I J C T A, 9(2) 2016, pp. 513-519

Semi Autonomous Self Balancing Robot

Mithul Mathivanan[1], Mithun Chakaravarrti[1] and G. S. Mahalakshmi[1]

ABSTRACT

Robots and artificially intelligent agents always have an edge over human in terms of accuracy and consistency. We intend to develop a semi-autonomous 2-wheeled Self balancing robot which can be easily manoeuvred through narrow spaces as well as on rough terrain. This robot can be used as a probe during disasters like earthquake to squeeze in through the rubble and help the rescuers with identifying people. The bot houses a Raspberry Pi along with IMU sensor (10 DOF) to observer its current orientation. The motors help the bot move back and forth like an inverted pendulum to achieve stability.

***Keywords*:** Raspberry Pi, IMU, PID

INTRODUCTION

This report documents the design and implementation of a self-balancing robot, which is an unstable system; the basic model is that of an inverted pendulum balancing on two wheels. This work details the derivation of the model of the system and lays out the framework of the robots control system. It shows the full implementation of a control system stabilizing the robot. The robot is built using Raspberry Pi, a credit-card sized computer. The Raspberry Pi is equipped with a 700 MHz Low Power ARM1176JZFS processor running Raspbian operating system. The control scheme used is a Proportional Integral Differential (PID) Controller for state feedback control. The PID is setup to drive the tilt angle to zero. The initial focus is on stabilizing the unstable system and rejecting disturbances. A complementary filter is used to reconstruct an angle estimate to use in the PID controller. The information used to estimate the current system states comes from a 9 Degrees of Freedom (10 DOF) Inertial Measurement Unit (IMU) that consists of a 3-axis Accelerometer, 3-axis Gyroscope, 3-axis Magnetometer. Infrared (IR) sensors are used as distance sensors to detect obstacles and thereby avoid them.

March 25, 2015

RELATED WORKS

Li Chaoquan *et al.* (2011) devised a blend controller based on states feedback control embedded with the PID speed synchronization in addition to a Hybrid Filter (Kalmann+RC network) for their self-balancing robot. This helps the bot remain steady even under impact disturbances. The problem with the blend controller is that the estimated covariance matrix tends to underestimate the true covariance matrix and therefore risks becoming inconsistent in the statistical sense without the addition of "stabilising noise".

Another research paper which helped in defining the vision was by A.N.K.Nasir et al. (2010) where they made a comparative assessment of the LQR and PID control scheme. This paper helped us to learn that the LQR design does not put any restrictions on the input signal amplitude. Thus making PID the obvious choice for the controller.

A 2-Wheeled Self-Balancing Robot with the Fuzzy PD Control Method was devised by Junfeng Wu *et al.* (2012) in which the precompensation of the Fuzzy Logic based controller was proposed. Fuzzy Logic

* Department of Computer Science and Engineering, Anna University, Chennai - 600025, *E-mails: mithulmathi@gmail.com; mithun2595@gmail.com; gsmaha@annauniv.edu*

Controller is experimental. Manual Tuning is required. This is extremely time consuming. Intuitive Fuzzy-PID like design does not outperform the conventional controllers. The performance robustness is not taken into account with FLC. Robustness is not a natural property of Fuzzy logic.

MATERIALS AND METHODS

Raspberry Pi serves as a processing unit for our system. The robot collaborates with its components through the Pi. The robot houses two 500 rpm centre shafted motors. These motors deliver a stall torque of 6 kg.cm. The speed of the motors and the direction of rotation are controlled by the Raspberry Pi using the L293D Dual Half H-Bridge Motor Controller. The Controller takes in input through 4 pins that control the direction in which the motor spins. The speed on the other hand is controlled using Pulse Width Modulated (PWM) signals generated by the Raspberry Pi. In addition to this the robot is equipped with a 10 DOF Inertial Measurement Unit (IMU) sensor, which is essentially a Gyroscope, Accelerometer, Magnetometer and Barometer combined together in the same chip and 2 Infrared (IR) sensors that act as distance sensors to detect the presence of obstacles. The IMU sensor using an i2c interfacing with the Pi, helps to observe the orientation of the robot. A combination of the gyroscope and accelerometer is used through a Complimentary filter and a Kalman filter to get a reliable orientation of the robot and remove any noise that would otherwise be present. The IR sensors return a binary value denoting the presence or absence of an obstacle. The robot uses a PID controller to compute the required speed of the motor from the angle perceived.

