Attitude Control and Stabilization of a Two-Wheeled Self-Balancing Robot

Omer Saleem Bhatti*, Khalid Mehmood-ul-Hasan, Muhammad Anas Imtiaz***

** Electrical Engineering Department, FAST - NU, Lahore, Pakistan,(e-mail: omersaleembhatti@gmail.com, anas_imtiaz@yahoo.com)*
*** Electrical Engineering Department, UET, Lahore, Pakistan, (e-mail: kmhasan@uet.edu.pk)*

Abstract: The paper demonstrates development of attitude control and stabilization technique of a self-balancing robot. The main aim is to ensure its vertical stability, even in the presence of an external bounded impulsive force. By electronically programming a hard-coded vertical reference position, the proposed robotic system can be balanced at the desired set-point angle. The orientation and the extent of inclination of the robot body in either direction are measured with inertial-sensor based feedback. The proposed system uses a combination of first-order spatial filters in order to remove the noise and to merge the analog sensor readings. Comparative performance analysis is also done between a simple PID controller and an auto-tuned PID controller for the optimization of attitude control and stabilization of the self-balancing platform.

Keywords: Attitude control, auto-tuned PID, inertial sensors, filters, PID, self-balancing robot.

1. INTRODUCTION

Two-wheeled balancing robots have immense significance in the area of robotics and control systems engineering. They offer to develop an intricate control system that is capable of maintaining stability of an otherwise unstable system. This balancing robotic system imitates the behavior of an inverted pendulum and in effect works on the same principle as the Pole and Cart theory. Hence, these principles are taken into account while designing a robot that is capable of balancing upright on its two wheels that are aligned on the same axle. The two wheels are situated below the base and allow the robot chassis to maintain an upright position by moving in the direction of tilt, either forward or backward, in an attempt to keep the centre of the mass above the wheel axles. These robots are highly non-linear and under-actuated. Since they are able to balance themselves on only two co-axial motorized wheels, it is very easy for them to maneuver on various terrains. Without active control, these systems become unstable and collapse. Apart from balancing the posture in a stable upright fashion, they are also able to regain their posture and stand erect, even when a bounded external force is applied to them. This force acts as a disturbance to the system. These robots sense their inclination (rotational pitch angle) continuously, compare it with the set-point reference provided by the user and correct their orientation by keeping it at the desired pitch angle. The system also keeps track of the maximum recovery pitch angle (the threshold angular displacement of the robot from the vertical before it collapses). Inverted pendulum being an inherently unstable system tends to fall in either direction. A conceptual view of the proposed robotic system is shown in Fig. 1. The balancing torque is given by (1).

$$T = Mg\sin(\theta) \quad (1)$$

where,

M = moment arm (perpendicular distance between center of mass and distance from pivot)
g = acceleration due to gravity
θ = inclination (angle with the vertical)

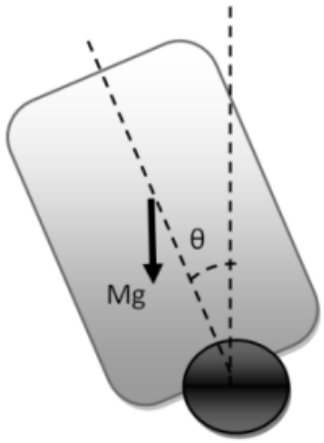

Fig. 1. Conceptual view of the robot.

When θ = 0 degree, the robot is in balanced position and no balancing torque in needed. With θ> 0 or θ< 0, the balancing torque moves the robot in the direction against falling torque. In this way, the robot tries to retain its balanced position. Development of a flexible self-balancing robotic platform comprises of several essential units. These units include a reliable systems model, sensors, signal processors, a stable control scheme and actuators. These units have been properly discussed by (Nawawi et al., 2007). In the past, the researchers have extensively used MATLAB toolbox to efficiently model and control a self-balancing robot (Araghi, et al., 2011).Interactive software tools and virtual prototyping techniques, such as ADAMS, can be used to build and

simulate a stable mechanical system model (Qian Hao et al., 2007). Several non-linear control schemes have been proposed and their performance has been verified by rigorous experimentation. Neural network controllers have been used in mobile inverted pendulum experiments to control the pendulum angle and the position of the cart (S. Jung et al., 2007). Simulation results with PID backstepping control algorithms have proven that with three control loops, this algorithm can offer a quicker response to balance the two-wheeled platform (Nguyen Gia Minh Thao et al., 2010). State-feedback controllers and Linear Quadratic Regulators (LQR) have also been experimentally validated to provide robustness for the balance control of a self-balancing robot (Solis and Takanishi, 2010; Junfeng and Wanyang, 2011).

