Journal of Intelligent Systems and Control
https://www.acadlore.com/journals/JISC

Design and Control of a Bluetooth-Enabled Two-Wheeled Self-Balancing Vehicle

Fuchun Jiang[1,2], Huangjie Guo[1*], Chenwei Feng[1,2], Xinjie Yang[1], Yau Hee Kho[3]

[1] School of Opto-Electronic and Communication Engineering, Xiamen University of Technology, 361024 Xiamen, China
[2] Fujian Key Laboratory of Communication Network and Information Processing, Xiamen University of Technology, 361024 Xiamen, China
[3] School of Engineering and Computer Science, Victoria University of Wellington, 6140 Wellington, New Zealand

* Correspondence: Chenwei Feng (1428044469@qq.com)

Received: 03-15-2025 **Revised:** 05-25-2025 **Accepted:** 06-05-2025

Citation: F. J. Jiang, H. J. Guo, C. W. Feng, X. J. Yang, Y. H. Kho, "Design and control of a Bluetooth-enabled two-wheeled self-balancing vehicle," *J. Intell Syst. Control*, vol. 4, no. 2, pp. 84–104, 2025. https://doi.org/10.56578/jisc040202.

Abstract: Two-wheeled self-balancing vehicles inherently exhibit nonlinear, unstable, and strongly coupled dynamic characteristics, and their analysis and control remain of substantial relevance to military, industrial, and intelligent transportation applications. To address these challenges, a Bluetooth-enabled self-balancing vehicle system was designed with enhanced sensing, estimation, and hierarchical control capabilities. An improved Kalman filter (KF) algorithm was developed to overcome the limitations of conventional sensor fusion approaches. In the proposed method, gyroscope and accelerometer measurements were adaptively fused, enabling higher accuracy in attitude estimation while suppressing cumulative drift and transient disturbances. On this basis, a hierarchical proportional–integral–derivative (PID) control strategy was formulated to enhance responsiveness, stability, and tunability. Optimal attitude angles and reference velocities were processed within this framework to generate pulse-width modulation (PWM) signals for motor actuation. In parallel, a Bluetooth module was integrated to receive real-time commands from a mobile application, enabling precise execution of forward motion, reverse motion, and differential steering maneuvers. Experimental validation demonstrated that the system maintained stable posture, resisted external perturbations, responded rapidly to mobile control inputs, and executed commanded trajectories with high accuracy. The overall performance indicates that the proposed design provides a reliable and scalable platform for self-balancing vehicle research and offers potential applicability in human-robot interaction, intelligent mobility, and adaptive control studies.

Keywords: Bluetooth; Self-balancing vehicle; KF; PID control

1 Introduction

In recent years, two-wheeled self-balancing vehicles have been favored by more and more green travel enthusiasts due to their advantages of energy saving, environmental protection, simple structure, flexible operation, and convenient carrying, and have now become an important branch of mobile robot research [1]. The two-wheeled self-balancing vehicle system, as an important part of the intelligent vehicle system, has also attracted much attention due to its nonlinear, highly unstable, and strong coupling characteristics.

In the vehicle's self-balancing control process, the accuracy and real-time performance of the body attitude inclination measurement determine the control performance. Relying on the gyroscope or accelerometer only leads to large measurement error and vibration interference, often failing to meet the self-balancing control requirements of a two-wheeled vehicle [2]. Therefore, the study of data fusion algorithms that effectively remove the interference of sensor data has also become the focus of research. The KF algorithm is widely used as the basis of many attitude data fusion methods [3–5]. Wei et al. [6] proposed a KF fusion algorithm based on a data iterative method. In this algorithm, the output of the multisensor data fusion is introduced into the system again for iterative fusion filtering. Zhu et al. [7] used extended KF to realize the information fusion of the accelerometer and gyroscope. In the study by Srichandan et al. [8], the angle output of the inertial measurement unit (IMU) is provided to the KF, and its output is

used as the input of Q-learning to achieve more precise balance control. The complementary filter (CF) is also widely used in two-wheeled self-balancing vehicles due to its ability to effectively eliminate the gyroscope drift, suppress the high-frequency perturbation of the accelerometer, reduce the dynamic error of the output attitude angle, and improve the angle measurement precision [9, 10]. Huang et al. [11] used CF to fuse and optimize the inclination and acceleration sensor measurement value, thus obtaining the optimal estimation value which is consistent with the actual attitude information. After CF processed the output data of the gyroscope and accelerometer and the weighted average operation was performed, the cross compensation was used for correction according to the measurement error.

In addition, various control algorithms have been studied to solve the self-balancing and motion control problems of the vehicle. Unluturk and Aydogdu [12] proposed an adaptive fuzzy logic proportional–integral (PI) controller based on artificial neural network (ANN). Kim and Kwon [13] proposed a control scheme based on the state-dependent Riccati equation. Xu and Li [14] proposed a design methodology for the one-dimensional cloud model controller. Guo et al. [15] used a completely online, feedback-based Q-learning to achieve optimal control of a balancing robot, with both state feedback and output feedback considered. At present, the PID control algorithm has been widely used because of its simple principle, easy design, and independent parameters [16–18]. However, this algorithm also has shortcomings. It is mainly suitable for the basic linear and dynamic characteristics that do not change with the time control system. And the controller parameters mainly rely on manual debugging because it is time-consuming. Therefore, it is difficult for the PID controller to achieve precise control of the vehicle for a long time.

To improve the control effect of the PID controller in self-balancing vehicle systems, Singh and Bhushan [19] revised the transition probability of the ant colony optimization (ACO) and obtained an improved ACO, which in turn optimized the PID controller and obtained its optimal tuning parameters. Jiang and He [20] proposed an active disturbance rejection control strategy, which is tuned by an adaptive differential evolution algorithm, for the self-balancing control part of the vehicle and designed a PID control strategy incorporating the transient process for the steering control part. Shen et al. [21] designed the controller using the neuronal PID control algorithm. Ren and Zhou [22] proposed a cascade coupling control scheme based on the PID control algorithm. The speed loop and balance loop in the classical parallel coupling control method were decoupled to remove the independent speed PID control, with the former being set as the outer loop and the latter being set as the inner loop to realize the coupling. Pang et al. [23] proposed an adaptive sliding mode attitude control scheme by combining the minimum parameter learning method with the radial basis function neural network. Anisimov et al. [24] designed a fuzzy proportional–derivative (PD) controller based on the relation model and optimized the scaling factors of the controller using the cross-entropy optimization method. Ren et al. [25] proposed a fuzzy controller with learning ability based on the fuzzy rule and the operant conditioning (OC) learning mechanism, enabling the self-balancing vehicle to learn and obtain the fuzzy control rules autonomously and realize the automation of controller design.

To obtain accurate attitude information of the current vehicle body, an improved KF algorithm was adopted in this current study based on the measurement's iterative update to fuse the output data of the gyroscope and accelerometer. In addition, the prior covariance matrix modified by the Levenberg-Marquardt (L-M) method was introduced at the beginning of the iteration for updating. For the control problem, an improved PID control algorithm was designed based on the hierarchical control strategy. In the balance control layer, the angular velocity loop was strung into the angle loop. In the speed control layer, a control algorithm based on weighted PI was designed. Accurate control of speed was realized by adjusting the change speed of accumulation and the degree of response to the error. In the steering control layer, a controller output update algorithm based on weighted PD was designed. A certain degree of feedback control was introduced into the steering output, and finally a limiting operation was performed on the steering control output. The rest of this study is organized below. Sections 2 and 3 introduce the hardware and software design of the system. Section 4 presents the principle and design of the improved KF. Section 5 explains the improved PID algorithm in terms of balance, speed and direction control. Section 6 simulates and analyzes the improved KF and PID algorithms and tests the system. Section 7 concludes this study.

In the process of accurately acquiring the real-time attitude of a self-balancing vehicle in dynamic operation, traditional KF methods are often affected by estimation errors when sensor noise is time-varying or abnormal disturbances occur. To overcome this limitation, a measurement-driven iteratively updated KF algorithm was used in this study to fuse the outputs of the gyroscope and accelerometer. During the iterative initialization phase, a prior covariance matrix corrected by the L-M method was introduced, which enhanced the robustness of the algorithm against initial errors and improved the convergence speed and accuracy of attitude estimation. In terms of the control strategy, the insufficient adjustment capability of the traditional PID control in multi-task coupled scenarios for self-balancing vehicles was addressed through an improved PID control algorithm designed based on hierarchical control principles. At the balance control layer, a composite structure was adopted in which an angular velocity loop was integrated into an angular position loop to suppress attitude oscillation and improve disturbance rejection capability. At the velocity control layer, a weighted PI controller was designed to achieve precise speed control by adjusting the integral term's rate of change and the sensitivity to errors. At the direction control layer, a weighted PD-based controller output update algorithm was developed, incorporating a certain degree of feedback control into

the steering output, while output limiting was applied to ensure smooth steering maneuvers.

2 System Hardware Design

The system's hardware block diagram is shown in Figure 1. The system hardware mainly consists of a microprocessor, an attitude sensor module, a motor drive module, a Hall encoder module, and a master-slave integrated Bluetooth module. Among them, the attitude sensor is used to collect the inclination and angular velocity data of the self-balancing vehicle. The encoder is responsible for measuring the speed and direction of the direct current (DC) motor. As a high-performance, low-power 8-bit AVR microprocessor, ATmega328P is used to receive and process the data collected by the sensor module and then output the PWM signal to the motor drive module. At the same time, the microprocessor can also receive the control command of the mobile phone through Bluetooth, enabling forward motion, reverse motion, and steering. The main components of the system and their circuit design are explained below.

