Design and Control of a Two-Wheel Self-Balancing Robot using the Arduino Microcontroller Board

Hau-Shiue Juang[1] and Kai-Yew Lum[2]

Abstract— This paper reports the design, construction and control of a two-wheel self-balancing robot. The system architecture comprises a pair of DC motor and an Arduino microcontroller board; a single-axis gyroscope and a 2-axis accelerometer are employed for attitude determination. In addition, a complementary filter is implemented to compensate for gyro drifts. Electrical and kinematic parameters are determined experimentally; PID and LQR-based PI-PD control designs, respectively, are performed on the linearized equations of motion. Experimental results show that self-balancing can be achieved with PI-PD control in the vicinity of the upright position.

I. INTRODUCTION

In the past decade, mobile robots have stepped out of the military and industrial settings, and entered civilian and personal spaces such as hospitals, schools and ordinary homes. While many of these robots for civil applications are mechanically stable, such as Aibo the Sony robotic dog, or four-wheel vacuum cleaners, one that ordinary on-lookers would find awe-inspiring is the Segway personal transport, a mechanically unstable, two-wheel self-balancing vehicle that has seen deployment for law-enforcement, tourism, etc. This vehicle can be rightfully called a robot because, without the sensory capability and intelligent control that accompany every robot, the Segway can never stay upright.

While Segway may have been a well-known commercial product, research into the control of such a mechanical system has been diverse. A two-wheel self-balancing robot is very similar to the inverted pendulum, which is an important testbed in control education and research; see, for example [1], [2]. Besides the development of Segway, studies of two-wheel self-balancing robots have been widely reported. For example, JOE [3] and nBot [4] are both early versions complete with inertia sensors, motor encoders and on-vehicle microcontrollers. See also an updated reference at the nBot website [4]. Since then, there has been active research on the control design for such platforms, including classical and linear multivariable control methods [3], [5], [6], [7], nonlinear backstepping controls [8], [9], fuzzy-neural control [10] and combinations of the above [11]. A related and interesting work that is worth mentioning concerns balancing of a four wheeled vehicle on its two side-wheels, using classical control [12].

One of the key enabler for this research in the academia is arguably the increasing affordability of commercial off-the-shelf (COTS) sensors and microprocessor boards. While JOE featured a digital signal processor board, controller boards based on microprocessor such as the 68HC11, ARM and the ATmega series of the Atmel architecture have become the staple in recent years. Arduino is an open prototyping platform based on ATmega processors and a C language-like software development environment, and can be connected with a variety of COTS sensors [13]. It is fast becoming a popular platform for both education [14] and product development, with applications ranging from robotics [15], [16] to process control [17], [18] and networked control [19].

In this paper, we report a student project on the design, construction and control of a two-wheel self-balancing robot. The robot is driven by two DC motors, and is equipped with an Arduino Mega board which is based on the ATmega2560 processor, a single-axis gyroscope and a 2-axis accelerometer for attitude determination. To compensate for gyro drifts common in COTS sensors, a complementary filter is implemented [20]; for a single-axis problem such as this balancing robot, the complementary filter approach is significantly simpler than the Kalman filter. Two control designs based on the linearized equations of motion is adopted for this project: a proportional-integral-differential (PID) control, and a proportional-integral proportional-differential control based on linear-quadratic regulator (LQR) design. The approach is found to be robust to modeling errors which can be incurred during experimental determination of such electrical and kinematic parameters as moments of inertia and motor gains. Simulation and experimental results are presented, which show that stability of the upright position is achieved with PI-PD control within small tilt angles.

This paper is organized as follows: Section 2 describes the hardware and system architecture of the robot; Section 3 details the designs of the complementary filter, inner control loop to equilibrate the two motors, and balancing control; Section 4 presents the experimental results, followed by some concluding remarks in Section 5.

