**ROS-based Controller for a Two-Wheeled Self-Balancing Robot**

Juan Díaz-Téllez ¹, Ruben Senen García-Ramírez², Jairo Pérez-Pérez ³, Jaime Esteez-Carréon ⁴, Miguel Angel Carreón-Rosales⁵

Electronics Dept., National Technology Institute of Mexico/“Instituto Tecnológico de Puebla”, Pue MX

Email: ¹ juan.diaz@puebla.tecnm.mx, ² rubensenen.garcia@puebla.tecnm.mx, ³ jairo.perez@puebla.tecnm.mx, ⁴ jaime.estevez@puebla.tecnm.mx, ⁵ miguelangel.rosales@puebla.tecnm.mx

*Corresponding author*

---

**Abstract**—In this article, a controller based on a Robot Operating System (ROS) for a two-wheeled self-balancing robot is designed. The proposed ROS architecture is open, allowing the integration of different sensors, actuators, and processing units. The low-cost robot was designed for educational purposes. It used an ESP32 microcontroller as the central unit, an MPU6050 Inertial Measurement Unit sensor, DC motors with encoders, and an L298N integrated circuit as a power stage. The mathematical model is analyzed through Newton-Euler and linearized around an equilibrium point. The control objective is to self-balance the robot to the vertical axis in the presence of disturbances. The proposed control is based on a bounded saturation, which is lightweight and easy to implement in embedded systems with low computational resources. Experimental results are performed in real-time under regulation, conditions far from the equilibrium point, and rejection of external disturbances. The results show a good performance, thus validating the mechanical design, the embedded system, and the control scheme. The proposed ROS architecture allows the incorporation of different modules, such as mapping, autonomous navigation, and manipulation, which contribute to studying robotics, control, and embedded systems.

**Keywords**—ROS; Mobile robot; Two-wheeled self-balancing; ESP32; Segway.

---

**I. Introduction**

In recent years, autonomous mobile robots have been a matter of research in the academy and industry. Due to their maneuverability, precision, autonomy, loading capacity, and flexibility, the mobile robots have been used for different tasks, for example, in agriculture [18], [19], [20], drawing applications [25], [26], [27], personal mobility [36], [37], [38], cooperative tasks [21], [22], [23], [24], package delivery [28], [29], surveillance [30], [31], [32], [33] and search and rescue [34], [35].

Mobile robots have different configurations depending on the type of wheels used (drive, passive and omnidirectional) and design. For example, mobile robots with differential drive [1], [2], [3], [4], [5], [6] Ackermann steering drive [7], [8], [9], [10] and omnidirectional drive [11], [12], [13], [15], [16], [14].

Nevertheless, the development of new configurations has grown rapidly in the last decade.

In particular, mobile inverted pendulum robots (MIPR) have been widely accepted in different applications. Its two independent drive motors allow it to balance and drive in a straight line, turn on its axis, curve, and cross small slopes. From a control perspective, MIPRs are unstable, nonlinear, and underactuated systems [17].

In the scientific literature, we can find different control algorithms for mobile robots. Most dedicated to the design of linear and time-invariant controls (LTI). For example, in [39], [40], [41], [42], [43], [44] a proportional-integral-derivative PID control is proposed. An optimal quadratic linear regulator LQR at [45], [47], [48], [49], [50]. A cascade proportional-derivative control with a proportional-integral control is proposed in [41].

In [46], two controls for a MIPR are designed to compare their performance, a PID controller and an optimal LQR controller. The results show a settling time of 0.3 seconds in the PID, a steady state error close to zero with an overdamping of 20%. In addition, it only allows a deviation error of 5 degrees from the vertical axis of the pendulum. On the other hand, the optimal LQR control shows better results, reducing the overshoot observed in the PID and improving the response in a steady and transient state.

Because LTI controls are susceptible to external disturbances, unmodeled dynamics, and uncertainties, different nonlinear control strategies have been designed. For example, Backstepping [51], [52], [53], [54]. Sliding mode control (SMC) [55], [56], [57], [58], adaptive control [63], [60], [62], [61], [63], fuzzy logic control in [69], [67], [68], [66], and antidisturbance control [64], [65]. However, with the increasing complexity of the tasks assigned to this type of robot, it is necessary to implement effective communication between their processes.

