IoT Enabled Self Balancing Robot using LabVIEW

K Akhilesh Rao[1], Makani Ramakrishna[1], Jency Monica[1], Shriya V S[1], Harsha H[1*]
[1]Department of Electronics and Instrumentation, RV College of Engineering, Bengaluru-560 059

Abstract

Self-balancing robot finds applications in restaurant serving, advanced military surveillance, warehouse and transportation. Self-balancing two-wheeled mobile robot has advantages over that of the traditional four-wheeled in terms of zero turning radii which lends better manoeuvrability and flexibility even in narrow paths. Open literature indicates issues like jitter in the motion of robot due to inaccuracy in angle measurement for self-balance and issues of control system stability. A prototype model with myRIO module was designed and tested for effectiveness. Two angular position sensors, i.e. accelerometer and gyroscope were deployed with PID balancing controller and a complementary filter for fusing sensor data for increased accuracy of tilt measurement. A* path planning algorithm was implemented in LabVIEW software for finding the shortest path for robot to travel from start to end position. An obstacle detection system using an ultrasonic sensor was deployed for collision avoidance. Upright balance on two wheels was achieved 99% accuracy.

Keywords: *Two wheeled Self-balancing robot, LabVIEW, myRIO, PID, Complementary filter*

1.0 Introduction

Two wheeled robots is found advantageous for industries and domestic applications because of zero turning radius and compactness. Major areas of application of self-balancing robots include restaurants, military, warehouses and segway. The inherent Instability due to two wheels requires control algorithm for its standing upright and self-balancing. Two Wheeled Self-balancing robot using a Proportional-Derivative Proportional Integral (PD-PI) with Kalman filter which provides remote control application using IoT is developed [1]. Outer-loop control mechanism for two-wheeled mobile balancing robot, position-tracking control system applications using fuzzy adaptive algorithm are reported. It has both proportional-type feedback controller and self-tuning algorithm to enhance position-tracking performance in transient periods [2]. Double-loop control scheme based on active disturbance rejection

[*]Mail address: Harisha H., Assistant Professor, Department of Electronics and Instrumentation, RV College of Engineering, Bengaluru-560 059, Email: harshah@rvce.edu.in

control to implement a stable upright control of the robot for self-balance applications is developed [3-4]. Kinematic model of a two wheeled self-balancing robot and its control using Linear Quadratic regulator control techniques is reported [5]. It uses a complementary filter and Proportional Differential (PD) controller implemented using LabVIEW for efficient self-balancing robot [6]. Two-wheeled self-balancing robot using Arduino Microcontroller Board along with a single-axis gyroscope and two-axis accelerometer is demonstrated [7-8]. Two wheeled balancing mobile robot equipped with tilting mechanism in the lateral direction is developed. It can generate roll motions around the forward direction vector [9]. Kinematics model of a two wheeled self-balancing autonomous mobile robot is simulated in ADAMS along with experimental validation [10].

This research incorporates PID control algorithm using LabVIEW and NI myRIO to solve real time control of self-balancing and navigation application. The feedback for balance control is achieved by the fusion of accelerometer and gyroscope sensor outputs using a complementary filter for high accuracy measurement. Remote control of the robot is achieved using a dashboard app provided by National Instruments. A* Path planning method is used for the robot to traverse from point to point without any collision in the predefined map given as input to the system. For obstacles that are not present in the map, obstacle detection using ultrasonic sensor is combined with the path planning algorithm to continuously update the robot path.

2.0 Design for self-balancing

Design for self-balancing mainly comprises of sensor data acquisition, complementary filter for fusion of gyroscope and accelerometer output, self-balancing PID control, motion control along with self-balancing, path planning algorithm block and IoT through NI Dashboard. For achieving self-balancing, a PID based control system by using an IMU (Inertial Measurement Unit) consisting of an accelerometer and gyroscope as feedback was used. LabVIEW software was used for graphical programming.