II. STRUCTURE OF THE TWO-WHEEL BALANCING ROBOT

The structure of a self-balancing robot can be classified into three parts: sensors, motor and motor control, and develop board [4], [5]. Section II-A introduces the application and advantage of the sensors on the proposed balancing robot, and how these sensors are employed to obtain measurements of acceleration, distance traveled, and

[1]H.-S. Juang is a senior undergraduate student at the Department of Electrical Engineering, National Chi-Nan University, 54561 Taiwan.

[2]K.-Y. Lum is an associate professor at the Department of Electrical Engineering, National Chi-Nan University, 54561 Taiwan. `kaiyew.lum@ieee.org`

robot tilt angle. Section II-B describes motor selection and control for the balancing robot. Section II-C discusses the reason behind choosing the Arduino develop board, and how it is deployed.
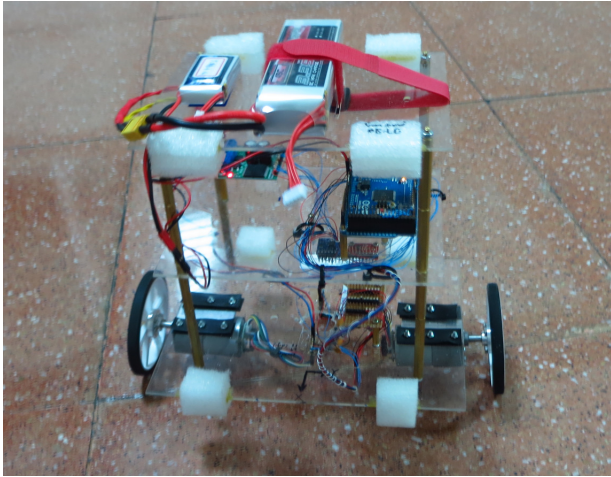

Fig. 1.   Balancing Robot

A. Selection and Application of Sensors

To balance the robot, data of the robot's tilt angle and angular velocity are needed. Using MEMS sensors including a gyroscope, an accelerometer and wheel-angle encoders, we can measure all the data needed for balancing control.

1) Gyroscope: The gyroscope is the sensor which can measure the angular velocity of the balancing robot, and send the data to the development board. The present robot uses the Parallax L3G4200D MEMS Gyroscope, which sends data via serial communication to the develop board, and has the advantage of three-axis angular measurement (although only one axis is used), low-power consumption, and low-cost. Theoretically, integrating the angular velocity will yield the angle θ directly; however, this process will also integrate noise in the gyroscope measurement. As a result, the value of θ will diverge. Section III-A will explain compensation of angular drift using the complementary filter approach.

2) Accelerometer: The accelerometer measures the total external acceleration of the balancing robot, which includes the gravitational and motion accelerations. In this project, we choose the LilyPad ADXL330 which features properties such as three-axis sensing, output-voltage signal conditioning, and low cost. Following the direction cosine method, one can use the X- and Z-axis gravitational acceleration measurements to calculate the tilt angle θ. Although this method gives θ quickly, it can be easily influenced by external forces and noise.

3) Encoders: The encoders return the rotation angles of individual motor shafts as digital signals, which are sent to the processor. After conversion based on gear ratio and wheel radius, the distance traveled can be calculated. The encoder chosen for this project is the Pololu 64 cycle-per-revolution hall effect encoder, which provides a resolution of 64 counts per revolution of the motor shaft.

B. Motor and Motor Control Board

Motor selection for the balancing robot emphasizes torque output instead of velocity, because it has to oppose the rotational moment that gravity applies on the robot. Hence, the motors need to provide enough torque to correct the robot's body pose back to a balanced state. Based on a preliminary calculation of the maximum torque required to correct the robot from a tilt angle of 20 degree, we choose a pair of the Pololu 67:1 Metal Gearmotor 37Dx54L mm with the above 64 CPR Encoder. Each motor provides a torque of 200 oz-in, sufficient for our purpose. The large currents drawn by the motors are supplied by the Pololu Dual VNH2SP30 motor control board, which can delivery a 14-Ampere continuous output current at a maximum operating voltage of 16 Volts.