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. ROS acts as a meta-operating system for robots, providing hardware
abstraction, low-level device control, inter-processes message-passing, and package management [81]. ROS allows you to manage the different layers and parallelize the processes and their communication with sensors, actuators, and processing units. Few papers cover ROS middleware for robot scalability. Some ROS-based works can be found at [72], [73], [74], [76], [77], [78]. For example, in [70] proposes implementing the ROS service to generate polynomial trajectories. Results are made in the open field for an agricultural robot. To the same extent, an adaptive Monte Carlo Localization (AMCL) algorithm is designed to navigate a mobile robot indoors in [80].

At [71], a ROS-based control for a UGV with a handling arm is used autonomously for easy pest detection and selective treatment tasks. In [79], a mobile robot is used to build a map of the robot’s physical environment. A SLAM algorithm and a Unity3D environment are used with the ROS middleware. In [75], a route planner is designed together with a control of pose regulation based on ROS for an Ackerman mobile robot. Physical tests demonstrate ROS-based navigation and control capabilities for agriculture applications.

The research topic and the contribution of this work focus on developing an open architecture based on ROS that allows the integration of different modules such as control, manipulation, navigation, and perception for mobile robots. This architecture contains sensors, a low-level controller, and a communication layer. This architecture reduces the development time to perform robotic applications due to the efficient reuse of standard modules and robot platforms. To validate the ROS-based architecture, a bounded saturation LTI controller is proposed. The proposed controller is compared with the classical PID and LQR approaches. The main contributions are numbered below,

- A ROS-Based embedded control is designed.
- Design of a low-cost prototype for educational purposes.
- Development of the embedded system, instrumentation, and control.
- The algorithm is relatively easy to implement in an embedded system.
- Real-time tests are performed to validate the control law.

It is worth mentioning that several scale prototypes developed at the Tecnológico Nacional de México/Instituto Tecnológico de Puebla TECNM/ITP, which serve as the subjects of the specialty module. It helps us introduce concepts of stability, modeling, nonlinearities, control, power electronics, programming, embedded systems, and computer-aided design CAD. The rest of the document is structured as follows. Section II presents the mathematical model of the two-wheeled self-balancing mobile robot. Section III is its mechanical and electronic design. In section IV, the controller is addressed. Section V numerical simulations and experimental results are illustrated. Finally, the conclusions and future work in section VI are given.

II. TWO-WHEELED SELF-BALANCING MATHEMATICAL MODEL

This section presents the nonlinear mathematical modeling of the two-wheeled self-balancing mobile robot. The nonlinear dynamic model of the WIPR is obtained employing Newton’s formulation.

\[
\begin{align*}
\dot{\xi} &= \frac{\left( m^2 g \sin 2\phi / 2 - (J + ml^2)(ml \sin \phi \dot{\phi}^2 - (J + ml^2)u \right)}{aml \cos^2 \phi - 4/3} \\
\dot{\phi} &= \frac{- (M + m)(ml \sin \phi) + (m^2 l^2 \dot{\phi}^2 \sin 2\phi) / 2 + (ml \cos \phi)u}{aml \cos^2 \phi - 4/3}
\end{align*}
\]

where \( \xi \) and \( \phi \) define the state variables of our system. \( \xi \) represents the displacement of the robot and \( \phi \) represents the angle of the pendulum. \( u \) represents the force exerted by the DC motors and, therefore, on our control input. The inertia of the robot is represented by \( J \), \( I \) represents the length of the bar at the centre of mass, \( M \) the mass of the robot’s base, \( m \) is the mass of the bar together with the person on board; finally \( g \) represents gravity.

III. TWO-WHEELED SELF BALANCING MOBILE ROBOT DESIGN

A two-wheeled self-balancing robot was designed and developed at the Embedded System Laboratory at the Instituto Tecnológico de Puebla to perform the necessary stability tests. Fig. 1 (b) shows the designed prototype. The system has been divided into mechanical structure, propulsion, embedded system, and communication system.

- Mechanical structure. The mechanical structure of the mobile mini robot consists of a 3 mm caliber acrylic with the following dimensions: 15.5 cm × 6 cm × 10 cm.
- Propulsion system. The propulsion system comprises two 5V permanent magnet direct current motors, and is fitted with a slotted infrared encoder sensor and encoder disk.