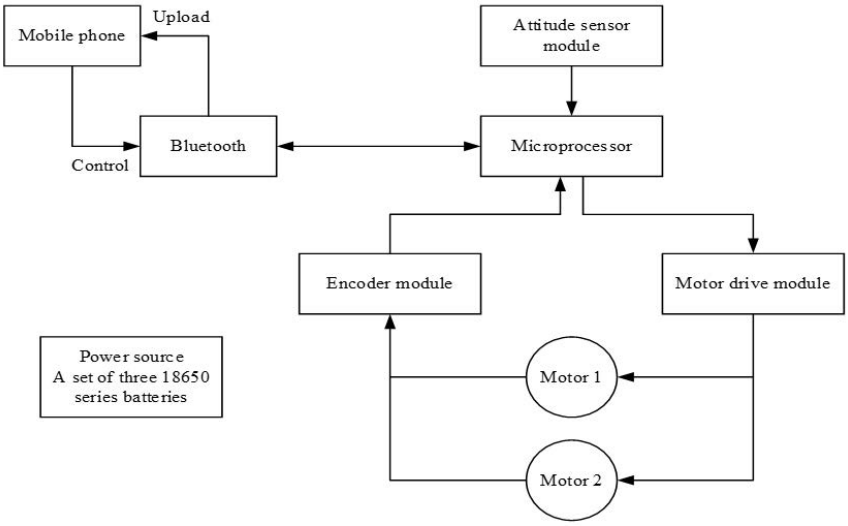

Figure 1. Hardware block diagram of the system

2.1 Bluetooth Module

The HC-06 Bluetooth module is equipped with both master and slave functionality and operates using the Bluetooth 2.0 serial protocol. An integrated printed circuit board (PCB) antenna is incorporated to ensure stable wireless communication within a range of approximately eight meters. In comparison to the HC-05 module, the HC-06 is configured to start up in the slave mode by default, which simplifies the initial setup process. The corresponding circuit diagram of the Bluetooth module is provided in Figure 2. Under typical operating conditions, a data transmission bandwidth of about 1 Mbps is achieved by this module, with communication latency generally ranging from several tens to hundreds of milliseconds. The actual performance values are determined by the communication environment and the size of data packets. To prevent the Bluetooth module from occupying the hardware serial port and thereby blocking the program upload via USB, a switch is used to control the activation of the Bluetooth output pin. The switch should be closed during Bluetooth communication and opened when the microcontroller is being programmed.

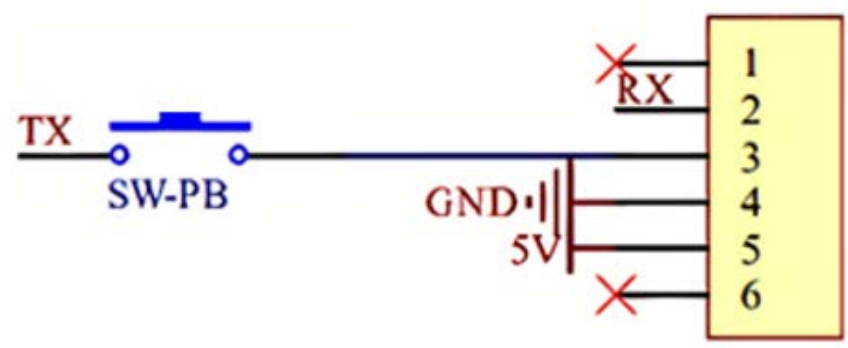

Figure 2. Circuit diagram of the Bluetooth module

2.2 Attitude Sensor Module

The MPU6050 is a six-axis motion sensor that integrates a three-axis gyroscope and a three-axis accelerometer, with communication performed via an inter-integrated circuit (I^2C) interface. To accommodate different motion environments, the sensor's measurement range can be defined by the user. The accelerometer range can be set to ± 2 g, ± 4 g, ± 8 g, or ± 16 g, while the gyroscope range can be selected from $\pm 250°/s, \pm 500°/s, \pm 1000°/s$, or $\pm 2000°/s$. The circuit of the MPU6050 attitude sensor is illustrated in Figure 3. To prevent circuit damage caused by transient pulse voltages that may exceed the rated output voltage during power supply interference, a 0.1μ F decoupling capacitor is connected between the power supply and ground terminals. As these pulse voltages are high-frequency signals, they are filtered out by the decoupling capacitor, thereby protecting subsequent circuits.

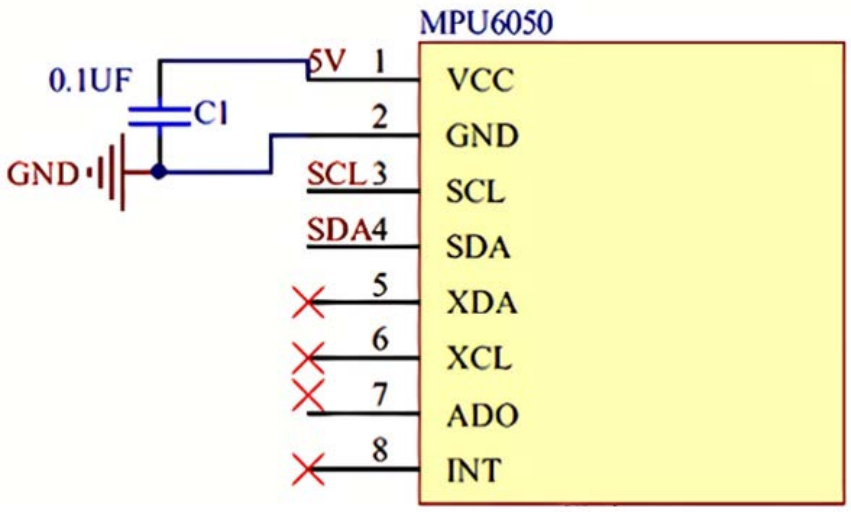

Figure 3. Circuit diagram of the MPU6050 attitude sensor

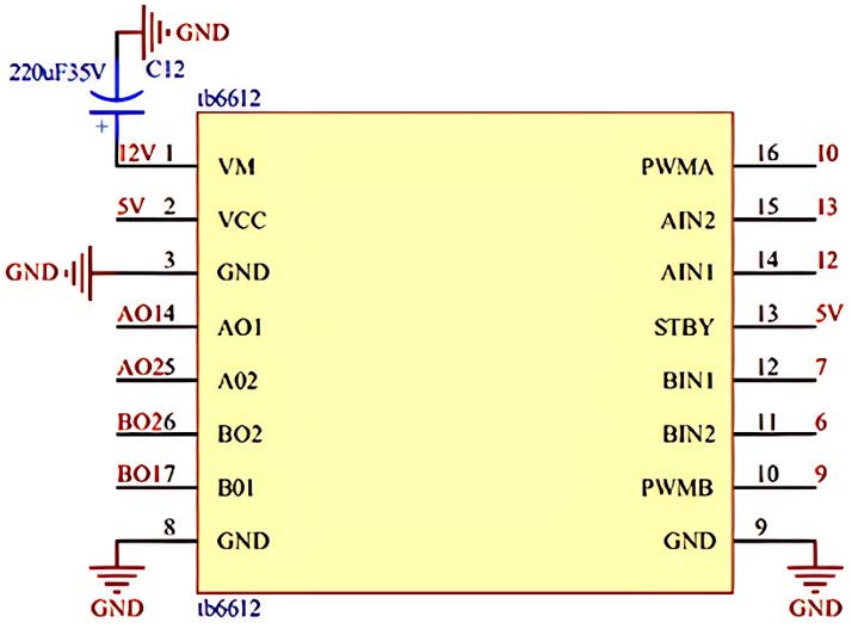

Figure 4. Circuit diagram of the motor drive module

2.3 Motor Drive Module

The TB6612 motor driver module is characterized by excellent power amplification performance. When the input remains within the rated operating range (typically VM $\leq$ 15 V and output current per channel $\leq$ 1.2 A), minimal heat is generated by the chip. The module supports PWM control frequencies of up to 100 kHz and is compatible with

both 3.3 V and 5 V logic level systems. The circuit of the motor driver module is presented in Figure 4. A 220μ F electrolytic capacitor is connected in parallel at the power supply input. This capacitor is primarily used to buffer current fluctuations caused by motor start/stop events and sudden load changes, while voltage dips and high-frequency noise induced by line inductance are suppressed. As a result, the supply voltage is effectively stabilized and noise is filtered out.