C. Arduino Mega Development Board

Selection of the development board is based on the following considerations:

1) Performance: The self-balancing robot needs almost real-time response to estimate and correct its tilt angle. Hence, the development board must provide a processing speed that is sufficiently fast to perform the processing tasks, including data acquisition, control computation and signal output, within the sampling time. Based on preliminary calculations, a sampling time smaller than 0.05 seconds is required. The Arduino Mega development board is equipped with the ATmega2560 processor, which features a maximum clock rate of 16 MHz. In our implementation, a sampling time of 0.01 seconds can be achieved with a 30% to 40% buffer.

2) I/O Pins: Another issue is the number of I/O pins available. On the robot, sensors are deployed to obtain measurements of its motion: a gyroscope and an accelerometer are used to estimate the tilt angle, encoders are used to obtained odometric measurements. In addition, a motor control board is interfaced with the development board for delivery of PWM signals. Base on the pin-outs of these sensors and the motor control board, it has been determined that at least thirty I/O pins are needed. The Arduino Mega features 54 general-purpose digital I/O pins of which 15 provide PWM output. Some of the digital I/O pins also support serial communication such as I2C and SPI, as well as interrupt handles. The board also features sixteen 10-bit analog inputs for 0-5V input, giving a quantization limit of 4.9mV.

3) Open Source: To alleviate difficulties in programming, a user-friendly development environment, useful function libraries and references are 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.
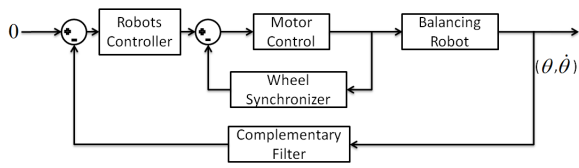
Fig. 2.   Architecture of the controlled system

4) Price and Expansions: Arduino boards are low-cost and expandable, where optional peripherals called shields can be purchased as and when needed. The accessibility of these products in terms of price versus functionality makes them ideal solutions for academic and student projects.

D. Power Supply

For power supply, the motors need a voltage between 12V to 16V, and the development board needs between 5V to 15V. Hence, two sets of batteries are incorporated: for the motors we select a 14.8 volt lithium battery, and for the development board, we choose a four-cell (4×1.5 volts) Ni-MH battery pack.

III. ANGLE ESTIMATION AND BALANCING CONTROL

The system architecture of the self-balancing robot is as shown in Fig. 2, where the forward loop comprises the robot, a balancing controller which delivers a motor-control signal, and a inner-loop controller for wheel synchronization. Feedback is provided through a complementary filter whose function is to provide an estimate of the robot tilt angle from gyroscope and accelerometer measurements. In the following sections, we shall first describe the complementary filter, followed by the wheel-synchronization controller, and finally the balancing controller.

A. Angle Estimation via Complementary Filter

In Section II, it has been seen that a gyroscope is used to measure the angular velocity of the robot, whereas an accelerometer measures the Y- and Z-components of gravitational acceleration, and encoders measure the distance travelled by the wheels. In the balacning control, the tilt angle is corrected to the upright position by using wheel motion. This requires a measurement of the tilt angle θ which must be accurate. The easiest way to obtain the angle is to use the gyroscope; since the gyroscope provides the angular velocity, integrating the latter will produce the angle. However, the gyroscope measurement contains noise. Hence the previous method will not only integrate the angular velocity, but also the noise. This will make the integrated data diverge from the correct angle. Another method is to apply the direction-cosine method to the gravity components measured by the accelerometer to obtain θ immediately; however, this is also affected by noise and the robot's accelerating motion.

