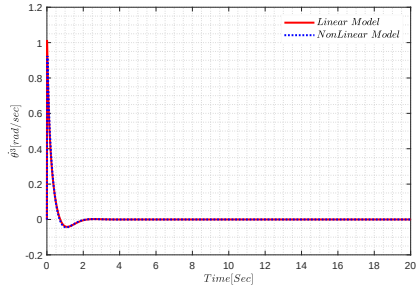

Fig. 17. The Inverted Body Angular Velocity $\dot{\theta}^3$ Using the Pole Placement Controller

While, Fig. 18 describes the control action in volts.It is clear that the linear model,the blue colour, needs initially about 3 volts to get the inverted body track the desired set points for the wheel displacement and the pendulum tilt angle. while the nonlinear model,orange colour, initially needs about 17 volts this is due to initial computation of the nonlinear model but quickly decays at less than 0.05 seconds to the 6 volts which is considered the initial voltage required for the stabilization.

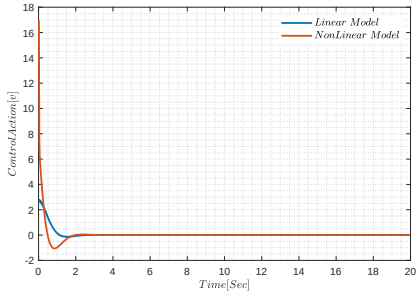

Fig. 18. The Control Voltage Using the Pole Placement Technique

2) LQR Controller Design:

The general equation for the optimal linear quadratic regulator cost function [35], [72], [82] is

$$J(u) = \int_0^\infty \left[x^T \mathcal{Q} x + u^T \mathcal{R} u \right] dt \quad (70)$$

Where $\mathcal{Q}$ and $\mathcal{R}$ are positive definite weighting matrices. And solving the Algebraic Riccati Equation (ARE) in (71) for $\mathcal{P}$,

$$A^T\mathcal{P} + \mathcal{P}A - \mathcal{P}B\mathcal{R}^{-1}B^T\mathcal{P} + \mathcal{Q} = 0 \tag{71}$$

which results in an asymptotically stable closed loop system. From the linear optimal control theory [35], the gain K used to minimize (70) is obtained for the optimal controller u by,

$$K = \mathcal{R}^{-1}B^T\mathcal{P} \tag{72}$$

, and the control action is found by,

$$u = -K\mathbf{x}, \tag{73}$$

The weighting matrices $\mathcal{Q}$ and $\mathcal{R}$ are adjusted utilizing Bryson's rule, as in [35]

1) $\mathcal{Q}_{i,i} = \frac{1}{max(\tilde{x}_i^2)}$
2) $\mathcal{R}_{i,i} = \frac{1}{max(\tilde{u}_i^2)}$

Recall that the desired maximum allowable error for each state is defined by $R_x^2 = 0.01\ cm$, $\theta^3 = 0.0175\ rad$, $\dot{R}_x^2 = 0.1\ m/sec$, $\dot{\theta}^3 = 0.1\ rad/sec$ and the max motor controller input is $12\ V$. For that by using the Bryson rules, $\mathcal{R} = 0.0069444$; and the matrix $\mathcal{Q}$ is summarized by,

$$\mathcal{Q} = \begin{bmatrix} 10000 & 0 & 0 & 0 \\ 0 & 3286.137 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \tag{74}$$

the optimal control gain matrix is found by using the "lqr" command in MATLAB,

$$\mathbf{K} = \begin{bmatrix} -1200.00381 & 1623.8364 & -647.06405 & 211.7751 \end{bmatrix} \tag{75}$$

and the controlled system, its closed loop eigenvalues are found from $A - KB$ as,

$$\mathbf{P}_{Closedloop} = \begin{bmatrix} -686.695 \\ -5.856 \\ -2.987 + 2.769i \\ -2.987 - 2.769i \end{bmatrix} \tag{76}$$

The simulation results are shown in Fig. 19 and Fig. 20 for both the displacement and tilt angle respectively. The similar initial and tracking conditions for the system states were defined in (68) and (69) respectively.

The most important note from the stated figures is that the linearized and nonlinear model responses are almost identical. In addition to that, the settling and zero steady-state errors are met, while the maximum overshoot is less than 25% from both the tilt angle and the linear displacement. Furthermore, it is obvious that the maximum overshoot for tilt angle is higher than the achieved in the pole placement technique in Fig. 15. Similarly, for the linear velocity, as shown in Fig. 21, initially it is zero and goes up to $0.32\ m/s$ before fast decay to zero at two seconds. Fig. 22 indicates the tilt angular velocity, which increases up to $0.65\ rad/s$ before decay at the same settling time as desired. Finally, Fig. 23 describes the control action or the DC motor control voltage over the simulation time. For real-time implementation, one should consider the maximum linear velocity for the selection of the DC motors and therefore the rated voltage.

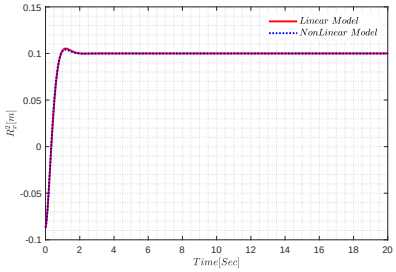

Fig. 19. The Robot wheel Position R_x^2 Using LQR controller

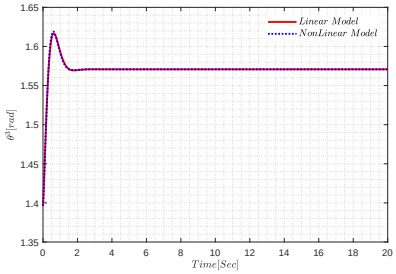

Fig. 20. The Inverted Body Tilt Angle θ^3 Using LQR controller

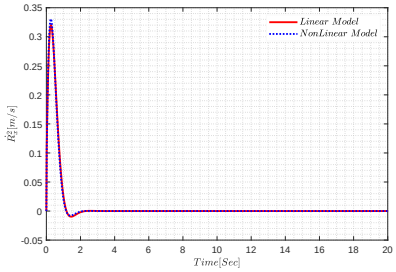

Fig. 21. The Robot wheel Velocity $\dot{R}_x^2$ Using LQR Controller.

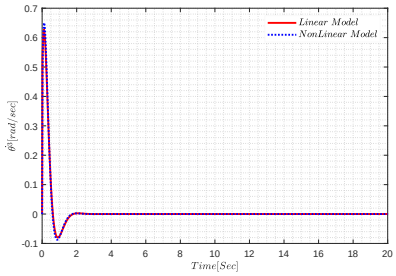

Fig. 22. The Inverted Body Angular Velocity $\dot{\theta}^3$ Using LQR Controller

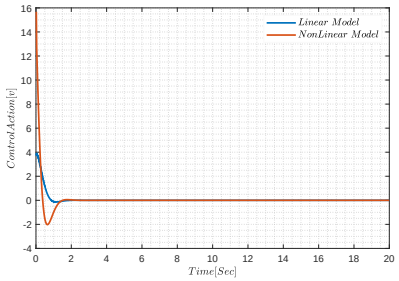

Fig. 23. The Control Voltage Using LQR Controller

3) Model Modification for Friction and Damping Forces:

For the realistic behavior of other external forces that cause losses in the control torque and affect robot stabilization. We added the friction forces. Assume a smooth Coulomb friction model will be considered [89]. Therefore, the dry friction force is given by:

$$F_{fric} = -\mu_k \left| F_N \right| sgn(\dot{R}_x^2) \tag{77}$$

where, μ_k is the dry friction coefficient, F_N is the instant normal force produced by the contact between the two surfaces interacting, and $\dot{R}_x^2$ is the relative velocity between them, whereas sgn is the sign function defined as: [89]

$$sgn(\dot{R}_x^2) = \begin{cases} -1 & \dot{R}_x^2 \leq 0, \\ 0 & \dot{R}_x^2 = 0, \\ 1 & \dot{R}_x^2 \geq 0. \end{cases} \tag{78}$$

The classical Coulomb friction model and Coulomb–tanh alternative model could do the same task. Because the alternative Coulomb–tanh model ensures continuity at zero velocity (represented by the $\dot{R}_x^2$ variable) and highly approaches the classical model as the friction coefficient K_{colmb} increases. It is obvious in Fig. 24 [90]. It shows that at K_{colmb} is the perfect case that is used with the Coulomb–tanh model.

$$F_{fr} = -\mu_k \left| F_N \right| \tanh\left(K_{colmb} \dot{R}_x^2 \right) \tag{79}$$

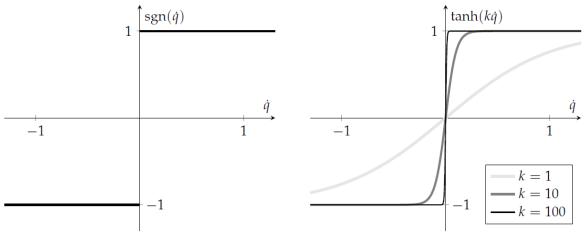

Fig. 24. The Friction Coefficient [89]

For the revolute joint damping due to the inverted mass T_{fr} is calculated as, $T_{fr} = -K_d(\dot{\theta^3} - \dot{\theta^2})$. [91] [92] Thus, the total modified force vector $\mathbf{Q}$,

$$\mathbf{Q} = \begin{bmatrix} m^2 g & -2\tau_w & 0 & -m^3 g & F_{fr} + \frac{T_{fr}}{R_w} & 2\tau_w - T_{fr} \end{bmatrix} \tag{80}$$

Recall the inclusion of the DC motor torque, τ_w, from (57) and [82]. That equation is simplified to be considered in the controller design of balancing robot stabilization.