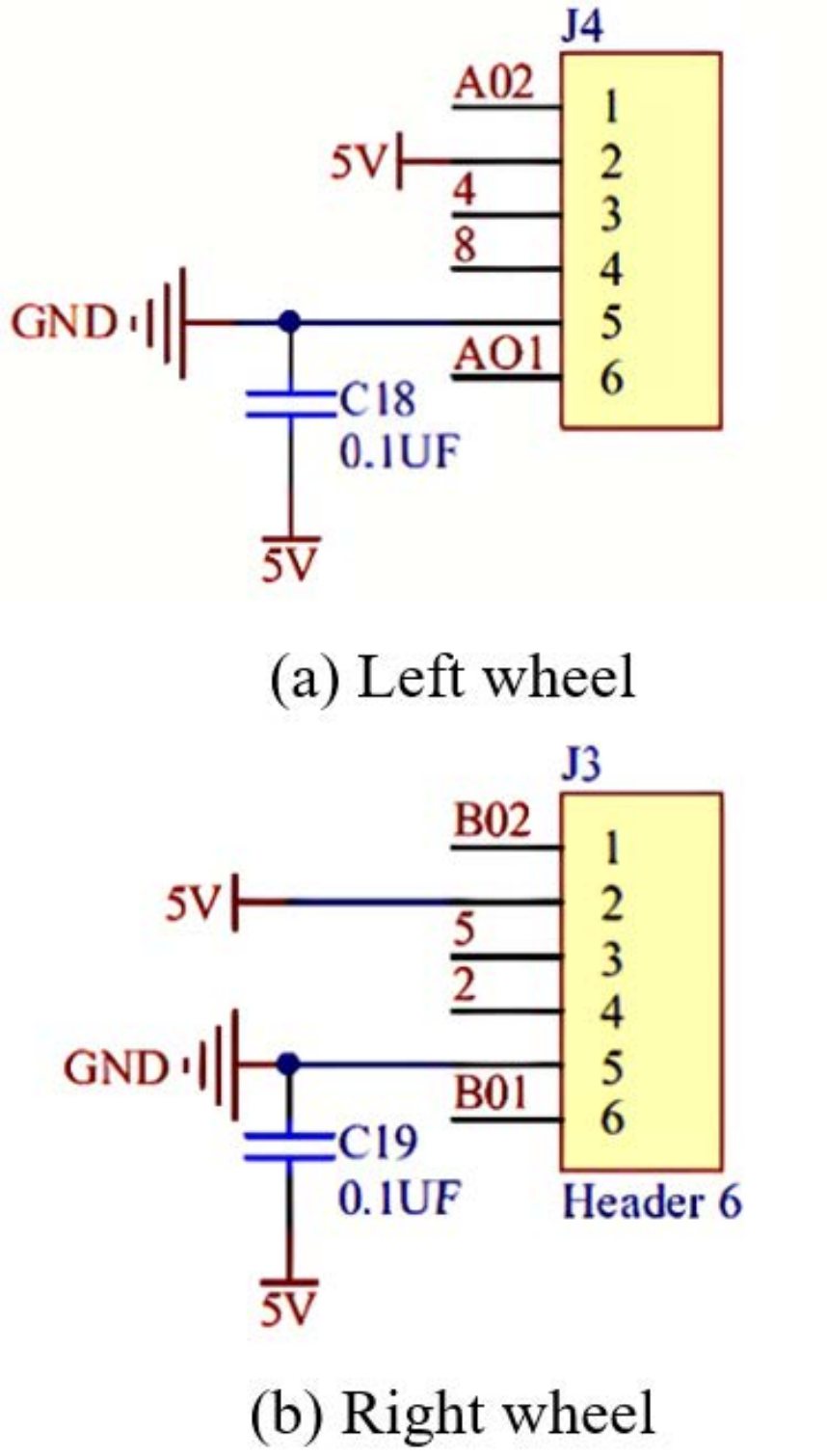

Figure 5. Circuit diagrams of the Hall encoder

2.4 Motor Encoder Module

The Hall encoder operates based on the Hall effect, through which angular displacement information from the motor shaft is converted into two square wave signals with a fixed phase difference (typically 90). By detecting edges and counting pulses from these two orthogonal signals, the motor's rotation direction, angular velocity, and relative displacement can be determined. This module is characterized by a typical resolution of 11 pulses per revolution (PPR) and a maximum response frequency that typically exceeds 100 kHz, with digital-level output signals provided. The interface circuit for a dual-wheel Hall encoder is shown in Figure 5, where typical configurations for power decoupling and signal output terminals are illustrated.

2.5 Power Buck Module

The LM2596 is a buck switching integrated voltage regulator integrated circuit (IC) capable of providing fixed output voltages of 3.3 V, 5 V, and 12 V, as well as an adjustable output voltage. In this self-balancing vehicle system, the power source is composed of three series-connected 18650 lithium-ion batteries. Their nominal voltage is approximately 11.1 V, rising to a maximum of about 12.6 V when fully charged, which is used as the input for the LM2596. This module supports a maximum continuous output current of 3 A, with typical conversion efficiencies exceeding 80%, and the output ripple voltage is generally maintained within tens of millivolts. The LM2596 buck module is used to step down the battery voltage to 5V, which powers the microcontroller and the motor driver module. Meanwhile, the attitude sensor and Bluetooth modules are supplied by the microcontroller's internal voltage regulator, where the 5 V input is converted to 3.3 V. A 12V-to-5V buck circuit based on the LM2596 chip is illustrated in Figure 6. Electrolytic and ceramic capacitors are placed at the input and output terminals, respectively, to suppress voltage fluctuations and reduce output ripple.

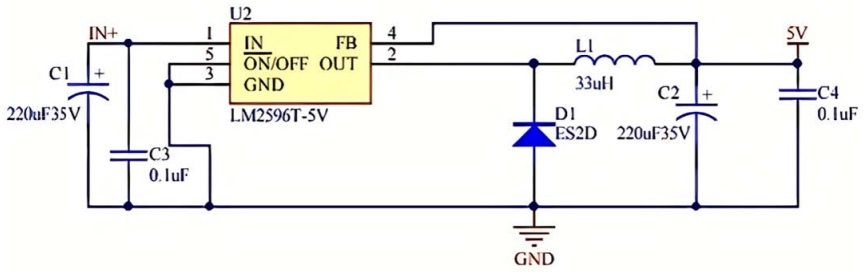

Figure 6. Circuit diagram of the 12V-to-5V buck module

3 System Software Design

The software architecture of the self-balancing vehicle system is illustrated in Figure 7. Upon system startup, the serial port and all peripheral modules are first initialized. Raw acceleration and angular velocity data are obtained by the attitude acquisition module through the MPU6050 sensor, with data being read in real time via the I^2C interface within the timer interrupt service routine. To improve the accuracy of the attitude angle estimation, an improved KF algorithm was employed for multi-source sensor fusion. The state vector consists of angle and angular velocity. By adjusting the process noise covariance matrix Q and the observation noise covariance matrix R, vibration interference from the accelerometer under dynamic conditions is effectively suppressed, resulting in more stable optimal estimates of the pitch angle.

The motor speed is calculated in real time by capturing the quadrature pulse signals output from the Hall encoder, while the rotation direction is simultaneously determined. The filtered angle estimates and the measured speed are combined to form the system state feedback, which is then processed by an improved PID controller. A cascade control structure is adopted, where the angular velocity is regulated by the inner loop and the angle is controlled by the outer loop. Error integral limiting and a differential lead strategy are implemented to balance response speed and stability. The output from this controller is a PWM duty cycle signal, which is used to drive the motor for balance regulation. Additionally, once the Bluetooth module is initialized, pairing can be performed through the mobile application. Connection with the module is established through the application, enabling commands to be transmitted via Bluetooth. This allows remote dynamic configuration of PID controller parameters (such as K_p, K_i, and K_d), facilitating adaptation to different operational conditions.

3.1 System Initialization Program

The system initialization program mainly includes serial port initialization, MPU6050 and I^2C bus initialization, timer interrupt and external interrupt settings. Serial port initialization is mainly to define the serial port input and

output mode as well as the type of data to be transmitted and to set the baud rate of serial port data transmission according to the demand. The information collected by the built-in gyroscope and accelerometer is transmitted between the MPU6050 and the single-chip microcomputer (SCM) through I^2C communication. Before that, the I^2C bus should be initialized first. The MPU6050 is then reset, its measurement range is configured, and the accelerometer and gyroscope are enabled, allowing the attitude data to be read from the device.

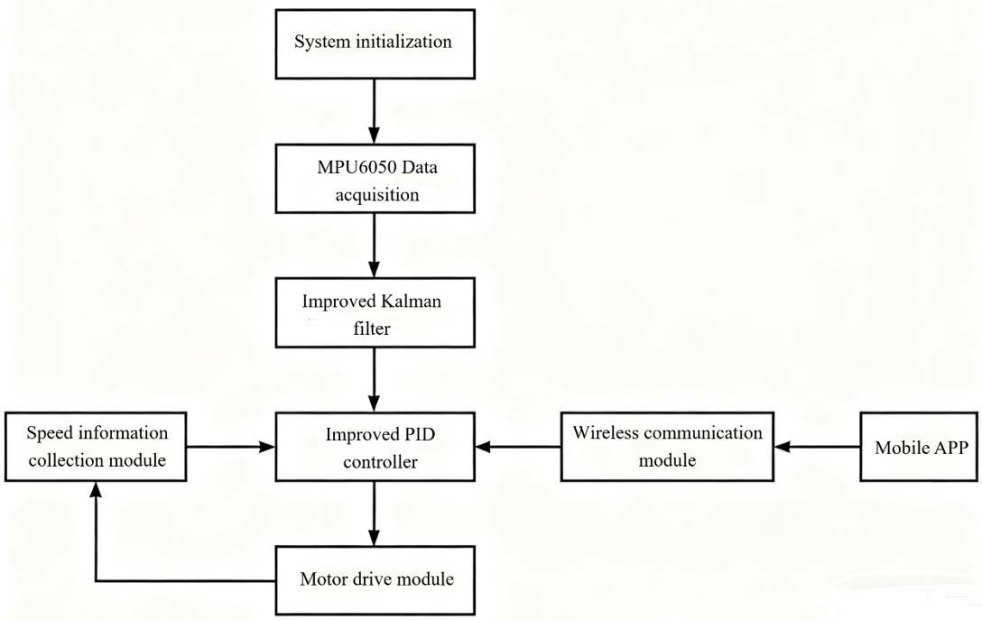

Figure 7. Software design block diagram of the system

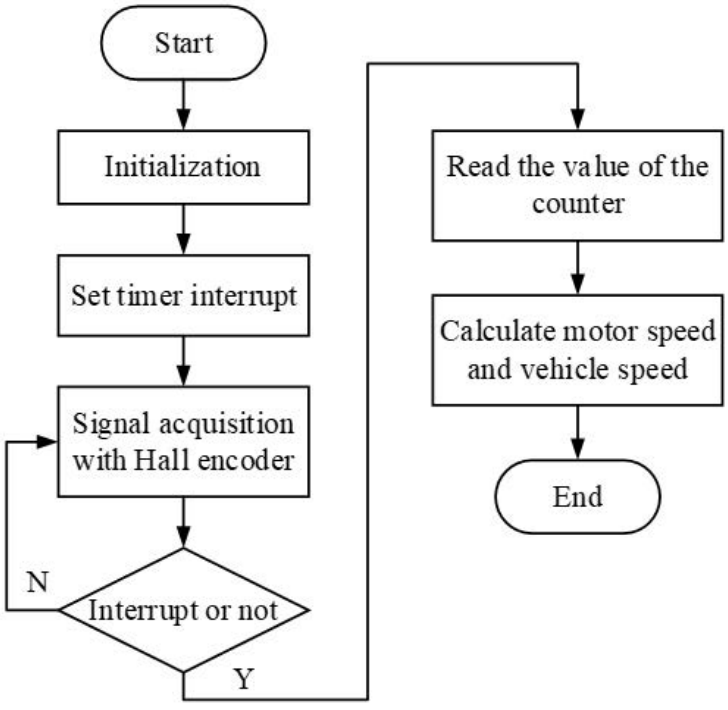

Figure 8. Flowchart of the encoder speed measurement

In addition, to ensure the normal acquisition of attitude data and motor speed, it is necessary to set the timer interrupt and external interrupt during initialization. The specific realization is to set and enable the interrupt duration and interrupt mode of the timer interrupt. In this system, since the acquisition frequency of the attitude sensor is 200 Hz, the interrupt is set to 5 ms. Besides, the interrupt triggering mode in this system is set to be triggered when the

level of the corresponding channel of the encoder changes.

3.2 Motor Speed Measurement Program

The process of encoder speed measurement is shown in Figure 8. Due to the high motor speed output of the self-balancing vehicle system, the method of accumulating the pulse number to measure the motor speed was selected; that is, the motor speed is obtained by calculating the number of accumulated pulse signals per unit time, and then the direction of the DC motor rotation is obtained according to the relationship between the lagging and leading of the two signals. The speed and direction of the vehicle can be obtained by further calculation. Compared with the previous method of measuring only the falling or rising edge of the encoder output signal as a pulse count, this design adopts the change count of the output signal; that is, the falling edge and the rising edge are all counted as a pulse count, which makes the precision of the speed measurement increase by four times.