In this work, the stabilization of the robot to its vertical position against external disturbances is addressed. Consider the free-body diagram in Fig. 1 (a). The following equations give the equations that describe the dynamics of the inverted pendulum angle and the displacement of the mobile robot.

![Fig. 1. Two-wheeled self-balancing a) Free-body diagram. b) Implemented prototype.](image)
The supply voltage is 7 V, with a 1000 mah 2s Zippy Lipo Battery and an L298N H-bridge as the power stage. The L298N has digital inputs with levels from 0 to 5 volts for speed control through the PWM signal and the direction of rotation of the motor. The duration of the Lipo Battery is approximately 1 hour of autonomy of the robot without load.

- Embedded system. The embedded system is composed of sensing, control and actuation. A 32-bit dual-core ESP32 microcontroller DEVKIT V1 model, with a 240 MHz clock frequency and 16 independent channels to generate pulse width modulation PWM signals with 16 bits resolutions. This chip contains wireless communication modules such as WiFi and Bluetooth. An MPU6050 Inertial Measurement Unit with 6 degrees of freedom (DoF) that combines a 3-axis accelerometer and a 3-axis gyroscope is used as the primary sensor. The MPU6050 is connected to a I²C communication interface. A complementary filter is used to estimate the pitch angle \( \phi \). In addition, the reading of the encoder coupled to the motor allows the wheel angle \( \theta \) to be obtained.

- Communication system. The communication is carried out through distributed computing through ROS, allowing different tasks to run in parallel on other processors, computers, or servers. Fig. 2 shows the proposed communication: the mobile robot reads the sensors, calculates the control algorithm, and publishes its pose through a publisher node. The computer subscribes to the node, graphs it in real-time, saves and displays the information, and finally publishes the desired navigation commands. Communication is done through the WiFi port of the ESP32 to the PC ground station. The measurement of state, filtering, and control algorithm calculation is performed at 100 Hz.

### IV. CONTROL DESIGN.

The system consists of three blocks, an approximate linearization control, an assignment control, and a PID for the dynamics of the DC motor. Fig. 3 shows the control algorithm proposed. The control law is calculated, and the control signal obtained \( u \) is converted by the assignment control block into desired movements of the robot, and the speed in both motors \( \Omega_i \) passes to a PID controller that stabilizes speed of the DC motors. The PID controller is parametrized by PWM signal \( \sigma_i \). The subscript \( (r,l) \) is for the left and right wheels, respectively.

#### A. Control Law

The main objective of the control law is the auto-balance, regulate the position of the bar \( \phi(t) \) to the desired reference \( \phi_d \) while the angular velocity \( \dot{\phi}(t) \) approaches zero when time tends to infinity. Mathematically is expressed as:

\[
\phi(t) \to \phi_d, \quad \dot{\phi}(t) \to 0, \quad t \to \infty.
\]  

Let us consider a simplified model of the system described above by decoupling the orientation dynamics. In this case, we present the following second-order system.

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{g \sin x_1 - aml x_2^2 \sin 2x_1 - a \cos x_1 u}{4l/3 - aml \cos^2 x_1}
\end{align*}
\]

where \( x_1 = \phi(t) \) is the pitch angle, measured in radians concerning the vertical, \( x_2 = \dot{\phi}(t) \) is the associated angular velocity, and \( a = \frac{1}{m+\frac{1}{4}} \) is a constant.

Considering that the robot remains operating close to the equilibrium point, that is, on the vertical axis, i.e., \( x_\delta = [x_{\delta,1} \ x_{\delta,2}] = [0 \ 0] \), \( u_\delta = 0 \) the Jacobian linearization around the equilibrium point results in the LTI form:

\[
\dot{x}_\delta = Ax_\delta + Bu_\delta + \zeta(t)
\]

with

\[
A = \begin{bmatrix} 0 & 1 \\ \frac{g}{4l/3-aml} & 0 \end{bmatrix}
\]

\[
B = \begin{bmatrix} 0 \\ \frac{-a}{4l/3-aml} \end{bmatrix}
\]

where \( X_\delta \) is the desired angle, \( \zeta(t) \) represents endogenous and exogenous disturbance. The new state variables being defined as \( x_{1\delta} = x_1 - X_\delta, x_{2\delta} = x_2, u_\delta = u_1 - U \) and \( U = \frac{a}{g}X_\delta \).