To solve the problem of angle measurement, we design a complementary filter [20], a block diagram of which is shown in Fig. 3. The complementary filter uses the measurements of the gyroscope and accelerometer, $\dot{\theta}$ and θ respectively, as input to estimate the bias in the gyroscope-measured angular
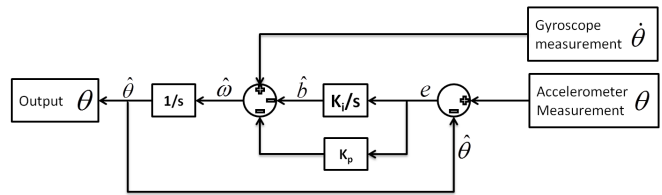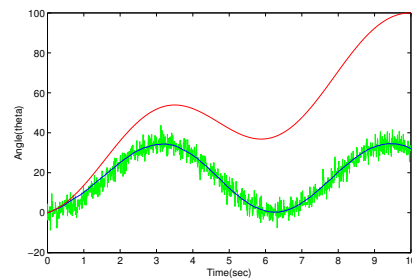

θ = measured angle of accelerometer
$\dot{\theta}$ = measured angular velocity of gyroscope
$\hat{\theta}$ = estimated angle of balancing robot
e = estimation error between $\hat{\theta}$ and θ measurement
$\hat{b}$ = estimated gyro bias
$\hat{\omega}$ = estimated angular velocity
k_p = proportional gain
k_i = integral gain

Fig. 3.   Block Diagram of the Complementary Filter

velocity. Subtracting the estimated bias $\hat{b}$ from $\dot{\theta}$ yields the estimated angular velocity $\hat{\omega}$; integrating $\hat{\omega}$ provides the estimated angle $\hat{\theta}$.

After tuning the filter, its performance under noise is first tested in simulation as shown in Fig. 4(a), which compares the three ways to estimate θ: direct integration of the measured angular velocity results in divergence, whereas the direction-cosine matrix method produce a stable estimate but which is severely affected by noise. The best estimate of the angle θ is provided by the complementary filter. The same result is also observed in experiment, as shown in Fig. 4(b).

(a) Simulation
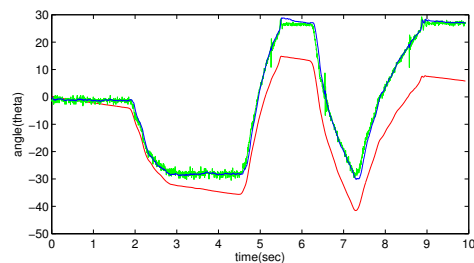
(b) Experiment at a sampling time of 0.01sec

Fig. 4.   Tilt angle estimation. Blue line: output value of complementary filter; red line: direct integration of gyroscope angular rate; green line: angle deduced from accelerometer measurement using the direction cosine method.

B. Inner-loop Control for Wheel-Synchronization

Ideally, if the same motor control signal is sent to the motors control board, both motors should rotate at the same speed. But in the real environment there are many reasons for which the motors rotate at different velocities under the same input signal, such as defects of the motors, terrain and hindrance on the ground. Thus, a method to synchronize both motors is needed. To solve this problem we design a wheel synchronization controller consisting of a simple PI controller with the block diagram given in Fig. 5. This controller adjusts the PWM inputs to the motors so that the difference between the left and right encoders tracks zero. Fig. 6 shows the experimental results of wheel travel distances with and without wheel synchronization, and Fig. 7 shows the motor control signals under wheel synchronization. It is clear that wheel synchronization is effective.
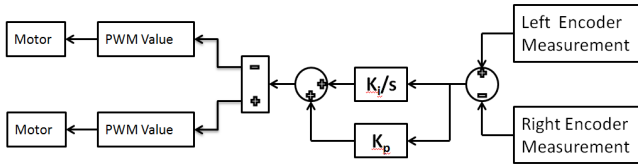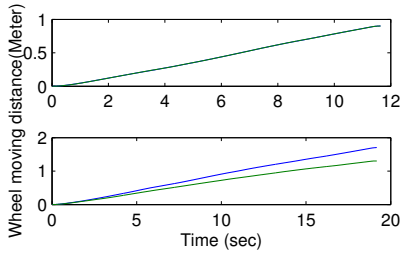

Fig. 5.   Block diagram of wheel synchronizer

Fig. 6.   Test results of of wheel travel distances. Top: The wheels' travel distances coincide when wheel synchronization control is employed. Bottom: At equal motor input, the wheels travel at different speed with without wheel synchronization.