2. EXPERIMENTAL SETUP

The voltage signal is the input and the rotational pitch angle serves as the output. The high level block diagram of the robot is shown in the Fig.2.

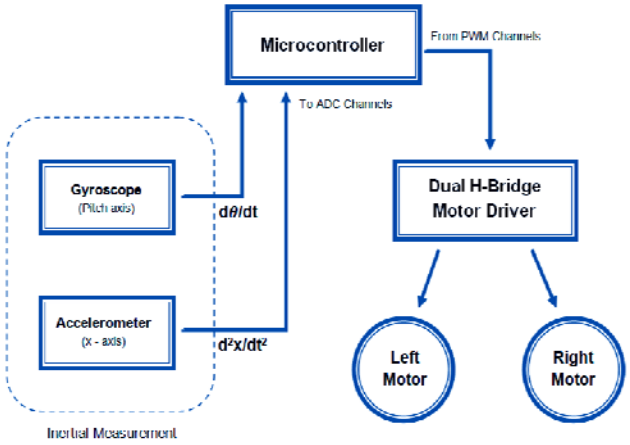

Fig. 2. High Level Block Diagram.

The inertial sensors (gyroscope and accelerometer) are used to provide analog signals regarding the attitude and orientation of the robot to the microcontroller, PIC18F452 (Microchip, 2006). The microcontroller processes them, compares them with the hard-coded equilibrium set-point, and then issues appropriate motor commands to actuate the DC geared motors via the power electronic motor driver circuit.

2.1 Sensors & Measurements

The information regarding the orientation and the attitude of the robot is measured with the aid of analog gyroscope, LPR550AL (ST Micro Electronics n.d.), and accelerometer, ADXL335 (Analog Devices (a), n.d.), sensors. The gyroscope tells us about the rate of change of angle ($d\theta/dt$) of the robot body in the forward/backward direction. The accelerometer tells us about the acceleration along the desired axis (d^2x/dt^2). The purpose of using both of these sensors, instead of one, is due to the fact that the accelerometer readings have noise while the gyroscope reading has an inherent drift. Hence, in order to overcome the individual short-comings of the two sensors, they are fused appropriately. This feedback provides reliable information regarding the robot's orientation. Using the basic trigonometric relations, this acceleration is used to compute angular displacement (θ) of the robot body along with the direction of tilt, as shown in Fig.3. If one axis (x-axis) is used to calculate the tilted angle of the accelerometer, the trigonometry relationship of (2) is used (Analog Devices (a), n.d.).

$$\theta = \sin^{-1}\left(\frac{V_{out}-V_{offset}}{S}\right) \quad (2)$$

where,

V_{out}= Accelerometer Output (Volt)
V_{offset}= Accelerometer Offset = 1650 mV
S = Accelerometer Sensitivity = 800 mV/g
θ = Angle of Tilt (radians)

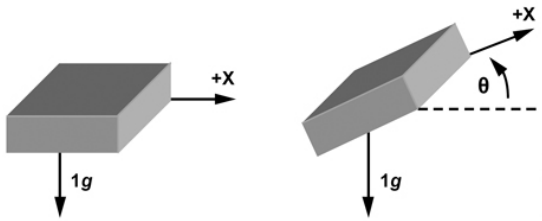

Fig. 3. Single Axis used for Tilt Sensing (Analog Devices (b), n.d.).

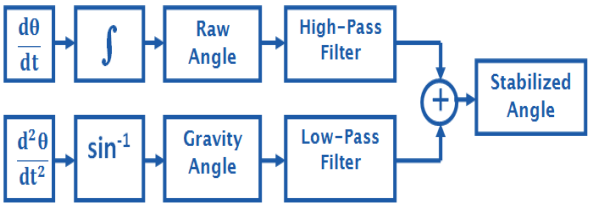

Fig. 4. Complementary Filter for Sensor Integration.

Once these values are read, they are fed directly to the microcontroller (PIC18F452) which initially converts them to equivalent digital values via the internal 10 bit ADC of PIC. The readings are stored in a 16 bit integer variable, namely "gyro_reading" and "accel_reading". The samples are taken and updated at a regular interval of 10 msec. This helps to ensure a reliable performance of robot.