3.3 Attitude Data Acquisition

The MPU6050 attitude sensor communicates with the SCM through the $\mathrm{I}^2\mathrm{C}$ bus. Since the internal key program is already encapsulated, the only thing that needs to be done is to read the attitude data from the register address of the desired information. Table 1 shows the register addresses for the accelerometer data and gyroscope x-axis data. It should be noted that the data read at this time is the original data and requires further data conversion before it can be used for subsequent processing.

Table 1. Summary of oilfield water-treatment processes and associated costs

Bit	**Register (hex)**	Bit	**Register (hex)**
ACEEL_XOUT $[15:8]$	0×3 B	GYRO_XOUT $[15:8]$	0×43
ACEEL_XOUT $[7:0]$	0×3C	GYRO_XOUT $[7:0]$	0×44

Table 2 shows the range settings for the accelerometer and gyroscope. The accelerometer used in this system is ±2g in the range, and the register AFS_SEL option is set to 00; that is, 0.0006 times the original data is the resulting acceleration value. Similarly, when the gyroscope range is set to ±2000°/s, the FS_SEL register option is configured to 11, and the corresponding angular velocity is obtained by dividing the raw data by 16.4. After data conversion, it also needs to be fused by the KF to get the optimal estimate of the attitude angle.

Table 2. Range settings for the accelerometer and gyroscope

FS_SEL [0:1]	**Range of Value (%/s)**	**AFS_SEL [0:1]**	**Range of Value (g)**
00	±250	00	±2
01	±500	01	±4
10	±1000	10	±8
11	±2000	11	±16

4 Improved KF Data Fusion Algorithm

At present, the attitude data measurement of the two-wheeled self-balancing vehicle is mostly obtained by IMU modules such as the gyroscope and accelerometer. The gyroscope has a good dynamic performance, but it is easy to be affected by noise and temperature, resulting in drift error. The accelerometer has good static performance, but the motion acceleration generated during movement interferes with the output signal [**?**]. Therefore, it is necessary to use the filtering algorithm to fuse the information obtained from each of the two, thus maximizing the suppression of the influence of noise on the measurement angle.

4.1 KF Algorithm

The two-wheeled self-balancing vehicle, which is a naturally unstable system, needs to be characterized by fast response and high adaptability to maintain the self-dynamic balance in the complex environment [**?**]. Therefore, the KF algorithm was improved in this study. That is, it was adapted to the dynamic environment with high fusion precision to further improve the filtering precision of the system. In this study, the KF algorithm was first introduced. The system state equation and observation equation are defined as follows:

$$\begin{cases} x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ y_k = Hx_k + v_k \end{cases} \tag{1}$$

where, x_k is the system state vector at moment k, u_{k-1} is the input control of the system, A is the state transfer matrix, B is the input control matrix, w_{k-1} is the process noise, y_k is the observation vector at moment k, H is the measurement matrix, and v_k is the measurement noise. The process of the KF algorithm is as follows:

Step 1: State prediction update.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{2}$$

Step 2: Calculation of the prior covariance matrix.

$$P_k^- = AP_{k-1}A^T + Q \tag{3}$$

Step 3: Update of the Kalman gain matrix.

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{4}$$

Step 4: Update of the posterior value of system state estimation.

$$\hat{x}_k = \hat{x}_k^- + K_k\left(y_k - H\hat{x}_k^-\right) \tag{5}$$

Step 5: Update of the posterior covariance matrix.

$$P_k = \left(I - K_k H\right)P_k^- \tag{6}$$

The above steps complete a round of iteration.

4.2 Improved KF Algorithm

As can be seen from the above process, the current system state estimate utilizes only the state prediction $\hat{x}_k^-$.$\hat{x}_k$ is a further calibration achieved by integrating K_k with the latest measurement data based on $\hat{x}_k^-$. Therefore, the error between $\hat{x}_k$ and the true state of the system x_k is smaller than the error between $\hat{x}_k^-$ and x_k [20]. The filtering accuracy of the KF can be improved by replacing measurement update $\hat{x}_k^-$ with measurement update $\hat{x}_k$. The fundamental approach of the algorithm presented in this study is to perform measurement updates using method $\hat{x}_k$ rather than method $\hat{x}_k^-$. To enhance the performance and robustness of the filter, a covariance correction mechanism based on the L-M method was introduced. The specific implementation process is described as follows:

Step 1: State prediction update.

$$\hat{x}_{k,1}^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{7}$$

Step 2: Calculation of the prior covariance matrix.

$$P_{k,1}^- = AP_{k-1}A^T + Q \tag{8}$$

Step 3: Start of iteration. The prior covariance matrix modified was introduced using the L-M method.

$$P_{k,l}^- = \begin{cases} \left[I - P_{k,1}^-\left(P_{k,1}^- + \mu^{-1}I\right)^{-1}\right]P_{k,1}^-, & l = 1 \\ \left[I - P_{k,l-1}\left(P_{k,l-1} + \mu^{-1}I\right)^{-1}\right]P_{k,l-1}, & l > 1 \end{cases} \tag{9}$$

where, l denotes the l iteration realization in the measurement update, with $l = 1, 2, \ldots, L$; L denotes the maximum number of iterations; and $\mu > 0$ is the damping factor. This modification enhances numerical stability by introducing a regularization term involving the identity matrix I, thereby constraining the divergence of covariance matrix eigenvalues.

Step 4: Update of the Kalman gain matrix.

$$K_{k,l} = \frac{P_{k,l}^{-} H^T}{H P_{k,l}^{-} H^T + R} \tag{10}$$

where, H denotes the observation matrix, and R denotes the measurement noise covariance matrix.

Step 5: Update of the posterior value of system state estimation.

$$\hat{x}_{k,l} = \begin{cases} \hat{x}_{k,1}^{-} + K_{k,l}\left(y_k - H\hat{x}_{k,1}^{-}\right), l = 1 \\ \hat{x}_{k,l-1} + K_{k,l}\left(y_k - H\hat{x}_{k,l-1}\right), l > 1 \end{cases} \tag{11}$$

Step 6: Update of the posterior covariance matrix.

$$P_k = \left(I - K_{k,l} H\right) P_{k,l}^{-} \tag{12}$$

The iteration ends after reaching the maximum number of iterations. In practical applications, the number of iterations should be limited to maintain an appropriate balance between filtering accuracy and computational cost. Based on the experimental simulation results, the number of iterations L is typically set to two to four.

4.3 Mechanism of Action and Parameter Selection for the L-M Method

The adaptive adjustment between the gradient descent and Gauss-Newton methods is achieved by introducing a damping factor μ in the L-M method, thereby enhancing the numerical stability of covariance iteration when system models demonstrate strong nonlinearity or insufficient observational information. The effectiveness of this approach depends critically on the appropriate setting of μ: when μ is assigned a larger value, the algorithm converges toward gradient descent, ensuring convergence stability at the cost of slower speed; when μ is given a smaller value, the method approaches the Gauss-Newton algorithm, achieving rapid convergence while demonstrating increased sensitivity to initial conditions. In practical applications, the optimal value of μ can be determined through experimental tuning.

The setting of the iteration count L requires a balance between estimation accuracy and computational efficiency. Simulation results demonstrate that estimation accuracy can be significantly improved when L is set to two or more iterations; however, when L exceeds four iterations, further improvement in accuracy is typically limited while the computational load continues to increase. Therefore, L is generally set to two to four.

4.4 Experimental Validation and Performance Evaluation

A comprehensive evaluation of the overall performance of the improved KF algorithm was conducted by introducing the following multi-dimensional performance metrics in addition to the Mean Squared Error (MSE) indicator:

a) Steady-state accuracy: The average deviation between the estimated pitch angle and the true value is measured when the self-balancing vehicle remains in a static equilibrium state.

b) Dynamic response time: The time required for the attitude estimation to return to a stable state is recorded after the system is subjected to external disturbances.

c) Interference resistance: The algorithm's capability to maintain stable estimation performance is evaluated after transient pulse disturbances are applied to the system.

d) Convergence characteristics: The convergence time required to return to the equilibrium position is measured when the system starts from a large angular deviation.

e) Real-time performance metric: The ratio of the time consumed for a single filtering computation to the total control cycle is calculated and used as an evaluation criterion.

During the testing process, typical motion states of a self-balancing vehicle were simulated, including static equilibrium, small-amplitude oscillations (to simulate uneven road surfaces), and sudden external force disturbances. When comparisons were made between the improved algorithm and both standard KF and CF methods, the following results were observed: under transient disturbances of 5°, the overshoot was reduced by approximately 40%, with convergence to the stable region being achieved within 0.3 s.

In tests where the self-balancing vehicle was required to recover equilibrium from a 20° large-angle deviation, stable convergence was achieved by the improved algorithm within approximately 0.5 s. Throughout the recovery process, the condition number of the covariance matrix was consistently maintained at low values, demonstrating excellent numerical stability. These characteristics confirm that the improved algorithm effectively enhances the capabilities of the self-balancing vehicle system in rapid response, disturbance rejection, and stable estimation.

4.5 Algorithm Complexity and Real-time Analysis

Compared to the standard KF, L iterative updates are introduced at each time step in this algorithm. Each iteration requires one matrix inversion and three matrix multiplications, resulting in a computational complexity of $O\left(n^3\right)$, where n represents the state dimension. When implemented on embedded platforms with $n = 4$ and $L = 2$, a single filtering operation is completed in approximately 0.8 ms . This execution time meets the periodic requirements of most real-time control tasks, demonstrating that the improved algorithm maintains good engineering feasibility even in embedded systems with limited floating-point computational capabilities.

4.6 KF Design

The key to KF design is the selection of state variables, which affects the structure of the whole state equation. In this system, the inclination angle of the vehicle body was firstly selected as one state variable, and the zero drift value of the gyroscope was selected as another state variable. The state equation is as follows:

$$\theta_k = \theta_{k-1} + (\omega_{k-1} - b_{k-1})\, dt \tag{13}$$

where, θ_k denotes the angle prediction value at moment k, θ_{k-1} is the optimal angle estimation value at moment $k-1$, ω_{k-1} is the gyroscope measurement value at moment $k-1$, b_{k-1} is the gyroscope drift value at moment $k-1$, and dt is the sampling time. This design considers that the drift error changes very little between two adjacent moments, that is, $b_k = b_{k-1}$.