C. Dynamical Model

In order to design a controller for the balancing robot, a dynamical model is first established. Based on nonlinear models of the similar robots found in the literature [4], [3], [5], the following linearized model can be derived:

$$
\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{2k_m k_e(M_p l r - I_p - M_p l^2)}{Rr^2\alpha} & \dfrac{M_p^2 g l^2}{\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{2k_m k_e(r\beta - M_p l)}{Rr^2\alpha} & \dfrac{M_p g l \beta}{\alpha} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}
$$

$$
+ \begin{bmatrix} 0 \\ 2k_m(\dfrac{I_p + M_p l^2 - M_p l r}{Rr\alpha}) \\ 0 \\ 2k_m(M_p l - r\beta) \end{bmatrix} V_a, \qquad (1)
$$

where x and $\dot{x}$ are the horizontal displacement [m] and velocity [m/s], respectively, of the robot's center of gravity
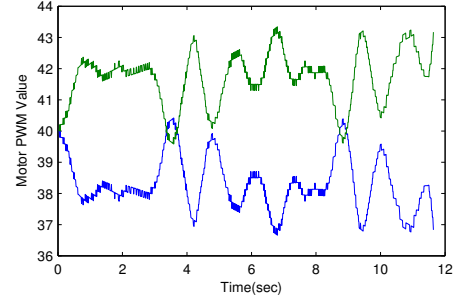

Fig. 7.   Motor control signals with wheel synchronization

(C.G.) ; θ is the tilt angle [rad], $\dot{\theta} = \omega$ is the angular rate [rad/s]; and V_a is the motor input voltage [V]. The model parameters are defined as:

$$
\alpha = I_p \beta + 2M_p l^2 (M_w + \frac{I_w}{r^2}),
$$

$$
\beta = 2M_w + \frac{2I_w}{r^2} + M_p,
$$

$M_p =$	mass of the robot's chassis,	[kg]
$I_p =$	moment of inertia of the robot's chassis,	[kg.m^2]
$l =$	distance between the center of the wheel and the robot's center of gravity,	[m]
$M_w =$	mass of robot's wheel,	[kg]
$k_m =$	motor's torque constant,	[N.m/A]
$k_e =$	back EMF constant,	[V.s/rad]
$R =$	terminal resistance,	[Ω]
$r =$	wheel radius.	[m]

The above physical parameters are determined experimentally, either by mechanical procedures (e.g. the trifilar pendulum method) or electrical measurement. The values obtained are listed below:

$$
\alpha = 0.0180, \qquad \beta = 1.7494,
$$
$$
M_p = 1.51 \quad \text{[kg]}, \qquad I_p = 0.0085 \quad \text{[kg.m}^2\text{]},
$$
$$
l = 0.0927 \quad \text{[m]}, \qquad M_w = 0.08 \quad \text{[kg]},
$$
$$
k_m = 0.2825 \quad \text{[N.m/A]}, \qquad k_e = 0.67859 \quad \text{[V.s/rad]},
$$
$$
R = 7.049 \quad \text{[Ω]}, \qquad r = 0.0422 \quad \text{[m]}.
$$

In practice, C.G. displacement cannot be measured directly. Instead, wheel displacement z and velocity v can be measured with encoders. These measurement output are related to the state variables in (1) by (see Fig. 8):

$$
\begin{bmatrix} z \\ v \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & l & 0 \\ 0 & 1 & 0 & l \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}. \qquad (2)
$$