2.2 Signal Conditioning

The accelerometer readings have noise while the gyroscope readings have an inherent drift. Therefore, before further processing, the digital signals corresponding to θ and $d\theta/dt$ are fed to the first order Median-Filter and Mean-Filter, to remove the random additive noise from the individual sensor readings. The simplest procedure was to take 100 samples, remove the upper and lower 15 samples and then take the average of the remaining 70 samples. The pseudo-code is as follows.

```
1. START;
2. TAKE 100 SAMPLES;
3. SORT VALUES IN ASCENDING ORDER;
4. REMOVE FIRST 15 VALUES;
5. REMOVE LAST 15 VALUES;
6. SUM REMAINIG 70 VALUES;
7. DIVIDE BY 70;
8. END.
```

These filtered values are then combined together via a first order digital complimentary filter, represented by (3).

$$f = (a)(y) + (1 - a)(x) \quad (3)$$

where,

a = τ / (τ + dt),
y = gyroscope reading,
x = accelerometer reading,
f = filtered-output

A properly implemented filter would combine the raw angle with the gravity angle, as shown in Fig.4. It fixes the inherent problems of sensor noise, drift and horizontal acceleration dependency. It helps rejecting all the short-term fluctuations. The time interval between successive program loops is known as the *Sample Period, dt*. The *time constant*, τ, is the time interval on which it operates on a given signal. So the signals that have time period smaller than this time constant are filtered out, while the longer signals stay unaltered. To reduce the gyroscopic drift, a lower time constant should be implemented. But on the other hand, this leads to a lot of horizontal acceleration noise. Hence a compromise is made by experimentally tweaking its value and adopting the best one for the application.

2.3 Closed Loop Control

This digital value is used henceforth for the purpose of comparison and correction of robots orientation and attitude to put it in its stable upright posture. The filtered output of the sensors, when the robot body is exactly in stable upright position, is taken as the equilibrium reference or equilibrium set-point by the control scheme. Once the robot is set into action, it continuously checks and compares its current state with the equilibrium set-point. The difference of these two entities generates the error signal, *e(t)*. The sign of this error signal denotes whether the robot is leaning forward (if $e(t) > 0$) or backward (if $e(t) < 0$). The magnitude of the *e(t)* specifies the extent to which it has fallen. These error signals, once computed, are stored. The current error is fed to the *P* controller after being multiplied with K_P. The *I* controller takes the sum of recent errors. Hence the ten recent errors are added over the time interval (between successive error readings) and sent to the *I* controller, where they are multiplied with K_I. The rate of change/difference between two recent errors is subjected to the *D* controller where they are multiplied with K_D. Eventually all these three terms are added and the output *u(t)* of PID control scheme is obtained, as illustrated in Fig.5.

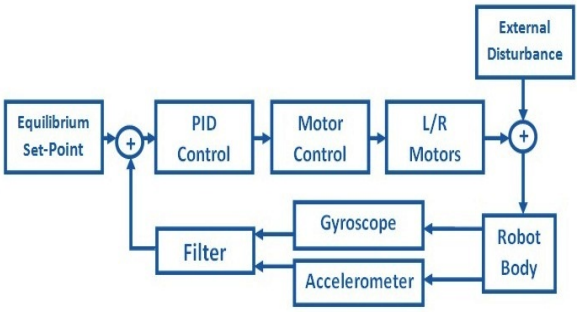

Fig. 5. Closed Loop Control Architecture.

The mathematical relationship of PID is shown in (4).

$$u(t) = K_P e(t) + K_I \int_0^t e(t)\, dt + K_D \frac{d}{dt} e(t) \quad (4)$$

where,

K_P = Proportional Gain
K_I = Integral Gain
K_D = Derivative Gain

The output of the PID controller is checked continuously, so that it may stay between a minimum and maximum value. This helps in avoiding the wind-up state. The tuning of K_P, K_D, K_I constants is usually done with the aid of simulations and rigorous experimentation. Now, the variation of the PID control output magnitude between these bounds helps in deciding the variation in the duty cycle of the Pulse-Width-Modulated (PWM) signal. The resultant PWM signal helps controlling the speed and direction of rotation of the DC geared motors, via an H-Bridge motor driver circuit. This way, if the robot body tilts in a given direction, the motors respond immediately by moving in the direction of inclination at an appropriate speed, in order to bring the wheels exactly below the centre of mass of the robot body. The 'Derivative' term amplifies higher frequency noise that is generated by the sensors. Thus, the higher values of the derivatives lead to large changes in the output of the PID controller. A practical solution that has been adopted to remove these high frequency components of noise is by putting a first order low pass filter programmatically on the derivative term. Consequently, the poles of the derivative term are tuned such that the noise does not affect the output.