It is assumed that $\dot{\theta^2}$ at the moment of an impulse disturbance on the system, that is, the moment that needs the largest control action (force), is $zero\ rad/s$. Substituting in (56), it shows the proportionality between the control output torque τ as an input and the motor driver output voltage to the DC motor, v.

$$\tau_w = \frac{NG\ k_t}{R_a} v \tag{81}$$

For that we conclude that the force vector is modified as the following matrix,

$$\bar{\mathbf{Q}}_B = \mathbf{B}_i^T \mathbf{Q} - \mathbf{B}_i^T \mathbf{M} \gamma \tag{82}$$

$$\bar{\mathbf{Q}}_B = \begin{bmatrix} (K_d \frac{(\dot{\theta^2} - \dot{\theta^3})}{R_w} - F_r tanh(K_f \dot{R_x}^2)((m^2 g) + (m^3 g)) + \\ L_p m^3 \dot{\theta^3}^2 cos(\theta^3) + \frac{2K_t N_G}{R_a R_w} V \\ \frac{(2K_t N_G V)}{R_a} - K_d(\dot{\theta^2} - \dot{\theta^3}) - L_p g m^3 cos(\theta^3) \end{bmatrix} \tag{83}$$

Now, the full nonlinear model has been defined on substitution in (40). In order to linearize the system as we assumed before, the equilibrium point of the tilt angle, $\theta^3 = 90 + \theta$. Thus, $sin(\theta + 90) = cos(\theta) = 0$, $cos(\theta + 90) = -sin(\theta) = -\theta$, $\dot{\theta^3} = 0$, $\dot{\theta^3}^2 = 0$ and θ is the drift from the equilibrium point.

these substitutions in the nonlinear model results in the linearized mass matrix, $\bar{\mathbf{M}}_L$, and the force matrix, $\bar{\mathbf{Q}}_L$, are defined by,

$$\bar{\mathbf{M}}_L = \begin{bmatrix} m^2 + m^3 + Izz^2/R_w^2 & -L_p m^3 \\ -L_p m^3 & m^3 L_p^2 + Izz^3 \end{bmatrix} \tag{84}$$

and

$$\bar{\mathbf{Q}}_L = \begin{bmatrix} K_d \dot{\theta^2} Rd - F_r tanh(K_f \dot{Rx^2})(m^2 g + m^3 g) - \theta^3 + \frac{2K_t N_G}{R_a R_w} V \\ \frac{2K_t N_G}{R_a} V - K_d \dot{\theta^2} - L_p g m^3 - \theta^3 \end{bmatrix} \tag{85}$$

After that by numerical substitution in the state space model in eq.58,

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1.9337 & -111.3709 & 0 \\ 0 & 84.7510 & -803.7338 & 0 \end{bmatrix} \tag{86}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 7.5240 \\ 137.5348 \end{bmatrix} and, \mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \tag{87}$$

Note that, our main goal is to stabilize the robot in the vertical position for that the system transfer function for angle related to input voltage is our interest,

$$G(s) = \frac{137.5s^2 + +9543s}{s^4 + 111.4s^3 - 84.8s^2 - 7885s} \tag{88}$$

and the system open loop poles for the modified model are,

$$\mathbf{P}_{open-loop} = \begin{bmatrix} 0 \\ -111.4968 \\ 8.4725 \\ -8.3466 \end{bmatrix} \tag{89}$$

it is obvious that system is not stable as there exist at least one pole in the right hand side of thae pole-zero map in S-plane. It is clearly shown from the root-locus graph in Fig. 27 of the system in (88). There are a pole at the origin point and another one at the point $(8.4725, 0)$. For this, the PID controller design is proposed to bring the system to the stability region and shape the system response.

4) Digital PID Controller Design:

We deal with SIMO (single input, u^c, multiple output, θ^3, R_x^2, planar system), as it has one input i.e. the torque, τ_w, or voltage, v, applied on the wheel and two outputs i.e. the wheel position and the main body angles, R_x^2 and θ^3 respectively. Various methods exist for determining the discrete time equivalent of a continuous controller. Given that there is no exact digital equivalent of a continuous controller a continuous controller has access to the complete time history of the error signal e(t). while a digital controller has access only to the samples of this signal [93] [94] [95] . We assume that the Analog PID controller transfer function is $G_c = \frac{u(s)}{e(s)}$,

$$G_c = k^p + k^i \frac{1}{s} + k^d s \tag{90}$$

And needs to be replaced with a discrete-time controller. It accepts the samples from the difference between the sensor angle and set point, which is the controller input signal at the sample kt_s, $e_{(kt_s)}$, from a sampler and, by using past values of the control signal $u_{(kt_s)}$ and present and past samples of the input $e_{(kt_s)}$, will compute the next control signal, $u_{(kt_s+t_s)}$, to be sent to the wheels DC motors. The equivalent control action is composed of three terms in the time domain,

$$u_t^c = u_t^p + u_t^i + u_t^d \tag{91}$$

By discretizing each control action term separately. For the proportional term, the next control sample can be computed as,

$$u^p_{(kt_s+t_s)} = k^p e_{(kt_s+t_s)}. \tag{92}$$

While the integral term can be approximated as,

$$\begin{aligned} u^i_{(kt_s+t_s)} &= \int_0^{(kt_s+t_s)} e(t)\,dt \\ &= \int_0^{kt_s} e(t)\,dt + \int_{kt_s}^{(kt_s+t_s)} e(t)\,dt \end{aligned} \tag{93}$$

The integral term $\int_{kt_s}^{(kt_s+t_s)} e(t)\,dt$ could be approximated using Tusons rule or the trapezoidal method for the numerical integration for one step of time. [93] [87]

$$u^i_{(kt_s+t_s)} = u^i(kt_s) + (k^i \frac{t_s}{2}(e_{(kt_s+t_s)} + e_{(kt_s)})). \tag{94}$$

Similarly, the next control sample for the derivative term is found by using the two point difference form in the term, $u^d(t) = k^d \frac{de(t)}{dt}$ to get,

$$u^d_{(kt_s+t_s)} = k^d \frac{e(kt_s+t_s) - e(Kt_s)}{t_s} \tag{95}$$

Now, the PID controller in discrete form is found by using, the operator z^{-1}, the backward shift operator. So that $u(z)$ is the transform of $u(kTs + Ts)$. And by this notation, the $z^{-1}u(z)$ will be the transform of $u(kt_s)$. With this definition, proportional ,integral and the derivative terms are defined as in (96),(97) and (98) respectively.

$$u^p(z) = k^p e(z) \tag{96}$$

$$u^i(z) = z^{-1}u(z) + k^i \frac{t_s}{2}(e(z) + z^{-1}e(z)) \tag{97}$$

$$\begin{aligned} u^i(z) &= k^i \frac{t_s}{2} \frac{1+z^{-1}}{1-z^{-1}} e(z) \\ u^d(z) &= k^d \frac{1}{t_s}(e(z) - z^{-1}e(z)) \\ u^i(z) &= k^d \frac{1}{t_s}(1 - z^{-1})e(z) \end{aligned} \tag{98}$$

Now recall (91), the controller discrete time transfer function, [2]

$$u^c(z) = (k^p + k^i \frac{t_s}{2} \frac{1+z^{-1}}{1-z^{-1}} + k^d \frac{1}{t_s}(1 - z^{-1}))e(z) \tag{99}$$

The system digital PID controller is defined as in Fig. 25. Note that, when the sampling time of the integral term is selected as $1\ ms$ according to the (80 MHz FPGA clock), the discrete time controller perfectly tracks the continuous time controller output.

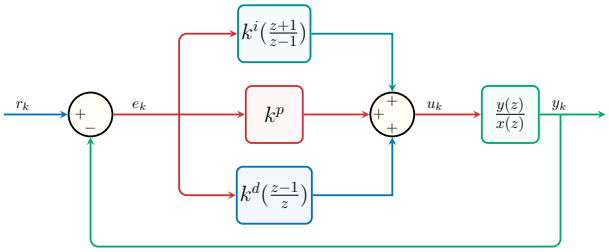

Fig. 25. Digital PID Controller Block Diagram

We used the MATLAB pidtool to tune system controller parameters according to the background experience stated by [87], [88],and [93], in Table IV. Where, D, i and D-by Var denotes decrease, increase and decrease by varying respectively. This table summarizes the different effects of each term of the PID controller on the rise time, settling time, maximum overshoot and the steady state error of the system response. E.g. the k^p cause the system to respond faster while the k^i parameter eliminates the steady state error.

TABLE IV. SUMMARY OF PID TERMS EFFECTS

Parameter	R.Time	OS	S.Time	SS Error
k^p	D	i	$D-by\ Var.$	D
k^i	D	i	i	$Vanish$
k^d	$D-by\ Var.$	D	D	$D-by\ Var.$

Tuning k^p, k^i and k^d so as to obtain the best parameters that meet the desired response was done. The results are summarized in Table VI. Which indicates that the settling time is $0.196\ sec$, the percentage overshoot is 19% with zero steady state error. And, the rootlocus of both the uncontrolled and the controlled self-balancing robot model are shown in Fig. 27 and Fig. 28. The rootlocus in Fig. 28 show that th system is stable.

Matlab-simulink as [73], Fig. 26, is used to test and simulate the designed both continuous time and the digital PID controllers on the tilt angle linearized transfer function. The simulink model is established. According to [93], [2] and in (99), the parameters of the digital PID controller defined in simulink model are $K_{p1} = k^p, K_{i1} = k^i \frac{t_s}{2}$, and $K_{d1} = k^d \frac{1}{t_s}$.

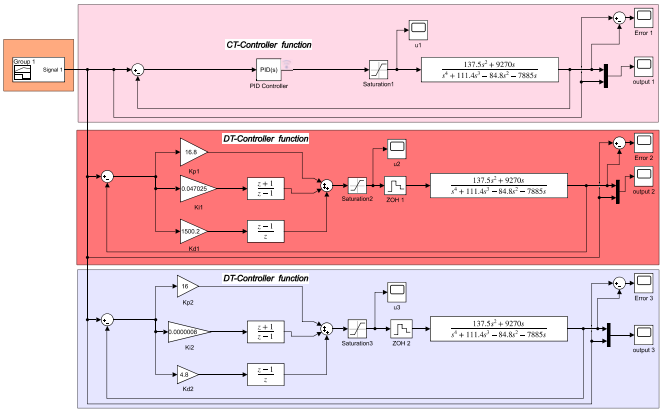

Fig. 26. Simulink Model of Tilt angle controller

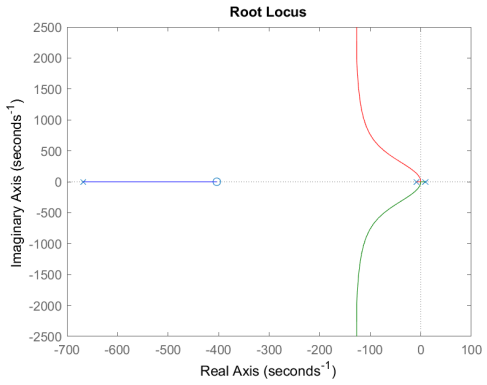

Fig. 27. The Uncontrolled System Rootlocus Plot

In this model, saturation function is utilized to limit the controller output voltage to be within $\pm 12v$. Also, a zero order hold function is used to give a continuous control action as an input to the CT-Transfer function. Also, the system transfer function is altered by an input signal to test its response with small deviations away of the equilibrium balancing angle (1.57 rad).

