EasyChair Preprint

№ 2067

Path Tracing in holonomic drive system with Reduced Overshoot using rotary encoders

Divyanshu Tak, Ayush Jain, Paras Savnani and Akash Mecwan

December 1, 2019

Path Tracing in holonomic drive system with Reduced Overshoot using rotary encoders

Divyanshu Tak,
Electronics and Communicaton Branch,
Institute of technology, Nirma University, Ahmedabad, India.
16bec037@nirmauni.ac.in

Ayush Jain,
Electronics and Communication Branch,
Institute of technology, Nirma University, Ahmedabad, India.
mr.ayush141@gmail.com

Paras S Savnani,
Mechanical Branch,
Institute of technology, Nirma University, Ahmedabad, India.
savnani5@gmail.com

Dr. Akash Mecwan,
Electronics and Communication Branch,
Institute of technology, Nirma University, Ahmedabad, India.
Akash.mecwan@nirmauni.ac.in

***Abstract*—This paper focuses on path tracing of a holonomic three wheel Omni drive system. Any path (curved, linear) is first sampled, where the coordinates of the path are stored in the local memory, at particular sampling rate which are then retrieved dynamically during the runtime to move the robot on the similar path. The task of moving the drive system from one coordinate to another is done by Deduced reckoning algorithm. This paper also proposes algorithm for minimizing overshoot (deviation from actual path) while moving in a curved trajectory. The proposed algorithm helps in keeping track of the robot very efficiently. The hardware employed to test the proposed algorithm used consists of a three wheeled Omni base, three high torque DC motors, three 6" Omni wheels, highly efficient DC motor drivers, high precision rotary encoders and a microcontroller powered by ARM® Cortex®-M3 processor. The robot controlled by the proposed algorithm was tested on wooden field while plying on curved paths.**

Keywords—Omni Direction, IMU, Rotary Encoders, Holonomic Drive, Deduced Reckoning.

I. INTRODUCTION

Holonomic drive systems[5] are primarily used in robotics and Omni directional movement applications , as the robot can move in any direction and also change its direction of motion without changing its orientation beforehand , this feature allows for better maneuverability in curved paths .This paper presents integration of path tracing in holonomic drive[8] with algorithm for minimum overshoot of the robot which provides better solutions for applications where reconfiguring the path of the robot, according to the changes in the conditions, is not possible. Path tracing[8] is done in two steps, the first being the sampling of the path where any path is broken down into straight lines and the corresponding coordinates, orientation and other metadata is stored. Then the robot is moved along straight lines, coordinates of which were stored during sampling along with other data using Deduced reckoning algorithm which is integrated with minimum overshoot algorithms to provide robustness and independence from any changes in the surface of the path. Along with providing robustness against changes the minimum overshoot algorithm also provides smoothness and stability to the system, as holonomic drives are prone to overshoot while changing direction of motion. The algorithms were tested on a Three wheel base made of mild-steel, electronic hardware involved microcontroller [11], gyroscope [10], encoder, motors [2].The existing methods involve the use of path/curve equations which govern the heading of the bot along the curved path but have a problem of overshoot due to slip, inertia and inability of motors to suddenly accelerate and decelerate, this prevents the bot to perform smooth path tracing. The seamless integration of deduced reckoning and minimum overshoot algorithm provide better results for most robotics applications as it enables the bot to reduce overshoot by performing sudden acceleration and deceleration where sharp turns are required.

THREE WHEEL OMNI DRIVE

A. Maneuverability

As the word 'OMNI' (meaning- everywhere) suggests, the drive can move everywhere. But what stands out is the ability to maneuver sharp turns with or without changing the orientation. The drive developed as such can move in any direction without changing the orientation as well as it can move in a straight line while changing its direction, such that when both of these properties are combined, the developed drive system proves better performance around curved trajectory along with straight trajectory and ability to move in any direction.