2.4 Motor Control

The motor control is probably the simplest of all the tasks. For driving the motor, L298 based dual H-Bridge motor driver circuit is used as shown in Fig.6. It can fully control and drive two motors simultaneously. Also the motors require a unique PWM signal. This signal is fed to the motor driver circuit in order to control the speed of the motor rotation. The pulse length can be varied to change the speed of the motor. Generally, the PWM frequency is about 1000 Hertz, with a period cycle of 1.0 msec.

2.5 Power Source

To provide DC power to all the electronic devices explained earlier, a DC battery has been utilized.

Fig. 6. H-Bridge Motor Control Circuit.

A Sealed-Lead-Acid (SLA) battery has been used. The specifications of the battery are 12V and 1300mAh.Each of the two motors require at most 500mA of current while

operating, whereas the sensors and the digital circuit requires approximately 100mA of current. Since the total current required by the system is roughly 1100mA, thus the battery utilized can provide a standby time of 70 minutes, before discharging completely.

2.6 Robot Structure

The material used in the design is able to offer durability, strength, maintainability, energy efficiency and operating capability of the robot. These parameters are also responsible to contribute in the final size and hence the weight of the robot. The weight is an important factor in designing the wheel and base structure of the robot. If the weight applied at the base is very high, the wheels and the hubs holding them would bend outward, making it quite difficult to maintain the robot in upright posture. The total weight of the robot is 1.513 Kg. The robot contains two horizontal plates. One is very near to the floor level and other is right above it. The two plates are separated by spacers. The battery of the robot is installed beneath the lower plate, geometrically placed in between the co-axial wheels. The power electronic circuitry is placed right above this lower plate. The upper plate contains the microcontroller circuitry on it. The computer aided design of the proposed robot structure is shown in Fig.7.

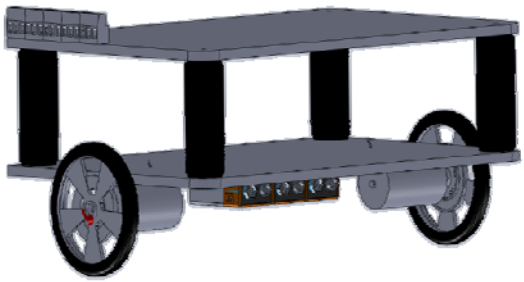

Fig. 7. Computer aided design of the robot.

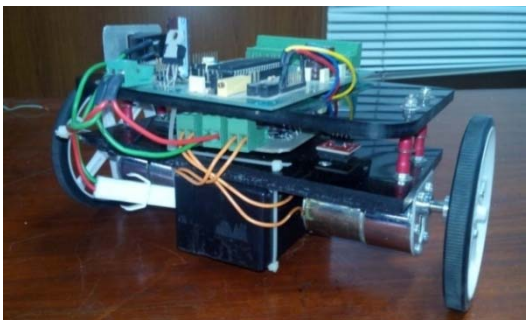

Fig. 8. Fabricated Structure of Robot.

Plexi-Glass of 0.75mm thickness is used to build the chassis of the robot because it is strong, durable and light in weight. A larger moment of inertia enhances the static stability. After experimentation, a mechanically balanced structure is fabricated that is easier to control via the embedded system. It is shown in Fig.8. The final dimensions of the robot are 18.5cm × 10.5cm × 10.0cm.

The flow of sub-routines is illustrated in Fig. 9.

