



# A Cascade PD Controller for Heavy Self-balancing Robot

Michał Okulski<sup>(✉)</sup> and Maciej Ławryńczuk<sup>(✉)</sup>

Institute of Control and Computation Engineering,  
Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland  
michal@okulski.pl, M.Lawrynczuk@ia.pw.edu.pl

**Abstract.** This paper describes control system for dynamic equilibrium finding of a self balancing (two-wheeled) heavy robot. Two cascade-connected Proportional-Derivative (PD) controllers are used to balance the robot and keep the desired driving speed (or standing still). A simple and efficient algorithm for tilt calculation takes data from three sensors: a gyroscope, an accelerometer and a contactless magnetic encoder. The PD controller output is combined with manual (remote) driving signals to control Electronic Speed Controllers (ESC) of two high-torque electric motors. Finally, experimental results and recommendations to cope with difficulties are discussed.

**Keywords:** Self-balancing robot · Cascade PD controller  
Gyroscope · Accelerometer · Feedback control

## 1 Introduction

The inverted pendulum is a classical benchmark in control theory [1, 3]. A two-wheeled self-balancing robot shown in Fig. 1 is a practical utilisation of the inverted pendulum concept. Similar robots are considered in numerous publications, e.g. [2, 5, 6]. Usually, off-line simulation based on model state equations [6] works well. However, the theoretical model must be extended to take into account features of the real robot. A more complex mathematical model, which includes friction, is described in [9], but still, there are some factors which are difficult to model, e.g. loosely fit gearbox elements, soft tires, subtle error between actual and calculated tilt angle [4, 11]. Different control algorithms may be used for the self-balancing robot: the classic PID controller [6], a fuzzy controller [10], the LQR controller [7] or a neural-network controller [8].

This paper presents an approach to design, implement and tune a cascade PD controller, without identifying details of the mathematical model. The controller is applied to an original laboratory two-wheeled self-balancing robot depicted in Fig. 1b. Two online experiments are discussed in this paper:

1. General stability of the robot when desired velocity is equal to 0 m/s.  
Expected result: the robot should stay still.

2. Dynamic ability to find a new equilibrium, when the robot's centre of mass has been moved.

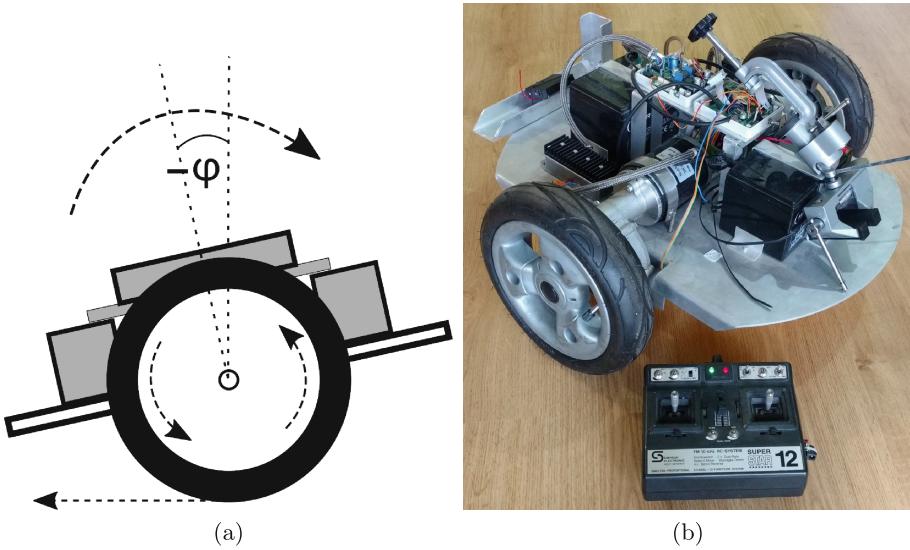


**Fig. 1.** The self-balancing robot: (a) a schematic, (b) the real robot described in this paper

## 2 Laboratory Self-balancing Robot

### 2.1 Hardware

A heavy, two-wheeled, self-balancing robot shown in Fig. 1b has been built. Robot chassis is made of custom bent aluminium sheet (4 mm). There are two electric motors: one per each wheel. Each motor (24 V DC, 200 W) is integrated with a 1:12 gearbox. Wheels are tight-coupled to the gearbox output shafts. Additionally, a single neodyme magnet is attached directly to the motor shaft – it is used for contactless magnetic encoder, only one in this experiment (the left motor only). The battery (24 V, 12 Ah) is composed of two serial connected Pb accumulators. The complete chassis (including motors, wheels, batteries) weight approx. 30 kg. Robot control system, depicted in Fig. 2, contains the following logic blocks:

- (a) 3 sensors: LSM303 – a 3-axis accelerometer and 3-axis magnetometer (magnetometer is not used in this experiment), L3G4200D – a 3-axis gyroscope and AS5040 – a contactless magnetic motor shaft encoder,

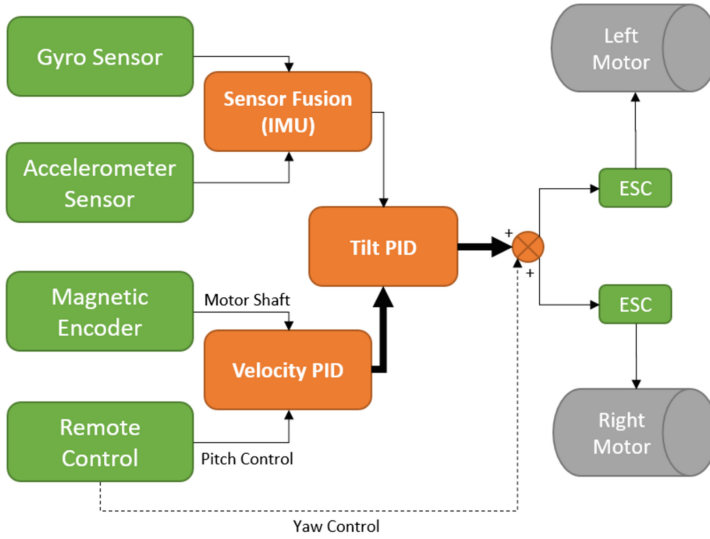


**Fig. 2.** Control system overview

- (b) a custom implementation of Inertial Measurement Unit (IMU) to perform a sensor fusion and calculate the robot's tilt angle,
- (c) two cascade PD controllers to control the velocity and the tilt angle of the robot,
- (d) an external remote controller (a radio transmitter) to allow operator force yaw (turns) and force speed and driving direction (forward or backward, with variable speed).

Electronic system of the robot is divided into a few separate circuits: a motherboard, a power adapter board and two PWM-controlled ESCs (Electronic Speed Controllers). The heart of the motherboard is a 8-bit microcontroller (AVR core): Atmel's AT90CAN128, running at 14.7456 MHz. The sensors are attached directly to the microcontroller's interfaces: IMU sensors via the I<sup>2</sup>C bus, motor encoder via the SPI serial interface.

### 3 Inertial Measurement Unit (IMU)

A 3-axis Micro-Electro-Mechanical-System (MEMS) digital gyroscope of the L3G4200D type and a 3-axis MEMS digital accelerometer of type LSM303 are used. All three axes from the accelerometer and just the pitch axis of the gyroscope are used in this IMU. Signals of both used MEMS sensors are noisy. Additionally, the MEMS gyroscope result value can be shifted by a constant component, due to temperature drift. The low-pass filter has been used in this IMU to decrease the amount of sensors' white noise. A simplified, discrete, single-

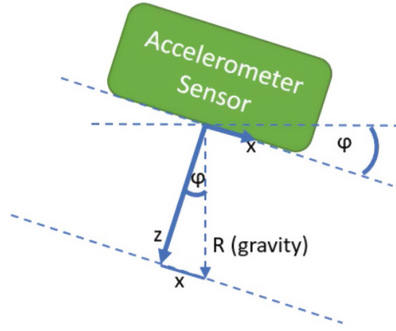


**Fig. 3.** Accelerometer  $x$  and  $z$  values against the gravity vector

dimension version of a low-pass is calculated from

$$y(k) = \frac{\alpha y(k-1) + x(k)}{\alpha + 1} \quad (1)$$

where  $y(k)$  is the current output value,  $y(k-1)$  is the previous output value,  $x(k)$  is the current sensor input value and  $\alpha$  is a gain of the filter.

### 3.1 Accelerometer Data

To obtain a tilt angle from an accelerometer sensor (in static conditions) first the length  $R(t)$  of a gravity vector at the time  $t$  is calculated

$$R(t) = \sqrt{x(t)^2 + y(t)^2 + z(t)^2} \quad (2)$$

and then the tilt angle is found

$$\varphi_{\text{acc}}(t) = \arcsin\left(\frac{x(t)}{R(t)}\right) \quad (3)$$

Relation between the gravity vector and the searched tilt angle is presented in Fig. 3.