(A) Complementary Filter

Both the accelerometer and gyroscope data are used for obtaining the angular position of the robot. The gyroscope does this by integrating the angular velocity over time. The angular position with the accelerometer is obtained by determining the position of the gravity vector (g-force). In both these cases, there is a problem, which makes the data very hard to use without filter.

Figure 1: 10DOF IMU

-A1 The problem with accelerometers: As an accelerometer measures all forces that are working on the object, it will also see a lot more than just the gravity vector. Every small force working on the object will disturb our measurement completely. If we are working on an actuated system (like the quadcopter), then the forces that drive the system will be visible on the sensor as well. The accelerometer data is reliable only on the long term, so a “low pass” filter has to be used.

-A2 The problem with gyroscopes: In one of the previous articles explained how to obtain the angular position by use of a gyroscope. We saw that it was very easy to obtain an accurate measurement that was not susceptible to external forces. The less good news was that, because of the integration over time, the measurement has the tendency to drift, not returning to zero when the system went back to its original position. The gyroscope data is reliable only on the short term, as it starts to drift on the long term.

Figure 2: Infrared Sensor -B PID Control System

The control algorithm that was used to maintain it balance on the autonomous self-balancing two wheel robot was the PID controller. The proportional, integral, and derivative (PID) controller, is well known as a three term controller. The input to the controller is the error from the system. The Kp, Ki, and Kd are referred as the proportional, integral, and derivative constants (the three terms get multiplied by these constants respectively). The closed loop control system is also referred to as a negative feedback system. The basic idea of a negative feedback system is that it measures the process output y from a sensor. The measured process output gets subtracted from the reference set point value to produce an error. The error is then fed into the PID controller, where the error gets managed in three ways. The error will be used on the PID controller to execute the proportional term, integral term for reduction of steady state errors, and the derivative term to handle overshoots. After the PID algorithm processes the error, the controller produces a control signal u. The PID control signal then gets fed into the process under control. The process under PID control is the two wheeled robot. The PID control signal will try to drive the process to the desired reference set point value. In the case of the two wheel robot, the desired set-point value is the zero degree vertical position. The PID control algorithm can be modelled in a mathematical representation. PID is used to calculate the correction term :

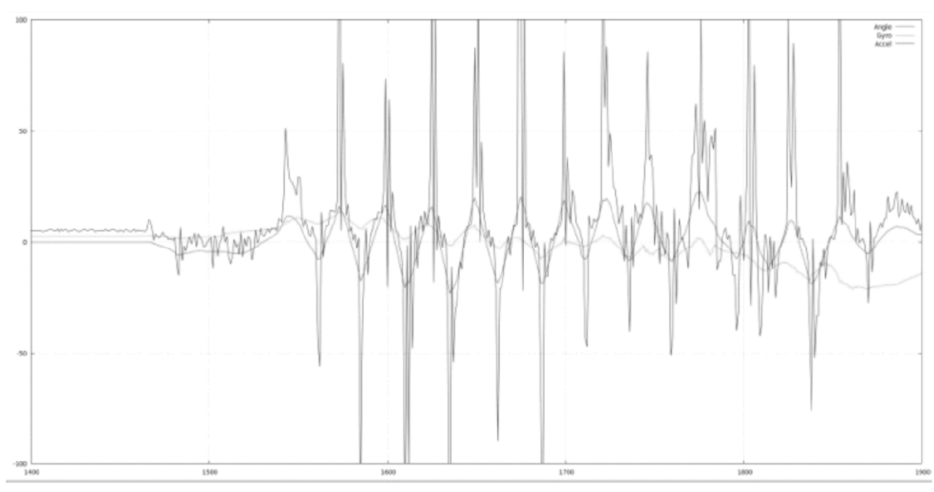

Figure 3: Comparison of computed angle with Accelerometer angle and Gyroscope angle

$$Correction = Kp * error + Ki \int error + Kd \frac{d(error)}{dt}$$

Kp , Ki and Kd are constants which are set experimentally. If only the first term had been used to calculate the correction, the robot would have reacted in the same way as in the classical line following algorithm. The second term forces the robot to move towards the mean position faster. The third term resists sudden change in deviation. The integral term is simply the summation of all previous deviations. Call this integraltotalerror. The derivative is the difference between the current deviation and the previous deviation. Following is the code for evaluating the correction. These lines should run in each iteration:

totalerror+= *correction*;

previousdeviation= *deviation*

correction = Kp * deviation + Ki * total error

+ Kd * (deviation – previous deviation)