B. Omni wheels

Holonomic drive systems requires special type of wheels , the wheels used in tank drive systems (commercial automobiles) cannot be used as the wheels used in these drives can be moved freely and without hindrance only in one direction , i.e.: forwards , backwards not along the axis of the wheels. Omni wheels are more suitable for holonomic applications as these wheels have beads/rollers attached on the circumference of the wheel to facilitate motion parallel to the axis of the wheel as shown in figure 1. We shall use the term 'rollers' throughout the paper to maintain clarity and conciseness. The rollers on the wheel rotate on their axis thus allowing the wheel to move parallel to its axis. This type of wheels can be acquired from any robotics shop or online vendors.

Figure 1 Omni Wheel

For higher controllability, the use of smaller wheels is advised, whereas higher speeds can be achieved using bigger wheels provided that the motors used can provide enough torque. We observed that there is a trade-off between speed and controllability, and the wheel size should be selected accordingly. The robot used for testing and verification of this paper uses a 6' wheels for a combination of speed and higher controllability.

C. *Microcontroller*

The controller will be responsible for all the calculations and execution of commands by either issuing them to the motor driver controller or by producing the desired PWM outputs. The more number of motors in this structure strains on the use of a motor driver with inbuilt microcontroller which will accept the issued commands and execute them.

The controller must be chosen according to the specifications of the design and must have sufficient pins and ports along with the hardware according to the designed skeleton. For example, if the drive is being controlled with an analog joystick, then the main controller must have an inbuilt ADC, if the feedback of angle is taken using Inter Integrated Circuit (I2C) then it is preferable to use a controller with inbuilt hardware I2C circuit. If the motor controller accepts commands using some standard communication protocol, then the hardware required for communication is also required to be taken into consideration. Any oscillator generating a clock above 1MHz will suffice for the calculations required to drive and control the structure.

D. *Gyroscope for feedback*

Feedback is added in the system to correct the small errors caused due to inertia or unbalanced system or unequal rpm of the motors. This feedback helps in maintaining the orientation of the robot while moving. The said feedback is in terms of current angle. The control algorithm then notes the current angle of the robot and makes the adequate changes in the speed of individual wheels to make the robot maintain its desired orientation. The feedback is quite necessary for driving three-wheel Omni drive and hence, the sensor providing the feedback must be as accurate as possible.

The use of gyroscope for angle feedback has proved to be very useful due to its accurate output and relatively error free transfer of data (it does not send data using analog values which will lead to quantization error). The gyroscope used for verification and test purpose was MPU6050 which works on I2C protocol for data transfer giving the change in angle between consecutive fetch cycles.

The new angle is calculated by adding the difference in angle acquired from the gyroscope to the present angle as,

$$\theta = \theta + \partial\theta \quad (1)$$

II. DRIVE CONTROL

The control of three wheel Omni drive rests upon basic but powerful trigonometric equations. In this drive system each wheel has separate control equation which is responsible for controlling the speed of that particular wheel relative to other two wheels to make the robot move in a particular direction

A. *Omni Drive*

Equations are a must as far as we want to drive our robot with/without control section embedded in it. These equations are not as simple as one might think and neither are they very hard to derive. Since our structure is not conventional with all wheels parallel to each other, the equations must include trigonometric components.

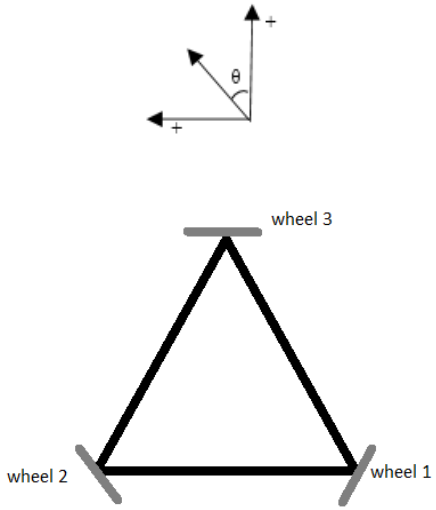

Figure 2 Robot Structure and convention

As shown in figure 2, we shall regard the wheel in lower right corner as wheel 1, the wheel in lower left corner as wheel 2 and the wheel on top as wheel 3. The axes are also shown alongside the figure with θ measuring from x-axis to y-axis. The convention is chosen for simplicity, one might completely ignore this convention to adapt a new convention and still, in essence, arrive at the same equations. For the following derivation, we assume that the heading of robot remains same and the equations can be tweaked for changing the heading and orientation of the robot.