The desired angle set point (0 degrees in this case) is given as input to the control system and feedback of the current angular position of the robot with respect to vertical axis is taken from the gyroscope and accelerometer. The feedback is subtracted from the desired set point (generally 0 degrees for self-balancing) to produce an error signal which is given to the PID controller input. Based on the error signal, the PID controller produces an output which acts as the duty cycle for the pulse

width modulation (PWM) signals which controls the speed of the motors connected to the wheels in order to produce a balancing action. The motors rotate in the same direction in which the robot tends to fall to produce a balancing action. Fig. 1 shows basic block diagram of the robotic system.

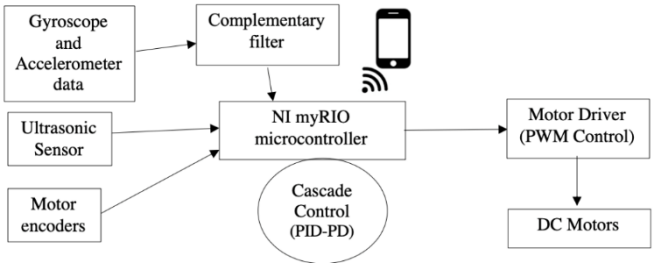

Fig. 1. Block Diagram of the Robot

2.1 Gyroscope data acquisition

In this system, the L3G4200D gyroscope is used which is connected to the NI motor control board which is connected to one of the ports of NI myRIO 1900. The L3G4200D gyroscope sensor uses I^2C protocol to send raw angular velocity data to myRIO. The gyroscope acts as the I2C slave and myRIO acts as I2C master. The angular velocity data sent by the gyroscope is in the form of LSB.

The 16-bit L3G4200D gyroscope output is in signed integer 2's complement format and the sensitivity of the sensor is 0.00875 degree per second / Least Significant Bit (dps/LSB) from the datasheet. Thus, angular velocity in degree per second can be obtained by multiplying the raw output of the gyroscope with 0.00875 dps/LSB. The angular velocity data is sent through an integrator block to obtain angular displacement.

2.1.1 Gyro drift / zero rate problem in gyroscope

The L3G4200D gyroscope output has a non-ideal characteristic called zero-rate level offset error. Even if the gyroscope is stationary (called zero rate input) the output contains some non-zero offset (zero-rate level). With this error as input to integrator input, the output starts to increase at a constant rate as offset error would be constantly integrated over time. Thus, angular displacement keeps increasing even when the gyroscope is stationary.

2.1.2 Solution to Gyro drift problem

In order to eliminate gyro drift the gyroscope is kept stationary and output of the integrator is allowed to increase (due to gyro drift) for a

known period of time (say, 60 seconds). The angular displacement output is noted down after a known time period and is divided by the time elapsed. It gives mean offset error rate of the gyroscope for zero displacement. The mean offset error is subtracted from the angular velocity before it is fed to the integrator. It removes the gyro drift problem to a certain degree, but the output still have small low frequency errors (Fig. 2).

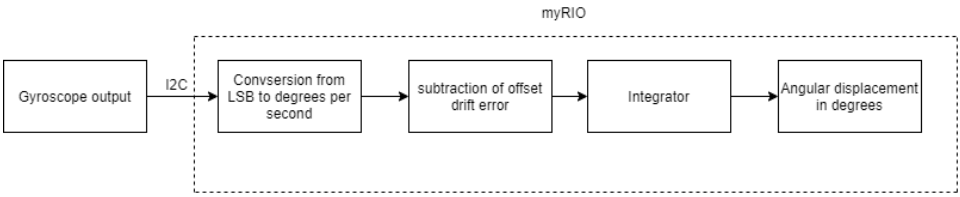

Fig. 2. Procedure of acquiring data from gyroscope

2.2 Complementary filter for fusion of accelerometer and gyroscope

The gyroscope output alone cannot be used as a feedback element for angular displacement computation in the control system due to the gyro drift error and the output of a gyroscope is reliable only for a short period of time as it starts to drift from the true value when used for long duration. Another device that can be used in place of a gyroscope for angular displacement calculation of the robot is accelerometer (Fig.3). The accelerometer alone does not give accurate output as it is affected by all the forces that act on it. Every small force acting on the gyroscope disturbs its output and hence it can be considered highly sensitive for the present application as stable feedback is needed for self-balancing of the robot. Thus, the accelerometer output data can be useful only in long term as it has high frequency error.