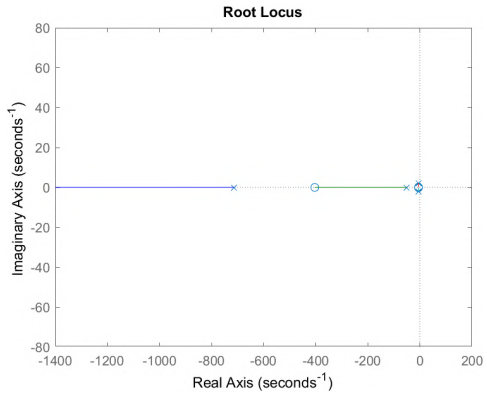

Fig. 28. The System with The Controller Rootlocus Plot

Furthermore, we used signal input and its unit in rad, and we made small perpetuation by $0.117\ rad$ from equilibrium linearization point $1.5707\ rad$ (90 deg.) of the tilt angle as shown in Fig. 29, then by using the standard PID Controller block with the tuned parameters in Table V. The output of the controller block is constrained with the saturation limits $\pm 12v$, the motor-rated voltage.

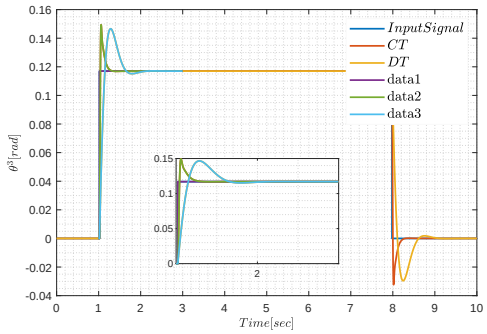

Fig. 29. Step Response PID DT-CT

TABLE V. TUNED PID PARAMETERS

Parameter	Value	Description
k^p	16.8	proportional gain
k^i	94.05	integral gain
k^d	0.7501	derivative gain

The response is displayed by scope function block as shown in Fig. 29 which indicates both responses of closed loop system for continuous and discrete time controller. It is clear that the corresponding the D-PID Controller to the CT-PID has a slightly small reduction in $Overshoot$ but is slower in terms of settling and rise time. This also reflects the controller output signal due to this error is shown in Fig. 30.

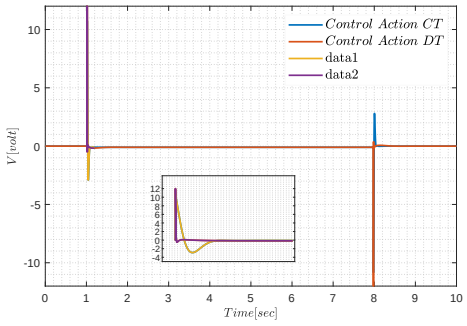

Fig. 30. Control Action DT-CT.

TABLE VI. CONTROLLER DESIGN STEP RESPONSE

Parameter	Value	Description
R_{time}	0.0204 sec	rise time
S_{time}	0.196 sec	settling time
$OS\%$	19%	maximum overshoot
SS_{error}	0	steady state error
G_m	$-17.7db$ at 10.6 rad/sec	gain margin
P_m	88 $deg.$ at 79.1 rad/sec	phase margin

C. Hardware System - Electrical circuit

The hardware configuration is shown in both Fig. 31 and Fig. 32. It consists of ,

1) Two DC motors: They are driven using a dual DC motor driver that is supplied with power from a rechargeable battery.
2) The dual cytron DC motor driver: It is controlled by the 3.3 v TTL FPGA Kit through DIO port and the driver PWM pins.
3) DC motor Hall effect encoders: They are installed on the motor shaft to give feedback about its angular position and velocity.
4) Accelerometer and gyro sensors: They give the feedback about the robot body tilt angle θ^3 and its angular velocity $\dot{\theta}^3$. Additionally, they take their power from the FPGA Kit through the AO port.
5) Robot main controller: It includes the hardware reprogrammable circuit, the FPGA, which is designed and tested using PC along with the FPGA kit interfacing software LabVIEW with FPGA module. As shown in Fig. 33, it includes FPGA target, a real time processor, the I/O digital and analog ports in addition to the ethernet cable to for communication with PC.
6) DC-DC converter: It is used to supply the 24V to the sbrio 9631 FPGA kit from the 12v power supply.
7) Fig. 32 shows the robot CAD model, which was implemented in the real-world prototype. It includes all the stated components in addition to the two robot wheels, the main inverted body (chassis), the 12V power supply, and the bolts and connectors that were used to assemble the whole robot structure.

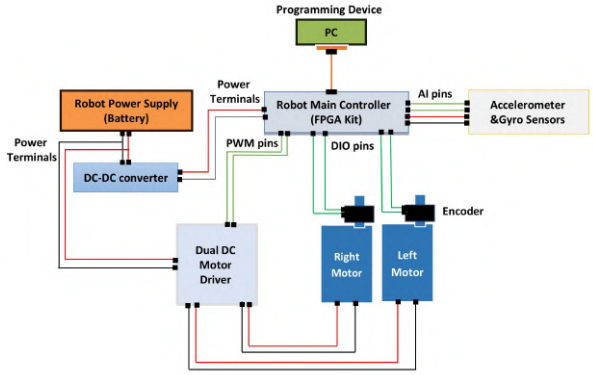

Fig. 31. System Hardware Block Diagram

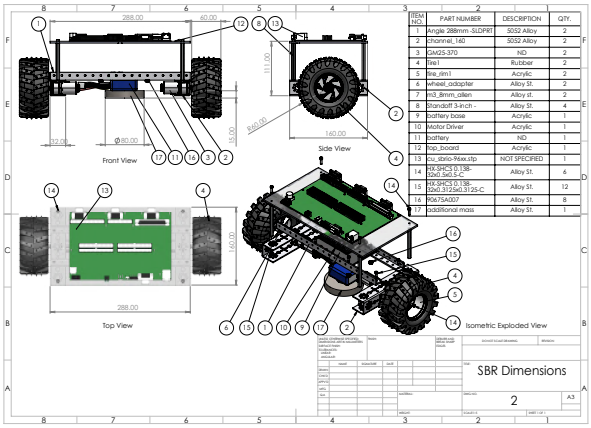

Fig. 32. The SBR CAD Model

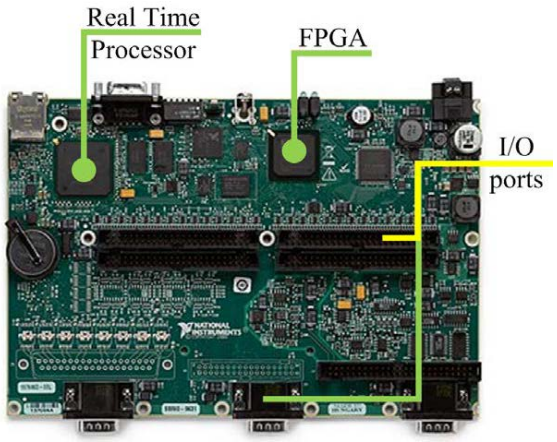

Fig. 33. Sbrio 9631 FPGA Board

The accelerometer MEMS sensor is considered the tilt angle feedback. And its output signal must be conditioned to give accurate results. The tilt angle of the robot describes the angular position of the robot's main body. A low-power MEMS-based analog ADXL335 accelerometer that measures the static acceleration of gravity as well as the dynamic resulting acceleration

from motion or vibration in the three axes $(\ddot{R}_x, \ddot{R}_y, \ddot{R}_z)$, and with these three components utilization, the tilt angle is found [96], [97]. The sbRio 9631 FPGA analog input port (AI) has the main specifications in Table VII. It's analog to digital converter resolution is 16 bits, the conversion time is $4\mu s$ and the analog input ranges are of $\pm 10,\ \pm 5,\ \pm 1,\ \pm 0.2V$. These data are utilized in accelerometer measurements and calibration.

TABLE VII. SPECIFICATIONS OF ANALOG PORT IN SBRIO9631

Property	Data	unit
ADC resolution	16	$bits$
Conversion time	$4.00 - (250)$	$\mu s - (kS/s)$
Input ranges	$\pm 10,\ \pm 5,\ \pm 1,\ \pm 0.2$	V

The traditional calibration method specified by the sensor datasheet to obtain acceleration values is implemented by using the equations from (100) to (102). To find the tilt angle about the z-axis, see (103). However, this method is not accurate, as the sensitivity factor makes significant acceleration errors that, by extension, enlarge the error in the tilt angle, which is not acceptable in our application. Therefore, we have decided to develop an alternative solution to address this issue.

$$\ddot{R}_x = (V_{x(out)} - V_{x(offset)})/K_s \tag{100}$$

$$\ddot{R}_y = (V_{y(out)} - V_{y(offset)})/K_s \tag{101}$$

$$\frac{\ddot{R}_x}{\ddot{R}_x} = \frac{g.sin\,(\alpha)}{g.cos(\alpha)} \tag{102}$$

$$\alpha = tan^{-1}(\ \frac{\ddot{R}_x}{\ddot{R}_x}\ ) \tag{103}$$

Where, α, is the inclination angle, and K_s, is the sensor sensitivity. While, $V_{x(offset)}, V_{y(offset)}$ are determined from sensor calibration with the help of datasheet stated in [97].

In our proposed method, at first, we changed the settings for analog inputs to get its raw ADC values coming from the accelerometers. There exist x-y-z axes for ADXL335. For each axis, two measurements are taken, at maximum positions$(+90)$ and (-90) degrees. Each single measurement takes 1000 samples of data and then the calculation of the mean value of these samples is made. The data taken are summarized in Table VIII.

TABLE VIII. ACCELEROMETER RANGE OF MEASUREMENT

Direction	X-Accele.	Y-Accele.	Z-Accele.
$+g$	6022.39	12083	11916.7
$-g$	3968.63	8057.39	8172.55

Following that step, we make a linear equation that maps the incoming ADC Values from the accelerometer to an angle about each axis $(Angle_x, Angle_y, Angle_z)$ shown in Fig. 34. this is done by developing a mapping function for each axis as sensor is linear as follows,

$$Y = a(x - x_{low}) + y_{low} \tag{104}$$

Where,

$$a = \frac{y_{high} - y_{low}}{x_{high} - x_{low}} \tag{105}$$

and, x is input data from channel AI0, x_{low}, is the AI0 Value corresponding to minimum Value of channel Input and y_{low}, is the minimum mapping value corresponding channel Input. Table IX states the calculated valves for the accelerometer constant a.

TABLE IX. ACCELEROMETER CONSTANTS FOR EACH AXIS

Direction	X-Accele.	Y-Accele.	Z-Accele.
a	0.09738	0.0496819	0.05341

The output value from each equation is divided by -100 to get the values of each accelerometer channel in "g". The angle about x-axis can be found by,

$$Angle_x = atan2(-y_g, -z_g) * 57.2957795 + 180 \tag{106}$$

and the angle about y-axis,

$$Angle_y = atan2(-x_g, -z_g) * 57.2957795 + 180 \tag{107}$$

similarly the angle about z-axis,

$$Angle_z = atan2(-y_g, -x_g) * 57.2957795 + 180 \tag{108}$$

The advantage of using dual axis measuring method is the ability to distinguish between each quadrant and to measure angles throughout the entire 360 degrees by examination of the sign of the measured acceleration values on each axis [96].

