MODELLING, SIMULATION, AND FABRICATION OF TWO WHEELED AUTONOMOUS SELF-BALANCING ROBOT WITH OBSTACLE DETECTION

A self-balancing robot is an intrinsically unstable system that is based on an inverted pendulum design [1]. Unlike conventional 4 wheel or 4 legged mobile robots that consume a lot of space, two wheel self-balancing robots consume less space and have several other advantages which include: movement on zero radius curve, high tolerant to impulsive force, small foot print to move on dangerous places, and greater stability over slopes [2]. As such, commercial products such as the Segway [3] have been developed. Besides the development of Segway, studies of two-wheel self-balancing robots have been widely reported. For example, JOE [6] that is used for radio control, made by Swiss Federal Institute of Technology Switzerland and nBot [9] are both early versions complete with on-vehicle microcontrollers. There has also been active research on the control design for such platforms, including classical and linear multivariable control methods [6], [8], nonlinear back stepping controls [10] and Fuzzy Adaptive PID Control [5]. In this project, a model of a self-balancing robot was developed to understand the effectiveness of the PID control algorithm.

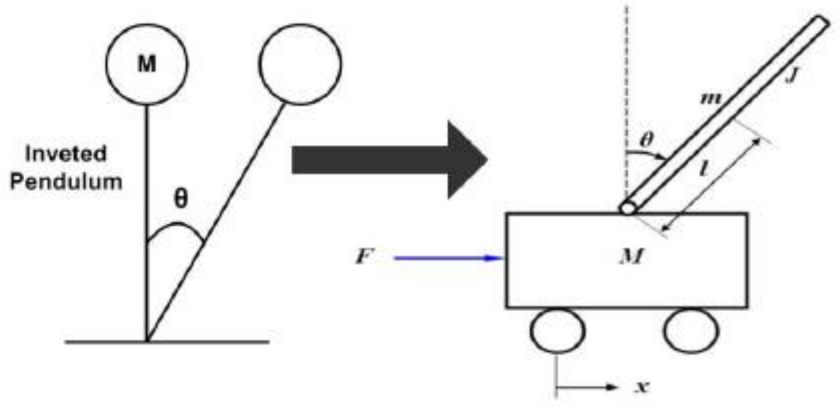


Fig. 1. An inverted pendulum model of a self-balancing robot

The basic idea behind a self-balancing robot is simple: move the actuator in a direction to counter the direction of fall, much like in balancing a broom at the tip of your finger. In practice, however, this requires two feedback sensors: an angle sensor to measure the tilt of the robot with respect to gravity, and an accelerometer to calibrate the gyroscope. Used individually, the accelerometer is prone to error in the short term due to horizontal acceleration and vibrations of the robot. The gyroscope, on the other hand, is a good predictor of tilt in the short term but has significant drift in the long term. Usually, a math filter (either Kalman filter or complimentary filter) is used to mix and merge the two values in order to have the correct value. Due to the complexities associated with the Kalman filter (it is very cumbersome, difficult to understand, and challenging to implement on an Arduino microcontroller), for this project, the complimentary filter was chosen to minimize the high frequency noise of the accelerometer and low frequency noise of the gyroscope to give the best accurate angle of inclination as shown below:

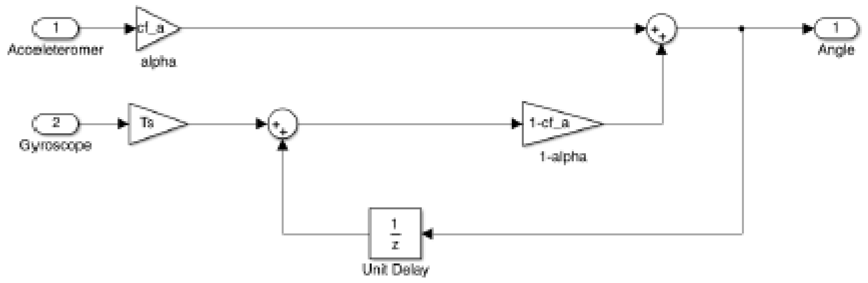


Fig.2. Block diagram showing how the complementary filter fuses accelerometer and gyroscope data.