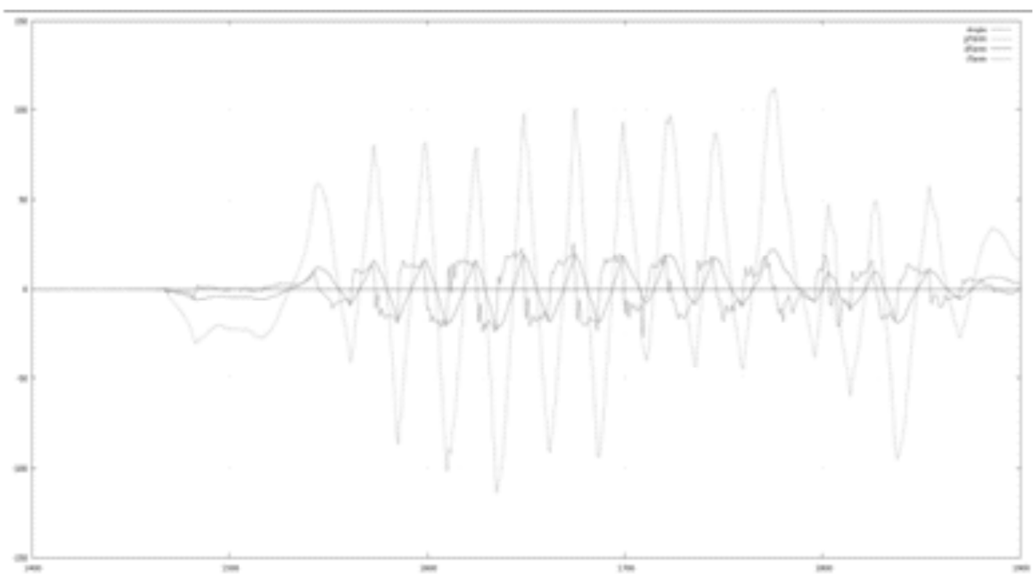

Figure 4: Calculation of PID terms with respect to computed angle

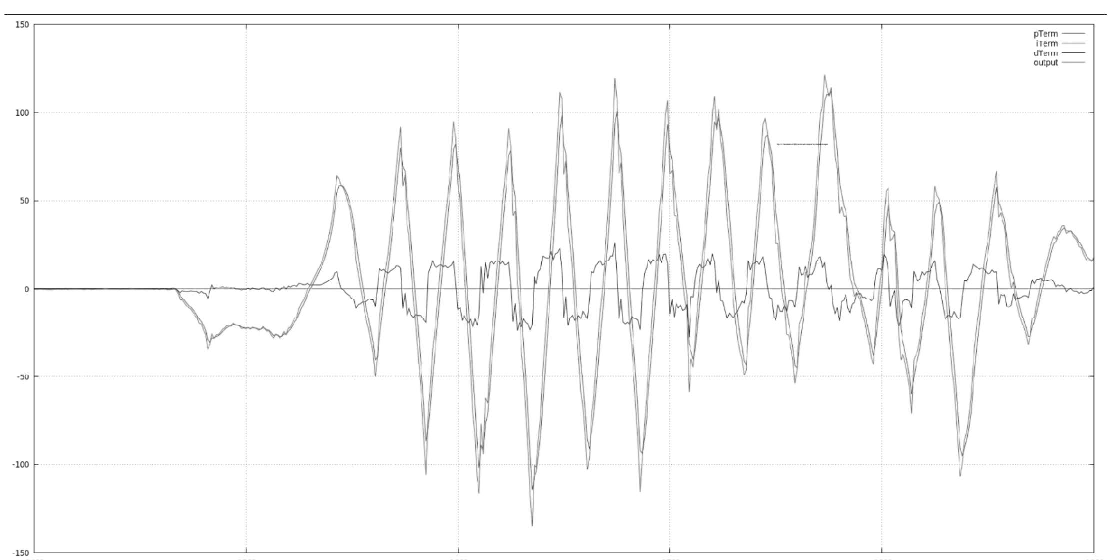

Figure 5: Resultant output correction computed from PID values

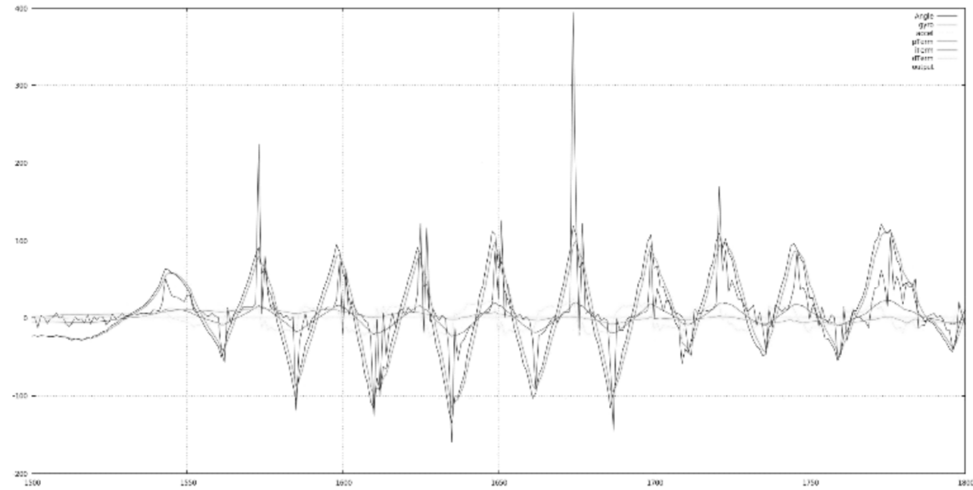

Figure 6: All computed values

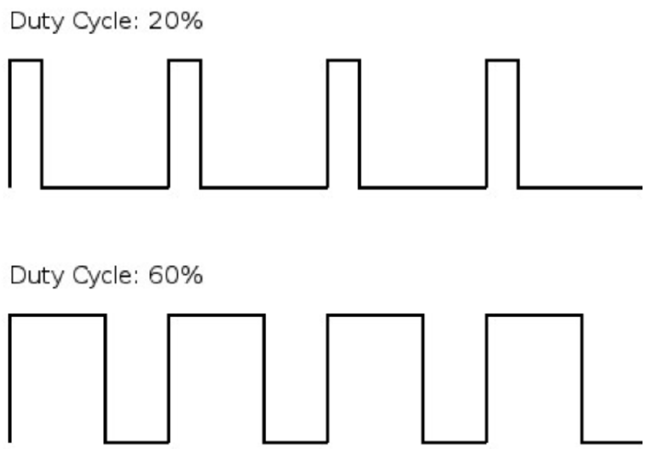

Figure 7: Comparison of output signal with different PWM duty cyclevalues

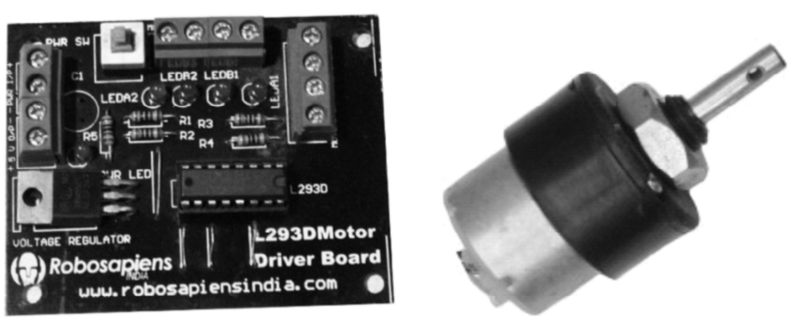

Figure 8: (a)L293D Motor Driver (b)500RPM Center Shafted DC Motor

Controlling the Motors

The raspberry pi is not capable of supplying enough power to operate the motors, thus a motor driver board is used. The L293D is a dual H-bridge motor driver board that allows the raspberry pi to control the direction and speed of the motors while powering the motors using an external DC power supply. The motor driver board has two control pins for each motor that determine the direction in which the motor rotates. The raspberry pi controls the speed of the motor by using Pulse Width Modulation (PWM) signals. The width of the signal determines the speed at which the motor operates.

(D) Interfacing

The robot accepts the input from the Distance sensor for detecting and avoiding obstacles as well as IMU sensor for detecting the orientation of the robot and computed the required speed and direction of the motors using the PID controller and sends the output in the form of PWM signals to the motor driver to control the speed and the direction of rotation of the motors.

The robot can be controlled using a Native Mobile Application/Laptop that communicates with the robot via Wi-Fi provided both the Mobile device/Laptop and the Wi-Fi dongle connected to the Raspberry Pi are connected to the same WiFi network.