Following that we designed a digital low pass filter, by using bilinear transformation from (109) up to (116). For the sensor output angle filter,

$$H\,(s) = \frac{\omega}{s + \omega} = \frac{2\pi f_c}{s + 2\pi f_c} \tag{109}$$

Using bilinear transformation

$$z = \frac{1 + \delta t/2}{1 - \delta t/2s} \tag{110}$$

then

$$s = \frac{2}{\delta t}\frac{(1 - z^{-1})}{(1 + z^{-1})} \tag{111}$$

$$H(z) = \frac{\omega}{\frac{2}{\delta t}\frac{(1-z^{-1})}{(1+z^{-1})} + \omega} \tag{112}$$

now,simplify and substitute for

$$H(z) = \delta t\omega\frac{(z+1)}{(\delta t\omega + 2)z + (\delta t\omega - 2)} \tag{113}$$

$$\omega = 2\pi f_c \tag{114}$$

Based on the sampling time $\delta t = 10\ ms$ and the cut off frequency $f_c = 5\ HZ$. Note that the f_c is selected by different

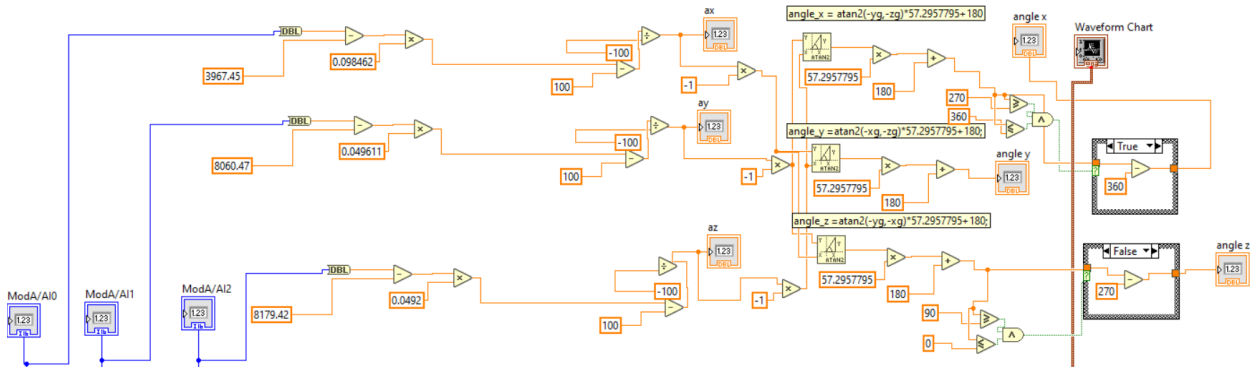

Fig. 34. Accelerometer reading vi

trial and error notability, by the substitution of these values in (113) which results in the filter equation in (116),

$$\frac{y(n)}{x(n)} = H(z) \tag{115}$$

$$y(n) = 0.951219y(n-1) + 0.02439x(n-1) + 0.02439x(n) \tag{116}$$

The last stated equation, (116), is implemented in Labview vi to filter out the undesired frequencies. which is indicated in the for the fully SBR real-time Vi in Fig. 37, i.e. to find the angle about X-axis (angle x) the analog accelerometer value corrosponding to this axis is read through the AI0 (MODA/AI0) then this value calculated as in (104) and then by finding the angle by (106). This is done for the three axis angles to test and validate the developed method to find the tilt anhle from the low cost accelerometer. The result of this implementation in NI-LabVIEW with FPGA is shown in Fig. 35. The figure indicates the plot of both the real (red color) and the filtered (blue color) tilt angle for different orientations of the sensor about the x-axis.

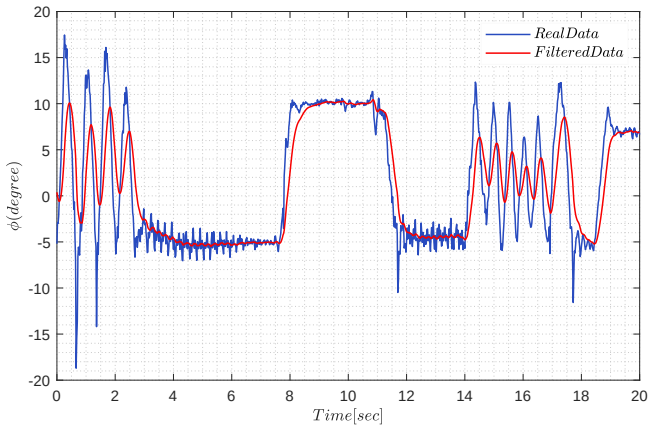

Fig. 35. The Reselt of the Tilt Angle with the designed filter for Adxl335

D. Stabilization Controller Implementation

In this context, we describe the full program used in the implementation of the stabilization D-PID controller on Sbrio 9631, The FPGA VI side is indicated by Fig. 36. It includes four ”while loop” And one Single cycle timed loop. Each loop has its specific task and they are explained in the following lines,

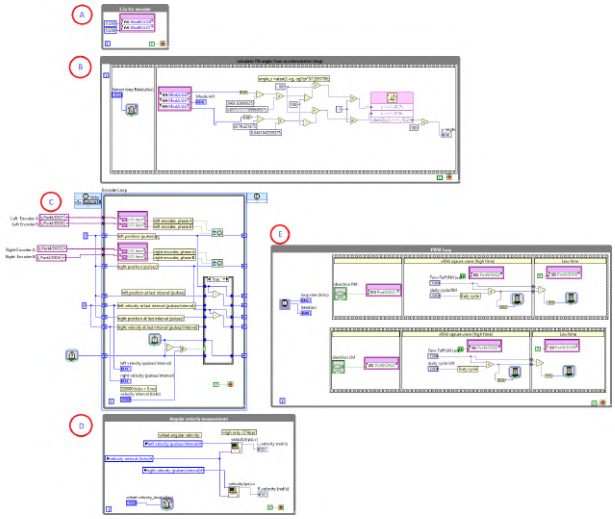

Fig. 36. SBR PID Balancing FPGA Block Diagram vi

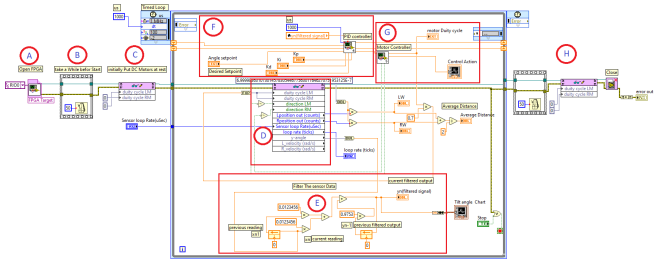

Fig. 37. The RT SBR PID Balancing Block Diagram vi

1) The first loop is titled "3.3V for encoder" in Fig. 36A. This loop Keeps supplying an analog voltage of 3.3 volt to the 374 ppr DC motor Hall effect encoders drawn From the DC-DC Converter that supplies the Sbrio 9631. The indicated binary value "10200" is related to the raw value for the 16 bit digital to analog converter on the RIO board which produce the required 3.3v for the encoder. This can be done by setting the calibration mode property to the analog output ports (AO0, AO1) to "Raw" in the NI 9263 Module Properties dialog box. That is done to make the FPGA I/O node, only, take a binary value when writing to the FPGA module. In order to convert the required output voltage into binary values before writing to the AO module. By using the formula in (117),

$$BV = (V \times 10^9 – Offset) \div LSB_{Weight} \quad (117)$$

The LSB_{Weight} is calculated by,

$$LSB_{Weight} = (OutputSpan \div 2^{DACResolution}) \times 10^9 \quad (118)$$

Where, BV, is the digital value required to be written on the AO port to the FPGA I/O Node. And V, is the corresponding voltage Value that is needed at the output channel. while, Offset, this value is = 0. The LSB_{Weight}, is the value obtained from the DAC resolution and this $DACResolution = 16bit$. Eventually, the typical $Output\ Span$ is $21.4V$ for the NI 9263 Module that is included in the 9631 RIO board.
2) The second loop, Fig. 36 B named by "calculate Tilt Angle". This loop is used to read the incoming analog voltage from the ADXL335 accelerometer from x, y, and z, which are connected to AI0, AI1, and AI2, respectively.The calibration process stated in section II-C is implemented and the resulted offset values for each AI port are added in this loop in fixed point format. A flat sequence structure is used along with the loop timer to synchronize the sampling rate. The loop timer receives the value desired for sampling time from the host (Real Time Processor). Also, the inverse tan function in fixed point format is utilized to get the tilt angle around the Y-axis. which is the installed orientation of the sensor on the robot.
3) "Encoder Loop" in Fig. 36 C is a single cycle timed (SCT) loop that is used to detect the incoming quadrature signal from the two channels encoder and calculates the number of pulses per interval of ticks of the FPGA Clock, where the tick is $(1 \div 400000$ sec). In the execution of each iteration of this loop, both left and right DC motor encoder signals are detected and used for the next loop iteration through shift registers indicated by the blue color.
4) After SCTL calculates the left and right velocity (pulse/interval), the angular velocity is determined in the while loop shown in Fig. 36 D, "Angular velocity Measurement." It uses two subVIs that change the number of pulses per interval into radians per second. It also includes a loop timer to synchronize the reading of each wheel's angular velocity with the real-time VI execution.
5) The final loop is the DC pulse width modulation loop (PWM), as shown in Fig. 36 E. This PWM loop generates an output voltage at the indicated digital output ports (P9-8, P9-6) of the sbrio for both motors. This loop also controls the direction of each motor by enabling/disabling the P9-7 and P9-5 digital outputs according to the motor driver's working principle. The system consists of two flat sequence structures, each containing two loop timers. They are used to control the high and low intervals of the PWM signal. It receives the period time (Ton+Toff) according to the desired working frequency along with the control input, the duty cycle resulted from the PID controller in the RT part, to generate a signal from 0 to 3.3 V. Thereafter, it is regulated by the Cytron motor driver to a signal between 0 and 12 volts, which is the rated voltage of the DC motors.

Referring to the Host Vi indicated in Fig. 37, it is composed of a timed loop that works at a rate of 1 msec and it includes the PID controller for the robot stabilization. the following lines discuss what it could do along with the FPGA Vi. At first, Step "A" in the red circle in Fig. 37, the FPGA target calls the bit file uploaded on the flash memory. Then the output is the reference FPGA target that allows opening the FPGA and getting or sending data to the AI/A/O and the DI/DO ports to do the specified functions.