For best possible result the outputs of both gyroscope and accelerometer are combined using a complementary filter. Since the accelerometer has high frequency error and gyroscope sensor has low frequency error, the output of the accelerometer is passed through a low pass filter which filters out high frequency noise and output of the gyroscope is passed through a low pass filter. Output of the filters are summed to give the output of the complementary filter. In this way, by fusing the data of accelerometer and gyroscope, both high and low frequency noise error gets eliminated. The output angle measurement of the complementary filter can be mathematically written as equation (1).

Filtered Angle=d x (Gyroscope Angle)+(1-d) x (Accelerometer Angle) ----- (1)

(Gyroscope Angle) = (Last Measured Filtered Angle) + (Gyro output x t)

Where d = Impulse response of a first order high pass filter = exp (-t/T), T is time constant = 0.75 seconds and t is sampling time = 0.005 seconds

Using these values,

filtered angle = 0.9934 x (gyroscope angle) + 0.0066 x (accelerometer angle)

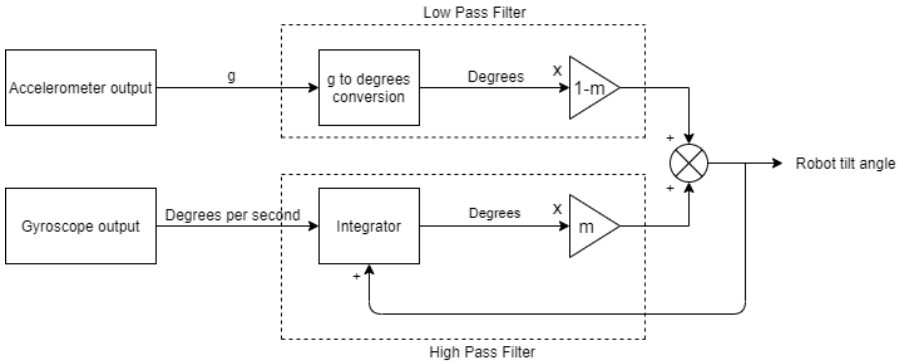

Fig. 3. Block Diagram of Complementary filter

2.3 PID control for self-balancing control

A PID controller (Fig.4) is used for implementing the control action for balancing the robot and keeping it upright. The input to the PID controller is an error signal obtained by subtracting the feedback from the complementary filter from the desired set-point. Proportional, integral and derivative action is performed on this error signal and a correcting signal output is produced by the controller in the range of -1 to +1 as a PWM range. This output of the controller acts as the duty cycle for the pulse width modulated signal (PWM signal) which is given to the motors connected to the wheels. Any PID Controller output can be mathematically determined using mathematical formula as in equation (2).

$$P = Kp * e + Ki * E + Kd * edt \quad \text{----------} \quad (2)$$

where P = Output of PID controller (correcting signal)

Kp = Proportional Gain, Ki = Integral Gain, Kd = Derivative Gain, E = Integrated error signal, e = Error signal, edt = Differentiated error signal

The proportional gain Kp, derivative gain Kd and integral gain Ki for the PID controller are determined using the Ziegler-Nichols technique. As per the Ziegler-Nichols technique, the integral and the derivative gain are kept at 0 and proportional gain is incremented in small steps until stable and continuous oscillations are achieved by the system. The proportional gain value at which these oscillations are achieved is called the ultimate

gain, Ku and the oscillation period is Tu. These two values are used to set the gain values using the relation:

Kp = 0.6 x Ku, Ki = 1.2 x Ku/ Tu, Kd = 3 x Ku x Tu /40

In this way the upright position of the robot is maintained by controlling the motors based on the tilt angle of the robot using a simple PID controller.

