ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

Self-Balancing Robot

Sivakumar S[1], Thariqul Abrar M[2], Thanesh T[3], Balamurugan H[4], Bharath K R[5]

Assistant Professor[1], UG Student2,3,4,5
Department of Electrical and Electronics Engineering
Saranathan College of Engineering

Abstract— This paper reports the outline, construction and control of a two-wheel Self-Balancing robot. It aims to study physical and dynamical characteristics that a two wheeled robot exhibits and to implement different control techniques to improve its performance. The robot focuses to attain a balance state by itself when it is subjected to any disturbance. The design framework contains a couple of DC gear motors and an Arduino microcontroller board; a three-axis accelerometer and gyroscope comprises sensor with six DOF (degree of freedom) which measures the angle acts as feedback sensor along with them a PID control algorithm is implemented.

***Index Terms*—**Accelerometer, DC motors, Gyroscope, Robot control, PD control, PI control angles.

I. INTRODUCTION

SELF –BALANCING ROBOT resembles inverted pendulum design configuration which rely upon dynamic balancing of this system. This robot basis gives good strength and ability because of their compact size and power prerequisites. Such robots have their applications in observation and transportation which is an important test bed in control education and research. Specifically, the attention is on the electro-mechanical systems and control calculations required to empower the robot to see and act continuously for a powerfully evolving world.

The essential requirement for the control of the system needs angle measurement .A stable angle point is marked as a set point and the strategy is to drive the DC motors backward or forward so as to maintain stable angle position, this working can be effectively found in Segway. The robot uses MPU-6050. This sensor provide input value to Arduino for all angle variations. Arduino is based on ATmega processors an open prototyping platform which has C based software development environment, and can be connected with a variety of sensors. It is an emerging platform for both education and product development, robotics etc.

In this journal we report a project design, and control of a two-wheel self-balancing robot. Robot has two gear motor which is attached firmly to the robot body.The estimation of PID parameters i.e. Kp, Ki and Kd have been obtained and applied to the Arduino. The PID Algorithm loop programming has been composed to change the digital data from the accelerometer to a speeding up magnitude vector. The magnitude is then compared to a predetermined mathematical function to infer the angle of tilt of the platform. The angle of tilt is then converted to angle of rotation for the servos to act on. The experimental outcomes are exhibited, which demonstrate that stability of the upright position is accomplished with PID control within little tilt

II. FUNDAMENTALS

A. *PID Controller:*

PID Controller is most normal control calculation utilized as a part of robots & industrial automation over 95% of the modern controllers are of PID integrated. PID controllers are utilized for more exact and precise control of different parameters such as regulation of temperature, pressure, speed, flow and other process variables. Due to robust and functional simplicity, these have been implemented to plenty of industrial applications where a more precise control is the foremost requirement. The entire idea of this algorithm revolves around manipulating the error. The error as is evident is the difference between the Process Variable and the Set point. It gets the input parameter from the sensor which is referred as actual process variable.

It also accepts the desired output, which is referred as set point, and then it calculates and combines the proportional, integral and derivative responses to compute the output.

As the name suggests, PID algorithm consists of three basic coefficients: proportional, integral and derivative which are varied to get optimal response. Fig. 1 represents closed loop PID controller block diagram. The proportional corrects instances of error, the integral corrects accumulation of error, and the derivative corrects present error versus error the last time it was checked. The effect of the derivative is to counteract the overshoot caused by P and I. When the error is large, the P and I will push the controller output. This controller response makes error change quickly, which in turn

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

causes the derivative to more aggressively counteract the P and I

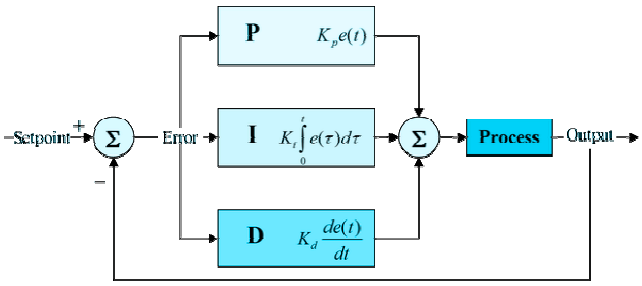

Fig.1 Block Diagram of PID Controller

Robot achieves its stability by tuning the controller with appropriate gain values (Kp, Ki, Kd).This robot uses trail & error method for tuning which is simple but this method is not suitable for all gain values changed step by step More
KP-will lead to oscillation and will generate an offset
KI- will counteract the offset. Higher Value of KI implies that the Set point will reach the PV too fast if this action is very fast, the process variable is prone to be unsteady KD-keeps this under control.