For obstacle detection, an ultrasonic sensor was used. The ultrasonic sensor simply converts electrical energy into ultrasonic sound waves and vice versa. It emits short, high-frequency sound pulses at regular intervals. These propagate in the air at the velocity of sound. If they strike an object, then they are reflected back as echo signals to the sensor, which itself computes the distance to the target based on the time-span between emitting the signal and receiving the echo. The selected HC-SR04 ultrasonic sensor generates ultrasonic waves at 40 kHz frequency.

DESIGN OVERVIEW

This section presents the robots hardware considerations.

Hardware:

(i) MPU6050 Accelerometer and Gyroscope
(ii) HC-SR04 ultrasonic sensor-virtually all materials which reflect sound can be detected with the HC-SR04 ultrasonic sensor, regardless of their color. Even transparent materials or thin foils represent no problem for an ultrasonic sensor. Ultrasonic sensors can also see through dust-laden air and ink mist.
(iii) 12V DC Motors
(iv) Wheels
(v) L298N Motor driver
(vi) Arduino Uno
(vii) Chassis

Selection of the Arduino Uno development board was based on the following considerations:

(i) Open Source: To alleviate difficulties in programming, a user-friendly development environment, useful function libraries and references was preferred. These requirements are well met by the Arduino development environment which is based on the C language. User-contributed function libraries like PWM control, I2C and SPI communication reduce difficulties in learning to program the Arduino boards. Most importantly, Arduino is open-source with a large user community and up-to-date discussion forums. This allows students to study other users' codes, compare results, and make modifications according to the project's needs.

(ii) Price and Expansions: Arduino boards are low-cost and expandable, where shields can be purchased as and when needed. The accessibility of these products in terms of price versus functionality makes it ideal solutions for this project.

BASIC LAYOUT OF THE DESIGN

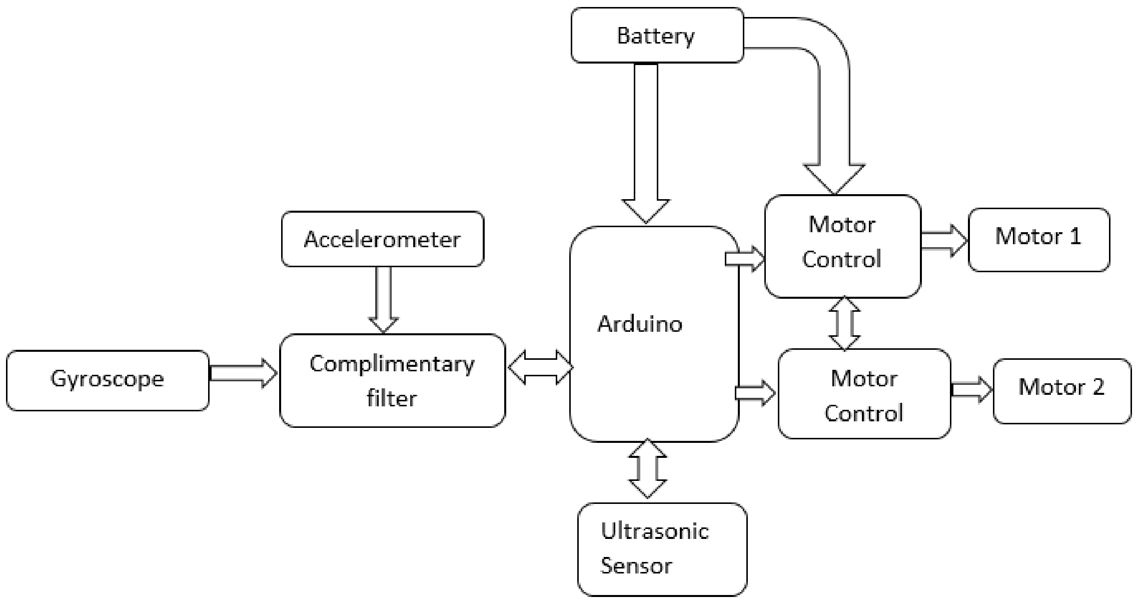


Fig. 3.