### 3.2 Gyroscope Data

A MEMS gyroscope captures the angular velocity  $\omega(t)$ . The integral

$$\varphi_{\text{gyro}}(t) = \int_0^t (\omega(\tau) - b) d\tau \quad (4)$$

is used to calculate the tilt angle at the time  $t$ , where  $b$  is the gyroscope bias component (initially set to 0) used to compensate sensor's drift. There are two issues of the MEMS gyroscope sensor: a white noise, which may be easily cancelled by a low-pass filter (Eq. (1)) and a temperature drift, which slightly shifts the sensor's output value  $\omega$ . Because  $\omega$  is continuously integrated over time (Eq. (4)), uncompensated drift finally makes the calculated angle  $\varphi_{\text{gyro}}$  useless.

### 3.3 IMU Sensors Fusion Algorithm

There are a few solutions that make it possible to precisely calculate the tilt angle using accelerometer and gyroscope sensors fusion:

- (a) The Kalman Filter – it is an optimal linear filter, but it requires a well-matched model of sensors (and their error covariances) [7],
- (b) A custom (simplified) algorithm, which uses the filtered accelerometer value for long-term gyroscope drift correction, and integrated gyroscope value (just slightly adjusted to match filtered accelerometer output) as the tilt angle value.

IMU described in this paper uses the latter approach. The accelerometer-based tilt angle is accurate in static conditions only. Every move, jerk or vibration of the robot, introduces additional forces, which are read by the accelerometer. That effect is not intended. It is important not to try to match the gyroscope-based tilt angle to the accelerometer-based tilt angle too strict. The algorithm implemented in this IMU is:

1. The tilt angle is calculated using the accelerometer (Eq. (3)) and the gyroscope (Eq. (4)).
2. The low-pass filter is used to calculate the angles (Eq. (1)); each filter is tuned individually.
3. The error is calculated:  $e_\varphi(t) = \varphi_{\text{acc}}(t) - \varphi_{\text{gyro}}(t)$ .
4. Let the  $\varphi(t) = \varphi_{\text{gyro}}(t)$ .
5. If  $|e_\varphi(t)| > 1.5^\circ$  then the  $\varphi(t)$  is adjusted:  $\varphi(t) = \varphi(t) + e_\varphi(t)/20$  and the error is cumulated in  $S_e$ .
6. If  $|S_e| > 50$  then the gyroscope bias is updated:  $b = b + S_e/1000$  and the absolute value of the cumulated error is decreased:  $S_e = S_e - 50\text{sign}(S_e)$ .

Every constant used in the algorithm has been found experimentally.

## 4 Cascade PD Controller

The controller has been intentionally designed without identifying robot's mathematical model. Instead, a set of simple and intuitive rules has been defined:

1. *If the robot tilts forward, then drive forward.* Do the same for the opposite direction: *If robot tilts backward, then drive backward.*
2. *If the robot still tilts forward, then drive forward faster.* Do the same for the opposite direction.
3. *If the robot's tilt angle is correct, but it is driving forward, then set the desired tilt angle a little bit to the back.* Do the same for the opposite direction.

Stabilization of two-wheeled self-balancing robot has been divided into two separate tasks – each of them is simple enough to be successfully solved by a

linear controller: a parallel variant of the classic PD controller, computed using formula

$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt} \quad (5)$$

where  $e(t)$  is the control error at the time  $t$  and  $K_P$  and  $K_D$  are the controller proportional and derivative gains, respectively. Two cascade PD controllers are used to stabilize and drive the robot described in this paper:

1. High level: the velocity PD controller – it keeps the desired velocity of the robot by adjusting the desired tilt angle,
2. Low level: the tilt PD controller – it keeps the desired tilt angle by controlling wheel rotation (direct control of electric motors).

#### 4.1 Velocity PD Controller

The controller output is computed using formula

$$u_v(t) = K_{Pv} e_v(t) + K_{Dv} \frac{de_v(t)}{dt} \quad (6)$$

The error is defined as

$$e_v(t) = x_{\text{enc}}(t) + r_{\text{velocity}}(t) \quad (7)$$

where  $x_{\text{enc}}(t)$  is an average amount of motor shaft rotation per second during the last 20 ms and  $r_{\text{velocity}}(t)$  is a manual remote control signal – used by the operator to force desired driving velocity vector.