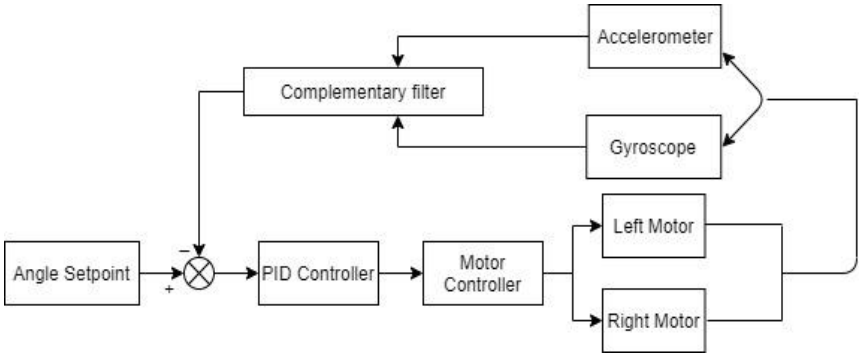

Fig. 4. PID control for self-balancing

2.4 Design for motion control

Motion control is the component which provides movement and manoeuvrability to robot. The signal for control of the robot movement is given from the NI dashboard application. In case of a four wheeled robot, PWM signals can directly be given to the motors connected to the wheels to move the robot. But in case of a two wheeled mobile self-balancing robot, the motors cannot be directly controlled as it will make the robot unstable by interfering with the self-balancing control loop. Thus, the motion control loop has to be cascaded along with the self-balancing loop in order to make the robot stay balanced while in motion. Motion control system has two main components: A PD control for forward and backward motion and A PD control for turning motion of the robot.

2.4.1 PD control for forward and backward motion

Easy way for a two wheeled self-balanced robot move forward or backward (Fig.5) is by introducing a small tilt error as input to self-balancing PID block. Due to the small angular error given to the balancing PID control block, the PID produces an output which moves the robot in the direction in which the robot is falling. In this way, by providing a very small positive or negative error input to the balancing PID block, the robot can be moved forward or backward as per the input given by the user. In order to calculate the distance moved by them-self balancing robot, an encoder is used which produces a given number of

pulses for one revolution. So, the distance to be travelled is given as the input setpoint to the system. This setpoint is subtracted by the feedback coming from the encoders to give an error signal. This error signal is fed to a PD controller which produces an output which is used as the tilt error signal to the balancing PID loop. Thus, as the robot gets closer to the setpoint, its speed will reduce because of the control action of the PD controller and it will gradually come to rest when the robot reaches the setpoint distance. This same principle cannot be used for turning motion as both wheels are controlled together by the balance control loop, but in a turning motion both the wheels have to be controlled independently.

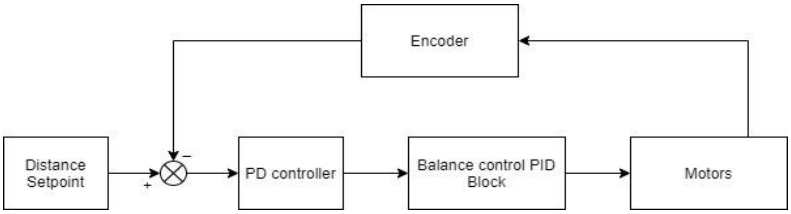

Fig. 5. PD control for forward and backward motion

2.4.2 A PD control for turning motion of the robot

In order to achieve turning motion, both the motors connected to the wheels are directly given PWM signal which is summed along with the output of the balancing PID control (Fig.6). The motors are made to rotate in opposite directions to each other in order to produce a rotating motion. The input setpoint to the system is given as the number pulses produced by the encoder for a 90-degree turn. The output of the encoder is given as the feedback. The error signal produced by subtracting the feedback from the setpoint is given to a PD controller which produces an output which is treated as duty cycle for the pulse width modulated signal (PWM signal) to be given to the motors. This PWM output is added to the PWM output of the balancing PID for one wheel and is subtracted from the PWM output of the balancing PID for the other wheel in order to rotate them in the opposite directions. The proportional and integral gain values for the PD controller are found out using the Ziegler-Nichols technique.