Eq. (13) in the matrix form can be expressed as follows:

$$\begin{bmatrix} \theta \\ b \end{bmatrix}_k = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ b \end{bmatrix}_{k-1} + \begin{bmatrix} dt \\ 0 \end{bmatrix} \omega_{k-1} \tag{14}$$

where, $A = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix}$ is the state transition matrix and $B = \begin{bmatrix} dt \\ 0 \end{bmatrix}$ is the control matrix. In addition, the measurement matrix is $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

The covariance matrix of the process noise is of the following form:

$$Q = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_\omega \end{bmatrix} \tag{15}$$

where, Q_θ and Q_ω are the covariances of the process noise of the vehicle inclination measured by the accelerometer and the angular velocity measured by the gyroscope, respectively. The two values represent the degree of trust of the KF for the data of the IMU module used, with smaller values representing a higher degree of trust.

The covariance matrix of the measurement noise is of the following form:

$$R = [R_\theta] \tag{16}$$

The larger R_θ is, the larger the component of the measurement value that is considered noise by the KF.

In this system, the angular velocity measured by the gyroscope is used as the input control quantity and the angle measured by the accelerometer is used as the measurement, and the optimal inclination angle is estimated by using the improved KF algorithmic process based on the iterative update of the measurement in this study.

5 PID Control Algorithm Based on the Hierarchical Control Strategy

The control of the two-wheeled self-balancing vehicle can be divided into three parts: balance, speed and direction control. To improve the control performance, stability and adjustability of the vehicle, a PID control algorithm was proposed in this study based on the hierarchical control strategy, which includes a balance control layer, a speed control layer and a direction control layer. This algorithm improves the structure of the traditional PID control algorithm and its design flow is shown in Figure 9.

5.1 Balance Control Layer

The balance control of the vehicle is actually to adjust its angle parameters according to the current inclination signal and angular speed signal of the vehicle through the motor drive of the wheel to keep the balance. This system adopts the method of angular velocity loop stringing into an angle loop to adjust the angle parameters of the vehicle.

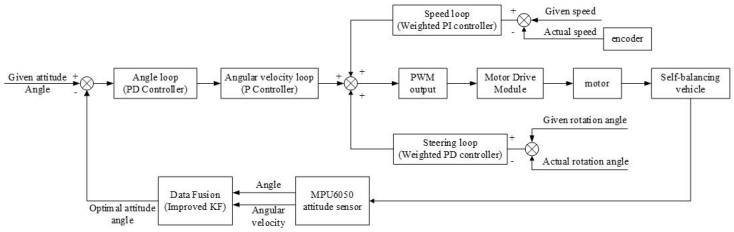

Figure 9. Block diagram of the improved PID control algorithm for the system

The change in proportional control output is proportional to the amount of deviation from its input and responds quickly to changes in error. When the error occurs, it produces a control effect to make the error change in a smaller direction, speeding up the response of the system. Derivative control improves the dynamic performance of the system by predicting the trend of the error through the time difference of the system and then controlling it in advance, which can reduce the overshoot and control the oscillation. In the self-balancing vehicle system, due to the existence of friction and other factors, the system may have the problem of integral saturation caused by the accumulation of deviation. Therefore, this system does not introduce the integral control in the balancing control. The output value obtained by the PD controller is defined as follows:

$$u_a = K_{p,a} \times (\theta - \theta_0) + K_{d,a} \times \dot{\theta} \tag{17}$$

where, $K_{p,a}$ and $K_{d,a}$ are the proportional coefficient and differential coefficient of the balance control layer, respectively, θ_0 is the angle setting value, θ is the optimized value of the vehicle body angle after the improved KF, $\dot{\theta}$ is the value of the vehicle body's angular velocity, and u is the force on the vehicle axle.

After passing through the PD controller, the angular velocity of the vehicle is updated at this time and is then strung into the angular velocity loop. By comparing the current angular velocity with the target angular velocity in the balance control, the vehicle can quickly reach the target velocity and output the PWM signal to drive the motor, keeping the angular velocity of the vehicle at 0. In addition, the target value of the angular velocity can be quickly adjusted to enable the system to better adapt to different operating conditions and more quickly restore the balance in the face of perturbations to improve the robustness of the system. The output value obtained by the angular velocity loop control is defined as follows:

$$u_g = K_{p,g} \times \left(\dot{\theta}_g - \dot{\theta}_0\right) \tag{18}$$

where, $\dot{\theta}_g$ is the updated angular velocity value after the PD controller, and $\dot{\theta}_0$ is the desired angular velocity value of the vehicle.

5.2 Speed Control Layer

The essence of the speed control of the vehicle is realized by controlling the rotational speed of the motor. PI control is one of the most commonly used control methods in speed control. In this study, a speed control algorithm based on weighted PI was further proposed. By adjusting the change speed of the accumulation and the degree of response to the error, the precise control of the vehicle speed is realized. The weighted operation was first performed using the following equation:

$$e = \alpha \cdot e_{\text{int}} + (1 - \alpha) \cdot e_v \tag{19}$$

where, e_{int} is the accumulation of the speed error in the previous moment; e_v is the deviation of the vehicle speed control in the current moment; α is the weight coefficient, which determines the degree of contribution of the previous accumulation to the new accumulation and then affects the degree of sensitivity of the accumulation to the system response; and e is the update to e_{int}.

Proportional control is used to reduce the deviation of the control system to maintain the balanced state of the vehicle in the process of movement. Integral control is used to form the control quantity for the continuous accumulation of error, eliminate the static error of the system, and enhance the anti-interference ability of the vehicle in the process of moving. And after the introduction of weighted operation, the degree of influence of the accumulation quantity can be adjusted according to the size of the error signal, which helps to better compensate for the error and

improve the precision and accuracy of the control system. The output value obtained after passing through the PI controller is defined as follows:

$$u_v = K_{p,v} \cdot e_v + K_{i,v} \cdot e \tag{20}$$

where, $K_{p,v}$ and $K_{i,v}$ are the proportional coefficient and integral coefficient of the speed control layer, respectively. By increasing the weight of the current error, the speed controller is more sensitive to the error change and can correct the error and adjust the speed more quickly.

5.3 Direction Control Layer

The essence of the direction control of the self-balancing vehicle is to realize the steering by controlling the difference between the rotation speeds of the two motors. In this study, a controller output update algorithm based on weighted PD was proposed. First, the proportional coefficient is multiplied with the steering angle deviation of the self-balancing vehicle. Therefore, the vehicle can be restored to the correct position when it deviates from the set rotation direction. In addition, to suppress the overshoot phenomenon during the direction adjustment of the self-balancing vehicle, the derivative coefficient is multiplied with the angle deviation of the adjacent moments of the self-balancing vehicle, as shown in Eq. (21). Then the steering control output is updated using the weighted operation, with the aim of introducing a certain degree of feedback control in the steering output and smoothing the fusion of the original output with the current steering error, which can reduce the rapid change of the control output and reduce the excessive response of the system to the steering error, as shown in Eq. (22). Finally, a limiting operation is performed on the steering control output. When the steering output deviates from the desired range, limiting its magnitude can prevent excessive steering angle from causing the vehicle to lose control or become unstable. In addition, through the limiting control, the system can respond more smoothly to the change of steering error, reduce the unnecessary oscillation phenomenon, and improve the overall stability and controllability of the system, which can make the system more robust in the case of noise, interference or other uncertainties.

$$u_0 = K_{p,d} \cdot e_v + K_{d,d} \cdot \Delta e_v \tag{21}$$

$$u_d = \beta \cdot u_0 + (1 - \beta) \cdot e_v \tag{22}$$

where, $K_{p,d}$ and $K_{d,d}$ are the proportional coefficient and derivative coefficient of the direction control layer, respectively, u_0 is the controller output, Δe_ν is the angle deviation of the vehicle at adjacent moments, u_d is the value after updating the controller output, and β is the weight coefficient.

5.4 Coupling Models and Collaborative Mechanisms in the Hierarchical Control System

Based on the separate design of controllers for the balance layer, the velocity layer, and the direction layer, a unified system coupling model was established to clarify the signal transmission relationships and dynamic interaction mechanisms among the control layers. In this model, the output of the balance control layer is used as the reference input for the velocity control layer, while the steering compensation from the direction control layer is incorporated into the balance control loop. The coupling relationships between different control layers are quantitatively described through the introduction of interaction coefficients, and system stability is achieved by adjusting the weighted allocation of these coefficients. Through this modeling approach, the dynamic interaction mechanisms between the control layers can be systematically analyzed, providing a theoretical foundation for parameter optimization of the hierarchical control system.

The system state vector is defined as:

$$\mathrm{x} = [\theta, \dot{\theta}, v, \omega, \phi]^T \tag{23}$$

where, θ represents the pitch angle, $\dot{\theta}$ denotes the pitch angular velocity, v indicates the vehicle body's linear velocity, ω signifies the steering angular velocity, and ϕ represents the heading angle. The dynamic equation of the system can be expressed as:

$$\dot{\mathrm{x}} = \mathrm{Ax} + \mathrm{B(u + d)} \tag{24}$$

where, $\mathrm{u} = [u_b, u_s, u_d]^T$ denotes the output vector of each control layer, d represents external disturbances, and A and B denote the system matrix and input matrix, respectively.

Each control layer achieves coordinated operation through the following mechanisms:

a) Feedforward coupling from the equilibrium layer to the velocity layer: The output ub of the balance controller also serves as the feedforward compensation term for the speed loop.

$$u_s = K_{pv}\left(v_{ref} - v\right) + K_{iv} \cdot \text{ WeightedIntegral } - \lambda u_b \tag{25}$$

where, λ is the equilibrium-speed coupling coefficient, which is used to suppress the interference of equilibrium regulation on vehicle speed.

b) Speed-to-balance-loop feedback compensation: The speed control output influences the angular velocity reference value of the balance loop through a weighting factor.

$$\dot{\theta}_{\text{ref}} = \gamma \cdot u_s \tag{26}$$

where, γ is the velocity-balance feedback coefficient.

Directional Layer Differential Integration: The direction controller output is distributed to the left and right motors via differential distribution:

$$\begin{aligned} u_L &= u_b + u_s + u_d \\ u_R &= u_b + u_s - u_d \end{aligned} \tag{27}$$

where, u_d represents the directional control output.