We verify that the system is controllable:

\[
C = [B; AB] = \begin{bmatrix} 0 & \frac{a}{g} \\ \frac{-a}{4l/3-aml} & 0 \end{bmatrix}
\]

since the determinant is different from zero, then the system is controllable. Linear feedback allows the poles of the closed-loop system to be located at preselected points in the complex plane.
We propose the following desired polynomial,

\[ p_d(s) = s^2 + 2\zeta w_n s + w_n^2 \]  

(10)

the constants \(\zeta, w_n\) are the damping factor and the undamped natural frequency, respectively.

The feedback control of the state vector is as follows,

\[ u = -k_1 x_1 \delta - k_2 x_2 s + \frac{g}{a} X_d \]  

(11)

of the desired polynomial, the feedback gains are left as

\[ k_1 = -\frac{2\zeta w_n(4l/3 - aml)}{a}, k_2 = -\frac{w_n^2(4l/3 - aml) + g}{a} \]  

(12)

Since, in practice, it is necessary not to saturate the actuators, we propose adding a bounded function to the control.

**Definition 1:** Given a positive constant \(M\), a defined by \(\sigma_M : \mathbb{R} \rightarrow \mathbb{R}\) is defined as:

\[ (1)\sigma_M = s \text{ if } |s| < M; \]
\[ (2)\sigma_M = M \cdot \text{sign}(s); \]  

(13)

Therefore, the implemented controller is:

\[ u_a = \sigma_M (-k_1 x_1 \delta - k_2 x_2 s + \frac{g}{a} X_d) \]  

(14)

B. Proportional-Integral-Derivative PID Motor Speed Control

A fast and smooth response is necessary for high performance of WIPRs rotational and translational control. The control goal is to design a PID in such a way that the PWM signal makes \(\lim_{t \to \infty} |\Omega_d(t) - \Omega(t)| = 0\). The mathematical model of the DC motor can be described as in [42], where its transfer function describes the angular velocity \(\Omega_i\) with the input voltage to the motor \(v\),

\[ \frac{\Omega_i(s)}{v(s)} = \frac{K_m}{LI s^2 + (RI + LB)s + RB + K_m K_a} \]  

(15)

where \(I\) represents the rotor’s moment of inertia, \(R\) the electrical resistance, \(L\) the electrical inductance, \(B\) represents the motor’s viscous friction constant, \(K_a\) represents the electromotive force constant, \(K_m\) represents the motor torque constant. The manufacturer’s parameters for the DC motor used are listed in Table I.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Moment of inertia</td>
<td>0.000031</td>
<td>kg·m²</td>
</tr>
<tr>
<td>R</td>
<td>Armature Resistance</td>
<td>0.425</td>
<td>(\Omega)</td>
</tr>
<tr>
<td>L</td>
<td>Electrical inductance</td>
<td>0.00018</td>
<td>H</td>
</tr>
<tr>
<td>B</td>
<td>Viscous Friction Constant</td>
<td>0.00028</td>
<td>N·m·s</td>
</tr>
<tr>
<td>(K_a)</td>
<td>Electromotive Force Constant</td>
<td>0.0292</td>
<td>N·m/A</td>
</tr>
<tr>
<td>(K_m)</td>
<td>Motor Torque Constant</td>
<td>0.0281</td>
<td>N·m/A</td>
</tr>
<tr>
<td>M</td>
<td>Mobile robot base mass</td>
<td>0.48</td>
<td>Kg</td>
</tr>
<tr>
<td>m</td>
<td>Pendulum mass</td>
<td>0.16</td>
<td>Kg</td>
</tr>
<tr>
<td>l</td>
<td>Pendulum length</td>
<td>0.25</td>
<td>m</td>
</tr>
<tr>
<td>J</td>
<td>Moment of inertia of the mobile robot</td>
<td>0.0043</td>
<td>kg·m²</td>
</tr>
<tr>
<td>g</td>
<td>Gravity</td>
<td>9.81</td>
<td>m/s²</td>
</tr>
</tbody>
</table>