Thereafter, a flat sequence structure, at step B, takes 50 ms before passing to the FPGA node that addresses the FPGA resources (duty cycle) at step C. That sets the motor driver to supply zero voltage to avoid any unexpected voltage at the start of the FPGA target. Following that, the host runs the timed loop with the stated loop time. In this loop, the host addresses the FPGA side, step D, for the accelerometer reading, PWM duty cycle, and the direction for both motors, sensor loop rate, and the wheel angular velocities. After getting the tilt angle reading, y-angle, from the FPGA accelerometer loop, it enters the designed filter, in step E, for the accelerometer reading as discussed in section (II-C).

Following this step, the D-PID Subvi as in Fig. 38 step F receives this reading and calculates the error between it and the desired tilt set point. Then it calculates the PID control action for stabilization. Note that at each loop iteration, the shift registers are used to store the previous values of the PID control output, such as the sum of errors for the integral part, for the next controller calculations. The PID output is passed to step G. At which the motor controller Subvi manages the output control action to set direction and the duty cycle for the PWM loop on the FPGA side for stabilization of the robot in the upright position. Finally, step H ensures a zero-duty cycle for robot motors and closes the FPGA reference target if the loop terminates.

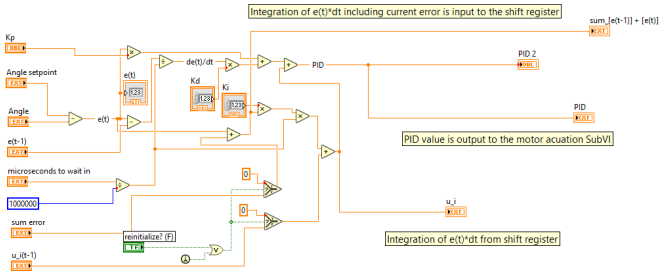

Fig. 38. D-PID Controller vi

The Practical HIL-Controller implementation of the robot stabilization on NI-Sbrio9631 is summarized in this paragraph. The PID-Controller vi in Fig. 38 was implemented on NI-LabVIEW to calculate the control action u^c each iteration of the controller loop, each t_s. Following this, fine-tuning of the D-PID parameters was done according to the designed continuous time and the discrete-time PID controllers with Matlab. We started with the proportional term, k^p, by slightly increasing its value till the robot started oscillating around the balancing set point.

At that step we carefully increased the proportional term of the controller as larger values caused the system to behave aggressively, oscillating around the set point. At this stage the damping-derivative term, k^d, was carefully increased by 0.001 till the robot started to balance without external aid. But still there was an error at the balancing step point. For that, a small integral term, k^i, was added to eliminate the steady-state error. Considering the table defined in Table IV for the effects of each term on the system response. The final implementation results, in Fig. 41, 43 and 45 showed the perfect parameters chosen to meet the response specification.

III. Results

The planar multibody model for the self-balancing robot was built. After that, Pole placement and LQR optimal controllers are designed and simulated with both the linearized and nonlinear models to follow a reference trajectory. The pole placement results ,for the wheel displacement R_x^2, the desired position was reference point $0.1\ m$. The rise time achived was one second while the maximum overshoot obtained was about 10% with the settling time of two seconds. while for desired tilt angle equilibrium point of $pi/2\ rad$ with initial error of $1.3963\ rad$ the response shows that the inverted body settles in nearly two seconds with maximum overshoot of 2% and zero steady state error. while for the LQR optimal controller, the settling and zero steady-state errors are met as specified in Table III, while the maximum overshoot is less than 25% nearly about 1% for the wheel position and about 3% for the pendillum tilt angle. And this is almost higher than the achieved in the pole placement technique. but both achieve the specified controller design parameters.

Following that step, design and simulation of the robot stabilization digital PID controller was addressed. And the results as listed in Table VI. The rise and settling times are 0.0204 and 0.196 sec respectively while the Maximum percentage overshoot is nearly about 19% which are also less than the specified design parameters. This is another indication that the controller design along with the multibody modelling with the coordinate partitioning method succeeded. After that utilization of FPGA technology in SbRio9631 single RIO board for practical hardware implementation of the PID controller on the self balancing robot prototype. Tuning trials are made for these implementations. Among these trials, we state the following successful three cases. Case 1, At first the robot is at a tilt angle of nearly -13 degrees, and the set point for stabilization is put to be $zero$. by using the PID Parameters $K_p = 12, K_i = 0.0001$ and $K_d = 0.00154$ the result is shown in Fig. 41. it shows that the robot is well stabilized after nearly 1sec with an angle error of -2 degrees. With the output to the DC motor, control action (Duty cycle), to stabilize the robot as in Fig. 42. The disturbance as shown in Fig. 41, was added between the period of 6 and 10 seconds to test the controller behaviour. That showed the robot tilt angle with steady-state error of $\pm 0.2^\circ$. In the Second Case, after some tuning with the proportional and the derivative gains then by slightly increasing the integral gain to the values of $K_p = 20, K_i = 0.00181$ and $K_d = 0.4$ it showed good and enhanced results. As it exhibits less steady-state error between $\pm 0.2^\circ$. Furthermore, the added disturbance at exactly $t = 7.5$ sec, shows that the robot stabilizes in about 4 seconds with the same steady state error. Fig. 43 is the corresponding DC motor voltage (Duty cycle). As seen, its duty cycle became between ± 0.8 and changes according to the exhibited tilt angle responding to the disturbance at the stated period of time. The last case, case 3, by changing the set point to $\theta^3 = -1.5^\circ$ the results are shown in Fig. 45 the robot exhibits a higher overshoot than in Case 1 and Case 2 But settles faster around the zero for the first 10 seconds then got settled around the $\theta^3 = -1^\circ$ and at the tenth second it settles between -1° and -1.5°. The corresponding Duty cycle is shown in Fig. 46. It is the signal sent to the DC motor driver so as to supply the value of voltage in the range of $\pm 12\ v$ corresponding to the error in the tilt angle of the inverted pendulum.

From both cases 1 and 2 for zero tilt angle set-point. It is notable, that the existence of the steady-state error in the 6 seconds before the external disturbance in Case 1, in Fig. 41, means that the controller is unable to fully reject the constant disturbances or uncertainties required to achieve the precise reference set point (zero tilt angle). This implies insufficient integral action.In contrast, Case 2 shown in Fig. 43 shows a smaller steady-state error (typically 0.2 degrees), as we finely tuned the controller parameters. which means better control accuracy and a greater ability to handle disturbances, likely

because of better tuning of the controller.

Furthermore, after fine-tuning the D-PID controller parameters on real-time FPGA hardware, the response to the selected parameters $K_p = 20, K_i = 0.00181$ and $K_d = 0.4$, case 2, in MATLAB simulink showed that the practical results nearly match the linearized model in both Fig. 39 and Fig. 40. It shows a higher overshoot in the linear model responding to the step signal but immediately settles with a very small steady-state error. This demonstrates the effects of the linearization process and other uncertainties in modelling and real hardware components.