Let us assume that we want our robot to move at an angle θ to x-axis with speed 'V' keeping its orientation unchanged. The angle θ as well as the speed are variables and so we can solve the equations without having to re-check them for different values.

The velocity at an angle θ, when broken into components along and perpendicular to wheel 3, resolves as $V.cos\theta$ î + $V.sin\theta$ ĵ, where î and ĵ are unit vectors along x and y axis respectively. As these wheels cannot apply force perpendicular to them, we shall equate them with components parallel to them. Thus,

$$v3 = v * sin\theta \quad (2)$$

Where $V3$ is velocity of wheel three. Similarly, when we resolve the along wheel 1 and 2, we get

$$v1 = v * \cos(\theta + 30) \quad (3)$$
$$v2 = v * \cos(\theta - 30) \quad (4)$$

These equations when simplified, give us three equations in which we have to substitute the value of speed and the angle at which the robot is supposed to move. The final equations are:

$$v1 = -\left(\frac{1}{2}\right) * v * sin\theta + \left(\frac{\sqrt{3}}{2}\right) * v * cos\theta \quad (5)$$
$$v2 = \left(\frac{1}{2}\right) * v * sin\theta + \left(\frac{\sqrt{3}}{2}\right) * v * cos\theta \quad (6)$$
$$v3 = v * sin\theta \quad (7)$$

(5), (6) and (7) are the ones commonly used and quite efficient when used with feedback to stop the unwanted drift in orientation.

B. Deduced Reckoning using rotary encoders

The holonomic drive is quite versatile due to its ability to change its direction of motion without changing its orientation and vice versa. But it is this versatility which makes it quite difficult to precisely calculate its coordinates using laser sensors and other non-contact based distance measurement techniques as it would require the robot to know its surroundings even before it sets sail. As the robot turns the walls around it might change in unexpected manner which shall require pre-known map of the surroundings as well as some quite complex equations and very precise measurements. To avoid this scenario altogether we have used rotary encoders even if they have drawback of incremental error along with the need to be continuously polled for change. But these drawbacks are overcome by its supreme advantage which is absolute reference which is independent of change in surroundings.

For simplicity, we shall consider the orientation of the robot to be constant while calculating the current coordinates of the robot. For The calculation let us assume that 'pulse_vert' is the number of pulses from the encoder facing the heading of the robot and 'pulse_horz' is the number of pulses from the encoder perpendicular to the heading of the robot. Thus, we can write it as:

$$\Delta dis_{vert} = Pulse\_vert * dis_{perpulse} \quad (8)$$
$$\Delta dis_{horz} = Pulse\_horz * dis_{perpulse} \quad (9)$$

Where, $dis_{perpulse}$ is the distance travelled by the encoder per pulse. It can easily be calculated as:

$$dis_{perpulse} = \frac{2\pi R}{Ppr} \quad (10)$$

Where R is the radius of the encoder and PPR stands for Pulse Per Revolution of the rotary encoder.
Now that we know about the distance travelled by robot in the form of Δdis_{horz} andΔdis_{vert}, we can update the current coordinates of the robot by following equations

$$X_{dis} = X_{dis} + \Delta dis_{vert} \quad (11)$$
$$Y_{dis} = Y_{dis} + \Delta dis_{horz} \quad (12)$$

C. Travelling in Straight Line

Using (8),(9),(11),(12) we are able to get the current coordinates of the robot travelling in any direction as the vector addition of horizontal and vertical components will always equal the distance travelled by the robot .