C. Allocation Control

According to some optimization algorithms, the allocation control calculates the angular speeds of the DC motors \(\Omega_i\) to the desired control inputs or commands. The limitations or constraint conditions of the actuator can be guaranteed,

\[ \Omega_{r,d} = u_a(t) + \epsilon_r \]  

(17)

\[ \Omega_{l,d} = u_a(t) + \epsilon_l \]  

(18)

where the variables \(\epsilon_r, \epsilon_l\) are zero if the robot goes in a straight line, and if the robot turns to the right or left, the values are different from zero.

V. SIMULATION AND EXPERIMENTAL RESULTS

Simulation and real-time tests are performed to validate the control architecture based on ROS design. The Kalman controller and filter have been programmed with C++ on the ESP32 microcontroller. The ROS communication between the different nodes has been implemented in C++ as well. Three scenarios have been addressed to validate, compare and measure the performance of the proposed controller. The first scenario demonstrates the ability of the control algorithm to stabilize the robot at the equilibrium point regardless of the initial conditions. In the second scenario, the orientation of the robot is regulated, that is, the position of the bar to the vertical axis. In the third scenario, we compare our controller with PID and LQR algorithms. We also measure the ability to reject endogenous and exogenous disturbances and follow different trajectories. The integral index of squared error ISE is used as a performance index. A saturation function has been used not to exceed the torques supplied to the DC motors. The saturation at the output \(u_a\) is in the range of -6 Newton to 6 Newton.

A. Simulations

In this subsection, we perform the numerical simulations of the proposed control algorithm in the MATLAB/Simulink environment.
1) First scenario: The Fig. 4 shows stabilize the pendulum to the vertical axis even when conditions are far from the equilibrium point. The initial values used were \((0.5, 0.2, -0.2, -0.5)\) radians. The response is smooth, fast, and without overshoot. The settling time is approximately 0.5 seconds.

2) Second scenario: Adjustments to different desired angles \(\phi_d\) are performed as a second test. The desired reference are \(\phi_d = (0.4, 0.6, -0.4, -0.6)\) radians. We have added an external disturbance to the robot dynamics \(\zeta(t) = 3 \sin 0.1 t \cos 10 t\) rad/sec possibly caused by the wind, friction with the surface, engine wear, or tire slip. Fig. 5 shows that the control can regulate to the desired reference despite external disturbance.

3) Third scenario: A comparative investigation in numerical form is carried out of the proposed algorithm. A classic PID control and an LQR are used to compare the proposed controller. The three algorithms control are simulated in Matlab/Simulink software. The integral square error (ISE), integral time square error (ITSE), integral absolute error (IAE), and integral time absolute error (ITAE) performance indexes are used to analyze the performance of each controller. The performance indexes are expressed by

\[
ISE = \int_0^T e_\phi^2(t) \cdot dt \\
ITSE = \int_0^T t e_\phi^2(t) \cdot dt \\
IAE = \int_0^T |e_\phi(t)| \cdot dt \\
ITAE = \int_0^T t |e_\phi(t)| \cdot dt
\]

PID type compensator design methods are very important in stabilizing real systems that operate near equilibrium points. From (7), (8), we calculate the open-loop transfer function.

\[
G(s) = \left(\frac{a}{4l/3 - aml}\right)s^2 - g
\]

The classical form of the PID compensator that stabilizes the output at the equilibrium point is given by the expression:

\[
u_{pid}(t) = k_1 e_\phi(t) + k_2 \int e_\phi(t) dt + k_3 \frac{de_\phi(t)}{dt}
\]

where \(e_\phi = \phi_d - \phi(t)\). The values obtained are \(K_1 = 93\), \(k_2 = 90\), \(k_3 = 10\) via Ziegler-Nichols self-sustaining oscillation.