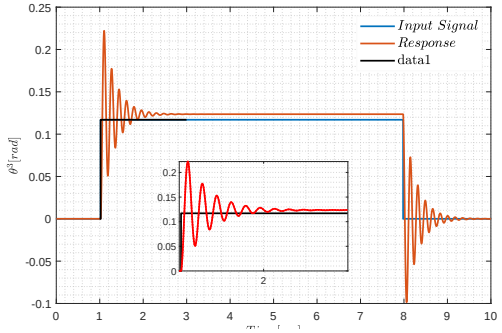

Fig. 39. Step response PID DT actual parameters in the theoretical modle

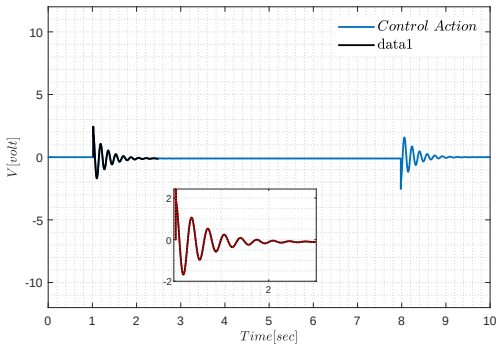

Fig. 40. Control Action

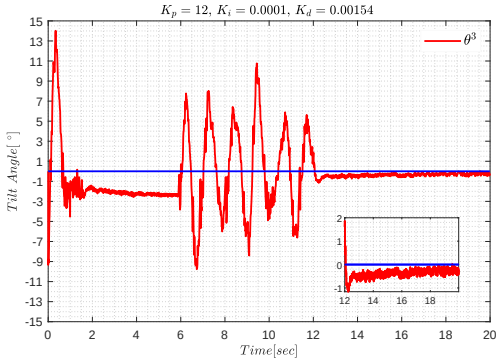

Fig. 41. Tilt Angle at the Setpoint $\theta^3 = 0^\circ$

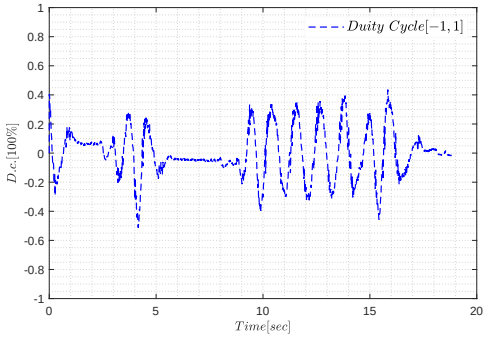

Fig. 42. Control Action (Duity cycle) at the Setpoint $\theta^3 = 0^\circ$

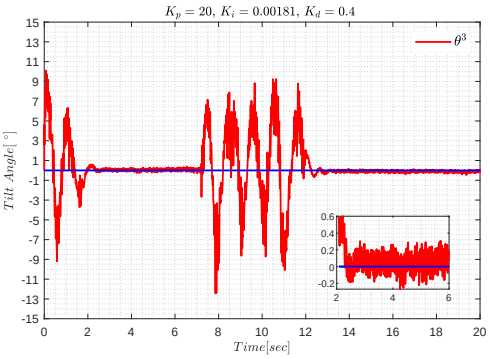

Fig. 43. Tilt Angle at the Setpoint $\theta^3 = 0^\circ$

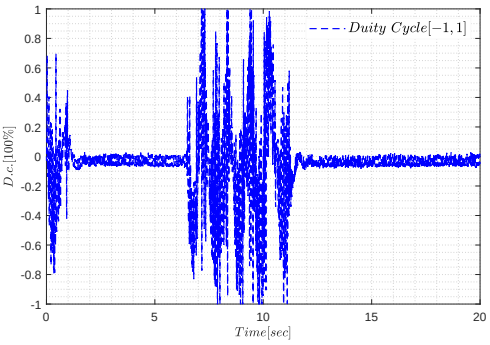

Fig. 44. Control Action (Duity cycle) at the Setpoint $\theta^3 = 0^\circ$

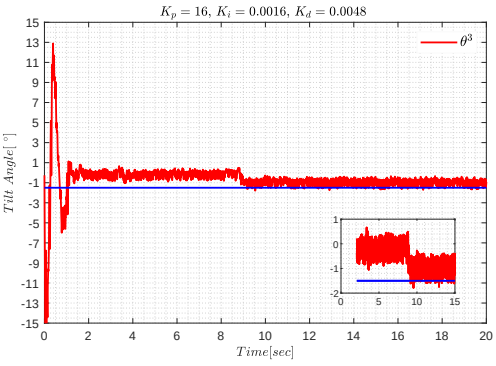

Fig. 45. Tilt Angle at the Setpoint $\theta^3 = -1.5^\circ$

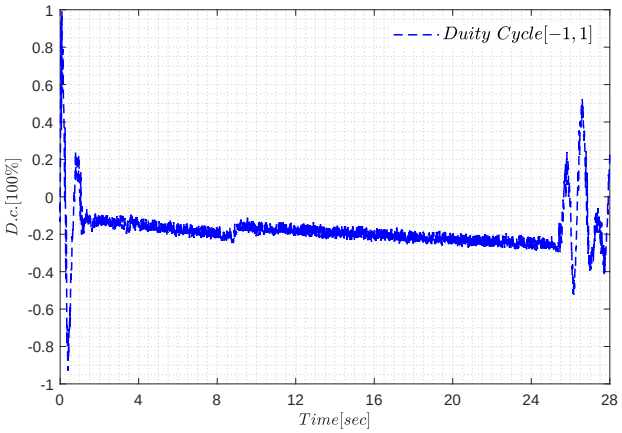

Fig. 46. Control Action (Duity cycle) at the Setpoint $\theta^3 = -1.5^\circ$

In addition to that for the practical implementation the robot started balancing from angle of nearly about -15 degrees and settled in nearly 2 seconds at the zero set point with error of nearly about ± 0.4 degrees. In addition to that external disturbance was added to the robot during the stabilization process that caused error in the tilt angle by 9 degrees and it showed a good response and fast settling nearly at 4 seconds to the set point. That ensures the effectiveness of the designed controller against the external unexpected disturbances for the proposed robot.

Also, we should state that dealing with FPGAs and the calculations based on the fixed-point format in dealing with variables and the reading from the analog port of the accelerometer values consumed relatively great experimental efforts to adjust the accelerometer reading. Also, the PWM frequency determination with the 3.3V TTL DIO of the sbrio 9631 for the smooth operation of the DC motors was another encountered problem. Furthermore, the availability of the expensive FPGA board was one of the limitations of the chosen approach. However, this limitation can be mitigated with adequate funding sources.

Furthermore, we tried to use the digital accelerometer and gyro sensors, which need I2C or SPI protocols that are not supported by the sbrio 9631 FPGA. However, we built the I2C protocol on the FPGA target and got an accurate filtered signal for both the tilt angle and its angular velocity. Due to the limited memory of 64 kb, the I2C consumed nearly all the available FPGA resources.

This resulted in the inability to perform any other functions on the FPGA, including the accelerometer, PWM, encoder, and velocity determination loops, as well as the PID controller, digital filter, and motor actuation subVis. But for large military and transportation sectors, it isn't a hard task to get these NI-based RIO FPGAs available. For the computational method of the augmented formulation of the multibody model with baumgrate stabilization trials for selection of the parameters α and β has been made till the best selection $\alpha = \beta = 10$ results in acceptable violations $\pm 10^{-6}$ and $\pm 10^{-4}$ for both holonomic and nonholonomic constraints.

IV. Conclusion

In conclusion, multibody modelling and simulation is a great approach for model-based controller design and simulation. We managed to use the coordinate partitioning method in the MBDs 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 managed to build the planar multibody model of the wheeled balancing robot.

Also, the validation of this model was done through the simulation of the dynamic model. Furthermore, state feedback and optimal controllers were designed and simulated for both linearized and nonlinear MBD models. The digital PID controller based on the linearized model was designed, simulated, and implemented on NI-SbRIO with the LabVIEW FPGA module. Case 2 shows the effectiveness of the designed controller. It results in steady-state error of $\pm 0.2^\circ$ at two second when started balancing from $\pm 15^\circ$. And on the application of external disturbance it recovers the balancing condition in 4 second. This indicates a good degree of model robustness although linearization and approximations of the DC motor model.

Which means better control accuracy and a greater ability to handle disturbances, mainly because of better tuning of the controller. Which also validate the dynamic modelling approach and the designed control system. The results of the experimental work ensure that the linearization of the multibody model is valid up to tilt angles of $\pm 15^\circ$ around the equilibrium set point ($\theta^3 = 0^\circ$) according to the limitations of DC motor torque output specifications. Additionally, we managed to get an accurate tilt angle measurement from cost effective ADXL335 sensor through software based filter design. Also, before the deployment, the sensor was calibrated manually as stated in (section III-C) to get the offsets error.

From this study we can conclude that fast stabilization of the under-actuated mechatronic systems is very important and has wide applications in the transportation, industrial and military sectors. In these sectors goods transportation in confined places and harbours beside the smart warehouses address these applications. Furthermore, the balancing and stabilization task is a crucial task, especially for human transportation using segways inside small tourist villages, hospitals, and hotels. This task is also particularly crucial for military applications, such as mobile robot gunners or snipers operating in high-risk environments.

All the mentioned mechatronics applications hold significant importance and necessitate a well-designed stabilization controller, which is the primary focus of our research. So, the followed approach could be extended for future work by applying the optimally designed LQR controller, but in the

presence of more available sensors, such as digital gyros, and another modern version of the NI-RIO FPGA board, such as the NI-myRIO 1900, that provides the interface for UART, I2C, and SPI protocols. In addition, it includes a Wi-Fi module, which is important for data transmission with PCs or sensors. The multi-body model parameter sensitivity analysis is another research point that would be addressed for the adopted application to ensure the model robustness. Finally, we can equip the wheeled self-balancing robot with it with a camera and AI detection tools to facilitate exploration utilizing FPGA technology.

APPENDIX

$$A_{32} = \frac{(\ L^2 {R_w}^2 g m^{3^2})}{a_1} \tag{119}$$

$$A_{42} = \frac{(L\ gm^3\ (I^2{}_{zz} + \ {R_w}^2 m^2 + \ {R_w}^2 m^3))}{a_1} \tag{120}$$

$$A_{34} = \frac{-k_t k_b {N_G}^2 (\ L^2 m^3 + I^3{}_{zz})}{a_2} \tag{121}$$

$$A_{44} = \frac{-k_t k_b {N_G}^2 L\ m^3}{a_2} \tag{122}$$

$$B_{31} = \frac{(k_t N_G R_w (m^3 L^2 + I^3{}_{zz}))}{a_2} \tag{123}$$

$$B_{34} = \frac{(k_t N_G\ L\ R_w m^3)}{a_1} \tag{124}$$

where, $a_1 = (I^3{}_{zz}\ I^2{}_{zz} + \ I^2{}_{zz} L^2 m^3 \ + I^3{}_{zz} {R_w}^2 m^2 \ + I^3{}_{zz} {R_w}^2 m^3 \ + \ L^2 {R_w}^2 m^2 m^3\ )$
$a_2 = R_a (I^3{}_{zz}\ I^2{}_{zz} + \ I^2{}_{zz} L^2 m^3 \ + I^3{}_{zz} {R_w}^2 m^2 \ + I^3{}_{zz} {R_w}^2 m^3 \ + \ L^2 {R_w}^2 m^2 m^3\ )$

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)

REFERENCES

[1] A. Hamed, E. Shaban, Abdelhaleem, and A. A. ghany, "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*, vol. 46, no. 3, pp. 743–751, 2022, doi: 10.1007/s40998-022-00488-3.

[2] E. Shaban, A. Hamed, A. Bassiuny, and A. Abdel ghany, "On implementation of nonlinear pid+ controller embedded on fpga module for industrial system," *International Journal of Dynamics and Control*, vol. 12, no. 7, pp. 2331–2340, 2024, doi: 10.1007/s40435-023-01338-8.

[3] A. A. Nada and M. A. Bayoumi, "Development of embedded fuzzy control using reconfigurable fpga technology," *Automatika*, vol. 65, no. 2, pp. 609–626, 2024, doi: 10.1080/00051144.2024.2313904.

[4] E. S. Ghith and F. A. A. Tolba, "Design and optimization of pid controller using various algorithms for micro-robotics system," *Journal of Robotics and Control*, vol. 3, no. 3, pp. 244–256, 2022, doi: 10.18196/jrc.v3i3.14827.

[5] D. Ziouzios, P. Kilintzis, N. Baras, and M. Dasygenis, "A survey of fpga robotics applications in the period 2010–2019," *Advances in Science Technology and Engineering Systems Journal*, vol. 6, pp. 385–408, 2021, doi: 10.25046/aj060344.

[6] Z. Chen, "Application and optimization of fpga-based real-time motion control systems in robotics," in *IET Conference Proceedings CP895*, vol. 2024, pp. 481–486, 2024, doi: 10.1049/icp.2024.4028.

[7] R. Tsutada, T.-T. Hoang, and C.-K. Pham, "An obstacle avoidance two-wheeled self-balancing robot," *International Journal of Mechanical Engineering and Robotics Research*, vol. 11, no. 1, pp. 1–7, 2022, doi: 10.18178/ijmerr.11.1.1-7.

[8] J. Ordóñez Cerezo, E. Castillo Morales, and J. M. Cañas Plaza, "Control system in open-source fpga for a self-balancing robot," *Electronics*, vol. 8, no. 2, 2019, doi: 10.3390/electronics8020198.

[9] C.-H. Huang, Y.-C. Chen, C.-Y. Hsu, J.-Y. Yang, and C.-H. Chang, "Fpga-based uav and ugv for search and rescue applications: A case study," *Computers and Electrical Engineering*, vol. 119, 2024, doi: 10.1016/j.compeleceng.2024.109491.

[10] A. Ghorbel, N. Ben Amor, and M. Jallouli, "Design of a flexible reconfigurable mobile robot localization system using fpga technology," *SN Applied Sciences*, vol. 2, no. 7, 2020, doi: 10.1007/s42452-020-2960-4.

[11] M. A. Pastrana Triana, M. S. Santana, J. A. Mendoza Peñaloza, L. H. Nunes de Oliveira, and D. M. Muñoz Arboleda, "Comparison of gmdh and perceptron controllers for mobile robot obstacle following/avoidance with hardware-in-the-loop validation," *Journal of Intelligent & Robotic Systems*, vol. 111, no. 1, pp. 1–23, 2025, doi: 10.1007/s10846-025-02239-y.