To achieve adaptive adjustment of control priorities, a time-varying weighting matrix is introduced:

$$\mathrm{W}(t) = \begin{bmatrix} w_b(t) & w_{bs}(t) & 0 \\ w_{sb}(t) & w_s(t) & 0 \\ 0 & 0 & w_d(t) \end{bmatrix} \tag{28}$$

Weight update rules are based on system status as follows:

$$w_b(t) = w_{b0} + \alpha \cdot |\theta| + \beta \cdot |\dot{\theta}| \tag{29}$$

$$w_s(t) = \frac{w_{s0}}{1 + \kappa \cdot |v - v_{\text{ref}}|} \tag{30}$$

where α, β, and κ are tuning parameters that ensure priority is given to maintaining balance during significant tilt angles and enhance speed tracking during substantial speed deviations.

The final control quantity is the weighted composite of outputs from each layer.

$$\mathrm{u}_{\text{final}} = \mathrm{M} \cdot \mathrm{W}(t) \cdot \begin{bmatrix} u_b \\ u_s \\ u_d \end{bmatrix} \tag{31}$$

where, M is the motor allocation matrix.

Through dynamic weight adjustment and the implementation of inter-layer feedforward-feedback mechanisms, cooperative control under balanced priority is achieved in this coupled model.

6 Experimental Simulation and System Test

6.1 Simulation and Analysis of the Improved KF

In the simulation process, the values of Q and R must be set reasonably to achieve a better filtering effect. The filter parameters of this system after debugging are $Q_\theta = 0.03, Q_\omega = 0.01$, $R_\theta = 200$, which can ensure that the system has fast dynamic response ability and can effectively suppress the interference of noise. Based on MATLAB, the algorithm simulation and analysis of the simulation data of the gyroscope and accelerometer in the self-balancing vehicle were carried out.

Figure 10 compares the effect of the KF algorithm, the iterative KF algorithm [6], and the proposed algorithm after fusion of gyroscope and accelerometer simulation data. As can be seen from Figure 10, if the attitude information measured by the accelerometer is used directly without the filtering algorithm, the interference noise is too large to accurately reflect the real value, which is not convenient for the subsequent control of the vehicle body. The KF and the iterative KF algorithm [6] can filter out the interference noise to a certain extent. The dynamic inclination curve's smooth filtering effect after using the algorithm in this study is more obvious, effectively reducing the noise interference and improving the precision of the vehicle body's attitude angle measurement. Figure 11 shows the angle estimation deviation corresponding to Figure 10 at each sampling moment, where $dt = 0.01$. From Figure 11, it can be seen that the proposed algorithm has an overall improvement in filtering precision compared to the other two algorithms, and the angular estimation deviation is smaller. Over this 10-s interval, the MSEs of the system after using the KF, the iterative KF algorithm [6], and the proposed algorithm are 0.5101, 0.4115, and 0.0135, respectively.

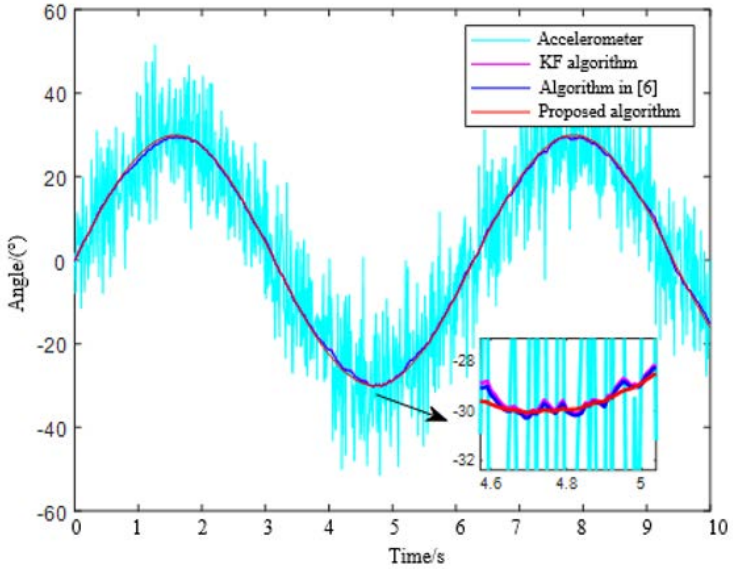

Figure 10. Comparison of data fusion effect using different algorithms under dynamic state

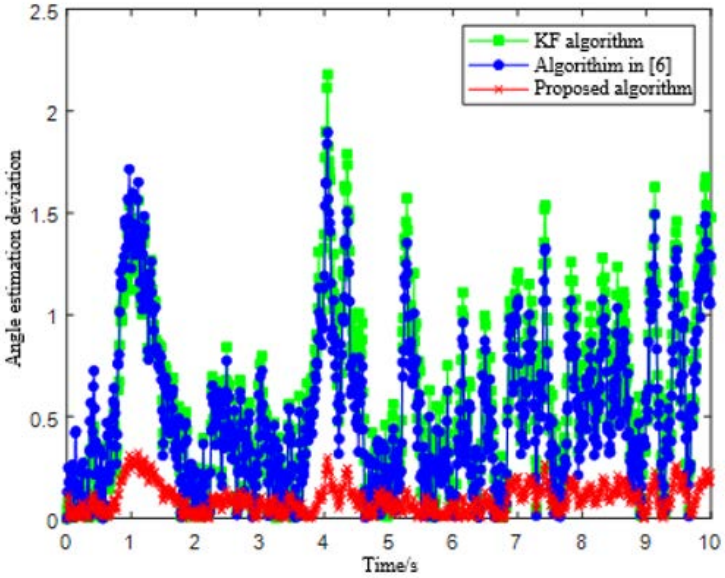

Figure 11. Comparison of angle estimation deviation after using different algorithms

The MSE effect of the system after 100 simulation experiments using the proposed algorithm and the other two algorithms is shown in Figure 12. It can be seen that compared with the other two algorithms, the proposed algorithm can realize continuous and stable attitude estimation, and the attitude angle estimation value of the vehicle is more precise, which improves the anti-interference of the two-wheeled balancing vehicle system. Comprehensive

experimental results show that the proposed method significantly improves the filtering precision under the dynamic state of the system, better tracks the angle change of the vehicle body in real time, and enhances the stability of the attitude detection system.

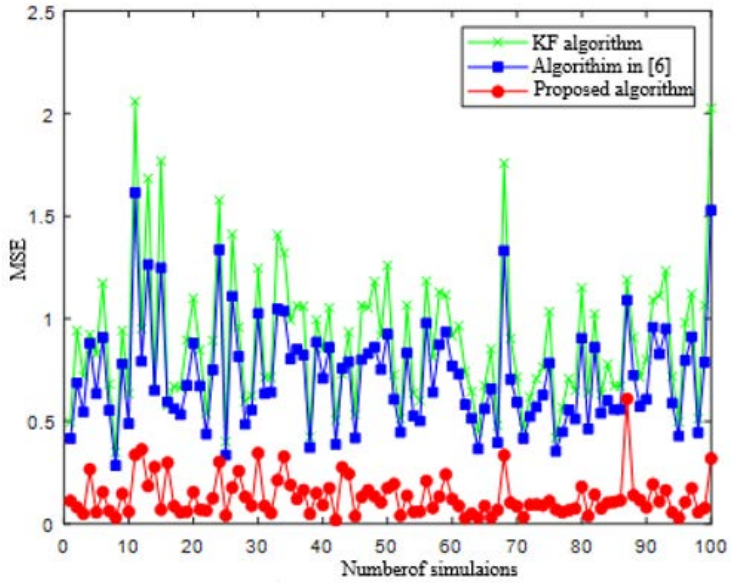

Figure 12. MSE comparison of the system after multiple simulations using different algorithms

6.2 Simulation and Analysis of the Improved PID Controller

Typical empirical ranges for PID parameter tuning can be found in the study by Wu [**?**]. The final tuned PID controller configuration parameters for the system are shown in Table 3. The signs of the parameters reflect the bidirectional control characteristics of the PID loops.

Table 3. Configuration parameters of the PID controller

Simulation Parameter	Value
Angle-loop proportional parameter $K_{p,a}$	55
Angle-loop derivative parameter $K_{d,a}$	-2
Angular-velocity loop proportional parameter $K_{p,g}$	3
Weight coefficient α	0.68
Speed-loop proportional parameter $K_{p,v}$	8
Speed-loop integral parameter $K_{i,v}$	17
Steering-loop proportional parameter $K_{p,\,\mathrm{d}}$	7
Steering-loop derivative parameter $K_{d,\,\mathrm{d}}$	5
Weight coefficient β	0.2

Figure 13 compares the effect of the PD control algorithm with the proposed algorithm on the balance control. The vertical axis represents the vehicle body's tilt angle (°), indicating equilibrium state, with a target value of 0°; the horizontal axis represents the time (s), recording the system recovery process following disturbance. It can be seen that, compared with the PD control algorithm, introducing the angular velocity loop reduces the oscillation of the vehicle during balancing, making the system more stable. Using the PD control algorithm and the proposed algorithm, the system basically restores to the balance position in about 1 s and 0.5 s, respectively. It shows that the proposed algorithm can accelerate the response time of the system and guarantee real-time vehicle control.

Figure 14 compares the effect of the PI control algorithm and the proposed algorithm on speed control. The vertical axis represents the vehicle speed (m/s), with the target speed being set at 1 m/s; the horizontal axis represents the time (s), recording the acceleration process from 0 to the target value. It can be seen that, compared with the PI control, the vehicle under the control of the proposed algorithm can reach the given speed faster and greatly reduce the overshoot, which is basically 0. It indicates that the proposed algorithm has better control performance.

The direction control performance of the two-wheeled self-balancing vehicle is mainly manifested as the accuracy to achieve the given angle of steering. Figure 15 compares the effect of the PD control algorithm with the proposed

algorithm on direction control. The vertical axis represents the vehicle's steering angle ($^\circ$), with the target steering angle being set at 45°; the horizontal axis represents the time (s), recording the process during which the steering angle increases from 0° to the target value and subsequently stabilizes. It can be seen that, although the response time of the PD algorithm and the proposed algorithm in reaching the given angle is about 1.5 s , the adoption of the proposed algorithm greatly reduces the system's oscillation, improves the stability, and reduces the sensitivity to noise and interference.