B. Working of PID:

Function of Controller can be explained in less complexity with two example situation

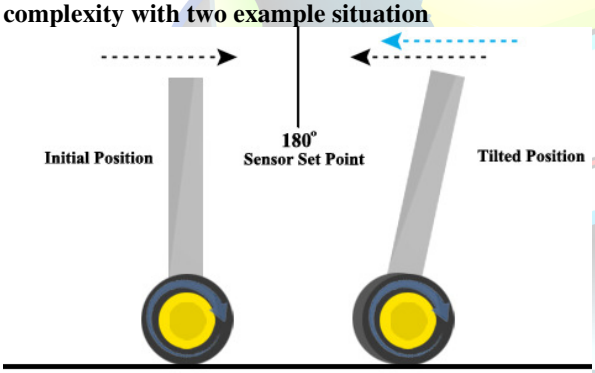

Fig. 2 Illustration of robot subjected to perturbation

In case (i) the robot is at its initial position; maintaining its stable state (ii) small perturbation which moves the robot from its initial state and tilted over its right. The robot is standing up and it is pushed towards right side. The position algorithm will make the reference change to the opposite side (left side), proportionally to the distanced travelled. Only the proportional component would make the robot to reach the original position, with a target angle of 180 degrees, but probably with some speed, that would make it go to the left side. So, using only proportional there are some disadvantages, the original position would overpasses if the robot is tilted too far from the original state, the target angle would be very large difference that makes the robot to fall.

Now, assume the robot is tilted too far away from the initial state. In this situation, before the fall, the robot would get its maximum speed to keep the new target angle. So, the solution is to scale down this speed, by adding or removing an offset to the target point using integral controller this component prevents the robot from falling down, this is done by varying the PWM value decided by the PID controller, which is the input for both motors. In Fig. 2 Illustration of robot subjected to perturbation, the black arrow indicates counteract of the robot to reach its initial state after small perturbation, this calculation has few parameters that helps for the steadiness of the robot. Different gain values are required in different situations since it is a trial and error method. In the tuning process the value of gain is repeatedly adjusted until the robot maintain its steady state. The perfect values of gain value is obtained from this tuning.

III. WORK PROGRESS

This section focus on mechanical construction and software configuration followed to build the robot.

Hardware Specification

The mechanical structure of the self-balancing robot is made with precise measurement and robustness. The structure has three segment: (i) the bottom part that carries the motors (ii) the second part to hold the electronic materials (Arduino, sensors, H-bridge, among others) (iii) the top part, where the user can place objects to test the equilibrium, or use it to install more additional electronic features.

C. Arduino Uno: Arduino Uno is a microcontroller card which has ATmega328 (Fig. 3). It has 14 digital input-output (6 of them can be used for PWM), 6 analog input, a 16 MHz quartz crystal, a USB connection port, an ICSP connection and a reset button. It also has 5V (operation voltage), 3.3V, GND and VIN pins in the power side. The input voltage can be between 7V and 12V.USB connection is used for providing supply voltage and to burn program coding into the board. In our project Arduino runs with a 12V supply provided by battery backup. Memory features, the Arduino has Flash Memory 32KB, SRAM 2KB, EEPROM 1KB. Arduino IDE complier is used to compile the Arduino code.