CAD MODELS

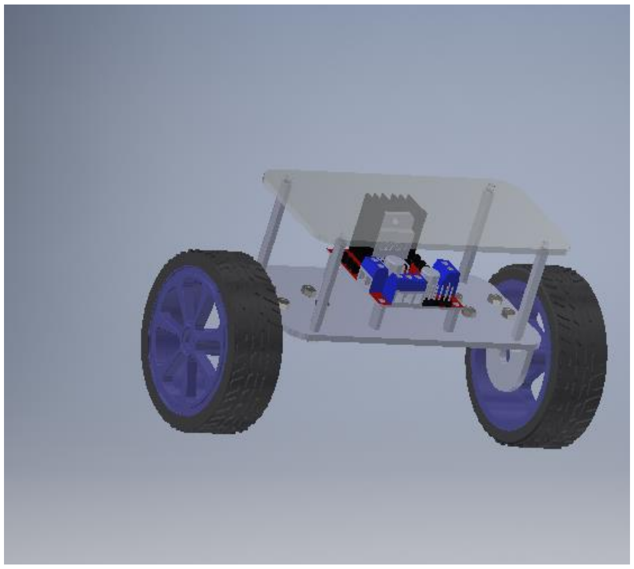


Fig.4. Chassis plus L298N motor driver

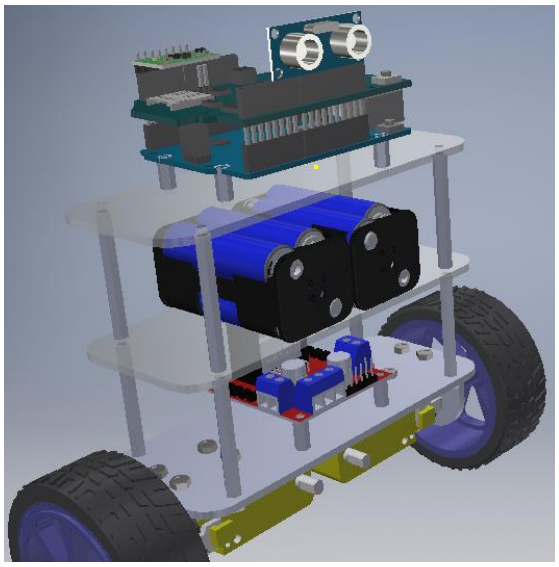


Fig.5. Fully assembled model

MECHANICAL DESIGN

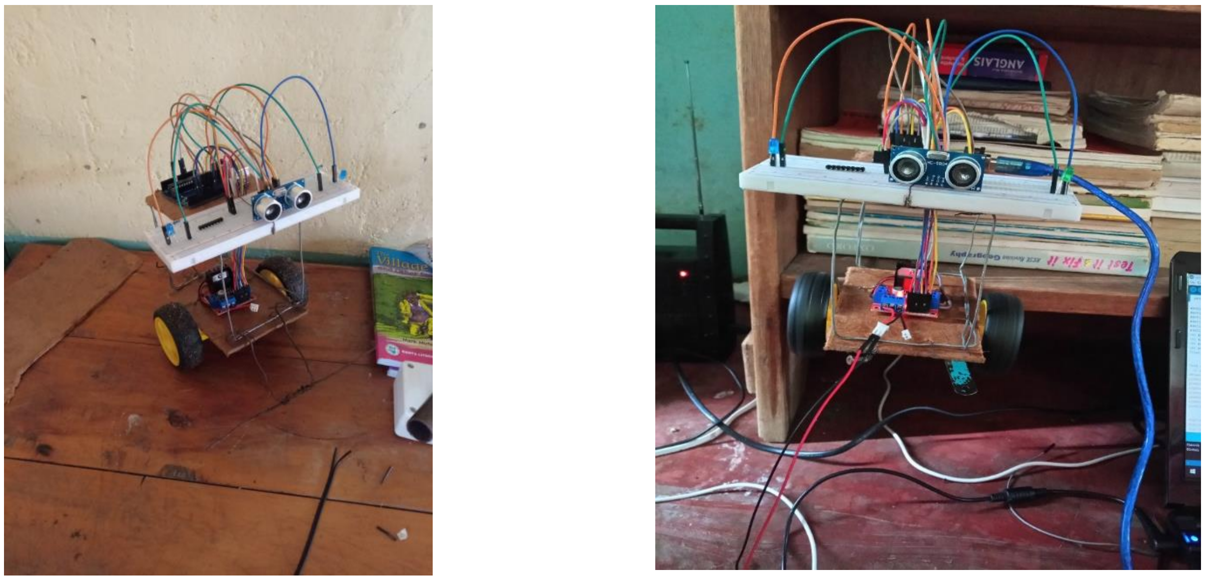


Fig. 6. Designed two wheel self-balancing robot

BALANCING AND OBSTACLE DETECTION PROCESS

For an inverted pendulum to balance, it has to stand 90 degrees upright. However, this system in itself is not balanced which means that the slightest disturbance from the equilibrium position results in a force away from the vertical axis that further destabilizes the system. Therefore, keeping balance requires precise control to correct any errors in tilt the instant they occur. To achieve this, PID (Proportional Integral Derivative) algorithm was employed. In this algorithm, the controller continuously measures a process variable, in this case the tilt of the robot, and calculates an error value (angle from the vertical). The controller then attempts to minimize this error over time by continuously adjusting a control variable (motor torque) according to the equation below:

$$u(t) = Kp * e(t) + Ki \int_{0}^{t} e(t)dt + Kd * \frac{d}{dt} e(t)$$

Where:

u(t) is the control variable
e(t) is the current error in the process variable
Kp, *Ki*, and *Kd* are coefficients that must be tuned to achieve the desired behavior of the controller.

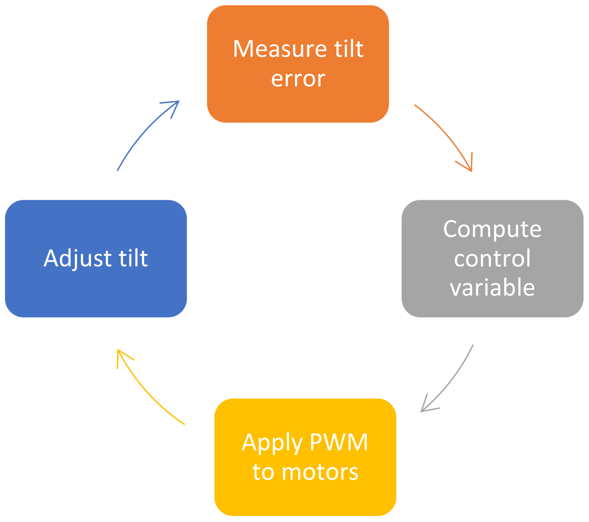


Fig.7. Steps involved in the feedback control loop of the self-balancing robot.

To determine the efficiency of the PID controller, the following model was designed and simulated in MATLAB’s Simulink environment. The sampling time is 5s and PID parameter group is Kp=60, Ki=70, Kd=1.4. Figure 8 shows the step response of the system. It can be seen the response speed of the PID is fast from the simulation curve, and it oscillates a few times, and has small overshoot

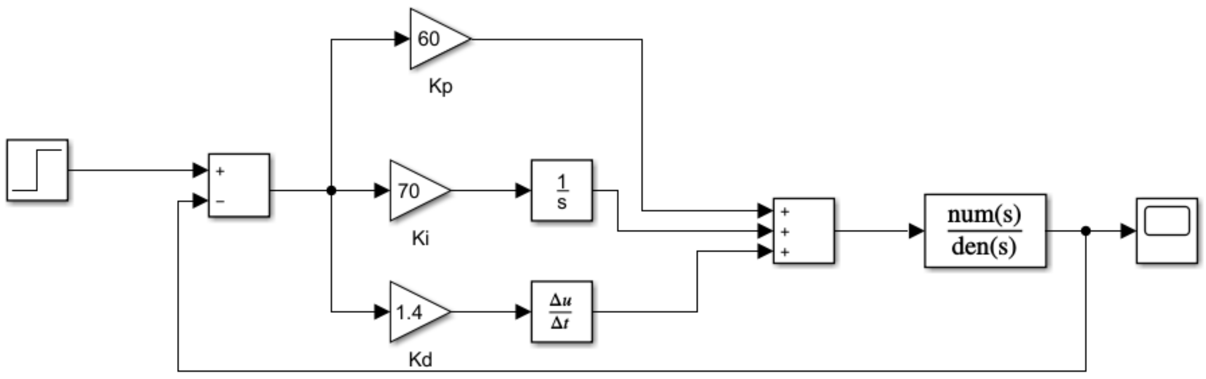


Fig. 8. Simulink diagram of the PID controller.