#### 4.2 Tilt PD Controller

The controller's error value is adjusted by the velocity PD controller output

$$e_\varphi(t) = \varphi(t) + \int_0^t u_v(\tau) d\tau \quad (8)$$

where  $\varphi(t)$  is the current tilt angle given by the IMU and  $u_v$  is the output of the velocity PD controller (Eq. (6)). The controller is computed from

$$u_\varphi(t) = K_{P\varphi} e_\varphi(t) + K_{D\varphi} \frac{de_\varphi(t)}{dt} \quad (9)$$

#### 4.3 Motor Controller

Electronic speed controller signals for left and right motor are calculated as follows

$$M_L(t) = u_\varphi(t) + r_{\text{yaw}}(t) \quad (10)$$

$$M_R(t) = u_\varphi(t) - r_{\text{yaw}}(t) \quad (11)$$

respectively, where  $u_\varphi(t)$  is the tilt PD controller output (Eq. (9)) and  $r_{\text{yaw}}(t)$  is a manual remote control signal – used by the operator to force yaw (turn left or right).

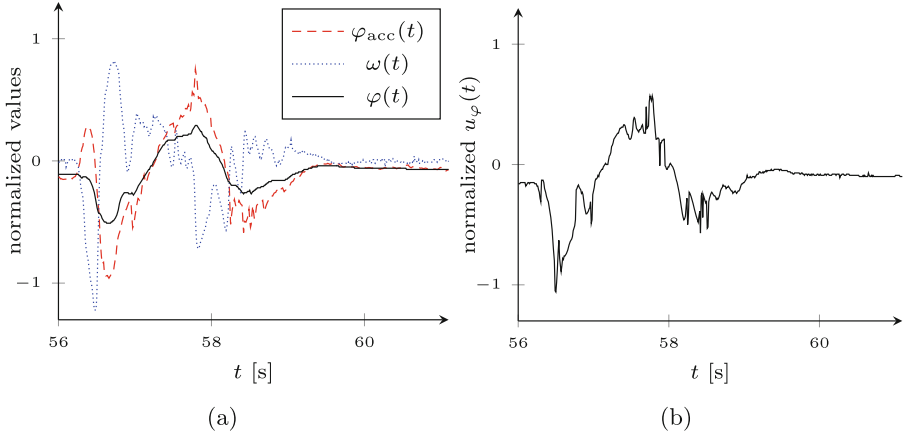


**Fig. 4.** Process of robot stabilization after knocking it out: (a) IMU data, (b) control signal from the tilt PD controller

#### 4.4 Adjusting PD Controllers' Gains

Both PD controllers have been tuned experimentally online using spare analog channels of the remote control transmitter. Many adjusting approaches have been tested experimentally, but finally the following algorithm has been used:

1. Set the  $K_P = 0$  and the  $K_D = 0$ .
2. The  $K_D$  is being increased as long as there are no high frequency motor vibrations.
3. The  $K_P$  is being increased as long as the overall process is still stable. Higher  $K_P$  value gives shorter total stabilization time.

## 5 Experimental Results

### 5.1 General Stability

The goal of this experiment is to tune the robot's control system and check whether it is able to stay still without (or with just minimal) tilt oscillations and if it is able to get back to stable state after a manual knock out.

First of all the IMU had to be tuned. After a series of tests, all the coefficients described in Sect. 3.3 have been set up. Then the tilt PD controller (Eq. (9)) has been tuned, using steps described in Sect. 4.4. Telemetry data from this experiment is presented in Fig. 4. Robot typically comes back into a stable state after approx. 3s.