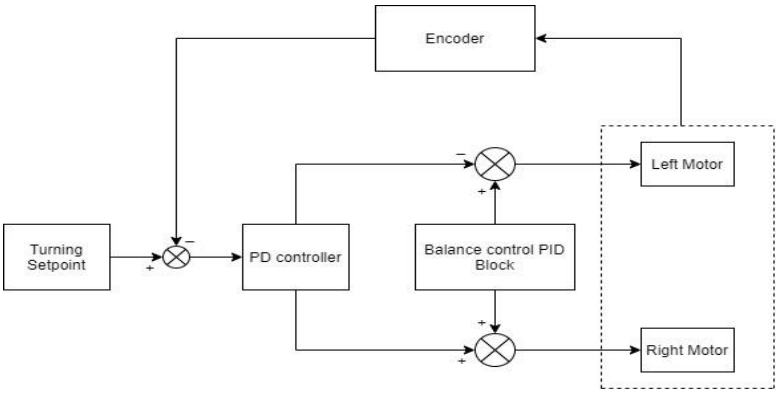

Fig. 6. PD control for turning motion

2.5 A* path planning algorithm

Path planning technique is being used in this project in an attempt to make the robot completely autonomous. Basically the A* path planning algorithm is used which requires a predefined map input. Path planning is combined with obstacle detection in order to avoid the obstacles which are not present in the map. A* is a popular path planning algorithm widely used in video game development and also autonomous vehicles. It takes input as map and gives the shortest route between any two coordinates in the map as per the user's input. The map is divided into grids or coordinates for simplifying the search area. The algorithm comprises of an open list and a closed list. The open list is a set of all the coordinates or grids in the vicinity of the present coordinates where the robot is, where the robot can go next. The closed list comprises of all the coordinates or grids that has the robot has already travelled and is not supposed to travel in the future.

The A star algorithm uses the following simple formula for path planning-

$F= G + H$

Where F is the overall cost to travel to an adjacent grid

G is the cost to move from start node to the next adjacent node

H is the cost to move from that node to the destination node without considering any obstacles in the way to the destination node.

All the nodes in the open list will be evaluated as per the above formula and the node with the lowest F score will be considered as the next move towards the destination and it will be included in the closed list. In case two nodes in the open list have the same F score, then both the paths will be evaluated and the F scores of both the paths are compared on reaching

the destination, in case both the routes are able to reach the destination. The one with the lower F score is considered as the shorter path. This process keeps on continuing until the destination coordinate is added to the closed list which indicates that the robot has the reached the destination (Fig.7).