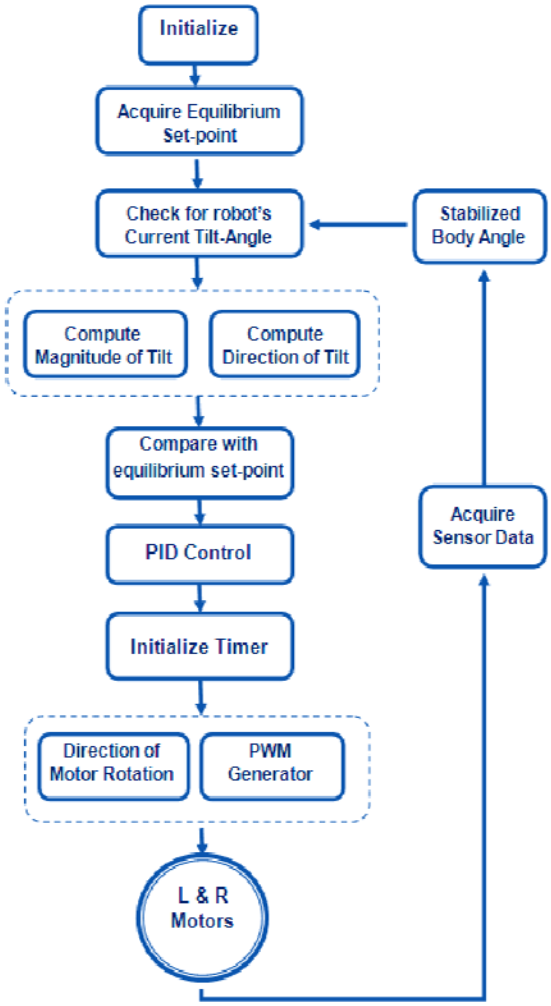

Fig. 9. Flow-chart of sub-routines.

3. CONTROL SYSTEM DESIGN

For a better understanding of the firmware that is responsible for the control of robots position, it is divided in a number of sub-routines.

3.1 PID Control

Before implementing the PID routine in software, it is mandatory to learn a couple of basics. A PID controller has basically three main components: Proportional controller, Integral controller and Derivative controller. Each of these terms is multiplied with a coefficient, namely K_P, K_D and K_I. The variation in controller gain 'K_P' has a direct impact at the robot's behavior. A higher value of K_P usually ensures a faster controller response. However, a fairly large value of K_P leads to undesirable oscillations along with system overshoot. The derivative term helps speeding up the *P* controller's response to a change of input. Consequently, *D* controller causes the robot to reach the equilibrium state faster. Finally, the *I* controller serves to reduce the steady-state error of the *P* controller. The steady-state error is made negligible, but the equilibrium state is reached somewhat slowly. The flow chart of the software routine of PID control is shown in Fig. 10.

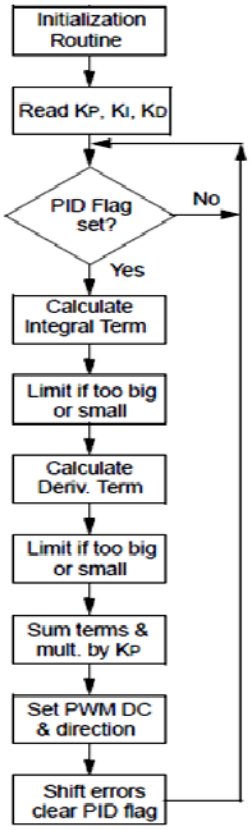

Fig. 10. Flow-chart of PID control routine (Microchip, n.d.).

Initially, the PID controller was implemented using the manual tuning, also known as the simple hand-tuning, technique. Adjusting the parameters of K_P, K_D and K_I is an important task. Hence the K_P, K_D and K_I constants are set to zero in the program. The robot/system is powered up using a 12V power supply. If the base does not move and the robot topples over freely, it validates that all constants are read as zeros. The K_P constant is gradually increased in small steps, until there is a little oscillation in the base. As a rule of thumb, K_I must not exceed 10% of the K_P. The K_I is then increased until the robot platform balances itself for a long duration, while still oscillating about its mean position. When the K_I has been optimized, the robot body will not only be balanced, but also its oscillations about the equilibrium position will be small and smooth. Eventually, the K_D is increased in the similar fashion as the other two constants until the platform becomes stable. Once these K_P, K_D and K_I co-efficient are found experimentally, they are hard-coded in the PID software routine. The routine outputs the direction of the motor and it also calculates the duty cycle of the PWM signal that is to be provided to the motors (Thomas Bräunl, 2008).

3.2 Automatically Tuned PID Control

Tuning the coefficients K_P, K_I and K_D manually is always a nuisance. Therefore the industrial applications employ the auto-tuning feature when using the PID controller. This feature aids in adjusting the three parameters automatically. In the proposed robot, the relay method is employed to auto-tune the parameters. Theoretically, it is quite similar to Zeiger-Nicholas Frequency Domain (ZNFD) method. As shown in Fig.11, the PID controller is being replaced by the relay right before the plant. While tuning the parameters via ZNFD method, the K_I and K_D are made zero, whereas K_P is manually adjusted to a point such that the closed-loop system starts oscillating in a periodic manner, neither decaying nor growing in magnitude. This value of K_P is recorded as K_U. The time period of these oscillations is measured and recorded as T_U. Then using the mathematical relation in Table 1, the three parameters are calculated. Unlike the ZNFD method, the relay method attempts to find the K_U and T_U on its own.