D. Balancing Control Designs

In this project, two control design strategies are studied: proportional-integral-differential (PID) control, and proportional-integral proportional-differential (PI-PD) control [4] based on linear-quadratic-regulator (LQR) design.
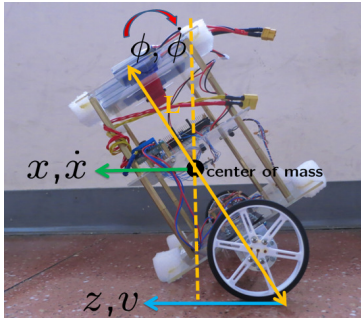
Fig. 8. Definition of motion variables

Fig. 10. Root-locus of the loop gain $L(s) = C(s)G(s)$, with closed-loop poles at a gain of $K = 45$ (diamond). A third pole is further in the left-half plane.

1) PID Control via Root-Locus Design: PID control is a fundamental method which has been extensively studied and implemented in many modern industrial applications. The PID control is chosen because it is easy to learn and implement, for which text-book design methods are available.
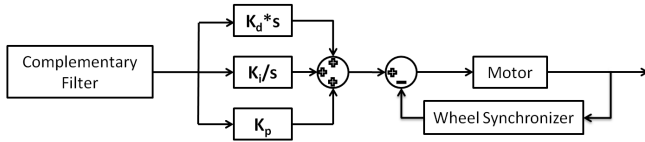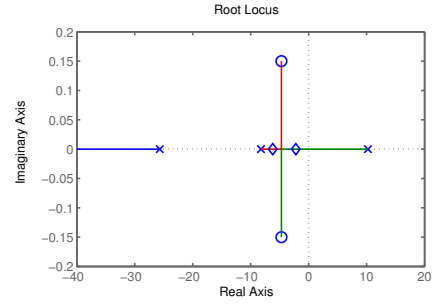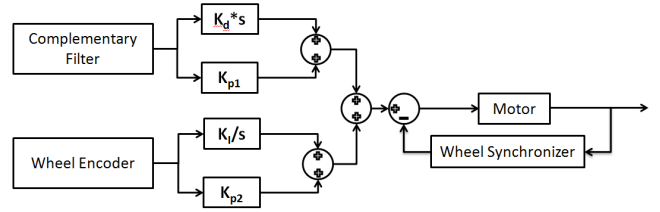

Fig. 9. Block diagram of balancing robot

The block diagram of the PID controller is as shown in Fig. 9, wherein the complementary filter generates estimates of the angular velocity $\hat{\omega}$ and angle $\hat{\theta}$. Instead of differentiating the angle measurement which will amplify noise, differential control is implemented by multiplying $\hat{\omega}$ with the differential gain K_d. Proportional control is given by $\hat{\theta}$ multiplied by the proportional gain K_p, whereas numerical integration of θ and multiplication by the integral gain K_i yields the integral control.

Taking the tilt angle θ as output, the transfer function from V_a to θ can be derived from the model (1) as:

$$\frac{\theta(s)}{V_a(s)} = G(s) = \frac{6.304s}{s^3 + 23.68s^2 - 135.7s - 2162}. \quad (3)$$

The transfer function of a PID controller can be expressed as:

$$K_p + K_d s + \frac{K_i}{s} = K\frac{(s + z_1)(s + z_2)}{s} = KC(s), \quad (4)$$

where $K = K_d$, $z_1 + z_2 = K_p/K_d$ and $z_1 z_2 = K_i/K_d$. Hence, PID design can be performed by adding two zeros and a pole at the origin to the loop gain $L(s) = C(s)G(s)$, and by determining the control gain K via root-locus analysis. In this design, the following zeros are chosen:

$$z_{1,2} = -4.7 \pm j0.15. \quad (5)$$

The resulting root locus of the loop gain is shown in Figure 10. It can been seen that the closed-loop poles enter the open left-half-plane for a large enough gain K. At a damping ratio equal to 1, we obtain the gain $K = 0.614$