[12] E. Galan-Uribe, L. Morales-Velazquez, and R. A. Osornio-Rios, "Fpga-based methodology for detecting positional accuracy degradation in industrial robots," *Applied Sciences*, vol. 13, no. 14, 2023, doi: 10.3390/app13148493.

[13] A. Fekik, H. Khati, A. T. Azar, M. L. Hamida, H. Denoun, I. A. Hameed, and N. A. Kamal, "Fpga in the loop implementation of the puma 560 robot based on backstepping control," *IET Control Theory & Applications*, vol. 18, no. 15, pp. 1877–1891, 2024, doi: 10.1049/cth2.12589.

[14] A. A. Nada, V. Parque, and M. A. Bayoumi, "Accelerating the performance of fuzzy-fpga based control in labview for trajectory tracking problems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3386–3391, 2023. doi: 10.1016/j.ifacol.2023.10.1486.

[15] C. E. Marquez-Garcia, J. Lopez-Gomez, F. Martinez-Solis, M. A. D. Vargas-Treviño, S. Vergara-Limon and V. M. Velazquez-Aguilar, "FPGA-based accuracy mechatronics of a feed-drive system with ball screw," *2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2021, pp. 1-6, doi: 10.1109/CCE53527.2021.9633104.

[16] M. C. García-López *et al*., "Dynamic performance by mathematical modeling and trajectory tracking control in an fpga-based architecture for an xyz cartesian system," *The International Journal of Advanced Manufacturing Technology*, vol. 136, pp. 3431–3450, 2025, doi: 10.1007/s00170-025-15049-1.

[17] R. W. Hamad, M. A. Qasim, and S. Raed, "Fpga-based charging stations for electric vehicles: A review," *JMCER*, vol. 2024, pp. 68–81, 2024.

[18] A. Natsheh *et al*., *Stand-alone Electric Vehicle Charging Station Using FPGA*, Okipublishing, 2022, doi: 10.55432/978-1-6692-0001-7_11.

[19] J. Koko, B. Onana Essama, J. Ngo Bisse, J. Atangana, and S. Ndjakomo Essiane, "A new hardware implementation of fuzzy logic control based on fpga for shunt active filter: Design and co-simulation," *World Journal of Artificial Intelligence and Robotics Researc*, vol. 14, 2025.

[20] A. A. Nada and A. H. Bashiri, "Integration of multibody system dynamics with sliding mode control using fpga technique for trajectory tracking problems," in *Dynamic Systems and Control Conference*, 2018, doi: 10.1115/DSCC2018-9108.

[21] M. E. Iranian, E. Zamiri, and A. de Castro, "Optimal implementation of d-q frame finite control set model predictive control with labview," *Electronics*, vol. 14, no. 1, 2024, doi: 10.3390/electronics14010100.

[22] S. Bourhnane *et al*., "Real-time Control of Smart Grids using NI CompactRIO," *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pp. 1-6, 2019, doi: 10.1109/WITS.2019.8723840.

[23] G. TATAR, H. KORKMAZ, N. F. O. SERTELLER and K. TOKER, "LabVIEW FPGA Based BLDC Motor Control by Using Field Oriented Control Algorithm," *2020 International Conference on Smart Energy Systems and Technologies (SEST)*, pp. 1-6, 2020, doi: 10.1109/SEST48500.2020.9203104.

[24] K. P. Seng, P. J. Lee, and L. M. Ang, "Embedded intelligence on fpga: Survey, applications and challenges," *Electronics*, vol. 10, no. 8, 2021, doi: 10.3390/electronics10080895.

[25] C. Wang and Z. Luo, "A review of the optimal design of neural networks based on fpga," *Applied Sciences*, vol. 12, no. 21, 2022, doi: 10.3390/app122110771.

[26] A. Nechi, L. Groth, S. Mulhem, F. Merchant, R. Buchty, and M. Berekovic, "Fpga-based deep learning inference accelerators: Where are we standing?," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 4, pp. 1–32, 2023, doi: 10.1145/3613963.

[27] F. Liu, H. Li, W. Hu, and Y. He, "Review of neural network model acceleration techniques based on fpga platforms," *Neurocomputing*, vol. 610, 2024, doi: 10.1016/j.neucom.2024.128511.

[28] R. Langley, "Development of a self-balancing robot utilizing fpga," *Engineering Honors Thesis, School of Engineering and Information Technology, Murdoch University, Perth, Western Australia*, 2016.

[29] P. Ponce-Cruz, A. Molina, B. MacCleery, P. Ponce-Cruz, A. Molina, and B. MacCleery, "Fuzzy logic type 1 and type 2 labview fpga toolkit," *Fuzzy Logic Type 1 and Type 2 Based on LabVIEW™ FPGA*, pp. 159–230, 2016, doi: 10.1007/978-3-319-26656-5_4.

[30] H. Y. Irwanto *et al*., "Model predictive control in hardware in the loop simulation for the onboard attitude determination control system," *Journal of Robotics and Control*, vol. 5, no. 2, pp. 533–541, 2024, doi: 10.18196/jrc.v5i2.21613.

[31] S. Sarathy, M. M. Mariyam Hibah, S. Anusooya and S. Kalaivani, "Implementation of Efficient Self Balancing Robot," *2018 International Conference on Recent Trends in Electrical, Control and Communication (RTECC)*, pp. 65-70, 2018, doi: 10.1109/RTECC.2018.8625624.

[32] B. N. K. Reddy *et al*., "An efficient approach for design and testing of FPGA programming using Lab VIEW," *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 543-548, 2015, doi: 10.1109/ICACCI.2015.7275665.

[33] Z. M. Kassas, "Methodologies for implementing fpga-based control systems," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 9911–9916, 2011, doi: 10.3182/20110828-6-IT-1002.00783.

[34] A. G. Agúndez, D. García-Vallejo and E. Freire, "An electric kickscooter multibody model: equations of motion and linear stability analysis," *Multibody System Dynamics*, vol. 62, pp. 493–524, 2024, doi: 10.1007/s11044-024-09974-4.

[35] A. Agúndez, A. Saccon, D. García-Vallejo, and E. Freire, "Dynamic inversion and optimal tracking control on the ball-plate system based on a linearized nonholonomic multibody model," *Mechanism and Machine Theory*, vol. 203, 2024, doi: 10.1016/j.mechmachtheory.2024.105795.

[36] A. A. Nada and A. H. Bishiri, "Multibody system design based on reference dynamic characteristics: gyroscopic system paradigm," *Mechanics Based Design of Structures and Machines*, vol. 51, no. 6, pp. 3372–3394, 2023, doi: 10.1080/15397734.2021.1923526.

[37] A. Agúndez, D. García-Vallejo and E. Freire, "Linear stability analysis of nonholonomic multibody systems," *International Journal of Mechanical Sciences*, vol. 198, 2021, doi: 10.1016/j.ijmecsci.2021.106392.

[38] Q. Bai, M. Shehata, A. Nada and Z. Shao, "Development of dynamics for design procedure of novel grating tiling device with experimental validation," *Applied Sciences*, vol. 11, no. 24, 2021, doi: 10.3390/app112411716.

[39] M. A. Özlen and O. Özener, "Integrated multibody dynamics and experimental validation study for a novel heavy-duty diesel engine valvetrain design," *Applied Sciences*, vol. 15, no. 6, 2025, doi: 10.3390/app15063262.

[40] S. Wang, Q. Cheng, L. Wang, J. Xu, X. Fang, and J. Kang, "Research on the rapid design technology of automotive inspection structure based on mbd model," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 8, pp. 1670–1686, 2022, doi: 10.1177/09544070211065201.

[41] X. Jin, H. Liu, G. Dong, K. Xu, and X. Ding, "Contact dynamics modeling and control during the combination process of air-ground robots," *Chinese Journal of Mechanical Engineering*, vol. 38, no. 39, pp. 1–17, 2025, doi: 10.1186/s10033-025-01178-x.

[42] C. M. Pappalardo and D. Guida, "On the dynamics and control of underactuated nonholonomic mechanical systems and applications to mobile robots," *Archive of Applied Mechanics*, vol. 89, pp. 669–698, 2019, doi: 10.1007/s00419-018-1491-6.

[43] G. Piva, D. Richiedei and A. Trevisani, "Model inversion for trajectory control of reconfigurable underactuated cable-driven parallel robots," *Nonlinear Dynamics*, vol. 113, pp. 8697–8712, 2025, doi: 10.1007/s11071-024-10753-1.

[44] A. A. Nada, "Simplified procedure of sensitivity-based parameter estimation of multibody systems with experimental validation," *IFAC-PapersOnLine*, vol. 54, no. 14, pp. 84–89, 2021, doi: 10.1016/j.ifacol.2021.10.333.

[45] F. Oexle, A. Benfer, A. Puchta, and J. Fleischer, "Jacobian-sensitivity approach for identifying machine dynamic model parameters of robots with flexible joints," *Procedia CIRP*, vol. 127, pp. 116–121, 2024, doi: 10.1016/j.procir.2024.07.021.

[46] B. Müller and O. S. and, "Dynamic multibody model of a turntable ladder truck considering unloaded outriggers and sensitivity-based parameter identification," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 30, no. 1, pp. 567–590, 2024, doi: 10.1080/13873954.2024.2361011.

[47] S. Yang, F. Wu, and J. Zhao, "Mbds: A multi-body dynamics simulation dataset for graph networks simulators," *arXiv*, 2024, doi: 10.48550/arXiv.2410.03107.

[48] S. I. Jang, S. Han, J.-G. Kim, J. Choi, S. Rhim, and J. H. Choi, "Sidnet: machine learning based system identification framework for rigid and flexible multibody dynamics," *Multibody System Dynamics*, pp. 1–22, 2024, doi: 10.1007/s11044-024-10043-z.

[49] T. Slimak, A. Zwölfer, B. Todorov and D. Rixen, "A machine learning approach to simulate flexible body dynamics," *Multibody System Dynamics*, pp. 1–27, 2025, doi: 10.1007/s11044-024-10049-7.

[50] A. A. Shabana, K. E. Zaazaa and H. Sugiyama, *Railroad vehicle dynamics: a computational approach*, CRC press, 2007, doi: 10.1201/9781420045857.

[51] A. A. Shabana, *Mathematical foundation of railroad vehicle systems: geometry and mechanics*, John Wiley & Sons, 2021.

[52] A. A. Shabana, *Dynamics of multibody systems*, Cambridge university press, 2020.