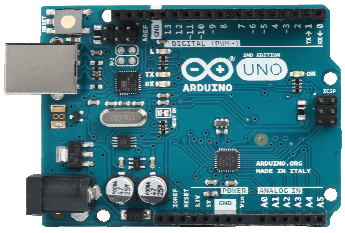

Fig. 3 Arduino Uno

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

D. Motors and H-bridge: Gear motor is a DC motor in which increase in torque decreases the speed of the motor. By using different combination of gears, its rpm can be reduced to any desirable value, but speed and rpm will be small when compared to a normal dc motor. The direction of the gear motor can be reversed by simply reversing the polarity and the speed of the motor can be controlled by changing the voltage level across it. Gear motor is also available in many range of rpm. In our project we are using 300 RPM gear motor. For interfacing motor with Arduino, dc motor driverL293D is used. It consist of two H-bridge designed using 4-transistor circuit that reverse the direction of rotation and control the speed of the motor. Hence controlling two motors simultaneously is achievable. The driver IC has 4 input pins, 4 output pins, 2 enable pins, Vcc and GND. Vcc is the supply voltage for the driver to operate. L293D will not use this voltage for driving the motor. For driving the motors there is additional provision Vss to provide motor supply. Vcc is supplied to the driver IC board using battery backup . In Fig. 4 image of L293D, Pins ENA, ENB are used to provide PWM signal for the motors received from Arduino digital pins . Pin IN1,IN3 are used to drive the motors in forward direction while IN2,IN4 are used to drive the motor in reverse direction.

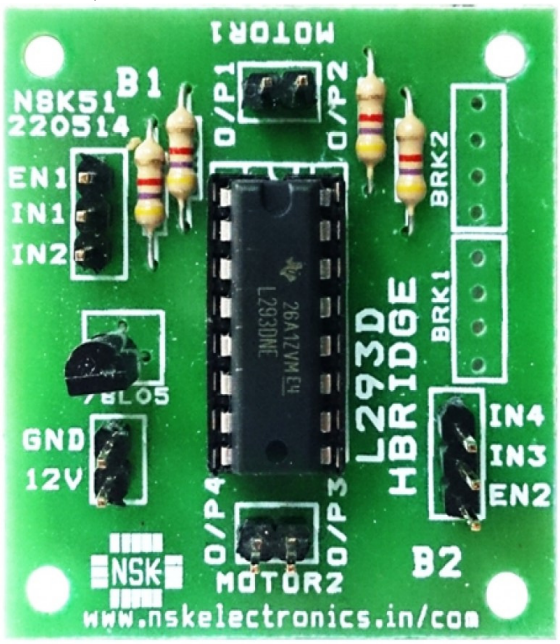

Fig. 4 Motor driver (L-293D)

E. MPU SENSOR:

MPU6050 is a combination of accelerometer and gyroscope sensor which is an imperative module for this project. The communication between IMU and Arduino is achieved with I2C at 400 Hz. SCL and SDA connections provide I2C communication's logic level of VLOGIC reference pin. VLOGIC pin is connected to 3.3V pin because 5V pin can cause damage to the sensor. MPU6050 converts the output of gyroscope and accelerometer to digital with 16bit ADC.MPU-6050's 16-bit ADC feature simplifies reading the data by converting analog data to digital data. The output of the sensor consists of 16-bit, which is 8-bit low and 8-bit high, for each axle. The sensor data, which is raw data is received by the Arduino with I2C communication. The received data is required to be converted to g-force or angular velocity.

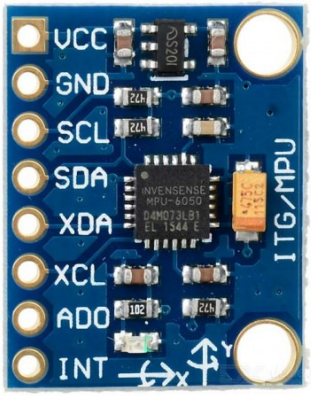

Fig. 5 MPU 6050 Sensor

B. Software Specification:

The project design created for this incorporates a set of standard libraries essentially to read the sensors and to control the motors. In these libraries, each one was tested