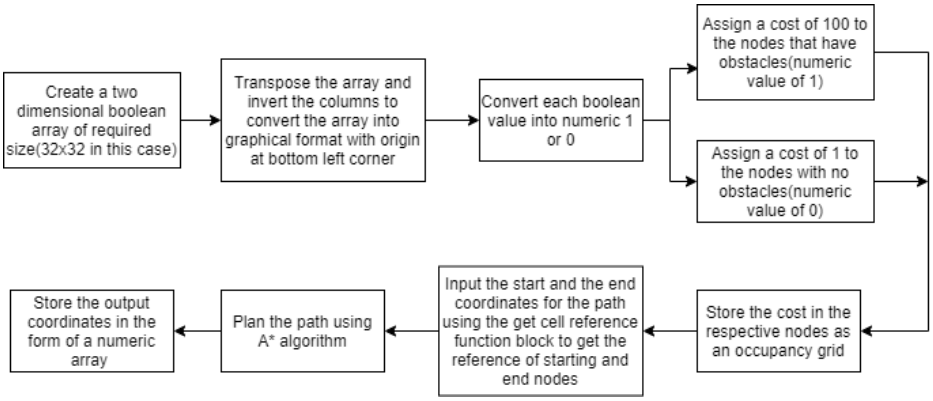

Fig. 7. A* Algorithm Implementation

2. 6 Obstacle Detection

An obstacle detection system is used to ensure that the robot will not collide with any object while moving to its goal in an optimum way. A robot gets data of the physical variables in its vicinity through sensors that are mounted on the robot. There are many sensors that can be used for the purpose of obstacle detection in the market like infrared sensors, capacitive sensors, inductive proximity sensors, ultrasonic sensors, lidar, hall effect sensors, sonar, radar, magnetic type proximity sensors and many more. But for our particular application, an ultrasonic sensor is used as distance of an obstacle can be measured with ease, is inexpensive and can also be easily mounted on top of the robot.

The range of operation of the ultrasonic sensor being used is about 10 cm to 30 cm. When the robot is moving on the desired path the ultrasonic sensor continuously keeps transmitting ultrasonic waves in the direction in which the robot is moving. Whenever an obstacle is encountered, the ultrasonic waves that were transmitted by the sensor are reflected by the obstacle and are received back and, is detected by the sensor. Based on the time required for the ultrasonic to hit the obstacle and come back to the sensor, the distance of the obstacle from the sensor/robot can be calculated. Distance is given by equation (3).

$$Distance = \frac{T*C}{2} \quad \text{----------} \qquad (3)$$

where, T = time needed for the ultrasonic wave to hit the obstacle and return

C = speed of sound, If the distance between the robotic system and the obstacle is less than or equal to 5 cm, then the motion of the robot is stopped, and the robot comes to rest.

2.7 NI Data Dashboard App for Internet of Things (IoT)

The NI Dashboard is an application by National Instruments which allows the user to create dashboards(or a front panel) consisting of different controls and variables such as charts, buttons, LEDs, textboxes, gauges etc., in order to remotely control a system. The concept of shared variables is used in order to send and receive data from the dashboard application. The variable which the user wants to access remotely is declared as a shared variable in LabVIEW application. These shared variables are sent through the network with the help of a shared variable engine (SVE). The shared variable engine uses the NI Publish/Subscribe protocol in order to transfer shared variable data through the network. In this type of communication, the sender "publishes" data into a particular class among several classes in the network and the receiver "subscribes" to a particular class of interest and extracts the data present in that particular class. In this project, the motion control of the robot is done using the app entirely using the concept of shared variables. Also, the data like the current angular displacement of the robot, distance of obstacle from the robot and PID response of the system can be directly viewed in the dashboard application.

3.0 Results and Discussion

It consists of outputs obtained from sensor data acquisition, complementary filter output, comparison of filtered and unfiltered sensor data, output response of self-balancing PID control loop and also output of the software implementation of the A* path planning algorithm.

3.1 Gyroscope and Accelerometer Response

Fig. 8, it is evident that the accelerometer response is not stable at the balance point of the robot, and it keeps drifting, hence it is alone not ideal for stabilising the position for self-balance of the robot. It is observed that accelerometer output does not drift over time and is constant at -140 degrees, but has a lot of noise which can make a self-balancing system unstable.

Fig. 9 understand the Gyroscope response shows a drift as an offset over a long period of time. There is a drift of nearly 1 degree in a time period

of 10 seconds which makes its output highly unreliable in the long term. Its response is reliable only for a short period of time.

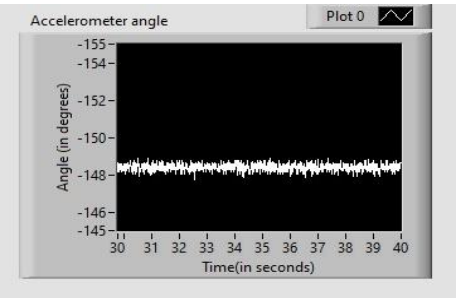

Fig. 8. Accelerometer response at balance point

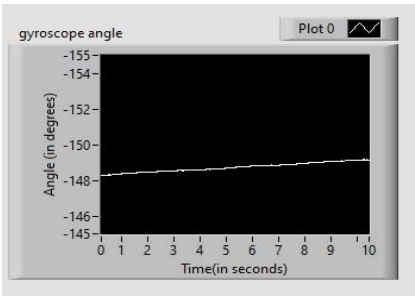

Fig. 9. Graph of Gyroscope drift over time

3.2 Complementary Filter Response

Fig. 10a shows LabVIEW based complementary filter output. The robot is kept stationary at an angle of -148.2 degree. Fig. 10b shows outputs of the sensors and filter at an interval of t = 0 to 40 seconds.