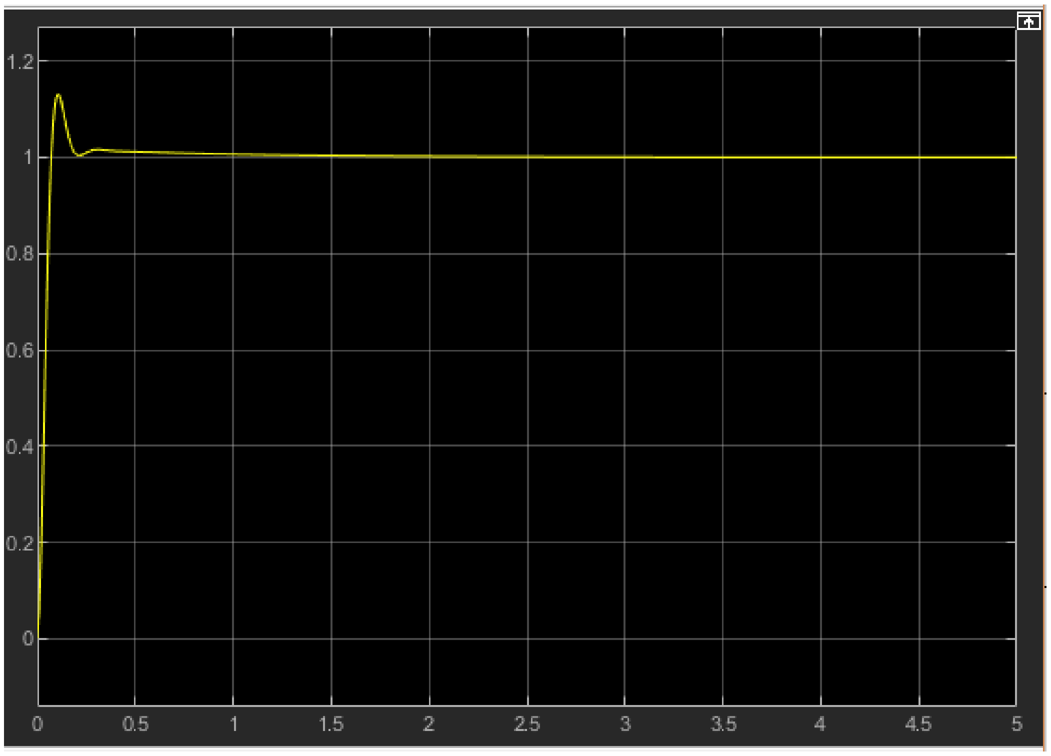
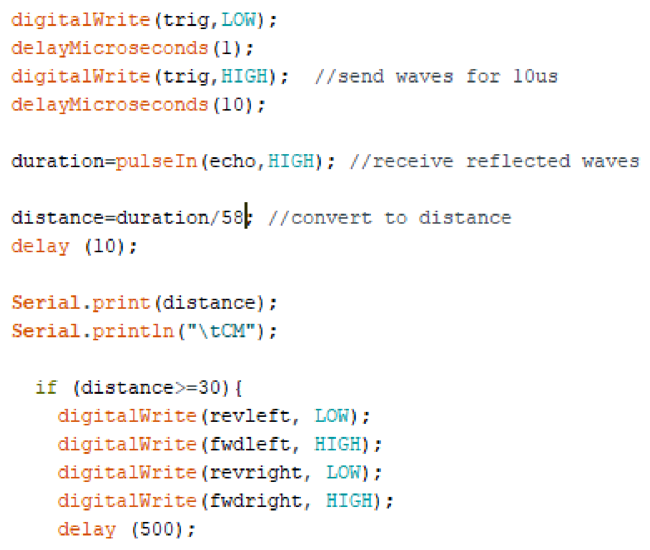


Fig. 9. Simulation results.