Now, let us assume that the robot is currently at coordinates (X1, Y1) and the destination coordinates are (X2, Y2).

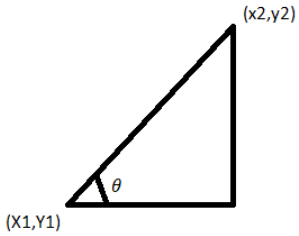

Figure 3 Straight line path

Using the Triangulation method depicted in Figure 3, the straight line between two coordinates represents the shortest path and θ represents the heading of the robot for travelling in straight line.

$$\theta = \tan^{-1}\left(\frac{Y2-Y1}{X2-X1}\right) \quad (13)$$

The θ calculated from (13) can be substituted in (5), (6), (7) to get the velocity values for individual wheels. By providing the speed values to the wheels the robot can be moved in straight line from (X1, Y1) to (X2, Y2).

D. Minimum Overshoot

Ideally, Speed and angle values calculated from (13), (5), (6), (7) at the starting of motion will make the robot reach the destination. But in Practical Scenario, because of irregularity in surface, different friction values for each wheel and uneven weight distribution the three wheel Omni drive system becomes prone to instability and often deviates from the path , this is where minimum overshoot algorithm comes into play.

The minimum overshoot algorithm comprises of mainly two parts namely, Path Overshoot, Speed Control. This algorithm is applied parallel to the perpetually running PID algorithms with the angle provided from gyroscope as the error variable to maintain the orientation of the robot during the motion and not allowing the robot to spin along the perpendicular axis of the wheel base.

1. Path Overshoot

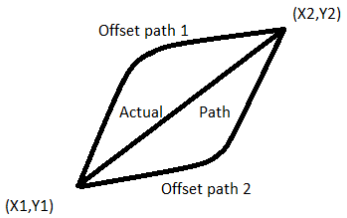

Figure 4 offset paths for straight line

Offset paths 1 and 2 depicts the possible overshoot that can occur during straight line motion. Taking help of Triangulation method in Figure3 and (13) to calculate $\Delta\theta$ Which is the offset heading.

$$\theta = \theta - \Delta\theta \quad (14)$$
$$\theta = \theta + \Delta\theta \quad (15)$$

Now, if the robot is on the offset path 1 then offset heading calculated can be substituted in (14) correct the path overshoot making the robot move on actual path as shown in Figure4 similarly , if the robot is on offset path 2 (14) can be used to correct the path error.
For further decreasing the overshoot the above mentioned correction method should be applied periodically or be placed in the main loop where the drive velocity is controlled for individual wheels , this will prevent path error to accumulate over the period of the motion resulting in closest path to the ideal straight line .

2. Speed Control

For speed control let us introduce some new variables 'x_start' , 'x_end' , 'y_start' , 'y_end' , 'diff_x' , 'diff_y'.

$$diff\_x = x\_end - x\_start \quad (16)$$
$$diff\_y = y\_end - y\_start \quad (17)$$

As indicated by the names variables 'x_start' , 'y_start' are updated at the starting of the main control loop using (11),(12) and similarly 'y_start' , 'y_end' , 'diff_x' , 'diff_y' are calculated at the end of the loop using (11),(12),(16),(17). The variables 'diff_x' and 'diff_y' represent the distance travelled by the robot during the execution of the main loop in programme, here time taken by the main loop in the programme is taken as a reference, these variables are then fed into PID algorithm as error parameter which will increase or decrease the speed of the robot against the preset speed limit, in the form of limit values of 'diff_x' and 'diff_y', hence preventing the robot from fluctuating from desired speed ,as depicted in the following flow chart.

The rotary encoders are enabled and the heading, required speed, Kp, preset ticks are passed as input parameters to the system. Current encoder ticks serve as initial ticks (x_start,y_start) and as the flow of program progresses different subroutines ,sensor updates and interrupt service routines ,after this interval encoder ticks are again accepted as final ticks (x_final,y_final).The error is calculated between the preset ticks and the tick difference which is the difference of final ticks and initial ticks, upon this error, PID is applied and corrected speed is calculated.