[53] K. Augustynek and A. Urbaś, "Numerical investigation on the constraint violation suppression methods efficiency and accuracy for dynamics of mechanisms with flexible links and friction in joints," *Mechanism and Machine Theory*, vol. 181, 2023, doi: 10.1016/j.mechmachtheory.2022.105211.

[54] I. Abdel-Hady, M. A. Bayoumi, N. A. Mansour and A. A. Nada, "Stabilization of Driving Velocity Constraints for Self-balanced Robot," *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 2354-2359, 2024, doi: 10.1109/CoDIT62066.2024.10708340.

[55] F. Marques, A. P. Souto, and P. Flores, "On the constraints violation in forward dynamics of multibody systems," *Multibody System Dynamics*, vol. 39, pp. 385–419, 2017, doi: 10.1007/s11044-016-9530-y.

[56] A. Nada and M. Bayoumi, "Development of a constraint stabilization method of multibody systems based on fuzzy logic control," *Multibody System Dynamics*, vol. 61, pp. 233–265, 2024, doi: 10.1007/s11044-023-09921-9.

[57] P. E. Nikravesh, *Computer-aided analysis of mechanical systems*, Prentice-Hall, 1988.

[58] P. Nikravesh, *Planar multibody dynamics: formulation, programming with MATLAB®, and applications*, CRC press, 2018.

[59] R. A. Wehage and E. J. Haug, "Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems," vol. 104, no. 1, pp. 247–255, 1982, doi: 10.1115/1.3256318.

[60] A. A. Nada and A. H. Bashiri, "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, 2017, doi: 10.1115/DETC2017-67476.

[61] J. Wu, X. Guo, D. Zhang, Y. Zhang, L. Li, and K. Guo, "A unified numerical solution framework for solving daes of multibody system dynamics with holonomic and nonholonomic constraints," *Nonlinear Dynamics*, vol. 113, pp. 14887–14916, 2025, doi: 10.1007/s11071-025-10893-y.

[62] M. Khoshnazar, M. Dastranj, A. Azimi, M. M. Aghdam and P. Flores, "Application of the bezier integration technique with enhanced stability in forward dynamics of constrained multibody systems with baumgarte stabilization method," *Engineering with Computers*, vol. 40, no. 3, pp. 1559–1573, 2024, doi: 10.1007/s00366-023-01884-x.

[63] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer methods in applied mechanics and engineering*, vol. 1, no. 1, pp. 1–16, 1972, doi: 10.1016/0045-7825(72)90018-7.

[64] J. G. De Jalon and E. Bayo, *Kinematic and dynamic simulation of multibody systems: the real-time challenge*, Springer, New York, 1994.

[65] S.-T. Lin and M.-W. Chen, "A pid type constraint stabilization method for numerical integration of multibody systems," *Journal of Computational and Nonlinear Dynamics*, vol. 6, no. 4, 2011, doi: 10.1115/1.4002688.

[66] G. V. Raffo, M. G. Ortega, V. Madero and F. R. Rubio, "Two-wheeled self-balanced pendulum workspace improvement via underactuated robust nonlinear control," *Control Engineering Practice*, vol. 44, pp. 231–242, 2015, doi: 10.1016/j.conengprac.2015.07.009.

[67] Y. Tian, L. Jiang, L. Ming and H. Bin, "LabVIEW and CRIO Linear Control of a Coaxial Two-wheeled Mobile Robot," *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, pp. 463-466, 2011, doi: 10.1109/ICMTMA.2011.402.

[68] Á. Odry, R. Fullér, I. J. Rudas and P. Odry, "Fuzzy control of self-balancing robots: A control laboratory project," *Computer Applications in Engineering Education*, vol. 28, no. 3, pp. 512–535, 2020, doi: 10.1002/cae.22219.

[69] T. Hoang, D. Kim, V. Pham, X. Dinh and H. X. Le, "Designing an adaptive controller for two-wheeled self-balancing mobile robot using hierarchical sliding control strategy and radial basis function neural network," *HUI Journal of Science and Technology*, vol. 57, pp. 39–48, 2021.

[70] G. P. Neves, B. A. Angélico, and C. M. Agulhari, "Robust h2 controller with parametric uncertainties applied to a reaction wheel unicycle," *International Journal of Control*, vol. 93, no. 10, pp. 2431–2441, 2020, doi: 10.1080/00207179.2018.1562224.

[71] J. Vries, *Redesign & implementation of a moment exchange unicycle robot*, University of Twente Student Theses, 2018.

[72] D. T. Tran, N. M. Hoang, N. H. Loc, Q. T. Truong and N. T. Nha, "A fuzzy lqr pid control for a two-legged wheel robot with uncertainties and variant height," *Journal of Robotics and Control*, vol. 4, no. 5, pp. 612–620, 2023, doi: 10.18196/jrc.v4i5.19448.

[73] K. Nader and D. Sarsri, "Modelling and control of a two-wheel inverted pendulum using fuzzy-pid-modified state feedback," *Journal of Robotics*, vol. 2023, no. 1, 2023, doi: 10.1155/2023/4178227.

[74] Y. Daud, A. Al Mamun, and J.-X. Xu, "Dynamic modeling and characteristics analysis of lateral-pendulum unicycle robot," *Robotica*, vol. 35, no. 3, pp. 537–568, 2017, doi: 10.1017/S0263574715000703.

[75] C.-H. Chiu and Y.-T. Hung, "One wheel vehicle real world control based on interval type 2 fuzzy controller," *Mechatronics*, vol. 70, 2020, doi: 10.1016/j.mechatronics.2020.102387.

[76] A. Abdelgawad, T. Shohdy, and A. Nada, "Model- and data-based control of self-balancing robots: Practical educational approach with labview and arduino," *IFAC-PapersOnLine*, vol. 58, no. 9, pp. 217–222, 2024, doi: 10.1016/j.ifacol.2024.07.399.

[77] N. L. Tao *et al*., "Optimization of hierarchical sliding mode control parameters for a two-wheeled balancing mobile robot using the firefly algorithm," *Journal of Robotics and Control*, vol. 6, no. 1, pp. 76–88, 2025, doi: 10.18196/jrc.v6i1.24192.

[78] A. García-Agúndez, D. García-Vallejo, and E. Freire, "Linearization approaches for general multibody systems validated through stability analysis of a benchmark bicycle model," *Nonlinear Dynamics*, vol. 103, no. 1, pp. 557–580, 2021, doi: 10.1007/s11071-020-06069-5.

[79] I. Roupa, S. B. Gonçalves and M. T. da Silva, "Kinematics and dynamics of planar multibody systems with fully cartesian coordinates and a generic rigid body," *Mechanism and Machine Theory*, vol. 180, 2023, doi: 10.1016/j.mechmachtheory.2022.105134.

[80] A. Nada and H. El-Hussieny, "Development of inverse static model of continuum robots based on absolute nodal coordinates formulation for large deformation applications," *Acta Mechanica*, vol. 235, no. 4, pp. 1761–1783, 2024, doi: 10.1007/s00707-023-03814-w.

[81] X. Bajrami, A. Pajaziti, R. Likaj, A. Shala, R. Berisha and M. Bruqi, "Control theory application for swing up and stabilisation of rotating inverted pendulum," *Symmetry*, vol. 13, no. 8, 2021, doi: 10.3390/sym13081491.

[82] M.-T. Vo *et al*., "Combining passivity-based control and linear quadratic regulator to control a rotary inverted pendulum," *Journal of Robotics and Control*, vol. 4, no. 4, pp. 479–490, 2023, doi: 10.18196/jrc.v4i4.18498.

[83] K. D. Papadimitriou, N. Murasovs, M. E. Giannaccini, and S. Aphale, "Genetic algorithm-based control of a two-wheeled self-balancing robot," *Journal of Intelligent & Robotic Systems*, vol. 111, no. 34, pp. 1–20, 2025, doi: 10.1007/s10846-025-02236-1.

[84] A. R. Hambley, *Electrical Engineering: Principles & Applications*, Pearson, 2021.

[85] A. A. A. M. A. Ashna Batool, Noor ul Ain and M. H. Shahbaz, "A comparative study of dc servo motor parameter estimation using various techniques," *Automatika*, vol. 63, no. 2, pp. 303–312, 2022, doi: 10.1080/00051144.2022.2036935.

[86] H.-G. Lee, *Linearization of nonlinear control systems*, Springer, 2022.

[87] I. Nagrath and M. Gopal, *Control Systems Engineering*, New Age International Pvt Ltd, 2025.

[88] R. Dorf and R. Bishop, *Modern Control Systems*, Pearson, 2022.

[89] J. A. Rojas-Quintero, F. Dubois, H. C. Ramírez-de Ávila, E. Bugarin, B. Sánchez-García, and N. R. Cazarez-Castro, "Analysis of a dry friction force law for the covariant optimal control of mechanical systems with revolute joints," *Mathematics*, vol. 12, no. 20, 2024, doi: 10.3390/math12203239.

[90] C. A. Manrique Escobar, C. M. Pappalardo, and D. Guida, "A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole," *Applied Sciences*, vol. 10, no. 24, 2020, doi: 10.3390/app10249013.

[91] M. D. Tran and V. H. Nguyen, "A novel integral interconnection damping assignment passivity-based control approach for underactuated inverted pendulum system," *Journal of Robotics and Control*, vol. 6, no. 2, pp. 649–665, 2025, doi: 10.18196/jrc.v6i2.24812.

[92] J. Huang, M. Zhang and T. Fukuda, *Robust and Intelligent Control of a Typical Underactuated Robot: Mobile Wheeled Inverted Pendulum*, Springer Nature, 2023.

[93] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Pearson, 2018.

[94] M. R. Islam, M. R. T. Hossain, and S. C. Banik, "Synchronizing of stabilizing platform mounted on a two-wheeled robot," *Journal of Robotics and Control*, vol. 2, no. 6, pp. 552–558, 2021, doi: 10.18196/jrc.26136.

[95] R. Kristiyono and W. Wiyono, "Autotuning fuzzy pid controller for speed control of bldc motor," *Journal of Robotics and Control*, vol. 2, no. 5, pp. 400–407, 2021, doi: 10.18196/jrc.25114.

[96] E. Li, J. Zhong, J. Jian, Y. Hao and H. Chang, "On enhancing the accuracy of inclinometer based on multiple dual-axis mems accelerometers fusion," *Journal of Mechanical Science and Technology*, vol. 39, pp. 1329–1337, 2025, doi: 10.1007/s12206-025-0227-0.

[97] K. H. Jatakar, G. Mulgund, A. Patange, B. Deshmukh, K. S. Rambhad, A. Saxena and V. Kalbande, "Vibration monitoring system based on adxl335 accelerometer and arduino mega2560 interface," *Journal of Algebraic Statistics*, vol. 13, no. 2, pp. 2291–2301, 2022.