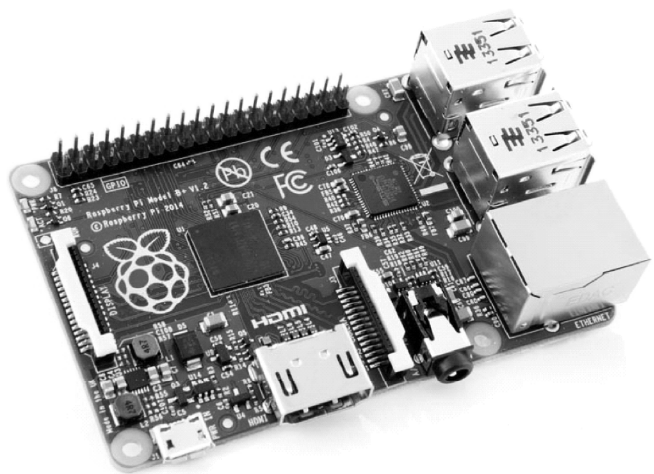

Figure 9: Raspberry Pi B+

RESULTS

The performance of the robot is assesed by calculating the deviation in angle during the duration of the run. This gives a measure of the stability of the robot using different algorithms and different external conditions.

Table I
Comparison of Angle deviation for Kalman and Complimentary Filters (P,I,D : 6,6,10)

Duration (seconds)	*Filter*	*Angle Deviation (degrees)*
2	Complimentary	6.5406130865527
2	Kalman	4.3579497081301
60	Complimentary	4.6049351507858
60	Kalman	3.3670694298121

Thus the Kalman filter is found to be more stable(almost 30% more stable) than the Complimentary filter which is more prominent for shorter runs. As the duration for which the measurements are taken increases, the difference in instability reduces since the bot gets closer to reaching a stable angle with minimal deviation in angle with time due to the dynamic detection of zero angle which is done during each session of the bot.

DISCUSSION

An algorithm for dynamic adjustment of Proportional, Integral and Differential constants (PID constants) and setting of zero angle (where the robot can stay stable) was implemented which over time observes its own movement during each session. Addition of an Artificial Intelligence component for the calculation of the zero angle and automated adaption of the PID values allows the robot to perform well even in case of change in weight distribution or external factors such as rough/uneven surfaces, wind etc.

This makes the robot more stable and resistant to its surroundings.

CONSLUSION

A self-balancing robot using Raspberry Pi as the core processor was built that uses PID controller for controlling the motor speed along with Kalman and Complimentary filters for estimation of angle. The bot was unable to perform in the presence of direct sunlight as the IR rays from the Sun interfere with the IR sensors placed on the robot. Usage of Ultra Sonic sensors would enable the robot to overcome this limitation. Due to limitations in terms of the motors power the robot was unable to handle large impacts.

Given the time and resources a robot with a single sphere can be built. This would enable the robot to move in any direction and make it even more versatile and compact.

REFERENCES

[1] G. X. Li Chaoquan and L. Kejie, *Smooth Control the Coaxial Self-Balance Robot Under Impact Disturbances*. International Journal of Advanced Robotic Systems, Vol. 8, No. 2, 2011.

[2] M. A. A. A.N.K.Nasir and R. Ismail, *The Control of a Highly Nonlinear Two-wheels Balancing Robot: A Comparative Assesment between LQR and PID-PID Control Schemes*. International Scholarly and Scientific Research and Innovation 4, 2010.

[3] W. Z. Junfeng Wu and S. Wang, *A 2-Wheeled Self-Balancing Robot with the Fuzzy PD Control Method*. Hindawi Publishing Corporation Mathematical Problems in Engineering, 2012.

[4] M. S. Team, *Simulink support package for legomindstormsnxt hardware (r2012a)*. Online, 2013. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/35206simulink-support-package-for-...lego-mindstorms-nxt-hardware-r2012a

[5] Y. Yamamoto, *Nxtway-gs (self-balancing twowheeled robot) controller design*. Online, 2009. [Online]. Available: *http://www.mathworks.com/matlabcentral/fileexchange/19147*

[6] M. Arvidsson and J. Karlsson, *Design, construction and verification of a self-balancing vehicle*. Online, 2012. [Online]. Available: *http://publications.lib.chalmers.se/records/fulltext/163640.pdf*

[7] S. Nilsson, *Sjalvbalancerande robot*. Online, 2012. [Online]. Available: http://sebastiannilsson.com/k/projekt/selfbalancing-robot/

[8] Segway, *Segway - the leader in personal, green transportation*. Online, 2013. [Online]. Available: *http://www.segway.com.10*