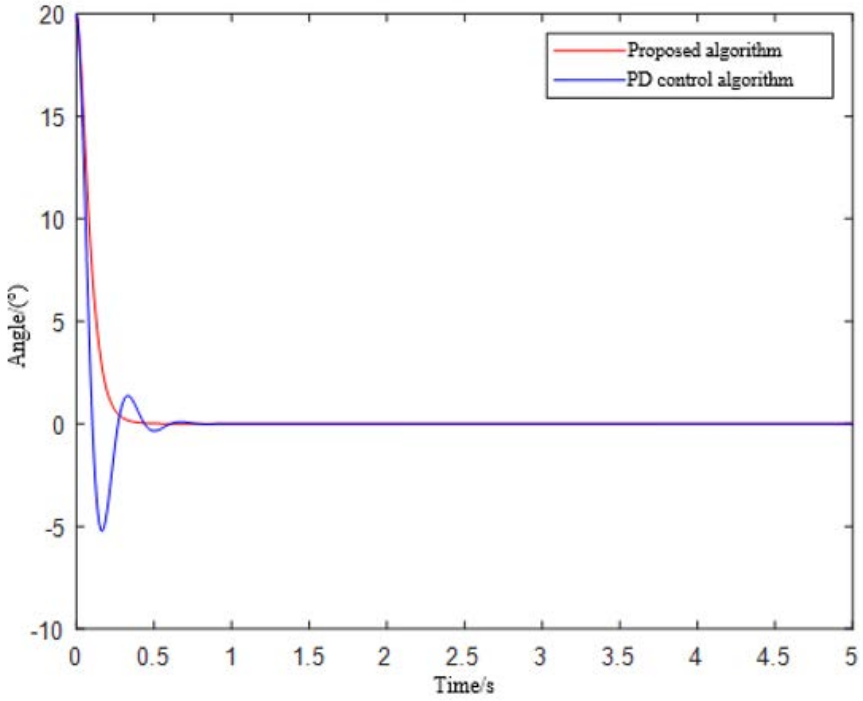

Figure 13. Comparison curves of the balance control effect using different algorithms

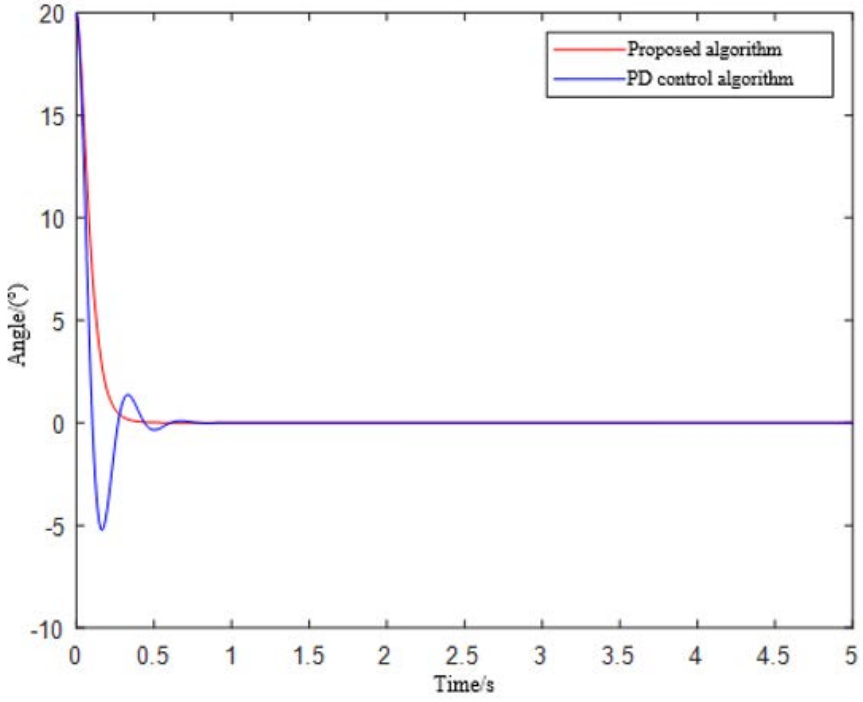

Figure 14. Comparison curves of the balance control effect using different algorithms

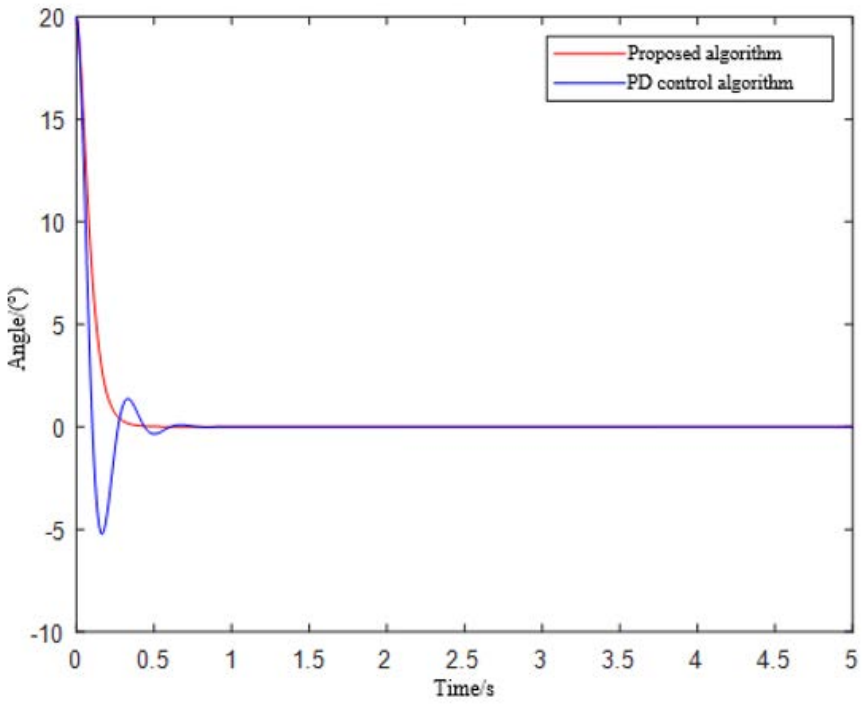

Figure 15. Comparison curves of the direction control effect using different algorithms

In summary, the effectiveness of this proposed algorithm in controlling the two-wheeled self-balancing vehicle is verified through experimental simulations, and the performance of this proposed control algorithm is better than the traditional PID control algorithm in the aspects of balance, speed, and direction control.

6.3 Performance Comparison of the Control Algorithm

Table 4 quantitatively compares the performance of the proposed control algorithm against traditional PD/PI methods across three control objectives: balance, speed, and direction. The comparison highlights the improvements in overshoot, steady-state error, and response time achieved by the new strategies, demonstrating their effectiveness in enhancing the stability and responsiveness of the two-wheeled self-balancing vehicle system.

Table 4. Quantitative performance comparison

Control Type	Comparison Algorithm	Overshoot (%)	Steady-state Error	Response Time (s)
Balance control (tilt angle)	PD control algorithm	8.2	0.8°	1.1
Balance control (tilt angle)	Proposed algorithm	0	0°	0.5
Speed control (vehicle speed)	PI control algorithm	15.5	0.06 m/s	0.9
Speed control (vehicle speed)	Proposed algorithm	0	0.01 m/s	0.4
Direction control (steering angle)	PD control algorithm	10.3	0.9°	2.2
Direction control (steering angle)	Proposed algorithm	0.8	0.2°	1.5

6.4 System Test

Based on the completed hardware and software design of the two-wheeled self-balancing vehicle system, comprehensive self-balancing and motion control tests were conducted. Each test item was repeated ten times to ensure statistical significance of the results. The physical prototype of the developed two-wheeled self-balancing vehicle is presented in Figure 16. Furthermore, comparative analysis between dynamic responses of control signals

and vehicle posture change curves, along with experimental verification against simulation results, will be conducted in subsequent research due to the current challenges in acquiring real-time data.

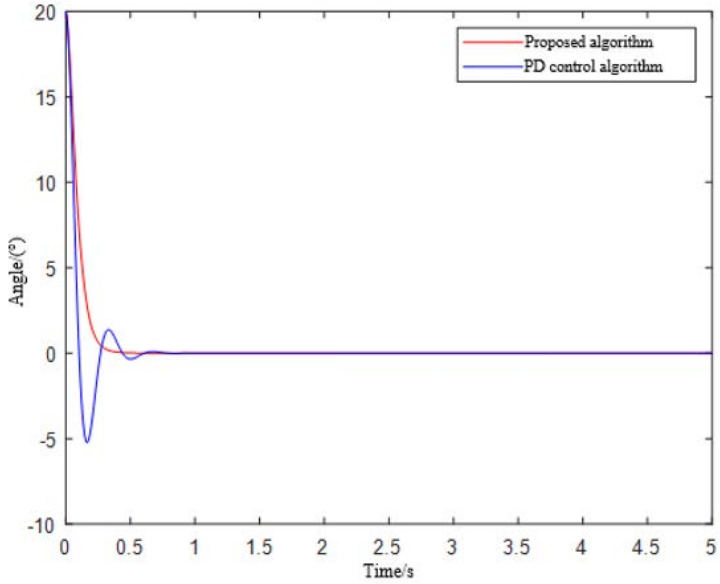

Figure 16. Physical picture of the two-wheeled self-balancing vehicle

All experimental procedures were carried out on smooth ceramic tile surfaces under indoor environmental conditions. The maximum surface tilt angle was verified to be less than 0.5° using a digital level instrument. The ambient temperature was maintained at $20 \pm 3^\circ\mathrm{C}$ throughout the testing period without external disturbances such as strong air currents. The power supply was provided by three freshly charged 18650 lithium-ion batteries connected in series configuration, with the initial voltage measured at 12.2 ± 0.2 V. Sensor data were acquired by the system through the Arduino's built-in ADC module, with the sampling frequency set at 200 Hz . The control algorithm was executed on a 5 ms cycle, with its execution being synchronized with the data acquisition process. Experimental data were transmitted in real time to the host computer via the Bluetooth module and were stored in CSV format for subsequent analysis.

During the self-balancing test, the power switch was first activated, and the vehicle was positioned on the test surface. To ensure testing accuracy, initial conditions were strictly controlled: the vehicle's initial pitch deviation must be maintained below 2°, while the surface inclination angle was kept within 0.5° to prevent hub overspeed caused by integral accumulation in the velocity loop. Subsequently, the motor start switch was engaged to activate the self-balancing mode. Test results demonstrated that stable equilibrium was consistently achieved within 1.2 s from the initial position across ten repeated experiments, with the steady-state balance error being maintained within $\pm 1.5^\circ$. Motion control testing was conducted through the Bluetooth interface of the mobile application. After the Bluetooth function was enabled in the application, the vehicle's Bluetooth module was searched for and paired with. Once the connection was successfully established, the module's indicator light remained steadily illuminated, confirming the establishment of a stable communication link. The testing protocol included forward movement, reverse movement, and steering functions. According to experimental data, while upright balance was maintained, control commands were accurately responded to by the vehicle: the precision of linear motion speed control reached ± 0.1 m/s, and the steering angle tracking error was maintained within 3°. Excellent repeatability and stability were demonstrated by the system throughout all ten repeated tests.