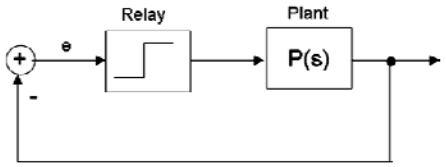

Fig. 11. Relay for auto-tuning of PID parameters (Dew, 2014).

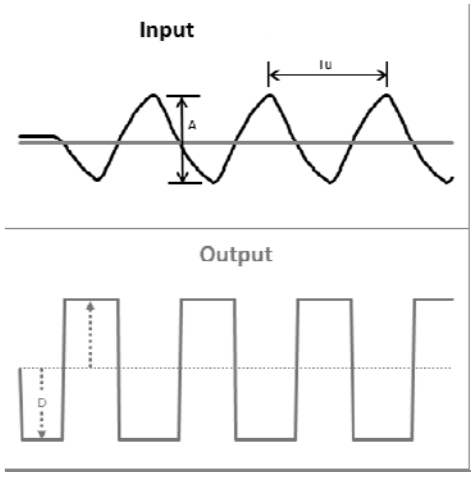

Fig. 12. Input and Output of Relay (Arduino PID Autotune Library, 2012).

Once found, this auto-tuner then performs back-calculations to tune the parameters. Referring to the Fig. 12, as we start from the steady state, the input to the relay is the error signal, e(t). The relay outputs a step response. The amplitude of this step response is 'D' and is fixed for a given relay. After each subsequent zero-crossing of the input, the output step changes its direction as shown in the Fig. 12. T_U is the time period of input signal, whereas the distance between the maxima and minima of the input signal is denoted by 'A' as shown in the Fig. 12. The value of A is variable and like T_U, it has to be found by the system. Finding the value of A and T_U is quite simple. When the oscillatory input signal is read by the microcontroller, its peaks are identified. In each sampling time frame window, the maximum value is found by reading and comparing the new value of the input signal with the previous largest value, and keeping the larger one as MAX (Arduino PID Autotune Library, 2012). Similarly the same new value is also compared with the previous smallest value and the smaller one amongst them is stored as MIN. The difference between the MAX and MIN is equal to the value of A, as shown in (5).

$$A = MAX - MIN \quad (5)$$

Table 1. PID parameters for the Auto-tuner (Arduino PID Auto-tune Library, 2012)

Control	K_P	K_I	K_D
P	$0.5K_U$	-	-
PI	$0.4K_U$	$0.48K_U/T_U$	-
PID	$0.6K_U$	$1.2K_U/T_U$	$0.075K_U/T_U$

The period T_U is found by detecting two zero-crossings and computing the time between them. For this purpose, the number of samples between the two zero-crossings is found and using (6), T_U is calculated.

$$T_U = n \times T_S \quad (6)$$

where,

T_S = sampling time
n = number of samples

Once these K_U and T_U are found, we next compute the K_U via (7). The PID coefficients are found using the mathematical relations given in Table 1.

$$K_U = \frac{4 \times D}{\pi \times A} \quad (7)$$

where,

D = Amplitude of relay output step response
A = peak-to-peak signal value of the input

4. RESULTS

The microcontroller communicates the robot body's tilt (pitch) angle with LABVIEW over serial link. The setup used for testing and recording the simulations results is shown in Fig. 13. The sampling time used in the application is 10 msec.

Two tests are commenced on the robot to control its attitude and stabilize it in the upright position. The first test is done by using a manually-tuned PID controller. The K_P, K_D and K_I co-efficient are found experimentally to be equal to12.05, 1.075and 0.355respectively. They are hard-coded in the PID software routine. The step-response of the system is shown in Fig. 14. The graph is plotted with robot body's tilt angle (degrees) and the time (seconds) along the x-axis.

The second test is done by using the automatically tuned PID controller. The K_P, K_D and K_I co-efficient are found using the relay method. These coefficients are adjusted automatically. The step-response of the system is shown in Fig. 15. The graph is plotted with robot body's tilt angle (degrees) and the time (seconds) along the x-axis. It can be clearly seen from the response(s) in Fig. 14 and Fig. 15, that the system energizes the motors of the robot to oscillate it back and forth. Once the robot has gained sufficient energy, the balance control scheme(s) implemented via the PID controller and its auto-tuning variant, tend to keep it erect.