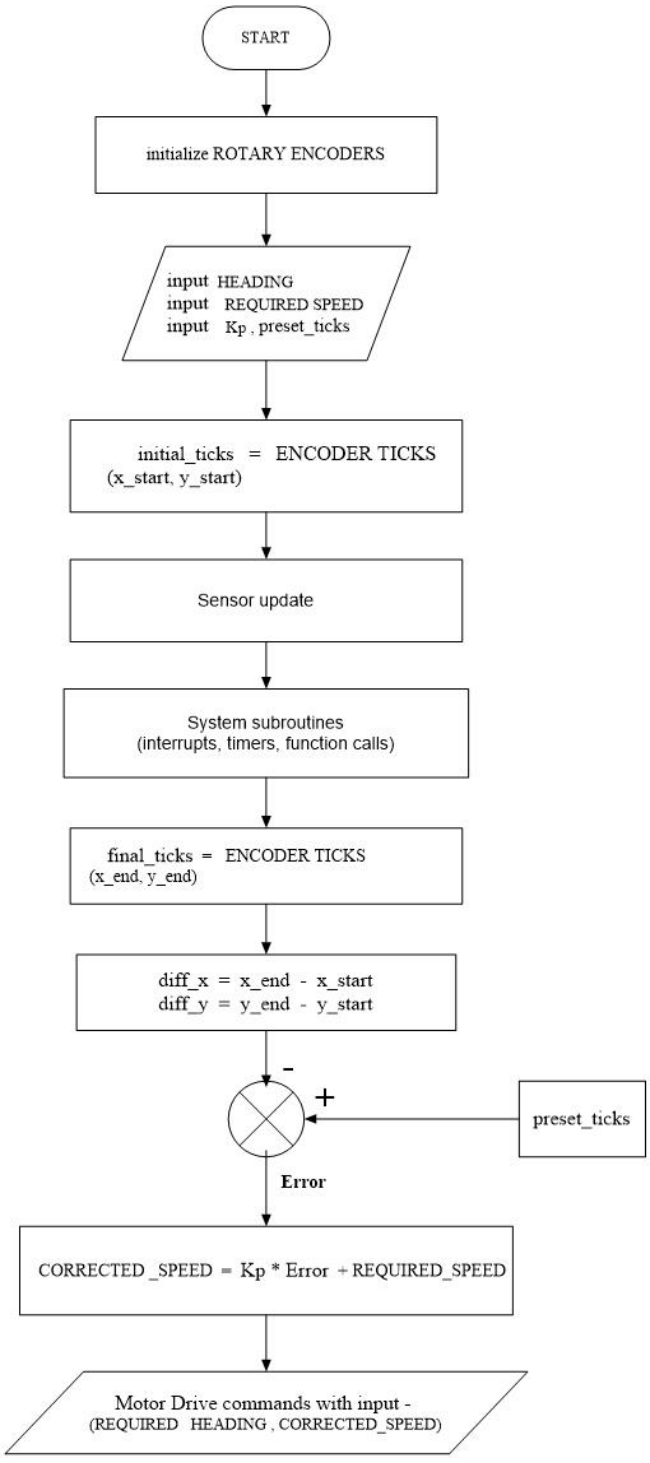

Figure 5 Flow chart for speed control

III. PATH SAMPLING

The idea behind path sampling is that every curved path can be formed from sequential juxtaposing of miniscule straight lines. Figure 6 depicts the curve made by combination of straight lines, the smoothness of the curve represented in Figure can be improved by increasing the sampling rate which

reduces the length of the straight lines, removes sharp edges and discontinuities hence resulting in a smooth curve.

Figure 6 Curve made of straight lines

The segmentation of the curve in straight lines helps to manoeuvre the robot into complicated curves by using simple and efficient straight line equations which consumes less computational power and provide reliability and robustness.