End-to-end communication latency was measured through timestamp comparison. The test results showed that at a distance of 0.5 m, the average communication latency was approximately 18 ms; at 3 m, the latency increased to about 25 ms; and at the maximum distance of 8 m, the latency reached approximately 45 ms. The latency at all tested distances meets the real-time control requirements of the system. In 500 data transmission tests, the packet loss rate remained within 0.2% at 0.5 m, increased to 0.8% at 3 m, and rose to 2.4% at 8 m. The test results confirmed that within the effective communication range of 8 m, the communication latency of the Bluetooth module was maintained below the system control cycle duration, while packet loss rates were controlled within acceptable limits. These performance characteristics satisfy the real-time control requirements for the self-balancing vehicle system.

7 Conclusion

In this study, a two-wheeled self-balancing vehicle system with Bluetooth control based on the ATmega328P microprocessor was designed. For the self-balancing vehicle attitude fusion problem, an improved KF algorithm based on iterative update of measurement was used to realize the data fusion of the gyroscope and accelerometer. Simulation results showed that the algorithm significantly improved the filtering precision of the system. For the self-balancing and motion control problem, an improved PID control algorithm was designed based on the hierarchical control strategy, which effectively addressed the susceptibility of the self-balancing vehicle system to external disturbances and poor responsiveness. After obtaining the system parameters suitable for actual operation through system debugging and analysis, the actual test showed that the Bluetooth-controlled self-balancing vehicle was able to execute the control commands issued by the mobile application while maintaining stable balance, achieving forward, backward, and steering motions with strong stability and anti-interference capability. Subsequently, the relevant methods of machine learning can be combined in the PID control algorithm to achieve real-time optimization of the PID controller parameters, thus achieving better control effects.

Funding

This research was funded by the Research Council of Shahid Chamran University of Ahvaz (Grant no. SCU.EM1403.72111).

Data Availability

The data used to support the research findings are available from the corresponding author upon request.

Conflicts of Interest

The author declares no conflict of interest.

References

[1] J. Tian, J. Ding, Y. Tai, and Z. Ma, "Control of different-axis two-wheeled self-balancing vehicles," *IEEE Access*, vol. 8, pp. 158 839–158 851, 2020. https://doi.org/10.1109/access.2020.3019538

[2] J. Velagic, I. Kovac, A. Panjevic, and A. Osmanovic, "Design and control of two-wheeled and self-balancing mobile robot," in *2021 International Symposium ELMAR, Zadar, Croatia*, vol. 2021, 2021, pp. 77–82. https://doi.org/10.1109/elmar52657.2021.9550938

[3] Z. Wang, J. Xu, Y. Li, Y. Zhang, and P. Shi, "Research and design of attitude detection system based on two-wheeled balanced robot," *Meas. Control Technol.*, vol. 38, no. 8, pp. 21–25, 2019. https://doi.org/10.19708/j.ckjs.2019.08.005

[4] Y. Su, T. Wang, K. Zhang, C. Yao, and Z. Wang, "Adaptive nonlinear control algorithm for a self-balancing robot," *IEEE Access*, vol. 8, pp. 3751–3760, 2020. https://doi.org/10.1109/access.2019.2963110

[5] C. Iwendi, A. Mohammed Alqarni, J. H. Anajemba, S. Ahmed Alfakeeh, Z. Zhang, and A. K. Bashir, "Robust navigational control of a two-wheeled self-balancing robot in a sensed environment," *IEEE Access*, vol. 7, pp. 82 337–82 348, 2019. https://doi.org/10.1109/access.2019.2923916

[6] Q. Wei, D. Chen, S. Lin, S. Qiu, and T. Zhang, "Simulation of multisensor data fusion based on iterative Kalman filter," *Comput. Technol. Dev.*, vol. 27, no. 9, pp. 137–140, 2017. https://doi.org/10.3969/j.issn.1673-629X.2017.09.030

[7] J. Zhu, H. Liu, X. Li, and D. Wang, "The multi-sensor information fusion research of selfbalancing two-wheeled electric vehicle based on EKF," *J. Electr. Eng.*, vol. 11, no. 6, pp. 25–32, 2016. https://doi.org/10.11985/2016.06.005

[8] A. Srichandan, J. Dhingra, and M. K. Hota, "An improved Q-learning approach with Kalman filter for self-balancing robot using OpenAI," *J. Control Autom. Electr. Syst.*, vol. 32, no. 6, pp. 1521–1530, 2021. https://doi.org/10.1007/s40313-021-00786-x

[9] Y. Huang, H. Chen, and L. Qin, "Design of self-balancing vehicle based on cascade PID control system," in *2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC)*. IEEE, 2022, pp. 1–4. https://doi.org/10.1109/ctisc54888.2022.9849694

[10] F. F. Rabbany, A. Qurthobi, and A. Suhendi, "Design of self-balancing virtual reality robot using PID control method and complementary filter," in *2021 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bandung, Indonesia*, vol. 2021, 2021, pp. 15–19. https://doi.org/10.1109/iaict52856.2021.9532576

[11] Y. Huang, C. Fu, Y. Jiang, and J. Lan, "Implement of two-wheeled self-balanced vehicle based on complementary filtering," *Autom. Instrum.*, vol. 38, no. 4, pp. 48–53, 2023. https://doi.org/10.19557/j.cnki.1001-9944.2023.04.011

[12] A. Unluturk and O. Aydogdu, "Machine learning based self-balancing and motion control of the underactuated mobile inverted pendulum with variable load," *IEEE Access*, vol. 10, pp. 104 706–104 718, 2022. https://doi.org/10.1109/access.2022.3210540

[13] S. Kim and S. Kwon, "Nonlinear optimal control design for underactuated two-wheeled inverted pendulum mobile platform," *IEEE/ASME Trans. Mechatron.*, vol. 22, no. 6, pp. 2803–2808, 2017. https://doi.org/10.1109/tmech.2017.2767085

[14] Z. Xu and Z. Li, "Application of cloud model controller in two-wheeled self-balancing robot," *Process Autom. Instrum.*, vol. 40, no. 5, pp. 70–74, 2019. https://doi.org/10.16086/j.cnki.issn1000-0380.2018100017

[15] L. Guo, S. A. A. Rizvi, and Z. Lin, "Optimal control of a two-wheeled self-balancing robot by reinforcement Q-learning," in *2020 IEEE 16th International Conference on Control and Automation (ICCA), Singapore*, vol. 2020, 2020, pp. 955–960. https://doi.org/10.1109/icca51439.2020.9264485

[16] T. Nikita and K. T. Prajwal, "Pid controller based two-wheeled self-balancing robot," in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India*, vol. 2021, 2021, pp. 1–4. https://doi.org/10.1109/icoei51242.2021.9453091

[17] M. Khaled Goher and O. Sulaiman Fadlallah, "Control of a two-wheeled machine with two-directions handling mechanism using PID and PD-FLC algorithms," *Int. J. Autom. Comput.*, vol. 16, pp. 511–533, 2019. https://doi.org/10.1007/s11633-019-1172-0

[18] E. Philip and S. Golluri, "Implementation of an autonomous self-balancing robot using cascaded PID strategy," in *2020 6th International Conference on Control, Automation and Robotics (ICCAR), Singapore*, vol. 2020, 2020, pp. 74–79. https://doi.org/10.1109/iccar49639.2020.9108049

[19] R. Singh and B. Bhushan, "Improved ant colony optimization for achieving self-balancing and position control for balancer systems," *J. Ambient Intell. Human Comput.*, vol. 12, pp. 8339–8356, 2020. https://doi.org/10.1007/s12652-020-02566-y

[20] L. Jiang and J. He., "Control strategy and dynamic simulation of two-wheeled self-balancing vehicle," *J. South China Univ. Technol. (Nat. Sci. Ed.)*, vol. 44, no. 1, pp. 9–15, 2016.

[21] H. Shen, X. Jiang, and T. Feng, "Neural control algorithm and simulation of two-wheel balancing vehicle," *Mach. Tool Hydraul.*, vol. 50, no. 19, pp. 159–166, 2022. https://doi.org/10.3969/j.issn.1001-3881.2022.19.028

[22] H. Ren and C. Zhou, "Control system of two-wheel self-balancing vehicle," *J. Shanghai Jiaotong Univ. (Sci.)*, vol. 26, pp. 713–721, 2021. https://doi.org/10.1007/s12204-021-2361-x

[23] H. Pang, M. Liu, C. Hu, and F. Zhang, "Adaptive sliding mode attitude control of two-wheel mobile robot with an integrated learning-based RBFNN approach," *Neural Comput. Appl.*, vol. 34, no. 17, pp. 14 959–14 969, 2022. https://doi.org/10.1007/s00521-022-07304-3

[24] D. N. Anisimov, T. S. Dang, S. Banerjee, and T. A. Mai, "Design and implementation of fuzzy-PD controller based on relation models: A cross-entropy optimization approach," *The European Physical Journal Special Topics*, vol. 226, no. 10, pp. 2393–2406, 2017. https://doi.org/10.1140/epjst/e2017-70069-y

[25] H. Ren, Q. Wu, and T. Shi., "Self balancing control of two-wheel self-balancing vehicle based on learning mechanism," *Mach. Des. Manuf.*, vol. 387, no. 5, pp. 283–286, 2023. https://doi.org/10.3969/j.issn.1001-3997.2023.05.057

[26] S. Q. Wang and W. T. Xiong, "The design of the two-wheel self-balancing vehicle system based on STM32," *Res. Explor. Lab.*, vol. 35, no. 5, pp. 146–150, 2016.

[27] Z. Liu, J. Feng, W. Li, and Q. Zhao, "Design of two-wheeled balance car system under multi-loop PID control," *Process Autom. Instrum.*, vol. 43, no. 11, 2022. https://doi.org/10.16086/j.cnki.issn1000-0380.2022030085

[28] H. Wu, *Principles of Automatic Control*. Wuhan: Huazhong University of Science and Technology Press, 2024. https://www.amazon.com.au/Principles-Automatic-Control-Fourth-Chinese/dp/7568095967