The rise time is experimentally found by looking at the time required by the response to reach from 10% to 90% of its step height in Fig. 14 and Fig. 15. Similarly, the settling time is found by observing the time taken by the response to fall within ±2% of the steady state value. Finally the percentage overshoot is calculated by using (8). The rise time (T_R), settling time (T_S), percentage overshoot (%OS) and steady-state error (e_{SS}) of the two responses are summarized in Table 2.

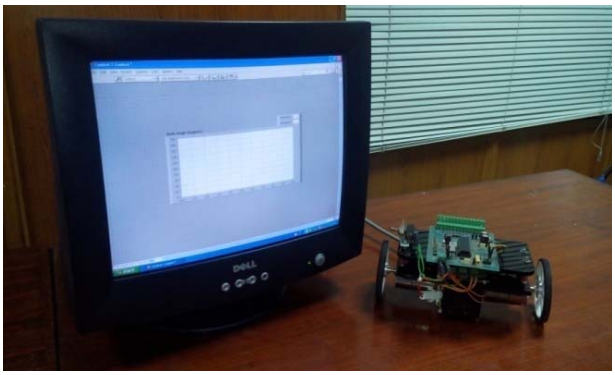

Fig. 13. Experimental Setup.

$$\%OS = \frac{\theta_{Max} - \theta_{SS}}{\theta_{SS}} \times 100 \quad (8)$$

where,

θ_{Max} = Highest peak value of the response in the graph
θ_{SS} = Steady-state value of response in the graph

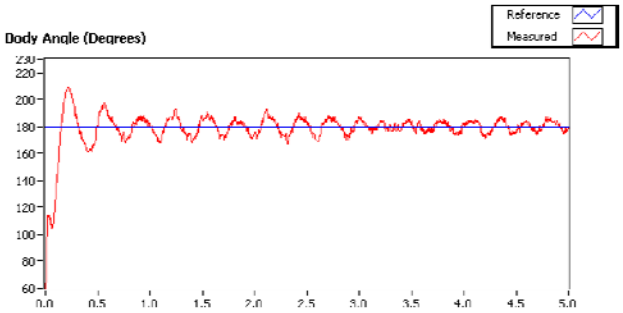

Fig. 14. Response with simple PID controller.

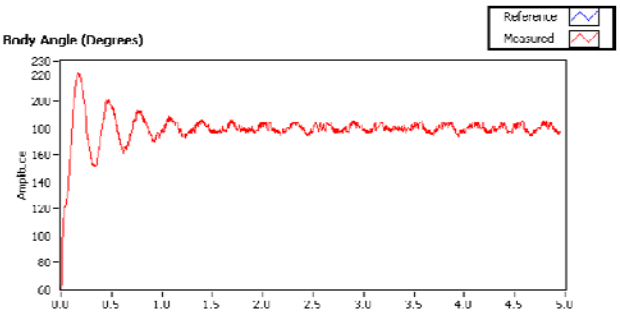

Fig. 15. Response with auto-tuned PID controller.

Table 2. Summary of robot's body angle response

Control Type	T_R (sec)	T_S(sec)	%OS	e_{SS}
PID	0.30	3.10	13.89%	$\pm 6^o$
Auto-PID	0.25	1.45	25.10%	$\pm 1^o$

5. CONCLUSION

The auto-tuning feature saves the trouble of going through the problematic and redundant task of manually tuning all the PID coefficients, but a comparative analysis of the responses in Fig. 14 and Fig. 15, reveals that the overshoot in the

system with auto-tuned PID is quite large. However, unlike the manually-tuned PID controllers, the auto-tuned system shows a smaller steady state error and smaller rise time and settling time. The steady state error is reduced by 5^o, whereas the rise time (T_R) and settling time (T_S) are reduced by 0.05 sec and 1.65 sec respectively.