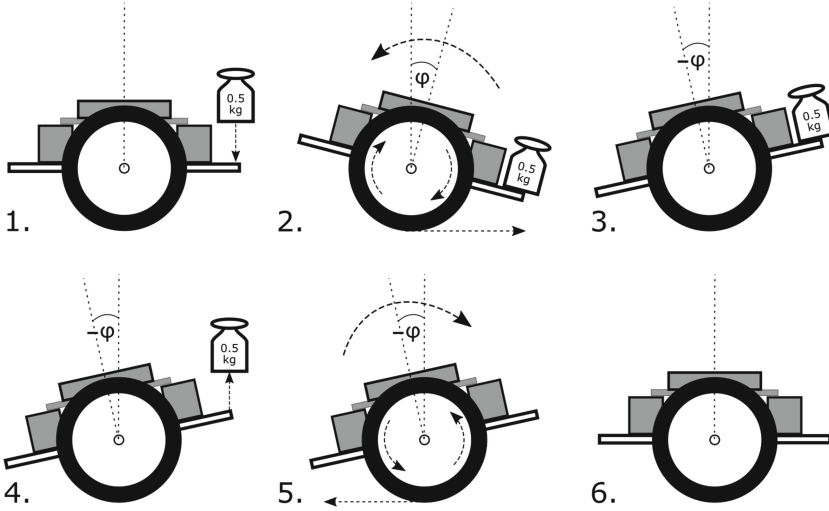


Fig. 5. Automatic equilibrium finding experiment steps

## 5.2 Automatic Finding of Equilibrium

The second experiment checks robot's ability to find a new equilibrium after its center of mass has been changed. This experiment requires fine tuning of the velocity PD controller (Eq. (6)). Figure 5 illustrates the steps performed to test the automatic equilibrium finding. The procedure is:

1. Start condition:  $\varphi = 0^\circ$  (robot's tilt angle),  $x_{\text{enc}} = 0$  m/s (robot's velocity measured by the encoder sensor). An additional mass (0.5 kg in this experiment) is attached at the edge of robot's chassis.
2. Expected result: robot tilts towards the added mass and the tilt PD controller start minimizing that tilt by running robot's wheels. Tilt angle quickly matches the desired value ( $0^\circ$ ), but robot is still driving while it should stay still. The desired tilt angle is adjusted by the velocity PD controller until the robot is driving.
3. Finally, robot's wheels are not driven by the tilt PD controller any more to keep the new tilt angle. A new equilibrium has just been found.
4. The 0.5 kg element is removed.
5. The tilt PD controller starts driving robot's wheels to keep current tilt angle. The velocity PD controller starts decreasing the target tilt angle because the encoder input is not equal to zero (robot is still driving). The tilt PD controller follows the new target tilt angle and robot decelerates.
6. Finally, the robot is standing still. A new equilibrium has just been found again.

Detailed telemetric data from that experiment is presented in Figs. 6 and 7. Finding new equilibrium usually takes approx. 2 s.