Fig. 11. The block diagram of PI-PD controller

2) LQR Design and PI-PD Control Implementation: As an alternative to PID control, we also develop an LQR design for the plant (1), which can be written in the form $\dot{\xi} = A\xi + Bu$, with the state vector $\xi = [x \ \dot{x} \ \theta \ \dot{\theta}]^T$. Given the covariance matrices Q and R, LQR control is given by

$$V_a = -K\xi, \qquad K = R^{-1}B^T P, \quad (6)$$

where P is the solution of the associated Ricatti equation $A^T P + PA - PBR^{-1}B^T P + Q = 0$. Choosing $Q = \text{diag}([0.00001, 0.0001, 10, 0.05])$ and $R = 0.001$, we obtain the following LQR gains:

$$\begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} = \begin{bmatrix} -10 & -33.8 & 1052.9 & 77.6 \end{bmatrix}, \quad (7)$$

with the closed-loop eigenvalues

$$(-446.5, -14.1, -0.06, -0.34).$$

Implementation of the input function V_a is achieved using the output variables in (2); hence,

$$V_a = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & -l & 0 \\ 0 & 1 & 0 & -l \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ v \\ \theta \\ \dot{\theta} \end{bmatrix}$$

$$\triangleq \begin{bmatrix} k_i & k_{p2} & k_{p1} & k_d \end{bmatrix} \begin{bmatrix} z \\ v \\ \theta \\ \dot{\theta} \end{bmatrix}. \quad (8)$$

In practice, the above LQR control can be implemented as a PI-PD controller as shown in Fig. 11, where the actual measurements used are the wheel velocity v measured with the encoder, and the tilt angle θ returned by the complementary

filter. Hence, in Laplace notation,

$$Va(s) = \frac{K_i}{s}v(s) + K_{p2}v(s) + K_{p1}\theta(s) + K_d s\theta(s), \quad (9)$$

where
$$v(s) = \frac{2.126s^2 - 140.7}{s^3 + 23.68s^2 - 135.7s - 2162}Va(s),$$

$$\theta(s) = \frac{6.304s}{s^3 + 23.68s^2 - 135.7s - 2162}Va(s).$$

IV. COMPARISON IN EXPERIMENT

Comparison between the PID and PI-PD controllers is conducted in experiment. From the results shown in Fig. 12, it can be seen that stability with PID control is marginal: beyond 25 seconds angular oscillations exceed the torque limit of the motors and cannot be contained. Moreover, it is found that the robot's position drifts during balancing due to C.G. misalignment. On the other hand, much improved stability is achieved by PI-PD control rather than the PID control. Moreover, PI-PD control is able to compensate for C.G. misalignment and allow the robot to return to its initial position.
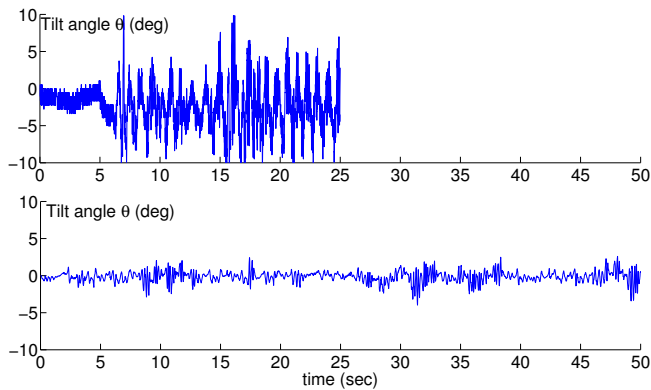

Fig. 12. Experimentally obtained history of tilt angle: PID controller (top), PI-PD controller (bottom).

V. CONCLUDING REMARKS

We have constructed a two-wheel self-balancing robot using low-cost components, and implemented a tilt-angle estimator and a stabilizing PI-PD controller for the balancing motion. The following are some future goals for improvement.