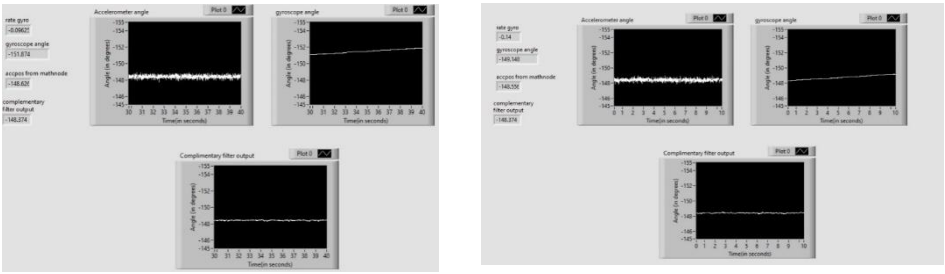

Fig. 10. Complementary filter output **a)** before gyro drift and **b)** after gyro drift

It is observed that the output of complementary filter remains unaffected by the gyroscope drift as well as the noise present in the accelerometer. In the interval t = 0 s to t =10 s, the gyroscope value starts drifting from its original value while the noisy output from the accelerometer remains constant at its original value. In the interval t=30s tot = 40s, the gyroscope output has drifted by around 4 degrees in a time period of 40s. In both these cases the output of the complementary filter remains constant at its original value and does not have any noise.

3.3 Self balance PID response to disturbance

The response is obtained by tuning the controller using Ziegler-Nichols technique. Using this technique, the ultimate gain Ku and the oscillation period Tu, with this ultimate gain value was found to be.

Ku=0.25, Tu=0.57 ms

As per Ziegler-Nichols technique,

Kp = 0.6 x Ku= 0.6 x 0.25= 0.15

Ki = 1.2 x Ku/ Tu= 1.2 x 0.25/0.00057= 526.31

Kd = 3 x Ku x Tu /40= 3 x 0.25 x 0.00057/40 = 10.6 $x10^{-6}$

Steady state error=0.5 degrees, Settling time=0.1 seconds, Rise time=12 milliseconds

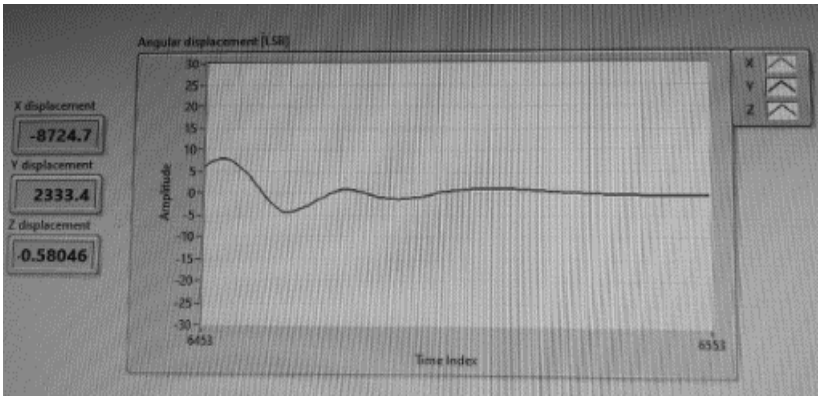

Fig. 11. PID Response of system to Disturbance

3.4 A* Path Planning Output

Fig. 12 shows A* algorithm output on LabVIEW. For implementing A* algorithm, the map input is created using a two dimensional Boolean array as shown in Fig. 12. Four numeric inputs above the map are for providing the start and end coordinates for the path planning. The blue coloured line in the intensity graph shows the path planned based on A* Algorithm and the arrays to the right of the intensity graph stores the coordinates of the path to be travelled as per route decided by the algorithm. Thus the Robot moves in the desired path without collision with obstacles as shown in the Fig. 12.