LQR is an optimal control, which requires complete knowledge of the state. The goal of the controller is to minimize the next performance index.

\[
J(x, t) = \frac{1}{2} \int_0^\infty (x^T Q x + u^T R u) dt
\]

Equation (25) represents a trade-off between the state’s distance from the reference and the cost of the control input. By choosing the matrices \(Q\) and \(R\), we can regulate the orientation of the TWSBR.

\[
u_{lqr} = K x_\delta = R^{-1} B^T P x_\delta
\]
where $k_1 = -55.784$, and $k_2 = -2.0445$, besides $P$ is the Riccati matrix equation solution:

$$A^T P + PA - PBR^{-1}B^T P = 0 \quad (27)$$

- Trajectory tracking of a square signal. The desired trajectory is defined by

$$\phi_d(t) = \begin{cases} 0.33 & : 0 \leq t \leq 3 \\ 0 & : 3 \leq t \leq 5 \end{cases} \quad (28)$$

The trajectory can be generated by a rider, ensuring the correct functioning of the robot. Fig. 6 shows the three control algorithms. All three controls are capable of following small time-varying references. The PID control performs poorly since its response is oscillatory and shows errors in a steady state. The LQR control presents a similar response to the proposed controller. However, calculating the gains is more tedious and expensive in processing time. Table II shows the performance indices for the three controls. It can be concluded that the proposed control has better performance, a quicker recovery rate, and less error fluctuation.

- Trajectory tracking of a sinusoidal signal. The desired trajectory is defined by

$$\phi_d(t) = 0.5 \sin t + 0.2 \cos 3t. \quad (29)$$

Fig. 7 shows the three control algorithms. The PID control shows an oscillatory response, contains fluctuations, and is not robust to external disturbances. The LQR control presents a similar response to the proposed controller. However, the calculation cost is high. The proposed control performs superior to the PID and LQR control; see table II. In addition to presenting advantages in practical implementation by not saturating the operating range in the actuators.

### B. Experimental Results

First, the angle $\phi(t)$ is estimated using the $MPU - 6050$ inertial measurement unit and the Kalman filter. The control is calculated by generating a virtual control input $\Omega_d$, which is a reference for the PID control. Finally, the PID control is calculated by reading the encoders of each DC motor; this PID control generates the PWM signals of the DC motors, Fig. 3 shows the proposed control scheme. Real-time tests are carried out to validate the control architecture based on ROS design. The control is implemented in the ESP32 microcontroller.

The ROS nodes, the complementary filter, and the control have been programmed in C++. Two scenarios have been addressed, the stabilization of the equilibrium point and the ability to reject external disturbances and keep the pendulum stabilized in a vertical position. Figs. 8 and 9 show the angular position and angular velocity. The Figs show that the control algorithm effectively maintains the equilibrium positions within about 0.1 radians (5 degrees). Despite the elements of friction, noise, and external disturbances, the control shows a smooth and robust response.

In a second scenario, the ability to reject disturbances is tested. Figs. 10 and 11 show how the system has been disturbed externally in 6 and 12 seconds. Despite being an LTI control, it manages to stabilize the bar to the vertical axis despite this disturbance. The robot’s operation can be
shown in the following link https://youtu.be/L1eUwG2GSfo. The control algorithm is lightweight, and robust, and can stabilize the self-balancing two-wheeled robot. Proposing ROS middleware to the embedded system allows adding modules such as navigation, manipulation, and perception. Which will allow surveillance, delivery, transfer of objects, and autonomous navigation tasks indoors or outdoors.

VI. CONCLUSION AND FUTURE WORK

This article designs a ROS-based architecture to control a two-wheeled self-balancing robot. The ROS architecture is open and allows the integration of different modules, such as mapping, navigation, perception, and manipulation, to develop new applications. This architecture can be a starting point for designing new robotics apps. The bounded saturation control shows robustness against small bounded external disturbances, as well as better performance in path-following control compared to classical PID and LQR controls. Besides that, its implementation is simple and light computing. However, we are the noise in the Kalman filter; friction and wear of the actuators can generate a small region of attraction and oscillation. In future work, control will be designed using ADRC’s disturbance rejection control methodology, which estimates endogenous and exogenous disturbances through an extended state observer and attenuates them under feedback.

REFERENCES


H. Tourajizadeh, M. Rezaei, and A. H. Sedigh, “Optimal Control of Screw In-pipe Inspection Robot with Controllable Pitch Rate,” J Intel
Juan Díez-Téllez, ROS-based Controller for a Two-Wheeled Self-Balancing Robot