1) Design improvement: Design improvement for better balancing motion will require optimization of the mechanical design such as relocating the center of mass, better sensor placement, modification of the complementary filter to reject disturbances due to translational motion, and a robust control design.

2) Remote control: Low-cost communicate hardware such Xbee may be added to the robot, so that the user can control its motion remotely and read the data via telemetry. Also, user-controlled motion such as forward, backward, turn right and left, may be implemented.

3) Obstacle avoidance and perimeter following: A possible extension is to combine the Arduino system and other sensors such ultrasonic and IR senses, GPS, digital compass and Camera to address other advanced applications, e.g. obstacle avoidance and perimeter following.

REFERENCES

[1] R. Fierro, F. Lewis, and A. Lowe, "Hybrid control for a class of underactuated mechanical systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 29, no. 6, pp. 649–4, nov 1999.

[2] K. Xu and X.-D. Duan, "Comparative study of control methods of single-rotational inverted pendulum," in *Proceedings of the First International Conference on Machine Learning and Cybernetics*, vol. 2, 2002, pp. 776–8.

[3] 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.

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

[5] R. C. Ooi, "Balancing a two-wheeled autonomous robot," Final Year Thesis, The University of Western Australia, School of Mechanical Engineering, 2003.

[6] N. M. A. Ghani, F. Naim, and P. Y. Tan, "Two wheels balancing robot with line following capability," *World Academy of Science, Engineering and Technology*, vol. 55, pp. 634–8, 2011.

[7] 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.

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

[9] G. M. T. Nguyen, H. N. Duong, and H. P. Ngyuen, "A PID backstepping controller for two-wheeled self-balancing robot," in *Proceedings fo the 2010 IFOST*, 2010.

[10] K.-H. Su and Y.-Y. Chen, "Balance control for two-wheeled robot via neural-fuzzy technique," in *SICE Annual Conference 2010, Proceedings of*, aug. 2010, pp. 2838 –2842.

[11] 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.

[12] D. Arndt, J. E. Bobrow, S. Peters, K. Iagnemma, and S. Bubowsky, "Two-wheel self-balancing of a four-wheeled vehicle," *IEEE Control Systems Magazine*, vol. 31, no. 2, pp. 29–37, April 2011.

[13] Arduino. [Online]. Available: http://arduino.cc/

[14] J. Sarik and I. Kymissis, "Lab kits using the Arduino prototyping platform," in *IEEE Frontiers in Education Conference*, 2010, pp. T3C–1–5.

[15] G. Guo and W. Yue, "Autonomous platoon control allowing range-limited sensors," *IEEE Trans. on Vehicular Technology*, vol. 61, no. 7, pp. 2901–2912, 2012.

[16] P. A. Vignesh and G. Vignesh, "Relocating vehicles to avoid traffic collision through wireless sensor networks," in *4th International Conference on Computational Intelligence, Communication Systems and Networks*. IEEE, 2012.

[17] M. J. A. Arizaga, J. de la Calleja, R. Hernandez, and A. Benitez, "Automatic control for laboratory sterilization process rsd on Arduino hardware," in *22nd International Conference on Electrical Communications and Computers*, 130-3, Ed., 2012.

[18] S. Krivic, M. Hujdur, A. Mrzic, and S. Konjicija, "Design and implementation of fuzzy controller on embedded computer for water level control," in *MIPRO*, Opatija, Croatia, May 21–25 2012, pp. 1747–51.

[19] V. Georgitzikis and I. Akribopoulos, O.and Chatzigiannakis, "Controlling physical objects via the internet using the Arduino platform over 802.15.4 networks," *IEEE Latin America Transactions*, vol. 10, no. 3, pp. 1686–9, 2012.

[20] A.-J. Baerveldt and R. Klang, "A low-cost and low-weight attitude estimation system for an autonomous helicopter," in *IEEE International Conference on Intelligent Engineering Systems*, sep 1997, pp. 391–5.