It is shown in this paper, that application of auto-tuned PID controllers tend to balance the two-wheeled self-balancing robots in a much more effective manner that the manually-tuned PID controllers. Scientifically speaking; with the application of the auto-tuned algorithm, the steady state error, rise time and the settling time of the dynamic system has improved. The comparison and hence the improvement in the transient as well as the steady state analysis of the system with manually tuned and the auto-tuned controller, also validates the proposed technique. This also manifests that the auto-tuned PID controllers can adapt to changes in the physical properties of the robot. That is to say, if the robot's battery drains over time, the robot will update its PID coefficients and would try to cope with the effect of battery loss to some extent. However in the case of ordinary PID controller, since the coefficients are hard-coded, the robot would get unstable and collapse under such circumstances. Similarly if the mass of the robot is changed during operation, the PID coefficients of system would adjust themselves automatically in order to continue stabilizing. The relay method for auto-tuning provides us with the ease of implementation and great flexibility in usage. It gives us fairly good values of K_P, K_D and K_I to balance and control the attitude of our robotic system. Thus, we are able to balance a two-wheeled platform in a very effective and an innovative way. However, in exchange, some performance sacrifices are made such as excessive overshoot.

There is still a lot of room for further research and enhancements that can improve the performance of this platform. Instead of using the manually-tuned PID and the auto-tuned PID controllers, adaptive fuzzy PID controller can be used to tune the K_P, K_D and K_I co-efficient in real time. Although the adaptive fuzzy PID control technique would be computationally expensive and slow. But being more sensitive to the changes it would yield a much better dynamic performance for the stability and balance control of the two-wheeled self-balancing robot.

REFERENCES

Analog Devices (a), 'AN-1057 Application Note,' [Online]. Available: www.analog.com/static/imported-files/application_notes/AN-1057.pdf. [Accessed May 2014].

Analog Devices (b), 'Small, Low Power, 3-Axis ±3 g Accelerometer ADXL335,' [Online]. Available: www.analog.com/static/imported-files/data_sheets/ADXL335.pdf. [Accessed May 2014].

Araghi, M.H., Kermani, M.R. (2011), 'Computer-Aided System Design for Educational Purposes: An Autonomous Self-Balancing Two-Wheeled Inverted Pendulum Robot,' IEEE *Conference Publications, 24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 001357- 001360.

Dew, 'Discrete-time PID Controller Implementation,' 4 May 2014. [Online]. Available: http://controlsystemslab.com/category/articles/control-engineering/pid/. [Accessed May 2014].

http://brettbeauregard.com/, 'Arduino PID Autotune Library,' 28 January 2012. [Online]. Available: http://brettbeauregard.com/blog/2012/01/arduino-pid-autotune-library/. [Accessed May 2014].

MicroChip, '*PIC18FXX2 DataSheet*,' 2006. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf. [Accessed May 2014].

Microchip, n.d. *'Software PID Control of an Inverted Pendulum Using the PIC16F684'*. [Online] Available at: http://ww1.microchip.com/downloads/en/AppNotes/00964A.pdf [Accessed May 2014].

Nawawi, S.W., Ahmad, M.N., Osman, J.H.S. (2007),'Development of a Two-Wheeled Inverted Pendulum Mobile Robot,' *IEEE Conference Publications, 5th Student Conference on Research and Development (SCOReD)*, pp. 1 – 5.

Nguyen Gia Minh Thao, Duong HoaiNghi, Nguyen HuuPhuc (2010),'A PID Back-stepping controller for two-wheeled self-balancing robot,'*IEEE Conference Publications, International Forum on Strategic Technology (IFOST)*, pp. 76 – 81.

Qian Hao, Liping Chen, WeiweiQiao, Peng Li, Songling Yang, Qifang Liu (2011),'Controlling Simulation Study On Two-Wheeled Self-Balancing Electrical Motorcycle Based on ADAMS And MATLAB,'*IEEE Conference Publications, Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC),* pp. 1704 – 1707.

Seul Jung, H. T. Cho, T. C. Hsia (2007), 'Neural network control for position tracking of a two-axis inverted pendulum system: Experimental studies,'*IEEE Transaction on Neural Networks*, vol. 18, no.4, pp. 1042-1048.

Solis, J., Takanishi, A. (2010),'Development of a Wheeled Inverted Pendulum Robot and a Pilot Experiment with Master Students,'*IEEE Conference Publications, 7th International Symposium on Mechatronics and its Applications (ISMA)*, pp. 1 – 6.

STMicroElectronics, 'LPR550AL,' [Online]. Available: http://www.starlino.com/wp-content/uploads/data/acc_gyro/LPR550AL.pdf. [Accessed June 2014]

Thomas Bräunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*, Springer, 3rd ed., 2008

Wu Junfeng, Zhang Wanying (2011),'Research on Control Method of Two-wheeled Self-balancing Robot,'*IEEEConference Publications, Intelligent Computation Technology And Automation (ICICTA)*, pp. 476 – 479.