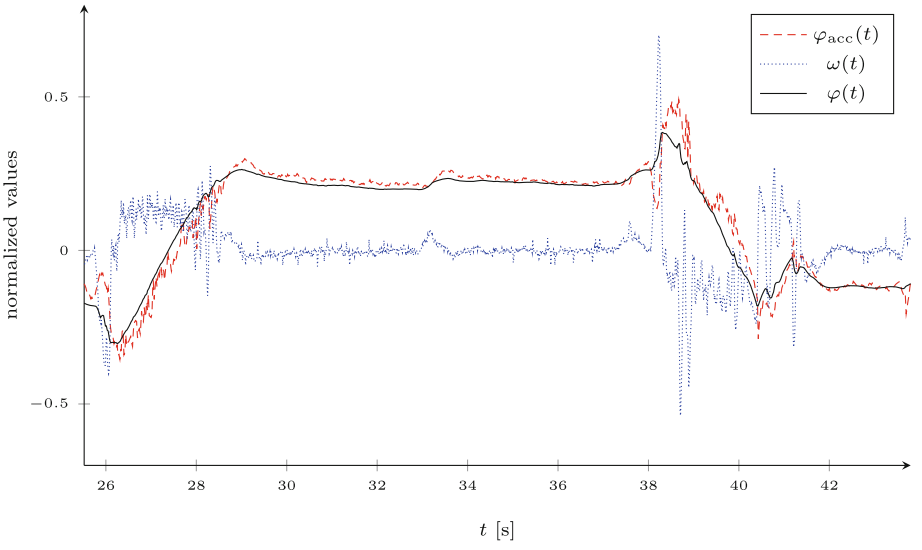


**Fig. 6.** Automatic equilibrium finding experiment: IMU data

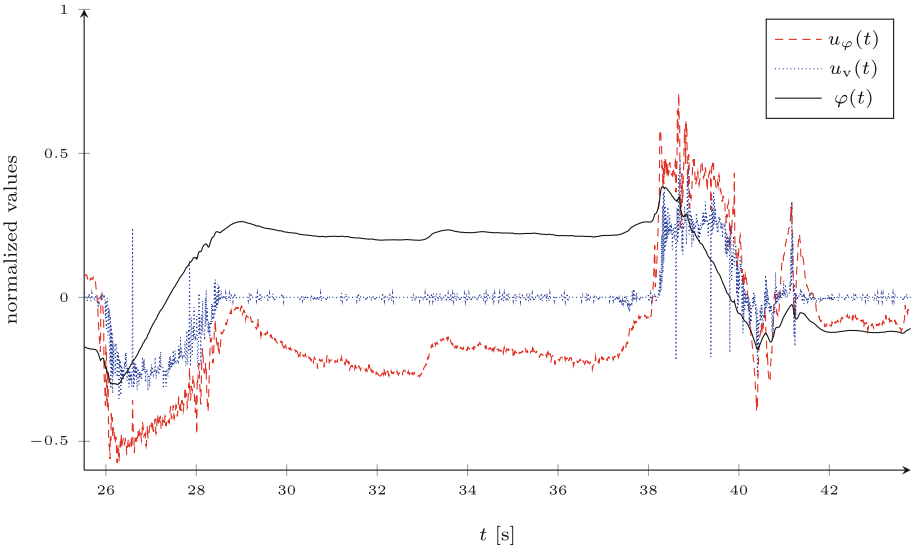


**Fig. 7.** Automatic equilibrium finding experiment: signals from both PD controllers against the tilt angle  $\varphi$

## 6 Conclusions and Future Research

Performed experiments have shown that the described cascade PD controller system works well in static conditions (stabilization) as well as in dynamic conditions (equilibrium finding). At first, the controllers have been tuned experimentally. Further optimizations have been applied to achieve the final results: decreasing IMU filtering delays (for more accurate tilt angle calculation) and increasing the frequency of the control loop (up to 100 Hz). Other minor adjustments: decreasing air pressure in tires (softer tires give more grip), limiting maximum motor signal (to avoid wheels' jerks). Suggested future researches: improve the IMU by applying a well-tuned Kalman Filter and use newer MEMS sensors. As a result, the whole control system is expected to be more aggressive: robot should accelerate, drive and decelerate much faster.

## References

1. Åström, K.J., Murray, R.M.: *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton (2007)
2. Bin, H., Zhen, L.W., Feng, L.H.: The kinematics model of a two-wheeled self-balancing autonomous mobile robot and its simulation. In: *International Conference on Computer Engineering and Applications*, pp. 64–68 (2010)
3. Franklin, G.F., Powell, J.D., Emami-Naeini, A.: *Feedback Control of Dynamic Systems*. Pearson, London (2015)
4. Güner, U., Canbolat, H., Ünlütürk, A.: Design and implementation of adaptive vibration filter for MEMS based low cost IMU. In: *9th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 130–133 (2015)
5. Mahler, B., Haase, J.: Mathematical model and control strategy of a two-wheeled self-balancing robot. In: *39th Annual Conference on IECON 2013*, pp. 4198–4203. Industrial Electronics Society (2013)
6. Majczak, M., Wawrzyński, P.: Comparison of two efficient control strategies for two-wheeled balancing robot. In: *The 20th International Conference on Methods & Models in Automation & Robotics (MMAR)*, Miedzyzdroje, Poland, pp. 745–746 (2015)
7. Sun, F., Yu, Z., Yang, H.: A design for two-wheeled self-balancing robot based on Kalman filter and LQR. In: *International Conference on Mechatronics and Control (ICMC)*, pp. 612–616 (2014)
8. Sun, C., Lu, T., Yuan, K.: Balance control of two-wheeled self-balancing robot based on linear quadratic regulator and neural network. In: *Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 862–867 (2013)
9. Teixeira, H.T., de Mattos Siqueira, V.S., Munaro, C.J.: Comparison of methods for estimation and compensation of friction applied to an inverted pendulum. In: *9th IEEE International Conference on Control and Automation (ICCA)*, pp. 818–822 (2011)
10. Yong, Q., Yanlong, L., Xizhe, Z., Ji, L.: Balance control of two-wheeled self-balancing mobile robot based on TS fuzzy model. In: *International Forum on Strategic Research*, pp. 406–409 (2011)
11. Yosi, S., Agung, N.J., Unang, S.: Tilt and heading measurement using sensor fusion from inertial measurement unit. In: *International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pp. 193–196 (2015)