To begin measuring the distance, the Arduino microcontroller sends a trigger signal to the ultrasonic sensor. The duty cycle of this trigger signal is 10μS as shown in the code below. When triggered, it generates 8 acoustic wave bursts and initiates a time counter. As soon as the echo signal is received, the timer stops. The output of the ultrasonic sensor is a high pulse with the same duration as the time difference between transmitted ultrasonic bursts and the received echo signal. The microcontroller interprets this time signal into distance as shown below:



```
digitalWrite(trig,LOW);
delayMicroseconds(1);
digitalWrite(trig,HIGH);  //send waves for 10us
delayMicroseconds(10);

duration=pulseIn(echo,HIGH); //receive reflected waves

distance=duration/58; //convert to distance
delay (10);

Serial.print(distance);
Serial.println("\tCM");

  if (distance>=30){
    digitalWrite(revleft, LOW);
    digitalWrite(fwdleft, HIGH);
    digitalWrite(revright, LOW);
    digitalWrite(fwdright, HIGH);
    delay (500);
```

In this project, as can be seen above, the maximum distance the robot should get to an obstacle was set to 30cm, below which the robot would stop, reverse, turn right then move forward if there is no obstacle. If there is one, the stop-reverse-turn right process would repeat itself as the function is contained within an if-else statement.

RESULTS

We came into this project expecting to build a two wheel self-balancing robot with obstacle detection capabilities. We encountered significant challenges, but we achieved our goal. However, it is still not perfect- the robot has minor wobbles which can be fixed.

Additionally, from the simulation results, it is evident that the PID controller - though it works- has the problem of frequent oscillation and overshoot.

FUTURE IMPROVEMENTS

(i) Implement Fuzzy logic for accuracy in control since the PID controller is oscillatory.

(ii) Use cameras for obstacle detection as they have a wide angle of view since with an ultrasonic sensor, if a target object is positioned such that the ultrasonic signal is deflected away rather than reflected back to the ultrasonic sensor, the calculated distance is incorrect. Also, in some cases, the target object is so small that the reflected ultrasonic signal is insufficient for detection, thus requiring use of multiple ultrasonic sensors positioned at different angles which increases the cost of the project. Additionally, some objects like fabric and carpet can absorb acoustic signals or the ultrasonic sensors can detect false signals coming from the airwaves compromising the accuracy of the retuned values.

(iii) Optimization of the mechanical design such as relocating the center of mass, better sensor placement, and modification of the complementary filter to reject disturbances due to translational motion.

(iv) Implement manual remote steering of the robot.

(v) Automatically tune *Kp, Ki,* and *Kd* using MATLAB's PID tuner as manual tuning is laborious and prone to errors.

<u>REFERENCES</u>

[1] Siddhesh Bhagat, Ram Jogal, Jugal Gharat, Vijay Jamariya, "Self Balancing Robot", International Journal for Research in Applied Science & Engineering Technology (IJRASET), 2018.

[2] Umar Adeel, K.S.Alimgeer, Omair Inam, Ayesha Hameed, Mehmood Qureshi, Mehmood Ashraf, "Autonomous Dual Wheel Self Balancing Robot Based on Microcontroller", Journal of Basic and Applied Scientific Research, 2013.

[3] Segway. Segway - the leader in personal, green transportation.Online, 2013. http://www.segway.com.

[4] Jian Fang, "The research on the Application of Fuzzy Immune PD Algorithm in the Two-Wheeled and Self-Balancing Robot System", International Journal of Control and Automation, Vol.7, No.10 (2014), pp.109-118.

[5] Congying Qiu and Yibin Huang, "The Design of Fuzzy Adaptive PID Controller of Two-Wheeled Self-Balancing Robot", International Journal of Information and Electronics Engineering, Vol. 5, No. 3, May 2015.

[6] F. Grasser, A. D'Arrrigo, S. Colombi, and A. C. Rufer, "JOE: A mobile, inverted pendulum," IEEE Transactions on Industrial Electronics, vol. 49, no. 1, pp. 107–14, 2002.

[7] X. Ruan and J. Cai, "Fuzzy backstepping controller for two-wheeled self-balancing robot," in International Asia Conference on Informatics in Control, Automation and Robotics, 2009, pp. 166–9.

[8] X. Ruan, J. Liu, H. Di, and X. Li, "Design and LQ control of a two-wheeled self-balancing robot," in Control Conference, 2008. CCC 2008. 27th Chinese, july 2008, pp. 275 –279.

[9] D. P. Anderson. (2003, Aug.) nBot balancing robot. Online. [Online]. Available: http://www.geology.smu.edu/ dpa-www/robo/nbot.

[10] people.ece.cornell.edu

[11] T. Nomura, Y. Kitsuka, H. Suemistu, and T. Matsuo, "Adaptive backstepping control for a two-wheeled autonomous robot," in ICROSSICE International Joint Conference, Aug. 2009, pp. 4687–92.