The path sampling is done by manually moving the robot along a curve path while simultaneously storing the coordinates using (11), (12) periodically at high frequency or high sampling rate, to increase the resolution and accuracy of the stored curve. After all the coordinates are stored Triangulation method and (13) are used , in the same sequence as the stored coordinates, to transform the distinct coordinate pairs into miniscule straight lines which eventually combine together to form a curve .

For applications with less memory constraints and more computational capabilities orientation of the robot at each coordinate and speed at each coordinates can also be stored to bolster the accuracy and reliability of the sampling.

IV. PATH FOLLOWING

Path following is done in robot by sequentially hopping from one straight line to another until the whole curve is traced. The motion in straight is performed by employing deduced reckoning algorithm along with reduced overshoot to accurately mimic the straight line path which is sampled.

V. RESULTS

The algorithms were tested on a Three wheel base made of mild-steel, electronic hardware involved microcontroller [11], gyroscope [10], encoder, motors [2]. Li-PO batteries (22.5V, 4500mAh).The surface involved a ply wood floor with metal/oil paint.

The robot weighed approximately 30 Kg. The initial tests involved analysis of performance without the use of algorithm, the robot was made to follow a ‘S’ shaped path, where sudden acceleration and deceleration were required for smooth direction change, but the robot could be accelerated from 0 to its maximum speed of 5 m/sec which required 2 sec pickup time due to slip of Omni wheels and low friction between wheels and floor, maintenance of desired constant speed was difficult. Upon implementation of Reduced Overshoot algorithm the pickup time and maintenance of constant desired speed was dramatically improved and the robot was able to decrease it’s speed from 5m/sec to 2m/sec in 1 sec during direction change and thus reducing the considerable effects of inertia easily in situations where sudden acceleration and deceleration are required like curve tracing.