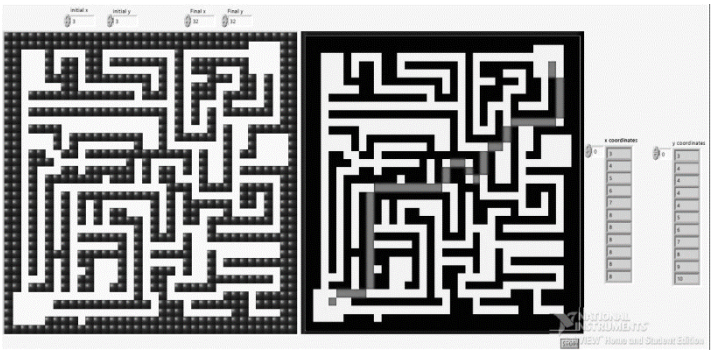

Fig. 12. End result of the A* path planning algorithm

4.0 Conclusion

A two wheeled self-balancing robot was designed using PID control algorithm with an inertial measurement unit in the feedback loop. Sensor fusion was achieved by using complementary filter, accelerometer and gyroscope. High accuracy of tilt angle measurement and overall stability was achieved. A* path planning algorithm was implemented in LabVIEW for finding shortest path for travel from point to point.

Reference

1. C Iwendi, M A Alqarni, J H Anajemba, A S Alfakeeh, Z Zhang, A K Bashir, Robust Navigational Control of a Two-Wheeled Self-Balancing Robot in a Sensed Environment, *IEEE Access*-2019, 7, 82337-82348

2. S Kim, C K Ahn, Self-Tuning Position-Tracking Controller for Two-Wheeled Mobile Balancing Robots, *IEEE Transactions on Circuits and Systems II: Express Briefs*-2019, 66 (6), 1008-1012

3. H Juang, K Lurrr, Design and control of a two-wheel self-balancing robot using the arduino microcontroller board, *10th IEEE International Conference on Control and Automation*-2013, 634-639

4. X Wang, S Chen, T Chen, B Yang, Study on control design of a two-wheeled self-balancing robot based on ADRC, *35th Chinese Control Conference, Chengdu*-2016, 6227-6232

5. O Jamil, M Jamil, Y Ayaz, K Ahmad, Modeling, control of a two-wheeled self-balancing robot, *International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE) Islamabad*-2014, 191-199

6. S Sarathy, M M M Hibah, S Anusooya, S Kalaivani, Implementation of Efficient Self Balancing Robot, *International Conference on Recent Trends in Electrical, Control and Communication (RTECC) Malaysia*-2018, 65-70

7. H Y Han, T Y Han, H S Jo, Development of omnidirectional self-balancing robot, *IEEE International Symposium on Robotics and Manufacturing Automation (ROMA) Kuala Lumpur*-2014, 57-62

8. T Feng, T Liu, X Wang, Z Xu, M Zhang, S Han, Modeling and implementation of two-wheel self-balancing robot equipped with supporting arms, *6th IEEE Conference on Industrial Electronics and Applications Beijing*-2011, 713-718

9. S Kwon, S Kim, J Yu, Tilting-Type Balancing Mobile Robot Platform for Enhancing Lateral Stability, *IEEE/ASME Transactions on Mechatronics*-2015, 20 (3), 1470-1481

10. H Bin, L W Zhen, L H Feng, The Kinematics Model of a Two-Wheeled Self-Balancing Autonomous Mobile Robot and Its Simulation, *Second International Conference on Computer Engineering and Applications Bali Island*-2010, 64-68