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

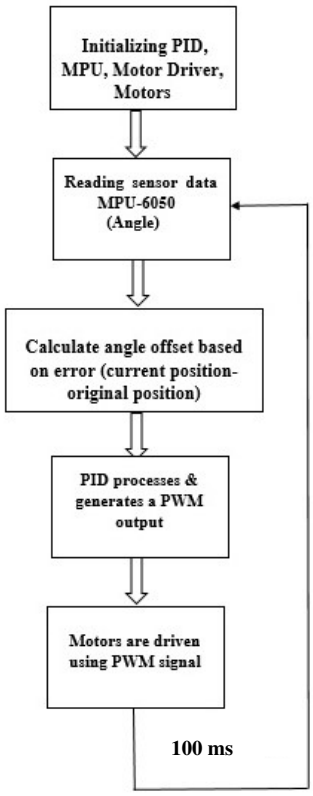

Fig. 6 Software Loop Schematic

The MPU sensor is calibrated by maintaining it to a stiff position or reference position. Then using sample sketch, the calibration value is noted which will be used as the set point for the MPU. Once the calibration is done the values are assigned to the main program and the motor drastically improves in the balancing process.

IV. SYSTEM OPERATION

To adjust the robot, maintain it in stable state we don't require the current angle of the robot we need to calculate the rate at which it is tilting. The MPU6050 measures the angular rate (rotational speed) along the three axes. We get the input gyro value about the X-axis and convert them to degrees/sec and then multiply it with the loop time to obtain the change in the angle. We have the angular velocity of the robot from the gyro sensor. Then multiply that value with the loop time to get the change in angle. Then we add that value into the previous angle to the get the current angle of the robot. In order to do this we need to know the loop time. Hence we use the function millis() which indicates the time taken from the start of the program. In our project milli second is 100 then loop time will have the difference between the Set Time & Previous Time. Then we use the function mpu.getRotationX() to get the angular velocity about X-axis, now the gyro holds the data in a 16bit register and we will be using the map() function to map it to a range between (-250,250). Then Gyro Angle will be equal to the sum off the previous angle and the Gyro Rate times the loop time in seconds. When the robot attains its steady state there will be deviation noticed in the robot. This drift is caused by the gyro. So, we cannot use the gyro Alone to measure the angle. Thus we are using both gyroscope and the accelerometer together. Any deviation in the sensor from the set point will drive the motor in either direction proportional to the direction of angle deflection. The PID manual tuning is done through step by step increment of all the three gain values Kp, Ki, Kd. For every change in the gain & angle the robot attains its stability by reducing its oscillation. Tuning the controller value and calibration of sensor will be a difficult task which need precision to attain the stability once the tuning is done the motor will attain the stable state

V. CONCLUSION

To make a self-balancing robot we first calculate and derive transfer function then check its real time response. But in our project only mini prototype is made so it is very easy to work without any equation Also, there is not much importance for precisions .Then the gyroscope is calibrated using library and set point is measured by placing it in vertically above the robot. We checked the controllability by providing various gain to the controller algorithm. Above steps are result in success, there were very close to build a self-balancing bot. The easy way to tune a controller is to tune the p, i and d parameters one at a time. The stability of the robot may be improved if you choose a enough rpm of motor and adjust its speed factor. So by implementation of all these concepts and avoiding the errors that we came across, the self-balancing bot is completely build. We can make Segway and SOHO security as a application of self-balancing. Further work will include by adding a RF control, thus allowing the robot to move front and back using a remote control. Also by improving the components of the robot to achieve higher speeds & high torque, automatic wheel chair can be designed for physically disabled people.

References

[1] R. Chan, K. Stol, and C. Halkyard, "Review of modelling and control of two-wheeled robots," *Annu. Rev. Control*, 2013.
[2] Ricardo Santos Martins and Francisco Nunes "Control System for a Self-Balancing Robot" *2017 4th IEEE Int. Conf.*, 2013.
[3] H. Juang and K. Lurrr, "Design and control of a two-wheel self-balancing robot using the arduino microcontroller board," *2013 10th IEEE Int. Conf.*, 2013.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

[4] Roland Pelayo, "How to build a self balancing robot" *maker.pro*. [Online]. Available: https://maker.pro/projects/arduino/build-arduinoselfbalancing-robot
[5] Albert Ko, H.Y.K. Lau and T.L. Lau, "SOHO security with mini self-balancing robots," 2005 *Industrial Robot: An International Journal 32/6*

Fig 7. Operation Block Diagram