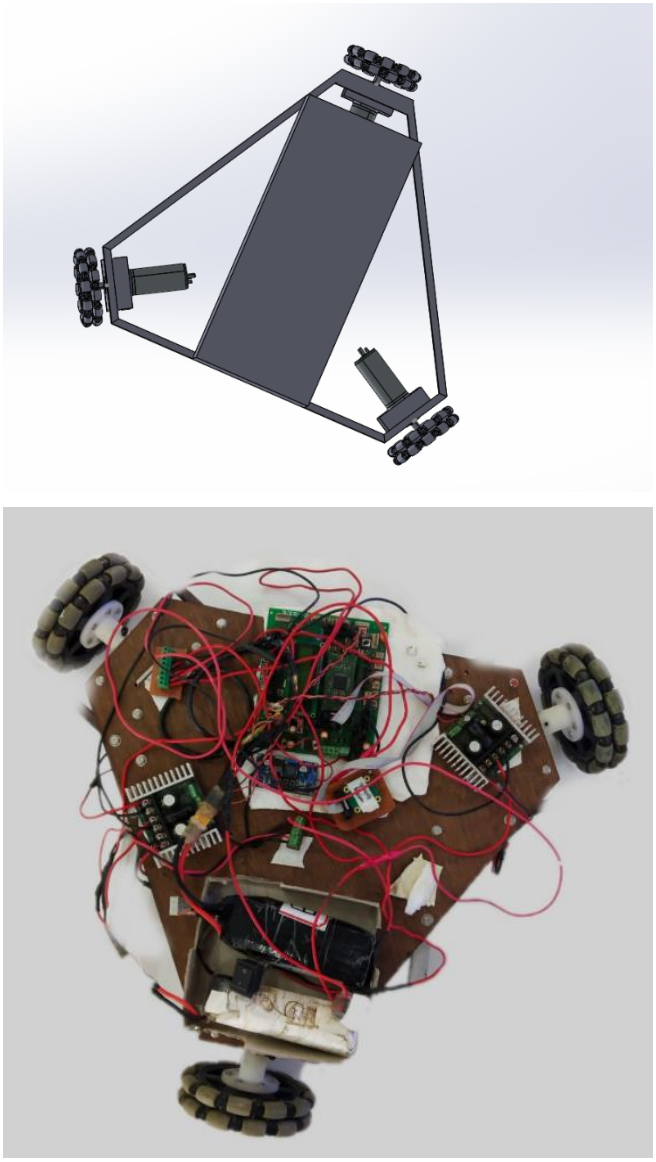

Figure 6 Software model and Real implementation

VI. CONCLUSION

The proposed algorithms when tested on specified hardware and structure provided improved results by damping the effects of inertia while changing paths and resulted in smooth motion of the robot throughout the curve. Our approaches were mainly focused on motion of three wheel Omni framework, but the proposed algorithms for speed reduction and curve tracing methods can also be implemented on other holonomic frameworks .The integration of proposed algorithm and three wheel framework along with specified hardware finds applications in various scenarios like industrial automation, domestic application robot for carrying

payloads from one place to another irrespective of the path between the two places. Another major benefit of the proposed method is that any path can be traced by only sampling the path once then the robot can adjust to the variations and changes that take place in the route dynamically in real time and without any reprogramming.

REFERENCES

[1] VexRobotics.com, 'Vex 6" omni directional wheel with Dual VersaKey Pattern', 2014. [Online]. Available: https://content.vexrobotics.com/vexpro/pdf/217-2585-Drawing20140818.PDF. [Accessed: 14 - Sept – 2017].

[2] -Maxon Motors, "Re 50, ø50, Graphite Brushes, 200W", May 2009.

[3] Dimensionengineering.com, 'Sabertooth 2x32 dual channel motor driver'.[Online].Available:https://www.dimensionengineering.com/datasheets/Sabertooth2x32.pdf . [Accessed: 23 – December – 2016].

[4] A. Pandey, S. Jha and D. Chakravarty, "Modeling and Control of an Autonomous Three Wheeled Mobile Robot with Front Steer," *2017 First IEEE International Conference on Robotic Computing (IRC)*, Taichung, 2017, pp. 136-142. doi: 10.1109/IRC.2017.67.

[5] H. Shah, K. Mehta and S. Gandhi, "Autonomous Navigation of 3 Wheel Robots Using Rotary Encoders and Gyroscope," *2014 International Conference on Computational Intelligence and Communication Networks*, Bhopal, 2014, pp. 1168-1172.

[6] Liu Peng-yu, He Yong-yi, "Omni-directional opera robot motion planning based on wavelet", *International Conference on Artificial Intelligence and Education*, pp. 525-528, Oct 29–30 2010.

[7] S. Ziaie-Rad, F. Janabi-Sharifi, M. M. Danesh-Panah, A. Abdollahi, "A practical approach to control and self-localization of Persia omni directional mobile robot", *International Conference on Intelligent Robots and Systems*, pp. 3473-3479, Aug 2–6 2005.

[8] Mark Ashmore, Nick Barnes, "Omni-drive Robot Motion on Curved Paths: The Fastest Path between Two Points Is Not a Straight- Line", *Proc. 15th Australian Joint Conference on Artificial Intelligence Canberra*, pp. 225-236, Dec 2–62002.

[9] M. H. Moradi, "New techniques for PID controller design," *Proceedings of 2003 IEEE Conference on Control Applications, 2003. CCA 2003.*, 2003, pp. 903-908 vol.2. doi: 10.1109/CCA.2003.1223130.

[10] L. Abraham and M. A. Babu, "Analysis of MEMS gyro sensors ADXRS 450 and ADXRS 649 using LabVIEW," *2014 First International Conference on Computational Systems and Communications (ICCSC)*, Trivandrum, 2014, pp. 144-149.

[11] NXP semiconductors - nxp.com, 'LPC1769 32-bit ARM Cortex-M3 microcontroller', May – 2017. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/LPC1769_68_67_66_65_64_63.pdf.